БАКАЛАВРИАТ И МАГИСТРАТУРА

МОДЕЛИРОВАНИЕ ХИМИКО-ТЕХНОЛОГИЧЕСКИХ ПРОЦЕССОВ

ПРИНЦИПЫ ПРИМЕНЕНИЯ ПАКЕТОВ КОМПЬЮТЕРНОЙ МАТЕМАТИКИ

Т. Н. Гартман, Д. В. Клушин

www.e.lanbook.com
ЭБС ЛАНЬ

Т. Н. ГАРТМАН, Д. В. КЛУШИН

МОДЕЛИРОВАНИЕ ХИМИКО-ТЕХНОЛОГИЧЕСКИХ ПРОЦЕССОВ

ПРИНЦИПЫ ПРИМЕНЕНИЯ ПАКЕТОВ КОМПЬЮТЕРНОЙ МАТЕМАТИКИ

ДОПУЩЕНО
Федеральным учебно-методическим объединением
по укрупненной группе специальностей
и направлений подготовки «Химические технологии»
в качестве учебного пособия для студентов высших учебных заведений,
обучающихся по направлению подготовки «Химическая технология»

ЛАНЬ®
САНКТ-ПЕТЕРБУРГ
МОСКВА
КРАСНОДАР
2020

УДК 66
ББК 35я73

Г 21 **Гартман Т. Н.** Моделирование химико-технологических процессов. Принципы применения пакетов компьютерной математики : учебное пособие / Т. Н. Гартман, Д. В. Клушин. — Санкт-Петербург : Лань, 2020. — 404 с. : ил. — (Учебники для вузов. Специальная литература). — Текст : непосредственный.

ISBN 978-5-8114-3900-3

В книге на примере пакета MATLAB рассмотрены основные аспекты применения современных пакетов компьютерной математики для моделирования химико-технологических процессов. Приведено описание интерпретируемого языка программирования MATLAB и на различных примерах проиллюстрированы его функциональные возможности. С использованием решателей (solvers) пакета MATLAB представлены программные коды решения многочисленных задач вычислительной математики и задач разработки компьютерных моделей химико-технологических процессов. Проанализированы возможности применения рассмотренных решателей MATLAB для решения типовых задач вычислительной математики численными методами.

Книга предназначена для изучения методов применения пакетов компьютерной математики, в частности пакета MATLAB, для технологических расчетов, математического моделирования и оптимизации при проектировании и управлении химико-технологическими процессами. Рекомендуется в качестве учебного пособия студентам и магистрам, обучающимся по направлениям подготовки и специальностям, входящим в УГСН «Химия», «Химические технологии», а также может быть полезна аспирантам, инженерам-химикам-технологам, занимающимся расчетами и расчетными исследованиями технологических процессов.

УДК 66
ББК 35я73

Рецензенты:
В. М. АРИСТОВ — доктор физико-математических наук, профессор Российского химико-технологического университета им. Д. И. Менделеева;
А. В. ТИМОШЕНКО — доктор технических наук, профессор кафедры химии и технологии основного органического синтеза, проректор по учебной работе МИРЭА — Российского технологического университета.

Обложка
П. И. ПОЛЯКОВА

ПРЕДИСЛОВИЕ

Применение современных подходов к исследованию технологических процессов, оптимизации энергоресурсосберегающих производств и их проектированию предполагает реализацию методов компьютерного моделирования как одного из важнейших инструментов, обеспечивающих получение достоверных решений перечисленных задач. Отличительная особенность процедуры компьютерного моделирования состоит в том, что для корректного применения вычислительных машин необходимо решить целый комплекс взаимосвязанных задач: математического моделирования технологических процессов, идентификации параметров математических моделей и определения оптимальных условий протекания процессов в производственных условиях. Основные аспекты решения задач такого типа для химико-технологических процессов были рассмотрены нами в учебном пособии: Гартман Т. Н., Клушин Д. В. Основы компьютерного моделирования химико-технологических процессов : учеб. пособие для вузов. — М. : Академкнига, 2008. — 416 с.

Практическое применение методов решения задач моделирования химико-технологических процессов возможно с использованием расчетных методов, которые в общем случае относятся к классу численных методов вычислительной математики. Эти методы широко представлены в настоящее время в так называемых системах компьютерной математики (СКМ), которые оформляются в виде пакетов компьютерной математики (ПКМ). Основным достоинством ПКМ является то обстоятельство, что численные методы решения математических задач представлены в них в виде решателей (solvers), в которых реализовано один или несколько алгоритмов решения типовых задач вычислительной математики. Такие алгоритмы разрабатываются, как правило, высококвалифицированными специалистами-математиками с учетом последних достижений в области численных методов вычислительной математики. В свою очередь, специалисты в предметных областях, в частности в химии и химической технологии, при расчетах могут концентрировать свое внимание на постановке и оценке результатов решения моделирования физико-химических задач, прилагая наименьшие усилия, связанные с решением вычислительных задач и их программированием на компьютерах.

Задачей настоящей книги является краткое ознакомление читателя с достоинствами и недостатками известных пакетов компьютерной математики (ПКМ) (глава 1), а также с принципами их применения при компьютерном моделировании химико-технологических процессов на примере ПКМ MATLAB (главы 2 и 3).

Настоящую книгу следует рассматривать как учебное пособие для химиков-технологов по применению пакета MATLAB при моделировании химико-технологических процессов. Поэтому книга написана на языке не специалиста в области программирования и вычислительной математики, а традиционного инженера, в частности инженера-химика и химика-технолога, и одна из глав посвящена описанию языка программирования пакета MATLAB (глава 2), а другая — применению решателей этого пакета для решения типовых задач

вычислительной математики и отдельных задач компьютерного моделирования химико-технологических процессов (глава 3).

В задачу настоящего пособия также входит ознакомление читателя с принципами корректного выбора и применения численных методов вычислительной математики (решателей пакета MATLAB) для решения задач компьютерного моделирования. Необходимо подчеркнуть, что, несмотря на то что математическое описание некоторых рассматриваемых в пособии процессов и явлений представлено в упрощенном виде, приведенные алгоритмы расчета и соответствующие реализованные программы с программными кодами языка программирования пакета MATLAB являются универсальными, легко масштабируемыми на более полные и точные математические описания процессов.

Многочисленные примеры решенных типовых задач программирования, вычислительной математики и моделирования химико-технологических процессов с подробным обсуждением постановки задачи, а также описанием и предоставлением программных кодов алгоритмов решения, анализом результатов решенных задач должны способствовать усвоению материала, представленного в пособии, и получению практических навыков работы с пакетом MATLAB при компьютерном моделировании.

Книга состоит из трех глав.

В главе 1 представлен краткий обзор основных пакетов компьютерной математики, используемых при компьютерном моделировании химико-технологических процессов, перечислены наиболее употребляемые численные методы вычислительной математики, применяемые в этом случае, а также дана характеристика одного из наиболее известных пакетов компьютерной математики (ПКМ) MATLAB.

В главе 2 приведены основные правила работы с пакетом MATLAB и изложены основы программирования на высокоуровневом интерпретируемом языке программирования этого пакета. Материал изложен в объеме, необходимом для применения пакета пользователем, имеющим ограниченные знания в области программирования.

Глава 3 посвящена описанию применения некоторых решателей пакета MATLAB для реализации численных методов вычислительной математики и их применению при компьютерном моделировании химико-технологических процессов. С использованием тестовых математических и расчетных химико-технологических задач показано, как применять решатели пакета MATLAB при решении типовых задач вычислительной математики и компьютерном моделировании химико-технологических процессов. Приведены программные коды решаемых с применением MATLAB задач, которые могут быть адаптированы пользователями для собственных целей.

В Приложении приведены результаты решения 16 задач компьютерного моделирования химико-технологических процессов в табличном виде, программные коды алгоритмов решения которых представлены на соответствующих рисунках в главе 3.

Авторы выражают надежду, что изучение представленного в пособии материала позволит читателям — студентам, магистрам и аспирантам, а также

специалистам-химикам и химикам-технологам, получить и углубить свои знания в области применения пакетов компьютерной математики, в частности пакета MATLAB, для реализации численных методов вычислительной математики при моделировании химико-технологических процессов.

Т. Н. Гартман, Д. В. Клушин

ГЛАВА 1. КРАТКАЯ ХАРАКТЕРИСТИКА СОВРЕМЕННЫХ ПАКЕТОВ КОМПЬЮТЕРНОЙ МАТЕМАТИКИ

1.1. Пакеты компьютерной математики, их разновидности и основные функциональные элементы

В последние 60 лет наблюдается бурное развитие мощностей цифровых компьютеров, резкое увеличение их быстродействия, с одновременным развитием систем отображения информации, что открыло возможность интерактивного графического взаимодействия исследователя и ЭВМ. За это время на компьютерах реализованы всевозможные модели процессов и явлений в различных областях знаний, что привело к разработке огромного количества программ численного решения типовых задач, освоение которых из-за сложности и трудоемкости уже не под силу каждому отдельному пользователю. В свою очередь, это потребовало изучения пользователями различных численных методов вычислительной математики, языков программирования и новых проблемно-ориентированных программных средств. Поэтому в настоящее время параллельно с развитием цифровых компьютеров и языков программирования ведутся работы по созданию проблемно-ориентированных инструментальных средств моделирования, в частности мощных специализированных систем компьютерной математики (СКМ), которые позволяют упростить реализацию математических моделей процессов и явлений на ЭВМ. На смену традиционным специализированным комплексам программ приходят пакеты компьютерной математики (ПКМ), которые позволяют решать всевозможные задачи, в том числе задачи моделирования в их исходной постановке, наиболее просто и, что особенно важно, с наименьшими затратами времени на программирование. Среди наиболее известных пакетов такого типа следует отметить MATLAB, Mathcad, Derive, Maple, Mathematica, FEMLAB, Python (Scipy + Numpy), GAMS.

Современные пакеты компьютерной математики (ПКМ) представляют собой программные продукты для решения разнотипных математических задач, в том числе и достаточно сложных, в которых используются последние научные достижения в области вычислительной математики, символьных вычислений, математического моделирования и оптимизации. ПКМ позволяют в полной мере реализовать преимущества передовых систем компьютерной математики (СКМ) и располагают эффективной интегрированной операционной средой, включающей полный набор средств для наиболее рациональных методов формулировки и решения широкого спектра математических задач и эффективного представления результатов вычислений. Они, как правило, используют собственный интерпретируемый язык программирования с синтаксисом, частично напоминающим известные универсальные языки программирования, обширную библиотеку решателей (solvers) для реализации численных методов вычислительной математики, развитые графические средства визуализации данных, совместимые с различными другими известными пакетами программ и

языками программирования, которые применимы для решения всевозможных математических задач в самых разных областях знаний и техники.

Интегрированная операционная среда типичного пакета компьютерной математики (ПКМ) обычно включает (рис. 1.1):

- интегрированную среду разработки;
- собственный или адаптируемый интерпретируемый язык программирования высокого уровня;
- развитые средства двухмерной и трехмерной графики;
- собственную и расширяемую библиотеку решателей для реализации численных методов вычислительной математики и оптимизации;
- библиотеку алгоритмов символьной математики;
- средство визуального объектно-ориентированного программирования для моделирования специальных систем, например Simulink в ПКМ MATLAB;
- интерфейсы взаимодействия: а) с различными универсальными языками программирования высокого уровня; б) с другими пакетами компьютерной математики (ПКМ); в) со всевозможными сторонними решателями для решения типовых задач вычислительной математики и оптимизации задач; г) с симуляторами (пакетами моделирующих программ (ПМП)), как, например, CHEMCAD для моделирования химико-технологических процессов).

Рис. 1.1

Основные функциональные элементы современного пакета компьютерной математики (ПКМ)

Интегрированная операционная среда ПКМ предназначена для организации взаимодействия между ее различными функциональными элементами и обеспечения пользователю возможности рационального и эффективного использования всех функциональных возможностей пакета компьютерной математики.

Интегрированная среда разработки служит для предоставления пользователю удобных графических средств взаимодействия с функционалом ПКМ и облегчения обучения исследователя для наиболее эффективного использования предоставляемых средств разработки.

Интерпретируемый язык программирования включает основные атрибуты универсальных языков программирования высокого уровня с собственным расширением отдельных операционных модулей и файлов, которые могут оформляться в виде скриптов и функций. Пользователю дается возможность как создавать (корректировать) отдельные файлы, так и библиотеки файлов, для решения конкретных специфических задач, что привело к созданию большого числа пакетов прикладных программ (toolboox) с использованием конкретного ПКМ для решения различных задач науки и техники.

Средства двумерной и трехмерной графики позволяют предоставить информацию об изучаемых свойствах объектов в наиболее удобно воспринимаемой человеком графической форме, что значительно упрощает нахождение зависимостей и трендов рассматриваемых параметров.

Библиотека решателей позволяет применять сложные алгоритмы численных методов вычислительной математики без необходимости их самостоятельного определения (программирования), что значительно повышает скорость разработки моделей и понижает порог вхождения для использования данного ПКМ исследователем-непрограммистом.

Библиотека символьной математики дает возможность получать общие решения различных уравнений, что позволяет исследователю сконцентрироваться на ключевых задачах моделирования процессов и объектов за счёт автоматизации сложных математических преобразований.

К средствам визуального объектно-ориентированного программирования относится, например, пакет Simulink в ПКМ MATLAB, который служит для имитационного моделирования систем, состоящих из графических блоков с заданными параметрами. Компоненты моделей являются графическими блоками, которые содержатся в ряде библиотек и с помощью мыши могут переноситься в основное меню и соединяться друг с другом требуемыми связями. Пакет основан на построении блочных схем путем переноса блоков компонентов в окно редактирования создаваемой пользователем модели.

Интерфейсы взаимодействия необходимы для предоставления возможности использовать другие ПКМ, которые наиболее подходят для решения подзадач моделирования, и впоследствии объединить их для получения единой модели.

Краткая сравнительная характеристика ПКМ различных типов приведена в таблице 1.1.

Таблица 1.1

Краткая сравнительная характеристика ПКМ различных типов

Система	Назначение и достоинства	Ограничения и недостатки	Совместимость
Derive	Умеренная математическая подготовка. Аналитические вычисления. Скромные требования к аппаратным ресурсам. Наличие русифицированных версий	Слабая графика и визуализация, отсутствие средств программирования. Слабая поддержка специальных функций в символьных расчётах	—
Mathcad	Система на все случаи жизни. Образцовый интерфейс. Ввод данных с помощью палитр математических знаков. Удачный набор операторов и функций. Множество примеров, электронных книг и библиотек	Ограниченные средства символьной математики. Примитивные средства программирования. Повышенные требования к аппаратным ресурсам	Может взаимодействовать с C/C++ и другими традиционными языками
Maple	Уникальное ядро символьных вычислений. Мощнейшая графика. Удобная справочная система. Развитые средства форматирования документов	Повышенные требования к аппаратным ресурсам. Ориентация на опытных пользователей и специалистов по математике	Возможность взаимодействия с C/C++, Fortran, Java, MATLAB и Visual Basic
Mathematica	Совместимость с разными компьютерными платформами. Уникальная трёхмерная графика. Развитые средства форматирования документов. Мировое лидерство.	Высокие требования к аппаратным ресурсам. Чрезмерная защита от копирования. Ориентация на опытных пользователей	Возможность использовать библиотеки, написанные на Java, .NET, C/C++ и Fortan
MATLAB	Научные расчёты, численное моделирование. Уникальные матричные средства, обилие численных методов, описательная графика, высокая скорость вычислений, лёгкость адаптации к задачам пользователя благодаря множеству пакетов расширения системы.	Чрезмерно высокие требования к аппаратным ресурсам. Скромные возможности символьных вычислений. Дороговизна как самой системы, так и пакетов расширения. Чрезмерная элитарность	Может непосредственно использовать библиотеки, написанные на C/C++, Fortran, Python, Perl, Java, ActiveX и .NET. Возможность генерации C/C++ кода непосредственно из MATLAB-кода

Продолжение табл. 1.1

Система	Назначение и достоинства	Ограничения и недостатки	Совместимость
FEMLAB	Пакет моделирования. Решает системы связанных нелинейных дифференциальных уравнений в частных производных методом конечных элементов в одном, двух и трёх измерениях. Простой и надёжный графический пользовательский интерфейс. Содержит специальные интерфейсы для химической технологии (Chemical Engineering)	Узкая направленность и малое время практической апробации системы	—
Python	Входит в тройку самых популярных языков программирования. Насчитывает огромное количество библиотек и расширений под любые задачи. Распространяется абсолютно бесплатно, в том числе и для коммерческого использования. Простой синтаксис и бесчисленное множество обучающих пособий	Для полноценного использования языка нужно обладать основами функционального и объектно-ориентированного программирования. Относительно медленные вычисления в циклах при больших объёмах данных	Python код может быть преобразован в C/C++, Java, CIL, VHCL, а также может быть вызван из семейства .NET. Программы, написанные на Python, могут быть скомпилированы в один исполняемый файл
GAMS	Высокоуровневая система для математической оптимизации. Совместимость с различными компьютерными платформами. Большое количество алгоритмов оптимизации сложных задач из различных областей. Наличие бесплатной версии с ограниченным функционалом	Малое количество обучающих пособий. Сложность решения задач, плохо описанных математически. Отсутствие средств визуализации	Имеет интерфейсы взаимодействия с такими языками, как C++, Java и Python, а также со средой .NET

Одним из важнейших преимуществ ПКМ является их относительно простое усвоение и обучение работе с ними специалистов в разнородных предметных областях, не являющихся профессиональными программистами. Практически все алгоритмы решения типичных задач вычислительной математики представлены в них в виде решателей, для применения которых специалисту или

научному работнику необходимо на используемом ПКМ языке программирования записать функцию, соответствующую постановке решаемой им технической или научной задачи. Для применения пакетов компьютерной математики инженеру необходимо изучить язык программирования, встроенный в ПКМ, и научиться пользоваться решателями для решения собственных математических задач.

Дальнейшие разработки в области усовершенствования пакетов компьютерной математики (ПКМ) должны привести к расширению их функциональных возможностей, использованию наиболее передовых компьютерных технологий для решения сложных математических задач и наиболее удобной форме их применения, в том числе и специалистами-непрограммистами.

1.2. Применение численных методов вычислительной математики при компьютерном моделировании химико-технологических процессов

Роль компьютерного моделирования в современных науке и технике настолько велика, что оно стало одним из основных инструментов научного познания и нашло широкое распространение при исследовании, оптимизации и управлении химико-технологическими процессами. В соответствии с системным подходом к моделированию процессов и свойствами комплексности, иерархичности и иерархической соподчиненности, при разработке компьютерных моделей наиболее распространенным является блочный подход. Блочный системный подход к созданию компьютерных моделей химико-технологических процессов предполагает разработку адекватных моделей процессов и явлений в отдельных блоках и ступенях иерархии совокупного процесса, а также учет взаимосвязей между блоками, так называемый процесс моделирования. При этом следует иметь в виду, что любая модель, в том числе и компьютерная, есть отражение существенных сторон реальной (или конструируемой) системы, в удобной форме представляющая информацию о ней.

Не менее важным аспектом моделирования является исследование разработанной адекватной компьютерной модели путем ее расчетных исследований и вычислительных экспериментов с ее использованием — так называемый процесс симуляции.

Необходимо подчеркнуть, что при компьютерном моделировании прежде всего речь идет о математическом моделировании с применением ЭВМ и что в этом процессе обязательно участвуют и взаимодействуют субъекты — специалисты в предметной области и в области математики, в частности вычислительной математики, а также объект исследования — оригинал и модель. В связи с этим нельзя забывать, что в большинстве случаев моделям присуща определенная доля субъективизма, поскольку практически в процессе исследования приходится иметь дело не с самим объектом (системой), а с представлениями о нем, т. е. с его моделью. Безусловно, по мере совершенствования модели и приближения ее к объекту объективная сторона модели становится преобладаю-

щей, происходит постепенное движение от относительной к никогда не достижимой абсолютной истине.

При рассмотрении общих методологических вопросов математического моделирования — основы компьютерного моделирования — доказана диалектическая связь причинности и случайности. Показано, что, несмотря на всеобщую причинную обусловленность процессов в реальных системах, причинность проявляется через случайность. Причем существует определенный уровень объективной случайности, раскрытие которой при данном уровне развития науки и техники может оказаться нецелесообразным, так как сопровождается непомерными затратами энергии и средств.

Все это должно найти отражение при построении математических моделей в виде определенного сочетания детерминистского и стохастического подходов. Естественно, соотношение этих подходов зависит от конкретных условий: степени знаний о процессе, его наблюдаемости (измеряемости), назначения модели и т. д.

Исходя из вышесказанного, в общем случае любая компьютерная модель должна быть детерминированно-вероятностной. Однако, несмотря на большое разнообразие подходов к моделированию химико-технологических систем, таких как, например, применение конечных автоматов, вероятностных марковских процессов, нейронных сетей, доминирующими остаются в настоящее время два подхода: теоретическое физико-химическое и эмпирическое регрессионное моделирование. Первый подход основан на знании механизмов протекающих процессов и на блочном принципе построения полной модели, а второй — на обработке большого объема экспериментальной информации о входных и выходных параметрах процесса и базируется на кибернетическом принципе «черного ящика» и применении методов регрессионного, корреляционного и дисперсионного анализа при построении моделей. На практике достаточно часто эти подходы приходится совмещать, так как математические физико-химические модели не всегда применимы для описания процессов на отдельных ступенях иерархии (для отдельных блоков) полной системы.

Необходимо подчеркнуть, что в настоящее время стремление к разработке теоретических физико-химических блочных математических моделей химико-технологических процессов преобладает, а регрессионные модели используются, когда знаний о теории и механизме протекающих процессов недостаточно.

При этом процесс моделирования можно представить в виде следующих этапов (рис. 1.2).

На всех перечисленных этапах компьютерного моделирования приходится решать расчетные задачи, причем, как правило, для решения используются численные методы вычислительной математики, что связано со сложностью решаемых задач и, как правило, их большой размерностью. При этом желательно выбирать оптимальные вычислительные алгоритмы, оформленные в различных пакетах компьютерной математики (ПКМ) в виде решателей с конкретным названием. Следует отметить, что отдельные решатели ПКМ могут включать в себя один численный алгоритм расчета, а другие — несколько раз-

личных алгоритмов, позволяющих решать конкретную задачу различными методами.

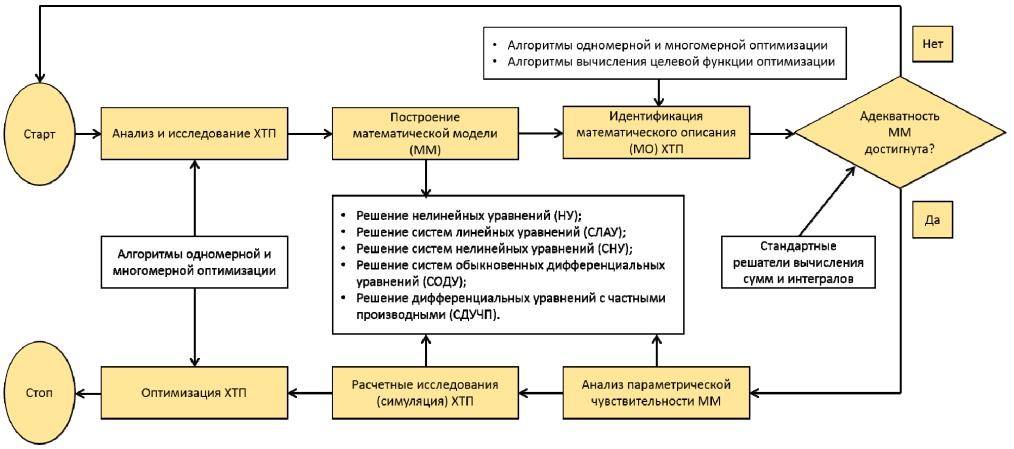


Рис. 1.2

Блок-схема процедуры компьютерного моделирования ХТП

Так, *на первом этапе при анализе и обработке экспериментальных данных* методами регрессионного и корреляционного анализа необходимо использовать алгоритмы одномерной и многомерной оптимизации.

На *втором этапе при построении математических моделей процессов* могут решаться следующие типовые задачи вычислительной математики:

- решение нелинейных уравнений (НУ);
- решение систем линейных уравнений (СЛАУ);
- решение систем нелинейных уравнений (СНУ);
- решение систем обыкновенных дифференциальных уравнений (СОДУ);
- решение дифференциальных уравнений с частными производными (СДУЧП).

На *третьем этапе решается задача структурной и параметрической идентификации* математического описания процесса, для чего необходимы алгоритмы одномерной и многомерной оптимизации для определения параметров (коэффициентов) моделей, а также алгоритмы вычисления целевой функции оптимизации в виде суммы или интеграла, оценивающих согласование расчетных и экспериментальных данных.

Четвертый этап связан с расчетом критерия рассогласования соответствующих расчетных по модели и экспериментальных данных, для чего необходимо использовать стандартные решатели вычисления сумм и интегралов.

На *пятом этапе происходит анализ изменения рассчитанных по модели результирующих режимных параметров в зависимости от колебания (изменения) параметров (коэффициентов) модели*. Для этого необходимы алгоритмы решения уравнений математического описания, перечисленные на втором этапе.

На *шестом этапе предполагается проведение расчетных исследований модели при изменении отдельных режимных параметров*, влияющих на состояние и результирующие показатели процесса, а также оценка этого влияния. Для этих целей также применяются алгоритмы, используемые на втором этапе процедуры моделирования.

Седьмой этап посвящен решению задачи оптимизации ресурсосберегающего процесса с выбранным критерием оптимальности (техническим, экономическим, термодинамическим и т. п.). Для решения этой задачи используются алгоритмы одномерной и многомерной оптимизации.

Таким образом, для применения пакетов компьютерной математики при моделировании химико-технологических процессов они должны располагать различными решателями для решения следующих типовых задач вычислительной математики численными методами:

- вычисление производных и интегралов (ИНТ);
- решение нелинейных уравнений (НУ);
- решение систем линейных (СЛАУ) и нелинейных уравнений (СНУ);
- решение систем обыкновенных дифференциальных уравнений (СОДУ) и дифференциальных уравнений с частными производными (СДУЧП);
- решение задач одномерной и многомерной оптимизации (ОПТ).

1.2.1. Пакет компьютерной математики MATLAB и применение его решателей для реализации численных методов вычислительной математики

Сложность решаемых с применением ЭВМ в настоящее время задач несопоставима с возможностями отдельного пользователя/программиста. Однако за время развития цифровых технологий наработано внушительное количество алгоритмов и программных продуктов для численного решения типовых задач разного рода. Такие программы интегрируются в специализированные пакеты программ, принимающие форму мощных специализированных систем компьютерной математики. Они позволяют решать, в частности, задачи моделирования материальных систем в их исходной постановке, с минимальными затратами времени на программирование или вовсе без таких затрат. Наиболее известные среди таких систем — MATLAB и MATHCAD, далее мы сосредоточимся на первой из них.

Система компьютерной математики (СКМ) MATLAB развивается долгое время и прошла путь от матричного программного модуля до универсального пакета компьютерной математики (ПКМ), ориентированного на работу на персональных компьютерах и имеющего мощные графические и диалоговые средства. Расширение этого ПКМ Simulink предоставляет удобные несложные средства, в частности визуального объектно-ориентированного программирования для блочного моделирования линейных и нелинейных динамических систем.

Достоинствами MATLAB являются открытость и расширяемость. Большинство команд и функций реализованы в виде m-файлов и файлов на языке C/C++, файлы доступны для модификаций. Пользователи могут создавать соб-

ственные отдельные файлы и библиотеки файлов для решения специфических задач. Такие адаптационные возможности MATLAB обусловили появление множества прикладных пакетов программ, расширяющих его возможности для решения научных и технологических задач.

MATLAB (матричная лаборатория) — один из наиболее проверенных временем комплексов автоматизации математических расчетов на основе матричных операций, которые широко используются для решения задач линейной алгебры и математического моделирования статических и динамических систем. Матричные операции являются основой автоматического составления и решения уравнений состояния химико-технологических систем.

В области математических вычислений среди возможностей MATLAB можно назвать:

- матричные, векторные, логические операторы;
- элементарные и специальные функции;
- полиномиальную арифметику;
- многомерные массивы;
- массивы записей;
- массивы ячеек.

В области численных методов MATLAB предоставляет возможность работы с:

- дифференциальными уравнениями;
- вычислениями одномерных и двухмерных квадратур;
- поиском корней нелинейных и полиномиальных алгебраических уравнений;
- оптимизацией функций нескольких переменных;
- одномерной и многомерной интерполяцией;
- прямыми обратными преобразованиями Фурье;
- решением уравнений с частными производными и др.

В области программирования ПКМ MATLAB имеет:

- свыше 600 встроенных математических функций;
- ввод/вывод двоичных и текстовых файлов;
- использование программ на C/C++ и Fortran;
- автоматическую перекодировку процедур в тексты программ на C и C++;
- типовые управляющие структуры.

Среди визуализационных возможностей MATLAB следует отметить:

- создание двухмерных и трехмерных графиков;
- визуальный анализ данных.

Именно благодаря графическим возможностям MATLAB осуществляется интерактивное взаимодействие пользователя с ПКМ в процессе моделирования. Расширение Simulink в составе MATLAB дает возможность имитировать реальные системы и устройства, задавая их моделями, состоящими из функцио-

нальных блоков. Имеется обширная и постоянно расширяемая пользователями библиотека блоков и средств задания и изменения параметров моделей.

К сервисным особенностям графической системы MATLAB относятся:

- расширенные возможности форматирования графики, в том числе форматирование линий, осей, маркеров опорных точек для графика нескольких функций, нанесение подписей и стрелок на график, построение легенды и шкалы цветов;
- перемещение графика в графическом окне;
- возможность создания нескольких графических окон;
- задание различных координатных систем и осей;
- широкие возможности использования цвета;
- простота построения трехмерных графиков с их проекцией на плоскость, построение сечений трехмерных фигур;
- возможность создания объектов для типового интерфейса пользователя.

Алгоритмический язык MATLAB является высокоуровневым интерпретируемым языком программирования, включающим основанные на матрицах структуры данных, широкий спектр функций, интегрированную среду разработки, объектно-ориентированные возможности и интерфейсы к программам, написанным на других языках программирования. Программы, написанные на MATLAB, бывают двух типов — функции и скрипты. Функции имеют входные и выходные аргументы, а также собственное рабочее пространство для хранения промежуточных результатов вычислений и переменных. Скрипты же используют общее рабочее пространство. Как скрипты, так и функции сохраняются в виде текстовых файлов и компилируются в машинный код динамически. Существует также возможность сохранять так называемые pre-parsed программы — функции и скрипты, обработанные в удобном для машинного исполнения виде. В общем случае такие программы выполняются быстрее обычных, особенно если функция содержит команды построения графиков. Основной особенностью языка MATLAB являются его широкие возможности по работе с матрицами, которые создатели языка выразили в лозунге «Думай векторами» (Think vectorized).

Для выполнения математических вычислений MATLAB предоставляет пользователю большое количество (несколько сотен) функций для анализа данных, покрывающих практически все области математики, в частности:

- матрицы и линейная алгебра — алгебра матриц, линейные уравнения, собственные значения и векторы, сингулярности, факторизация матриц и др.;
- многочлены и интерполяция — корни многочленов, операции над многочленами и их дифференцирование, интерполяция и экстраполяция кривых и др.;
- математическая статистика и анализ данных — статистические функции, статистическая регрессия, цифровая фильтрация, быстрое преобразование Фурье и др.;

- обработка данных — набор специальных функций, включая построение графиков, оптимизацию, поиск нулей, численное интегрирование (в квадратурах) и др.;
- дифференциальные уравнения — решение дифференциальных и дифференциально-алгебраических уравнений, дифференциальных уравнений с запаздыванием, уравнений с ограничениями, уравнений в частных производных и др.;
- разреженные матрицы — специальный класс данных пакета MATLAB, использующийся в специализированных приложениях;
- целочисленная арифметика — выполнение операций целочисленной арифметики в среде MATLAB.

MATLAB предоставляет удобные средства для разработки алгоритмов, включая высокоуровневые с использованием концепций объектно-ориентированного программирования. В нём имеются все необходимые средства интегрированной среды разработки, включая отладчик и профайлер. Функции для работы с целыми типами данных облегчают создание алгоритмов для микроконтроллеров и других приложений, где это необходимо. В составе пакета MATLAB имеется большое количество функций для построения графиков, в том числе трёхмерных, визуального анализа данных и создания анимированных роликов. Встроенная среда разработки позволяет создавать графические интерфейсы пользователя с различными элементами управления, такими как кнопки, поля ввода и др.

Программы MATLAB, как консольные, так и с графическим интерфейсом пользователя, могут быть собраны с помощью компоненты MATLAB Compiler в независимые от MATLAB исполняемые приложения или динамические библиотеки, для запуска которых на других компьютерах, однако, требуется установка свободно распространяемой среды MATLAB Compiler Runtime.

Пакет MATLAB включает различные интерфейсы для получения доступа к внешним подпрограммам, написанным на других языках программирования, данным, клиентам и серверам, общающимся через технологии Component Object Model или Dynamic Data Exchange, а также периферийным устройствам, которые взаимодействуют напрямую с MATLAB. Многие из этих возможностей известны под названием MATLAB API.

Пакет MATLAB предоставляет доступ к функциям, позволяющим создавать, манипулировать и удалять COM-объекты (как клиенты, так и серверы). Поддерживается также технология ActiveX. Все COM-объекты принадлежат к специальному COM-классу пакета MATLAB. Все программы, имеющие функции контроллера автоматизации (*англ.* Automation controller), могут иметь доступ к MATLAB как к серверу автоматизации (*англ.* Automation server).

Пакет MATLAB в Microsoft Windows предоставляет доступ к программной платформе .NET Framework. Имеется возможность загружать .NET сборки (Assemblies) и работать с объектами .NET классов из среды MATLAB. В версии MATLAB 7.11 (R2010b) поддерживается .NET Framework версий 2.0, 3.0, 3.5 и 4.0.

Пакет MATLAB содержит функции, которые позволяют ему получать доступ к другим приложениям среды Windows, равно как и этим приложениям

получать доступ к данным MATLAB, посредством технологии динамического обмена данными (DDE). Каждое приложение, которое может быть DDE-сервером, имеет своё уникальное идентификационное имя. Для MATLAB это имя — MATLAB.

В MATLAB существует возможность вызывать методы веб-сервисов. Специальная функция создаёт класс, основываясь на методах API веб-сервиса. MATLAB взаимодействует с клиентом веб-сервиса с помощью принятия от него посылок, их обработки и посылок ответа. Поддерживаются следующие технологии: Simple Object Access Protocol (SOAP) и Web Services Description Language (WSDL).

Интерфейс для последовательного порта пакета MATLAB обеспечивает прямой доступ к периферийным устройствам, таким как модемы, принтеры и научное оборудование, подключающееся к компьютеру через последовательный порт (COM-порт). Интерфейс работает путём создания объекта специального класса для последовательного порта. Имеющиеся методы этого класса позволяют считывать и записывать данные в последовательный порт, использовать события и обработчики событий, а также записывать информацию на диск компьютера в режиме реального времени. Это бывает необходимо при проведении экспериментов, симуляции систем реального времени и для других приложений.

Пакет MATLAB включает интерфейс взаимодействия с внешними приложениями, написанными на языках C и Фортран. Осуществляется это взаимодействие через MEX-файлы. Существует возможность вызова подпрограмм, написанных на C или Фортране, из MATLAB, как будто это встроенные функции пакета. MEX-файлы представляют собой динамически подключаемые библиотеки, которые могут быть загружены и исполнены интерпретатором, встроенным в MATLAB. MEX-процедуры имеют также возможность вызывать встроенные команды MATLAB.

Интерфейс MATLAB, относящийся к общим DLL, позволяет вызывать функции, находящиеся в обычных динамически подключаемых библиотеках, прямо из MATLAB. Эти функции должны иметь C-интерфейс. Кроме того, в MATLAB имеется возможность получить доступ к его встроенным функциям через C-интерфейс, что позволяет использовать функции пакета во внешних приложениях, написанных на C. Эта технология в MATLAB называется C Engine.

Для MATLAB имеется возможность создавать специальные наборы инструментов (toolbox), расширяющие его функциональность. Наборы инструментов представляют собой коллекции функций и объектов, написанных на языке MATLAB для решения определённого класса задач. Компания Mathworks поставляет наборы инструментов, которые используются во многих областях, включая следующие:

- цифровая обработка сигналов, изображений и данных: Signal Processing Toolbox, DSP System Toolbox, Image Processing Toolbox, Wavelet Toolbox, Communications System Toolbox — наборы функций и объектов, поз-

воляющих решать широкий спектр задач обработки сигналов, изображений, проектирования цифровых фильтров и систем связи;

- системы управления: Control Systems Toolbox, Robust Control Toolbox, System Identification Toolbox, Model Predictive Control Toolbox, Model-Based Calibration Toolbox — наборы функций и объектов, облегчающих анализ и синтез динамических систем, проектирование, моделирование и идентификацию систем управления, включая современные алгоритмы управления, такие как робастное управление, H-управление, ЛМН-синтез, μ-синтез и др.;
- финансовый анализ: Econometrics Toolbox, Financial Instruments Toolbox, Financial Toolbox, Datafeed Toolbox, Trading Toolbox — наборы функций и объектов, позволяющие быстро и эффективно собирать, обрабатывать и передавать различную финансовую информацию;
- анализ и синтез географических карт, включая трёхмерные: Mapping Toolbox;
- сбор и анализ экспериментальных данных: Data Acquisition Toolbox, Image Acquisition Toolbox, Instrument Control Toolbox, OPC Toolbox — наборы функций и объектов, позволяющих сохранять и обрабатывать данные, полученные в ходе экспериментов, в том числе в режиме реального времени. Поддерживается широкий спектр научного и инженерного измерительного оборудования;
- визуализация и представление данных: Virtual Reality Toolbox — позволяет создавать интерактивные миры и визуализировать научную информацию с помощью технологий виртуальной реальности и языка VRML;
- средства разработки: MATLAB Builder for COM, MATLAB Builder for Excel, MATLAB Builder for NET, MATLAB Compiler, HDL Coder — инструменты, позволяющие создавать независимые приложения из среды MATLAB;
- взаимодействие с внешними программными продуктами: MATLAB Report Generator, Excel Link, Database Toolbox, MATLAB Web Server, Link for ModelSim — наборы функций, позволяющие сохранять данные различных видов таким образом, чтобы другие программы могли с ними работать;
- базы данных: Database Toolbox — инструменты работы с базами данных;
- научные и математические пакеты: Bioinformatics Toolbox, Curve Fitting Toolbox, Fixed-Point Toolbox, Optimization Toolbox, Global Optimization Toolbox, Partial Differential Equation Toolbox, Statistics And Machine Learning Toolbox, RF Toolbox — наборы специализированных математических функций и объектов, позволяющие решать широкий спектр научных и инженерных задач, включая разработку генетических алгоритмов, решение задач в частных производных, целочисленные проблемы, оптимизацию систем и др.;
- нейронные сети: Neural Network Toolbox — инструменты для синтеза и анализа нейронных сетей;
- нечёткая логика: Fuzzy Logic Toolbox — инструменты для построения и анализа нечётких множеств;

- символьные вычисления: Symbolic Math Toolbox — инструменты для символьных вычислений с возможностью взаимодействия с символьным процессором программы Maple.

Помимо вышеперечисленных, существует множество других наборов инструментов для MATLAB, написанных другими компаниями и энтузиастами. В равной степени этой относится и к химии и химической технологии, в частности к компьютерному моделированию химико-технологических процессов. В последнее время они широко используются в пакетах моделирующих программ — симуляторах процессов типа CHEMCAD, HYSYS, ASPEN.

При разработке программных комплексов для компьютерного моделирования химико-технологических процессов необходимо использовать алгоритмы решения типовых задач вычислительной математики, представленные в предыдущем разделе на рисунке 1.2. В таблице 1.2 приведены некоторые наиболее известные решатели MATLAB, позволяющие реализовать соответствующие численные методы решения указанных задач.

Таблица 1.2

Некоторые решатели ПКМ MATLAB, используемые для реализации численных методов вычислительной математики при компьютерном моделировании ХТП

№	Типовая задача вычислительной математики	Используемые решатели	Комментарии
1	**2**	**3**	**4**
1	**Вычисление производных и интегралов (ИНТ)**		
1.1	Вычисление производных	—	Метод разделенных разностей
1.2	Вычисление интегралов	“trapz” “quad”	Метод трапеций Метод Симпсона
2	**Решение нелинейных уравнений (НУ)**		
2.1	Полиномиальных	“roots”	Определяются все корни уравнения, в том числе и комплексные
2.2	Трансцендентных	“fzero” “fminbnd”	Определяется один корень. Методом одномерной оптимизации определяется корень, обеспечивающий нулевое значение функции уравнения вида f(x) = 0
3	**Решение систем линейных (СЛАУ) и нелинейных уравнений (СНУ)**		
3.1	Линейных (СЛАУ)	“linsolve” а также стандартные функции: “inv”, “det”	СЛАУ записывается в виде матричного уравнения $\overline{\overline{A}}\,\overline{x} = \overline{b}$
		“polyfit”	Определение коэффициентов многочленов произвольной степени при аппроксимации данных, зависящих от одной независимой переменной

Продолжение табл. 1.2

№	Типовая задача вычислительной математики	Используемые решатели	Комментарии
3.2	Нелинейных (СНУ)	“fsolve”	СНУ записывается в виде $\overline{f}(\overline{x}) = \overline{0}$
		“fminsearch” “fmincon”	Методом многомерной оптимизации определяются корни, обеспечивающие нулевые значения всех функций системы нелинейных уравнений вида $\overline{f}(\overline{x}) = \overline{0}$
4	**Решение систем обыкновенных дифференциальных уравнений (СОДУ) и с частными производными (СДУЧП)**		
4.1	Обыкновенных (СОДУ)		
4.1.1	Нежестких	“ode45” “ode23” “ode113”	Явные методы
4.1.2	Жестких	“ode15s” “ode23s” “ode23t” “ode23tb”	Неявные методы
4.1.3	Краевые задачи	Все перечисленные выше решатели нежестких и жестких СОДУ совместно с решателями “fzero” “fminbnd” или “fsolve” “fminsearch” “fmincon”	Метод «стрельбы» для двухточечной краевой задачи Для двух СОДУ первого порядка или для СОДУ первого порядка произвольной размерности
4.2	С частными производными (СДУЧП)		
4.2.1	Нежестких		Явные разностные схемы
4.2.2	Жестких		Неявные разностные схемы
5	**Решение оптимизационных задач (ОПТ)**		
5.1	Одномерных	“fminbnd”	Определение глобального минимума
		“fminsearch”	Безусловная одномерная оптимизация без ограничений на параметры
		“fmincon”	Одномерная оптимизация с ограничениями 1-го и 2-го рода на параметры

Продолжение табл. 1.2

№	Типовая задача вычислительной математики	Используемые решатели	Комментарии
5.2	Многомерных	“lsqcurvefit”	Определение коэффициентов аппроксимационных регрессионных моделей, зависящих от одной независимой переменной
		“fminsearch”	Безусловная многомерная оптимизация с использованием алгоритма Нелдера — Мида
		“fmincon”	Возможность выбора алгоритма решения задачи нелинейного программирования с ограничениями в виде системы линейных равенств и неравенств, а также ограничений 1-го и 2-го рода на параметры

В главе 3 настоящей книги подробно рассматривается постановка приведенных типовых задач вычислительной математики и особенности их решения численными методами с применением решателей MATLAB. Одновременно на примере решения 16 задач моделирования химико-технологических процессов показано, как в конкретных случаях применяются представленные в таблице 1.2 решатели.

ГЛАВА 2. ОСНОВЫ ПРОГРАММИРОВАНИЯ НА ИНТЕРПРЕТИРУЕМОМ ЯЗЫКЕ ПАКЕТА MATLAB

Программные модули в среде MATLAB представляют собой *m*-файлы (файлы с расширением *.m), программные коды, которые должны быть написаны на специальном языке программирования MATLAB. Язык программирования MATLAB, так называемый ***m*-язык программирования**, является **высокоуровневым интерпретируемым языком программирования** и в общем случае оперирует с переменными, представляющими собой массивы и матрицы: название MATLAB — сокращение двух английских слов Matrix Laboratory. Поэтому матричные операции на *m*-языке записываются в наиболее компактной форме, что является одним из достоинств MATLAB, в частности при решении задач линейной алгебры. Обширный набор встроенных стандартных функций вычислительной математики и решателей (солверов), предназначенных для реализации численных методов решения математических задач, позволяет рассматривать его как один из наиболее эффективных и универсальных языков программирования для решения научных и технических задач. Простота графического представления результатов расчетных и экспериментальных исследований с применением языка программирования MATLAB — *m*-языка программирования — позволяет считать последний надежным средством при разработке и исследовании компьютерных моделей химико-технологических процессов, а также при применении их для расчетов, исследования, оптимизации и проектирования технологий химических производств.

2.1. Интегрированная среда MATLAB и работа в ней

Если на компьютере установлена операционная система Windows, для запуска среды MATLAB следует сделать двойной щелчок левой клавишей мыши по ярлыку на дисплее, соответствующему MATLAB. В результате после некоторых служебных сообщений появится основное окно среды MATLAB (рис. 2.1).

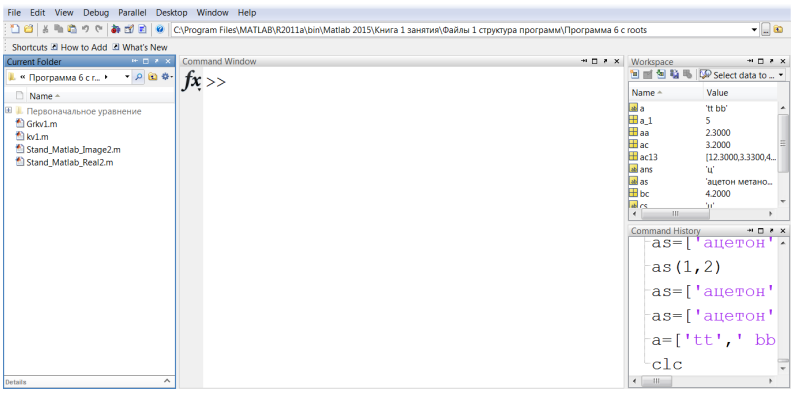


Рис. 2.1

Основное окно интегрированной среды MATLAB (по умолчанию)

В основном окне среды MATLAB можно писать, редактировать, отлаживать и выполнять программы, написанные на языке программирования MATLAB, и просматривать результаты их работы, не покидая среды MATLAB.

Для установки расположения окон в среде MATLAB **по умолчанию (default)** (рис. 2.1) на панели инструментов следует выбрать левой кнопкой мыши команду **Desktop — Desktop Layout — Default**.

Конфигурация основного окна среды MATLAB (по умолчанию) состоит из четырех различных по функциональному назначению окон:

- **Текущая папка (Current Folder)** — слева от центральной части экрана со списком файлов, в том числе и *m*-файлов с программными кодами, написанными на языке MATLAB, и папок, в которых хранятся файлы, в том числе и *m*-файлы;
- **Командное окно (Command Window)** — центральная часть экрана, предназначена для запуска программ (*m*-файлов) из **Текущей папки (Current Folder)**, выполнения программных кодов и операторов языка MATLAB, ввода данных для исполнения программ и вывода результатов их выполнения;
- **Рабочая память (Workspace)** — располагается справа сверху от центральной части экрана со списком и текущими значениями используемых переменных;
- **История команд (Command History)** — располагается справа снизу от центральной части экрана с выполненными *m*-файлами и программными кодами на языке MATLAB, сгруппированными по датам.

Над перечисленными четырьмя окнами (по умолчанию) располагается **Панель инструментов** с изображением кнопок и надписями над кнопками для управления интегрированной средой MATLAB.

Справа от изображения кнопок **Панели инструментов** отображается название текущей папки среды MATLAB и путь к ней, а в левом окне **Текущей папки (Current Folder)** — ее содержимое. Выпадающее меню, располагаемое справа от **Панели инструментов**, позволяет оперативно открыть папки в **Текущей папке (Current Folder)**, с которыми ранее работал и намерен работать пользователь MATLAB.

Следует отметить, что расположение окон среды MATLAB по умолчанию является наиболее удобным для пользователя, и потому в процессе работы с пакетом рекомендуется восстанавливать четырехоконную конфигурацию среды MATLAB, как было указано выше.

Для изменения типоразмеров отображаемой информации в основном окне MATLAB (рис. 2.1), в частности параметров шрифта, в среде MATLAB следует переустановить **Шрифты (Fonts)** с использованием команды настроек на **Панели инструментов** командой **File — Preferences — Fonts**.

Все действия (операции, вычисления, построение графиков) можно выполнять в командной строке (после знака >>) командного окна (Command Window) MATLAB, отделяя действия друг от друга точкой с запятой (;).

В случае отсутствия точки с запятой после каждого действия результат его выполнения будет выводиться на дисплей в Командное окно

MATLAB в строгом соответствии с действиями интерпретатора *m*-языка программирования.

Как и в любом алгоритмическом языке высокого уровня, используемые в программных кодах m-языка программирования переменные должны быть **представлены (описаны, декларированы) в виде целых, вещественных, логических переменных скалярного или матричного типа**. Последние чаще всего называют массивами, если это скаляр, то его размер (1 * 1), вектор-строка — размер (1 * *n*), вектор-столбец — размер (*m* * 1), произвольная матрица — размер (*m* * *n*), где *m* и *n* — конкретные значения чисел, характеризующие соответственно число строк и столбцов матрицы.

Для **представления** типов переменных в программе с использованием оператора присвоения им присваиваются значения числовых, логических или символьных констант — скаляров или массивов, после чего они воспринимаются в программном коде программы как переменные представленного (описанного) ранее типа с соответствующими правилами действий над ними.

Изменение представления переменных в программном коде возможно, как будет показано ниже, с использованием специальных стандартных функций *m*-языка программирования.

Для выполнения действий с переменными *m*-языка программирования используются следующие основные операторы:

- оператор присвоения “=”;
- оператор условного перехода “if”;
- оператор выбора “switch”;
- оператор цикла “for”;
- оператор цикла “while”;
- оператор построения двумерного (плоского) графика с использованием функции “plot”;
- оператор построения трехмерного (объемного) графика с использованием функций “mesh”, “meshc” и “surf”, “surfc”.

2.2. Представление (описание, декларирование) переменных с использованием оператора присвоения

Для обозначения **переменных** на языке программирования MATLAB используются **идентификаторы**, которые представляют собой набор символов, состоящий из прописных и строчных букв, цифр и символов, которые начинаются с буквы и не могут содержать пробелы.

В общем случае **переменные** могут быть **числовыми**, когда им присваиваются **числовые константы**, или **символьными**, когда им присваиваются **строковые константы**, которые должны быть записаны в кавычках.

В среде MATLAB константы представляют собой массивы из нескольких **обязательно** однотипных элементов (числовых или строковых), которые заключены в квадратные скобки и элементы которых отделяются друг от друга либо запятыми, либо пробелами, либо точками с запятой.

На рисунке 2.2 в командной строке Командного окна (Command Window) показано, что для представления переменной в виде скалярной величины ей необходимо присвоить числовую константу — массив размером (1 * 1), а для представления символьной переменной ей надо присвоить строковую константу, взятую в кавычки. Так как после операторов присвоения нет точки с запятой (;), то после их выполнения в командном окне появляется либо числовая переменная с соответствующим числовым значением, либо символьная переменная с соответствующим строковым значением без кавычек.

```
1. Представление числовой переменной в виде скалярной величины:
— присвоение переменной "ac" одной числовой константы в квадратных скобках — массив размером (1*1):
>> ac=[3.2]
ac = 3.2000

или

— присвоение переменной "ac" одной числовой константы без квадратных скобок — справедливо только для массива размером (1*1):
>> ac=3.2
ac = 3.2000

>> 2. Представлением символьной переменной в виде одной строковой величины:
— присвоение переменной "as" одной строковой константы в квадратных скобках — массив размером (1*1):
>> as=['Исходные данные']
as = Исходные данные

или
— присвоение переменной "as" одной строковой константы без квадратных скобок — только для массива размером (1*1):
>> as='Исходные данные'
as = Исходные данные
```

Рис. 2.2

Представление числовых и символьных переменных в виде скаляра

Для присвоения переменной (рис. 2.3) элементов числового массива размером (2 * 3): 2 — число строк матрицы; 3 — число столбцов матрицы, необходимо все числа записать в квадратных скобках и элементы каждой следующей строки отделить от предыдущей точкой с запятой. На рисунке 2.3 показано, как символьной переменной присваиваются строковые элементы массива размером (1 * 3).

2.2.1. Оператор создания числового массива с равномерным распределением элементов “linspace”

Элементы числового массива, например *zm*, в отличие от квадратных скобок, могут быть представлены стандартной функцией MATLAB — “linspace”:

zm = linspace (*a*,*b*,*n*)

с *n* элементами, равномерно распределенными между числами *a* и *b*, включая *a* и *b*.

```
1. Представление переменной в виде массива с числовыми элементами:
— присвоение переменной "ac13" числовых элементов массива размером (2*3) — элементы массива отделены пробелами:
>> ac13 = [12.3 -3.4 7;8.12 13 17]
ac13 =
   12.3000   -3.4000    7.0000
    8.1200   13.0000   17.0000

или
— присвоение переменной "ac13" числовых элементов массива размером (2*3) — элементы массива отделены запятыми:
>>ac13=[12.3,-3.4,7;8.12,13,1]
ac13 =
   12.3000   -3.4000    7.0000
    8.1200   13.0000    1.0000

2. Представление переменной в виде массива с строковыми элементами:
— присвоение переменной "as13" строковых элементов массива размером (1*3) — элементы массива отделены пробелами:
>> as13 = ['ацетон' ' метанол' ' вода']
as13 =
ацетон метанол вода

или

— присвоение переменной "as13" строковых элементов массива размером (1*3) — элементы массива отделены запятыми:
>>  as13 = ['ацетон',' метанол',' вода']
as13 =
ацетон метанол вода
```

Рис. 2.3

Представление числовых и символьных переменных в виде массивов

2.2.2. Оператор двоеточие (:) для создания числового массива с равномерным распределением элементов

Массив чисел может быть представлен и оператором присвоения (рис. 2.4), который представляет собой некоторую переменную (в данном случае *xx*) в виде массива, которому присваивается значение трех чисел: первое и третье — крайние значения элементов массива с уменьшающимся значением на 0.25 каждого следующего по порядку элемента, тогда среднее (второе) значение со знаком «–» ограничивается двумя двоеточиями **:-0.25:**, как в данном случае (рис. 2.4), а при возрастающих значениях среднее значение ограничивается, например, **:+0.25:** соответственно со знаком «+» или без знака.

2.2.3. Оператор “format” для фиксации определенного числа цифр после десятичной точки в случае изображения числовой константы

Следует обратить внимание на выводимый по умолчанию (default) формат чисел (десятичный формат с фиксированной запятой) — четыре цифры после десятичной точки (рис. 2.4, 1 и 2.4, 3). Для изменения формата выводимых чисел в командную строку следует записать: format bank — выводятся 2 цифры после десятичной точки (рис. 2.4, 2) или format long — 15 цифр после десятич-

ной точки (рис. 2.4, 4), format short (рис. 2.4, 3) соответствует формату вывода по умолчанию (default) (рис. 2.4, 3.1).

```
1. С форматом вывода чисел по умолчанию (“default”):
>>  xx=7.1:-0.25:6.5
xx =
   7.1000    6.8500    6.6000
2.  С форматом вывода чисел “bank” — 2 цифры после десятичной точки:
>> format bank
>>  xx=7.1:-0.25:6.5

xx =

       7.10        6.85        6.60
3.  С форматом вывода чисел “short” — 4 цифры после десятичной точки (совпадает с
выводом по умолчанию (“default”):
>> format short
>> xx=7.1:-0.25:6.5
xx =
   7.1000    6.8500    6.6000
4. С форматом вывода чисел “long” — 15 цифр после десятичной точки;
>> format long
>> xx=7.1:-0.25:6.7
xx =
  7.100000000000000   6.850000000000000
```

Рис. 2.4

Представление переменной в виде одномерного массива
с использованием оператора двоеточия с различными форматами выводимых чисел
(“bank”, “short” , “long”)

2.2.4. Операторы преобразования целых и вещественных переменных в символьные "int2str" и "num2str"

При формировании отчетов по проведенным вычислениям важное значение имеет вывод массивов из символьных элементов, в том числе целых и вещественных чисел, преобразованных в символьные константы.

Для преобразования целых чисел в символьные константы используется стандартная функция MATLAB "int2str()" (рис. 2.5, 1), а вещественных — "num2str()" (рис. 2.5, 2). В результате таких преобразований в отчетах о вычислениях могут создаваться символьные массивы и включающие в себя конкретные числовые значения в символьном виде. На рисунке 2.5, 2 массив sym_arr_P включает в себя три строковые константы, одна из которых (sym_P) является преобразованной в этом же пункте выше числовой константой. На этом же рисунке показано (рис. 2.5, 3), что функции преобразования "int2str()" и "num2str()" можно применить и ко всему числовому массиву, тогда все его элементы станут строковыми константами.

Важно отметить, что если после задания значений переменным с использованием оператора присвоения или оператора двоеточия не поставлена точка с запятой, то результат отображается в командном окне. В противном случае, при наличии точки с запятой, значение переменной не выводится ни в виде скаляра, ни в виде массива.

В общем случае для вывода символьной переменной в командное окно MATLAB (на дисплее) используется оператор disp со скобками, внутри которых размещается текст в кавычках, например, disp (‘Исходные данные для расчетов’) или disp ('Результаты расчетов').

```
1. Преобразование целой пременной "ics" в символьную 'sym_ics' стандартной функцией "int2str()" :
>> ics=780
ics =
   780
>> sym_ics=int2str(ics)
sym_ics =
780

2. Преобразование вещественной переменной “P" в символьную sym_P стандартной функцией "num2str()":
>> P=760.15
P =
  760.1500
>> sym_P=num2str(P)
sym_P =
760.15

— создание массива из строковых элементов "sym_arr_P" размером (1*3):
>> sym_arr_P=['Давление = ' sym_P ' мм.рт.ст.']
sym_arr_P =
Давление = 760.15 мм.рт.ст.

3. Преобразование массива вещественных чисел "zmas" массив из символьных переменных "sym_zmas" стандартной функцией "num2str()":
>> zmas=[1.3 2.3 4.2]
zmas =
   1.3000    2.3000    4.2000
>> sym_zmas=num2str(zmas)
sym_zmas =
1.3       2.3       4.2
- создание массива из строковых элементов "sym_text" размером (1*2):
>> sym_text=['symbol ' sym_zmas]
sym_text =
symbol 1.3       2.3       4.2
```

Рис. 2.5

Преобразование числовых переменных в символьные

2.3. Оператор присваивания «=»

Оператор присваивания присваивает переменной (в общем случае массиву, стоящему слева от оператора присваивания «=») значение выражения, стоящего справа. При этом все использованные в выражении переменные должны быть предварительно заданы.

Выражение может содержать арифметические операции, стандартные функции и логические операции. Как было указано ранее (раздел 2.2), если переменная слева от оператора присваивания «=» встречается в программе впервые, то ее тип задается в соответствии с типом результата выражения, стоящего справа от оператора присваивания «=».

На рисунке 2.6 показаны простейшие случаи присвоения переменным элементам одномерных и двумерных массивов, а также векторов-строк и векторов-столбцов, как числовых, так и символьных.

```
1. Переменной "z" присваивается одномерный числовой массив в виде вектора-строки:
>> z=[1.2,1.3e2,-7.4]
z =
   1.2000  130.0000   -7.4000
 или с пробелами между элементами вектора-строки:
>> z=[1.2 1.3e2 -7.4]
z =
   1.2000  130.0000   -7.4000

2. Переменной "sz" присваивается одномерный символьный массив с 4-мя элементами в
виде вектора-строки:
>> sz=['система ','метанол' '-' 'вода']
sz =
система метанол-вода

3. Переменной "zst" присваивается числовой массив в виде вектора-столбца:
>> zst=[1.2;1.3e2;-7.4]
zst =
   1.2000
  130.0000
   -7.4000
4. Переменной "zm" присваивается числовая массив в виде прямоугольной матрицы
размером (4*2):
>> zm=[1.3 2.154;-1.25 22.15; 3.15 -2.5;11.3 212.9]
zm =
   1.3000    2.1540
  -1.2500   22.1500
   3.1500   -2.5000
  11.3000  212.9000

5. Переменной "zk" присваивается числовая константа в виде квадратной матрицы:
>> zk=[1.3 2.4 7.2;8.3 -1.25 4.5;8.4 -2.56 1e2]
zk =
   1.3000    2.4000    7.2000
   8.3000   -1.2500    4.5000
   8.4000   -2.5600  100.0000
```

Рис. 2.6

Присвоение переменным элементов одномерных и двумерных массивов

Как следует из рисунка 2.6, 1, числа в массивах могут задаваться как в десятичном формате с фиксированной запятой (например, 1.2 = 1.2), так и в экспоненциальном формате с плавающей запятой (например, 1.3e2 = $1.3 * 10^2$ = = 130.0000).

2.3.1. Применение стандартных функций MATLAB к одномерным массивам

Применение оператора присвоения со стандартными функциями (табл. 2.1) для определения числовых характеристик одномерных массивов показано на рисунке 2.7. Как следует из этого рисунка, если в результате использования стандартной функции одномерный массив не присваивается конкретной переменной, то по умолчанию он автоматически присваивается системной переменной “ans”.

Таблица 2.1

Стандартные функции, применяемые к одномерным массивам

Функция	Описание для одномерного массива A
`prod (A)`	произведение элементов массива
`sum (A)`	сумма элементов массива
`mean (A)`	`mean (A)` = $\bar{A}$ — среднее арифметическое значение элементов
`std (A)` `std (A, flag)`	`std (A)` = `std (A,0)` = $\sqrt{\frac{1}{n-1}\sum_{k=1}^{n}(A_k - \bar{A})^2}$ — стандартное отклонение `std (A,0)` = $\sqrt{\frac{1}{n}\sum_{k=1}^{n}(A_k - \bar{A})^2}$ — среднеквадратичное отклонение
`length (A)`	возвращает число элементов массива
`max (A)`	возвращает наибольший элемент
`min (A)`	возвращает наименьший элемент
`sort (A)`	сортирует элементы по возрастанию
`dot (A1, A2)`	возвращает скалярное произведение векторов A1 и A2

```
1. Вычисление числа элементов вектора-строки "z":
>> z=[1.2,1.3e2,-7.4];
>> length(z)
ans =
     3
2. Вычисление числа элементов вектора-столбца "zst":
 zst=[1.2;1.3e2;-7.4];
>> length(zst)
ans =
     3

3. Вычисление суммы элементов вектора-строки "z":
>> z=[1.2,1.3e2,-7.4];
>> sum(z)
ans =
  123.8000

4. Вычисление произведения элементов вектора-строки "z":
>> z=[1.2,1.3e2,-7.4];
>> prod(z)
ans =
 -1.1544e+003

5. Вычисление минимального элемента вектора-столбца "zst":
 >> zst=[1.2;1.3e2;-7.4];
>> min(zst)
ans =
   -7.4000

6. Вычисление максимального элемента вектора-столбца "zst":
 >> zst=[1.2;1.3e2;-7.4];
>> max(zst)
ans =
   130
7. Вычисление среднего арифметического элементов вектора-строки "zst":
>>z=[1.2,1.3e2,-7.4];
>> mean(z)
ans = 41.2667
```

Рис. 2.7

Определение числовых характеристик векторов и присвоение их значений системной переменной "ans"

2.3.2. Применение стандартных функций MATLAB к двумерным массивам

Применение оператора присвоения со стандартными функциями для определения числовых характеристик двумерных массивов (табл. 2.2) показано на рисунке 2.8.

Таблица 2.2

Стандартные функции, применяемые к двумерным массивам

Функция	Описание для матриц A и B
`size (A)`	определение размера матрицы
`norm (A)`	определение евклидовой нормы матрицы
`cond (A)`	определение числа обусловленности матрицы
`A'`	транспонирование матрицы
`A(:,k)=[]`	удаление *k*-го столбца
`A(k,:)=[]`	удаление *k*-й строки
`V = A(:,k)`	создание одномерного массива *V* из *k*-го столбца
`V = A(k,:)`	создание одномерного массива *V* из *k*-й строки
`A(:,k) = V`	замена *k*-го столбца одномерным массивом *V*
`A(k,:) = V`	замена *k*-й строки одномерным массивом *V*
`eye(n)`	возвращает квадратную единичную матрицу размером *n*n*
`zeros(n,m)`	возвращает матрицу размером *n*×*m*, состоящую из нулей
`ones(n,m)`	возвращает матрицу размером *n*×*m*, состоящую из единиц
`rand(n,m)`	возвращает матрицу размером *n*×*m*, заполненную случайными числами, подчиняющимися равномерному распределению в интервале (0, 1)
`randn(n,m)`	возвращает матрицу размером *n*×*m*, заполненную случайными числами, подчиняющимися нормальному распределению с математическим ожиданием, равным нулю, и среднеквадратичным отклонением, равным единице
`[A,B]`	горизонтальное объединение матриц *A* и *B*
`[A;B]`	вертикальное объединение матриц A и *B*
`tril (A)`	формирует нижнюю треугольную матрицу
`trin (A)`	формирует верхнюю треугольную матрицу
`trace (A)`	след матрицы
`det (A)`	вычисляет детерминант квадратной матрицы методом Гаусса
`rank (A)`	возвращает ранг матрицы
`rref (A)`	приводит матрицу к треугольному виду методом Гаусса
`sum (sum (A))`	сумма всех элементов матрицы
`prod (A)`	произведение всех элементов матрицы
`inv (A)`	возвращает матрицу, обратную заданной

Здесь также результаты присваиваются системной переменной “ans” (рис. 2.8), и в дополнение к стандартным функциям, приведенным в таблице 1.1, представлены следующие стандартные функции для вычисления числовых характеристик прямоугольных и квадратных матриц:

- size (matr_prjam) — определение размера матрицы: числа строк и числа столбцов;

- norm (matr_prjam) — определение евклидовой нормы матрицы;
- det (matr_kvadr) — вычисление определителя квадратной матрицы;
- trace (matr_kvadr) — вычисление следа квадратной матрицы;
- eig (matr_kvadr) — вычисление собственных значений квадратной матрицы,

где matr_prjam — произвольная прямоугольная матрица; matr_kvadr — произвольная квадратная матрица.

```
1. Вычисление размера прямоугольной матрицы "zm":
>> zm=[1.3 2.154;-1.25 22.15; 3.15 -2.5;11.3 212.9];
>> size(zm)
ans =
    4    2
* Первое число — число строк матрицы, второе число — число столбцов матрицы.

2. Вычисление ранга прямоугольной матрицы "zm":
>> zm=[1.3 2.154;-1.25 22.15; 3.15 -2.5;11.3 212.9];
>> rank(zm)
ans =
    2

3. Вычисление евклидовой нормы прямоугольной матрицы "zm":
>> zm=[1.3 2.154;-1.25 22.15; 3.15 -2.5;11.3 212.9];
>> norm(zm)
ans =
  214.3615

4. Вычисление определителя (детерминанта) квадратной матрицы "zk":
>> zk=[1.3 2.4 7.2;8.3 -1.25 4.5;8.4 -2.56 1e2];
>> det(zk)
ans =
 -2.1262e+003

5. Вычисление следа квадратной матрицы "zk":
>> zk=[1.3 2.4 7.2;8.3 -1.25 4.5;8.4 -2.56 1e2];
>> trace(zk)
ans =
  100.0500

6. Вычисление собственных значений квадратной матрицы "zk":
>> zk=[1.3 2.4 7.2;8.3 -1.25 4.5;8.4 -2.56 1e2];
>> eig(zk)
ans =
  100.4913
    4.3844
   -4.8257
```

Рис. 2.8

Определение числовых характеристик матриц и присвоение их значений системной переменной "ans"

Следует отметить, что стандартные функции таблицы 2.1 для векторов можно также применять и к матрицам, что будет приводить к выполнению предусмотренных действий для каждого столбца матрицы. В ре-

зультате будет рассчитано столько значений стандартных функций, сколько столбцов в матрице.

2.3.3. Элементарные стандартные математические и собственные функции MATLAB

Элементарные стандартные математические функции приведены в таблице 2.3, а их применение — на рисунках 2.9, 9 и 2.9, 10. На рисунках 2.9, 1–2.9, 8 показано применение следующих стандартных функций MATLAB к массивам:

“sort” — сортировка элементов массива по возрастанию и убыванию;

“ ' ” — транспонирование матриц (кавычки);

“inv” — обращение квадратных матриц;

“eig” — вычисление собственных векторов и собственных значений квадратных матриц.

Таблица 2.3

Элементарные стандартные математические и собственные функции MATLAB

Функция	Описание функции
Тригонометрические	
`sin(x)`	синус числа *x*
`cos(x)`	косинус числа *x*
`tan(x)`	тангенс числа *x*
`cot(x)`	котангенс числа *x*
`sec(x)`	секанс числа *x*
`csc(x)`	косеканс числа *x*
`asin(x)`	арксинус числа *x*
`acos(x)`	арккосинус числа *x*
`atan(x)`	арктангенс числа *x*
`acot(x)`	арккотангенс числа *x*
`asec(x)`	арксеканс числа *x*
`acsc(x)`	арккосеканс числа *x*
Гиперболические	
`sinh(x)`	гиперболический синус числа *x*
`cosh(x)`	гиперболический косинус числа *x*
`tanh(x)`	гиперболический тангенс числа *x*
`coth(x)`	гиперболический котангенс числа *x*
`sech(x)`	гиперболический секанс числа *x*
`csch(x)`	гиперболический косеканс числа *x*
Экспоненциальные	
`exp(x)`	экспонента числа *x*
`log(x)`	натуральный логарифм числа *x*
Целочисленные	
`fix(x)`	округление числа *x* до ближайшего целого в сторону нуля
`floor(x)`	округление числа *x* до ближайшего целого в сторону -∞
`ceil(x)`	округление числа *x* до ближайшего целого в сторону +∞
`round(x)`	обычное округление числа *x* до ближайшего целого
`rem(x, y)`	вычисление остатка от деления *x* на *y*
`sign(x)`	сигнум-функция числа *x*, выдает 0, если $x = 0$, -1 при $x < 0$ и 1 при $x > 0$

Продолжение табл. 2.3

Функция	Описание функции
Другие	
`sqrt(x)`	корень квадратный из числа *x*
`abs(x)`	модуль числа *x*
`log10(x)`	десятичный логарифм от числа *x*
`log2(x)`	логарифм по основанию два от числа *x*
`real(x)`	вещественная часть комплексного числа *x*
`imag(x)`	мнимая часть комплексного числа *x*
`pow2(x)`	возведение двойки в степень *x*
`gcd(x, y)`	наибольший общий делитель чисел *x* и *y*
`rats(x)`	представление числа *x* в виде рациональной дроби

Простейшие действия над матрицами и векторами с применением стандартных функций (табл. 2.2) и элементарных математических функций (табл. 2.3) представлены на рисунке 2.9.

```
1. Упорядочивание элементов вектора-строки "z" по возрастанию:
>>z=[1.2,1.3e2,-7.4];
>> zuv=sort(z)
zuv =
  -7.4000    1.2000  130.0000
2. Упорядочивание элементов вектора-строки "z" по убыванию:
>>z=[1.2,1.3e2,-7.4];
>> zuu=-sort(-z)
zuu =
  130.0000    1.2000   -7.4000

3. Транспонирование вектора-строки "z":
>>z=[1.2,1.3e2,-7.4];
>> z_trans=z'
z_trans =
    1.2000
  130.0000
   -7.4000

4. Транспонирование вектора-столбца "zst":
>>zst=[1.2;1.3e2;-7.4];
>> zst_trans=zst'
zst_trans =
    1.2000  130.0000   -7.4000

5. Определение элементов транспонированной прямоугольной матрицы "zm":
>> zm=[1.3 2.154;-1.25 22.15; 3.15 -2.5;11.3  212.9];
>> zm_trans=zm'
zm_trans =
    1.3000   -1.2500    3.1500   11.3000
    2.1540   22.1500   -2.5000  212.9000

6. Определение элементов транспонированной квадратной матрицы "zk":
>> zk=[1.3 2.4 7.2;8.3 -1.25 4.5;8.4  -2.56  1e2];
>> zk_trans=zk'
zk_trans =
    1.3000    8.3000    8.4000
    2.4000   -1.2500   -2.5600
    7.2000    4.5000  100.0000

7. Вычисление элементов обратной матрицы "zk"(только для квадратных матриц):
```

Рис. 2.9 (начало)

Простейшие действия над векторами и матрицами с применением стандартных функций MATLAB

```
>> zk=[1.3 2.4 7.2;8.3 -1.25 4.5;8.4 -2.56 1e2];
>> zk_inv=inv(zk)
zk_inv =
   0.0534   0.1215   -0.0093
   0.3726  -0.0327   -0.0254
   0.0051  -0.0110    0.0101

8. Вычисление собственных векторов(msobs) и собственных значений(znsobs) матрицы
"zk" из предыдущей позиции (7) (только для квадратных матриц):
>> [msobs,znsobs]=eig(zk)
msobs=
  -0.0735  -0.5719    0.4204
  -0.0501  -0.8198   -0.9056
  -0.9960   0.0283   -0.0558
znsobs =
  100.4913        0        0
         0   4.3844        0
         0        0   -4.8257
*Первый столбец матрицы msobs — собственный вектор, соответствующий собственно-
му значению
                                znsobs(1,1)=100.4913
*Второй столбец матрицы msobs — собственный вектор, соответствующий собственно-
му значению
                                znsobs(2,2)=4.3844
*Третий столбец матрицы msobs — собственный вектор, соответствующий собственно-
му значению
                                znsobs(3,3)=-4.8257

9. Применение стандартной функции MATLAB (log10 — Табл. 1.4) к вектору-строке "z":
>>z=[1.2,1.3e2,-7.4];
>> z_log10=log10(z)
z_log10 =
   0.0792            2.1139            0.8692 + 1.3644i

10. Применение стандартной функции MATLAB (sqrt — Табл. 1.4) к прямоугольной матри-
це "zm" :
>> zm=[1.3 2.154;-1.25 22.15; 3.15 -2.5;11.3  212.9];
>> zm_sqrt=sqrt(zm)
zm_sqrt =
   1.1402           1.4677
   0 + 1.1180i      4.7064
   1.7748            0 + 1.5811i
   3.3615           14.5911
```

Рис. 2.9 (окончание)

Простейшие действия над векторами и матрицами с применением стандартных функций MATLAB

Как следует из рисунка 2.9, применение элементарных математических функций к вектору (рис. 2.9, 9) и матрице (рис. 2.9, 10) означает, что эта функция будет применена к каждому элементу соответствующего массива.

2.3.4. Создание собственной элементарной функции с использованием стандартной функции MATLAB “inline”

Для создания собственной элементарной функции используется стандартная функция MATLAB “inline”, в круглых скобках которой и в кавычках необходимо записать свою функцию с произвольным числом переменных (например, x, y, z на рис. 2.10, 1). При этом созданная таким образом собственная “inline”-функция может быть присвоена любой переменной, как, например, rast на рисунке 2.10, 1. При выполнении этой операции присваивания без точки с запятой (;) после операции в командной строке Командного окна (Command Window)

>>rast = inline (‘*x*^2 + *y*^2 + *z*^2’)

на дисплее появится следующее сообщение:

rast =

Inline function:

rast (*x*, *y*, *z*) = *x*^2 + *y*^2 + *z*^2

Это означает, что для использования в программе созданной таким образом собственной “inline”-функции rast (*x*, *y*, *z*) переменным *x*, *y*, *z* необходимо присвоить конкретные (фактические) числовые значения аргументов (рис. 2.10, 1), после чего они должны быть перечислены в круглых скобках после названия функции “rast”:

>>*zz* = sqrt (rast (*xs*, *ys*, *zs*)).

Последовательность перечисления аргументов в последнем выражении должна строго соответствовать последовательности перечисления переменных при создании собственной “inline”-функции, как это было представлено в командном окне (Command Window) (см. выше).

Обязательным условием применения “inline”-функции является задание аргументам конкретных числовых значений или вычисление конкретных числовых значений используемых аргументов (рис. 2.10, 1):

>>*xs* = 2; *ys* = 3.3; *zs* = 4.5.

На рисунках 2.10, 2 и 2.10, 3 приведены примеры создания собственных “inline”-функций для расчета объема и площади поверхности цилиндра.

```
1. Определение расстояния в трехмерной системе координат:
>> rast=inline('x^2+y^2+z^2')
rast =
     Inline function:
     rast(x,y,z) = x^2+y^2+z^2
>> xs=2;ys=3.3;zs=4.5;
>> zr=sqrt(rast(xs,ys,zs))
zr =
    5.9279

2. Определение объема цилиндра:
 >> volume=inline('pi*R^2*H')
volume =
     Inline function:
     volume(H,R) = pi*R^2*H
>> R=2.5;H=5;
>> z=volume(R,H)
z =
196.3495

3. Определение площади поверхности цилиндра
>> surface=inline('2*pi*r*h');
>> zz=surface(0.25,0.5)
zz =
    0.7854
```

Рис. 2.10

Создание собственной функции с применением стандартной функции MATLAB "inline"

2.3.5. Арифметические операции

Пакет MATLAB ориентирован на выполнение операций с массивами (векторами и матрицами). Операции со скалярными величинами рассматриваются как частный случай операций с массивами, т. е. как операции с элемента-

ми массива с одним конкретным индексом в круглых скобках (рис. 2.11, 1) в случае вектора или двумя индексами в круглых скобках в случае матрицы (рис. 2.11, 2).

В общем случае операции с матрицами выполняются в соответствии с правилами линейной алгебры (рис. 2.11), и применение элементарных математических функций (табл. 2.3) к массивам предполагает, что они действуют на все элементы массива (рис. 2.9, 9 и 2.9, 10, а также рис. 2.11, 3 и 2.11, 4). Однако в некоторых случаях, в частности при выполнении операции возведения в степень всех элементов массива (рис. 2.11, 5а и 2.11, 5б), необходимо перед знаком возведения в степень (^) после идентификатора массива поставить точку (.) — *z*.^2 (рис. 2.11, 5).

Основные правила записи арифметических операций представлены в таблице 2.4. **Следует отметить, что для переноса слишком «длинных» выражений на другую строку используется многоточие (…).**

Таблица 2.4

Правила записи арифметических операций

Арифметическая операция	Математика	MATLAB
Сложение	a + b	a + b
Вычитание	a – b	a – b
Умножение	ab	a*b
Деление	a / b	a / b
Возведение в степень	a^b	a^b

Как следует из рисунков 2.11, 6 и 2.11, 7, при сложении (вычитании) и перемножении матриц должны выполняться условия, связанные с их размерами. При сложении (вычитании) размеры складываемых матриц должны совпадать (рис. 2.11, 6), а при перемножении число столбцов первой из перемножаемых матриц должно быть равно числу строк второй из перемножаемых матриц (второго сомножителя) (рис. 2.11, 7).

В зависимости от последовательности сомножителей при перемножении векторов (рис. 2.11, 8 и 2.11, 9) в качестве результата можно получить либо число — скалярное произведение векторов (рис. 2.11, 8), либо матрицу (рис. 2.11, 9).

```
1. Операции с элементами вектора:
>> a=[1.2 -2.5,3.1 5.6 125.3];
>> ass=a(1)+a(2)
ass =
  -1.3000

2. Операции с элементами матрицы:
>> BB=[1.4 2.7;3.4 23.5;5.6 77.8]
BB =
   1.4000   2.7000
   3.4000  23.5000
   5.6000  77.8000
>> ib=3;jb=1;
```

```
6. Сложение матриц A размером (3*2) с матрицей B
размером (3*2):
(размеры матриц должны быть одинаковыми):
>>  A=[1 2.1;3.7 -2;7 -0.125]
A =
   1.0000   2.1000
   3.7000  -2.0000
   7.0000  -0.1250
>> B=[9.325 -7.1;13.7 -25;1.37 -5.125]
B =
   9.3250  -7.1000
  13.7000 -25.0000
   1.3700  -5.1250
```

Рис. 2.11 (начало)

Арифметические операции с векторами, матрицами и их элементами

```
>> BBR=BB(ib,jb)*BB(1,2)
BBR =
  15.1200

3. Сложение матрицы с числом:
(каждый элемент матрицы складывается
с числом):
>> A=[1 2.1;3.7 -2;7 -0.125]
A =
   1.0000   2.1000
   3.7000  -2.0000
   7.0000  -0.1250
>> A5=A+5
A5 =
   6.0000   7.1000
   8.7000   3.0000
  12.0000   4.8750
4. Деление матрицы на число:
(каждый элемент матрицы делится на
число):
 >> A=[1 2.1;3.7 -2;7 -0.125];
>> d=A/5
d =
   0.2000   0.4200
   0.7400  -0.4000
   1.4000  -0.0250
5. Возведение массива в степень
а) (каждый элемент матрицы возводится
в степень):
>> A=[1 2.1;3.7 -2;7 -0.125];
>> aaa=A.^2

aaa =
         1.0000   4.4100
        13.6900   4.0000
        49.0000   0.0156
б) (каждый элемент матрицы возводится
в дробную степень):
>> a=[1 2 3;5 6 7];
>> aa=a.^2.5
aa =
   1.0000   5.6569  15.5885
  55.9017  88.1816 129.6418
```

```
>> C=A+B
C =
  10.3250  -5.0000
  17.4000 -27.0000
   8.3700  -5.2500

7. Перемножение матрицы "A" размером (3*2) на мат-
рицу "D5" размером (2*5):
(число столбцов первой перемножаемой матрицы
("A») должно быть равно числу строк второй пере-
множаемой матрицы ("D5")):
>> A=[1 2.1;3.7 -2;7 -0.125]
A =
   1.0000   2.1000
   3.7000  -2.0000
   7.0000  -0.1250

>> D5=[1.1 7.3 -2.4 3.2 0.725;-3.2 3.25,7.21 2.15 -125.12];
(получается результирующая матрица "cc1" размером
(3*5)):
>> cc1=A*D5
cc1 =
  -5.6200  14.1250  12.7410   7.7150 -262.0270
  10.4700  20.5100 -23.3000   7.5400  252.9225
   8.1000  50.6938 -17.7013  22.1313   20.7150
8. Перемножение векторов "a" размером (1*4) на век-
тор "b" размером (4*1):
>> a=[1.3 2.5 7.1 4.2];
>> b=[0.1;7.5;4.1;1.25];
(получается результат —  скалярное число "c" с раз-
мером (1*1)):
>> c=a*b

c =
  53.2400
9. Перемножение векторов "b" размером (4*1) на век-
тор "a" размером (1*4):
>> a=[1.3 2.5 7.1 4.2];
>> b=[0.1;7.5;4.1;1.25];
>> c1=b*a
(получается результат —  матрица "c1" с размером
(4*4)):
c1 =
   0.1300   0.2500   0.7100   0.4200
   9.7500  18.7500  53.2500  31.5000
   5.3300  10.2500  29.1100  17.2200
   1.6250   3.1250   8.8750   5.2500
```

Рис. 2.11 (окончание)

Арифметические операции с векторами, матрицами и их элементами

2.3.6. Логические операции

Логические операции необходимы для проверки справедливости определенных условий, которые следует определять при выполнении условных операторов “if” и операторов цикла “while”. Так, в условном операторе “if” и операторе цикла “while” от результата логических операций зависит дальнейший ход вычислительного процесса.

В результате выполнения логических операций логической переменной присваивается одно из значений — 1 или 0. Логическая переменная получает значение 1, если результат выполнения логического отношения или логического выражения является истинным (**True**), и значение 0, если результат выполнения логического отношения или логического выражения является ложным (**False**).

В логических выражениях (рис. 2.12, 2) используются логические отношения (рис. 2.12.1) и логические операции (рис. 2.12). В логических отношениях выполняется сравнение двух операторов и определяется, истинно ли логическое отношение или ложно (рис. 2.12.1). Правила записи операций логических отношений приведены в таблице 2.5, а логических выражений — в таблице 2.6. Результатом выполнения операций логических отношений (рис. 2.12.1) и логических выражений (рис. 2.12, 2) всегда являются целые числа 1 или 0 (рис. 2.12).

Логические выражения включают в себя несколько логических отношений; основные виды логических выражений и их обозначения в среде MATLAB («и», «или», «не») приведены в таблице 2.6, а результаты их выполнения — в таблице 2.7.

Таблица 2.5

Правила записи операций отношения

Операция отношения	Математика	MATLAB
Равно	`a = b`	`a == b`
Неравно	`a ≠ b`	`a ~= b`
Больше	`a > b`	`a > b`
Меньше	`a < b`	`a < b`
Больше или равно	`a ≥ b`	`a >= b`
Меньше или равно	`a ≤ b`	`a <= b`

Таблица 2.6

Правила записи логических выражений

Тип выражения	Выражение	Логический оператор
Логическое «и»	*A* «и» *B*	`and (A, B)`
Логическое «или»	*A* «или» *B*	`or (A, B)`
Исключающее «или»	*A* исключающее «или» *B*	`xor (A,B)`
Отрицание	«не» *A*	`not (A)`

Таблица 2.7

Результаты логических операций над логическими переменными *A* и *B*

A	***B***	**«не» *A***	***A* «и» *B***	***A* «или» *B***	***A* исключающее «или» *B***
0	0	1	0	0	0
0	1	1	0	1	1
1	0	0	0	1	1
1	1	0	1	1	0

Следует отметить, что для массивов одинакового размера также определены логические операторы (выражения), они выполняются над каждым элементом массива. В итоге формируется результирующий массив логических значений, каждый элемент которого вычисляется путем выполнения логической операции вычисления значения логического выражения над соответствующими элементами исходных массивов (рис. 2.12, 2).

```
1. Логические отношения:
 а) для скаляров
>>x=5;z=3;
>> szx=x~=z
szx =
    1

б)
>>x=5;z=3;
>> szx=x==z
szx =
    0
в) для матриц
>>  a=[1 3;5 4];
>> b=[0.5 2;4 3];
>> s1=a>b
s1 =
    1    1
    1    1

>> a=[1 3;5 4];
>> b=[0.5 2;4 5];
>> s=a>b

s =

    1    1
    1    0
```

```
2. Логические выражения:
а)
>> z=5; x=3; d=1.25;
>> sa=and(d<z,d<x)
sa =
    1

б)
>> z=5; x=3; d=1.25;
>> sb=or(d<z,d>x)
sb =
    1
в)
>> z=5; x=3; d=1.25;
>> sc=or(d<z,d>=x)
sc =
    1

г)
>> z=5; x=3; d=1.25;
>> sc=or(d<=z,d>=x)
sc =
    1
д)
>> z=5; x=3; d=1.25;
>> sc=or(d==z,d>=x)
sc =
    0
```

Рис. 2.12

Логические отношения и логические выражения

2.3.7. Совместное выполнение арифметических и логических операций

При одновременном использовании в выражении логических и арифметических операций возникает проблема последовательности их выполнения.

В MATLAB принят следующий приоритет операций.

1. Логические операторы.
2. Логическая операция.
3. Транспонирование матриц, операции возведения в степень, унарный «плюс» и «минус».
4. Умножение, деление.
5. Сложение, вычитание.
6. Операции отношения.
7. Логическая операция «и».
8. Логическая операция «или».

Как упоминалось (на примере системной переменной ans), в MATLAB может использоваться целый ряд системных переменных, среди которых основными являются:

- ans — результат последней операции без знака присваивания;
- i, j — мнимая единица ($\sqrt{-1}$);
- pi — число π (3.141592653589793);

- inf — машинный символ бесконечности (∞);
- NaN — неопределенный результат (0/0, ∞/∞ и т. п.);
- realmin — наименьшее число с плавающей точкой (2.2251e – 308);
- realmax — наибольшее число с плавающей точкой (1.7977e + 308).

Перечисленные системные переменные пользователь может иногда использовать в выражениях, а в некоторых случаях система выдает их самостоятельно при выводе результатов расчетов.

2.4. Оператор условного перехода “if”

Структура простейшего условного оператора имеет вид (рис. 2.13.1):

```
if<условие>
<операторы>
end
```

где <условие> — это логическое выражение, результат вычисления которого может быть либо истинным (**True**) — соответствующая ему логическая переменная принимает значение 1, либо ложным (**False**) — соответствующая ему логическая переменная принимает значение 0; <операторы> — операторы *m*-языка MATLAB, которые должны выполняться до оператора end; если справедливо <условие>, результат вычисления логического выражения в условии равен 1 (условие справедливо).

Условия, представленные в виде одной операции отношения (табл. 1.5), называются **простыми условиями** (рис. 2.13.1а). Условия, записанные в виде нескольких операций отношения — в виде логических выражений, называются **составными условиями** (рис. 2.13.1б).

Следует отметить, что операторы между “if” и “end” выполняются только при справедливости условия после “if”, т. е. когда значение соответствующей <условию> логической переменной равно 1. В противном случае, если условие не выполнено, указанные операторы при выполнении программы игнорируются.

Часто используется расширенная форма условного оператора с “else”, имеющая вид (рис. 2.13, 2):

```
if <условие>
<операторы 1>
else
<операторы 2>
end
```

В отличие от предыдущего условного оператора с простым условием (рис. 2.13.1а) при невыполнении <условия> в этом случае выполняются все <операторы 2> после “else” до оператора “end”, а <операторы 1> после <условия> до “else” игнорируются. <Операторы 1> между <условием> после “if” и “else” выполняются при выполнении <условия> после “if”, при этом будут игнорироваться <операторы 2>.

Можно использовать и третью форму условного оператора с несколькими условиями (рис. 2.13, 3):

if <условие 1>
<операторы 1>
elseif <условие 2>
<операторы 2>
else
<операторы 3>
end

Такой оператор следует применять, когда в случае невыполнения <условия 1> необходимо проверить некоторое дополнительное <условие 2>. При этом операторы 1, 2 и 3 выполняются в строгом соответствии с перечисленными выше правилами.

```
1. Условный оператор без "else"
а) с простым условием и одной логической переменной — логическим отношением
>> a=3;b=5.1;c=7;
>> if b>a
z=a+b
end

z = 8.1000
б) с составным условием и двумя логическими переменными — логическим выражением
>> a=3;b=5.1;c=7;
>> if and(b>a,a<c)
z1=a-b
z2=sqrt(c-a)
end

z1 = -2.1000


z2 = 2

2. Расширенная форма условного оператора с "else"
>> a=3.2;b=7.3;c=2.3;d=8.3;
а) выполняется условие
>> if or(a>0,c<3)
res1=(a+b)^3.2
else
res2=(c+d)^2.3
end

res1 = 1.8527e+003

б) не выполняется условие
>> if and(a==0,c<3)
res1=(a+b)^3.2
else
res2=(c+d)^2.3
end
```

Рис. 2.13 (начало)

Основные виды условного оператора “if” и действия с ними

```
res2 = 228.1411

>> a=3.2;b=7.3;c=2.3;d=8.3;
if or(a>0,c>=3)
res1=(a+b)^3.2
elseif d==3.8
res2=(c+d)^2.3
end

res1 = 1.8527e+003

3. Расширенная форма условного оператора с двумя условиями и с "elseif"
>> a=3.2;b=7.3;c=2.3;d=8.3;
if and(a>0,c>=3)
res1=(a+b)^3.2
elseif d==3.8
res2=(c+d)^2.3
else
res3=a+b+c+d
end

res3 = 21.1000
```

Рис. 2.13 (окончание)

Основные виды условного оператора “if” и действия с ними

2.5. Оператор выбора “switch”

Структура этого оператора имеет вид (рис. 2.14.1):

switch <ключ>
case <значение 1>
<операторы 1>
……..
case <значение *N*>
<операторы *N*>
end

или в расширенном варианте оператора (рис. 2.14, 2):

switch <ключ>
case <значение 1>
<операторы 1>
……..
case <значение *N*>
<операторы *N*>
otherwise
<операторы>
end

“Switch” <ключ> — это переменная или выражение, значение которого представляет собой скаляр или строку. По значению <ключа> осуществляется выбор группы операторов, которые будут выполняться программой.

По значению “switch” (ключа) осуществляется выбор “case” (в общем случае из *N* возможных), операторы которого будут выполняться.

Если значение “switch” (ключа) не совпадает с одним из значений “case”-ов, то:

— для первого варианта оператора “switch” без “otherwise” (рис. 2.14.1) все операторы между “switch” и “end” игнорируются программой;

— для второго варианта оператора “switch” с “otherwise” (рис. 2.14, 2) будут выполняться операторы между “otherwise” и “end”.

Очевидно, что при совпадении значения “switch” (ключа) с одним из значений “case”-ов будут выполняться только операторы, следующие непосредственно за “case”-ом с этим значением до “case”-а с другим значением, или до служебных операторов “otherwise” или “end”.

Следует отметить, что значение “switch” (ключа) в “case” может задаваться не только одним значением (рис. 2.14, 2а), но и массивом, один элемент которого должен совпасть с предварительно заданным значением (рис. 2.14, 2б). При этом элементы массива в “case” берутся в фигурные скобки.

Последовательность элементов массива в “case” может задаваться стандартной функцией “num2cell” совместно с оператором двоеточия (рис. 2.14, 2б).

```
1. Оператор выбора switch без "otherwise"
>> z=3;
>> switch z
case 1
x1=5.3
case 2
x2=7.4
case 3
x3=55
case 4
x4=44
end

x3 = 55

2. Оператор выбора switch с "otherwise"
а) с одним числом в "case"
>> z=5;
>>switch z
case 1
x1=5.3
case 2
x2=7.4
case 3
x3=55
case 4
x4=44
otherwise
x5=555
end

x5 = 555
```

```
б) с массивом чисел в "case"
>> z=17;
>>switch z
case 1
x1=5.3
case 2
x2=7.4
case {3,12,17,25}
x3=z-17
case 4
x4=44
otherwise
xxx=555
end

x3 = 0

б) с массивом чисел в "case", создаваемым
стандартной функцией num2cell и операто-
ром двоеточия
>> z=14;
>>switch z
case 1
x1=5.3
case 2
x2=7.4
case num2cell(2:3:17)
x3=z+14
case 4
x4=44
otherwise
xxx=555
end

x3 = 28
```

Рис. 2.14

Основные виды оператора выбора “switch” и действия с ним

2.6. Оператор цикла “for”

Этот оператор служит для многократного выполнения группы операторов программы (<операторы>) с автоматически изменяющимся значением параметра цикла (<параметр цикла>), задаваемым оператором двоеточия от величины <*xn*> до величины <*xk*> с положительным или отрицательным шагом изменения параметра цикла <*xo*>.

Оператор цикла в общем случае начинается со служебного слова “for” и заканчивается служебным словом “end” и имеет вид:

for <параметр цикла> = <*xn*>:<*xd*>:<*xk*>
<операторы>
end

или, если шаг изменения параметра цикла <xd> равен 1, общий вид оператора цикла преобразуется следующим образом:

for <параметр цикла> = <*xn*>:<*xk*>
<операторы>
end.

Значения параметра цикла “for” представляют собой числовую последовательность, принадлежащую к одному и тому же типу данных и задаваемую оператором двоеточия:

<параметр цикла> = <*xn*>:<*xd*>:<*xk*>

с границами изменения <*xn*> и <*xk*> и шагом <*xd*>.

Параметр цикла в последнем соотношении называется счетчиком циклов, который может быть вещественным — принимать значения вещественных чисел (рис. 2.15.1) или натуральным — принимать значения натуральных чисел (рис. 2.15, 2). Для каждого значения счетчика цикла выполняются все <операторы>, расположенные в операторе цикла “for” после указания границ его изменения, вплоть до служебного слова “end”.

На рисунках 2.15.1 и 2.15, 2 приведены примеры соответственно с вещественным и натуральным счетчиком циклов, причем в последнем случае шаг изменения счетчика цикла (параметра цикла) по умолчанию равен 1, поэтому не представлен в операторе цикла (рис. 2.15, 2). Следует отметить, что параметры цикла, его границы и шаг нельзя изменять с помощью <операторов>, расположенных внутри цикла.

В операторе цикла “for” существует возможность не выполнять расчет <операторов>, например, при конкретном значении счетчика цикла (рис. 2.15, 3). Для решения этой задачи в число операторов цикла включается условный оператор “if”, который при выполнении условия равенства счетчика цикла нежелательному его значению с помощью выполнения оператора “continue” прерывает текущий шаг цикла. При этом все остальные шаги цикла выполняются (рис. 2.15, 3).

Для полного прерывания выполнения оператора цикла “for” используется оператор “break” (рис. 2.15, 4). В этом случае в число операторов цикла также включается условный оператор “if”, и если некоторые из переменных, используемых в операторах цикла, принимают значение, соответствующее условию оператора “if”, то выполняется оператор “break” и работа оператора цикла “for” прекращается (рис. 2.15, 4).

1. Оператор цикла "for" с вещественным счетчиком циклов:

```
>>ij=0;
>>for T=363.5:10.7:403.1
ij=ij+1
TV=T
PV=exp(18.3-3816.4/(-46.1+TV))
end
РЕЗУЛЬТАТ:
ij = 1
TV = 363.5000
PV = 531.6873
ij = 2
TV = 374.2000
PV = 786.9624
ij = 3
TV = 384.9000
PV = 1.1363e+003
ij = 4
TV = 395.6000
PV = 1.6042e+003
```

2. Оператор цикла "for" с натуральным счетчиком циклов:

```
>> for i=2:7
ii=i
dsum=ii^2
end
РЕЗУЛЬТАТ:
ii = 2
dsum = 4
ii = 3
dsum = 9
ii = 4
dsum = 16
ii = 5
dsum = 25
ii = 6
dsum = 36
ii = 7
dsum = 49
```

3. Оператор цикла "for" с натуральным счетчиком циклов и оператором "continue", прерывающем выполнение текущего шага цикла:

```
>> for i=2:7
if i==3
continue
end
ii=i
dsum=ii^2
end
РЕЗУЛЬТАТ:
ii = 2
dsum = 4
ii = 4
dsum = 16
ii = 5
dsum = 25
ii = 6
dsum = 36
ii = 7
dsum = 49
```

4. Оператор цикла "for" с натуральным счетчиком циклов и оператором "break", прерывающем выполнение цикла:

```
>> for i=2:7
ii=i
dsum=ii^2
if dsum>=25
break
end
end
РЕЗУЛЬТАТ:
ii = 2
dsum = 4
ii = 3
dsum = 9
ii = 4
dsum = 16
ii = 5
dsum = 25
```

Рис. 2.15

Оператор цикла “for” для выполнения операций заданное число раз в определенном интервале изменения переменной и операторs прерываний “continue” и “break”

2.7. Оператор цикла “while”

Оператор цикла “while” также служит для многократного выполнения группы операторов до тех пор, пока справедливо условие, соответствующее логическому выражению оператора “while”, и в общем случае имеет вид:

while <условие>
<операторы>
end.

Если логическое выражение в <условии> принимает значение «истина» (“true”) или 1, то все <операторы> между <условием> и служебным словом “end” будут выполняться.

В противном случае, когда логическое выражение в <условии> принимает значение «ложь» (“false”) или 0, <операторы> в теле цикла между “while” и служебным словом “end” не будут выполняться.

Для корректной организации работы с оператором цикла “while” необходимо, чтобы значение выражения и соответствующей переменной, проверяемой в <условии>, обязательно задавалось перед оператором цикла “while” и обязательно изменялось внутри цикла.

На рисунках 2.16, 1 и 2.16, 2 показано, как можно организовать работу оператора цикла “while” для задач, выполненных ранее с использованием оператора цикла “for” с вещественным счетчиком циклов (рис. 2.15, 1) и натуральным счетчиком циклов (рис. 2.15, 2).

```
1. Оператор цикла "while" с условием из вещественных чисел:
>> T=363.5;ij=0;TD=10.7;
>> while T<=403.1
ij=ij+1
TV=T
PV=exp(18.3-3816.4/(-46.1+TV))
T=T+TD;
end
РЕЗУЛЬТАТ:
ij = 1
TV = 363.5000
PV = 531.6873
ij = 2
TV = 374.2000
PV = 786.9624
ij = 3
TV = 384.9000
PV = 1.1363e+003
ij = 4
TV = 395.6000
PV = 1.6042e+003

2. Оператор цикла "while" с условием из натуральных чисел:
>> i=2;di=1;
while i<=7
ij=i
dsum=ij^2
i=i+di;
end
РЕЗУЛЬТАТ:
ij = 2
dsum = 4
ij = 3
dsum = 9
ij = 4
dsum = 16
ij = 5
dsum = 25
ij = 6
dsum = 36
ij = 7
dsum = 49
```

Рис. 2.16

Оператор цикла “while” для многократного выполнения группы операторов в соответствии с некоторым условием

2.8. Построение двумерных (плоских) графиков с использованием функции “plot”

2.8.1. Построение одной или нескольких функций на одном графике

Функции от одной независимой переменной могут быть представлены либо в табличном виде (первый тип данных), либо в виде аналитической зависимости (второй тип данных). Программа построения графика для первого типа данных приведена на рисунке 2.17, 1, а для второго типа данных — на рисунке 2.17, 2. Графическое представление таких функциональных зависимостей в интегрированной среде MATLAB сводится к следующему:

- подготовка числовых данных (A);
- создание графического окна и построение графика (B);
- задание параметров графика (C).

A. Подготовка числовых данных.

Числовые данные должны быть заданы в табличном виде и представлены в MATLAB в виде двух одномерных массивов одинаковой длины, один из которых соответствует набору данных по независимой переменной, второй — по зависимой.

```
1.Представление  табличных данных в виде двумерного графика;
>> x=[0 0.058,0.078,0.136,0.167,0.201,0.229,0.363,0.398,0.584,0.611,0.746,0.917,0.921  1];...
y=[0 0.118,0.153,0.252,0.295,0.353,0.381,0.501,0.526,0.653,0.666,0.759,0.907,0.913  1];...
x1=0:0.1:1;...
y1=x1.^1;...
plot(x,y,'k-.',x1,y1,'k-');...
text(0.83,0.83,'\rightarrow  особая точка');...
title('Представление  табличных данных y=f(x) в виде функции');...
xlabel('x');...
ylabel('y');...
grid on;

2.Построение  графика функции уравнения многочлена.
 >> x=[-2:0.1:1];...
 y=x.^6+3*x.^5+2*x.^4+7*x.^2+x.*6-5;...
x1=-2:0.1:1;...
y1=1-x1.^0;...
 plot(x,y,'k-.',x1,y1,'k-');...
grid on;...
title('Графическое  предс. функции уравн. x^6+3*x^5+2*x^4+7*x^2+6*x-5=0');...
xlabel('Аргумент  функции — x');...
ylabel('Функция  f(x)=x^6+3*x^5+2*x^4+7*x^2+6*x-5');...
axis([-2  1  -7  15]);...
text(-1.5,0,'\rightarrow  Корень 1');...
text(0.5,0,'\leftarrow  Корень 2');...
text(-0.485,-6.15,'\leftarrow  Минимум');
```

Рис. 2.17

Программы для построения двумерных (плоских) графиков с применением оператора “plot”

Первый тип данных с табличным представлением функциональной зависимости представляется в программе в виде одномерных массивов для независимой и зависимой переменных (рис. 2.17, 1).

Для дискретизации аналитических зависимостей (второй тип данных (рис. 2.17, 2)) для независимой переменной можно воспользоваться либо оператором двоеточие (:), либо функцией “linspace” (см. предыдущие разделы):

x=*xmin*:*xd*:*xmax*

или

x=linspace (*xmin*, *xmax*, *n*),

где *xmin*, *xmax* — минимальное и максимальное значения независимой переменной; *xd* — шаг дискретизации; *n* — число интервалов разбиения отрезка — *xmin*, *xmax*.

В результате получается одномерный массив, состоящий из определенного числа точек независимой переменной в заданном интервале (*xmin*, *xmax*) — (–2, 1) (рис. 2.17, 2). Следует отметить, что можно сформировать одномерный массив и для неравномерно распределенных в интервале значений независимых переменных.

В любом случае для полученных дискретных значений аргументов должны быть рассчитаны дискретные значения функций (*y*), как, например, в программе на рисунке 2.17, 2:

>> *x* = [–2:0.1:1]

y = *x*.^6 + 3 ∗ *x*.^5 + 2 ∗ *x*. ^4 + 7 ∗ *x*. ^2 + *x*. ∗ 6 – 5 [1]

В. Создание графического окна и построение графика.

Программа автоматически создает одно графическое окно при записи функции построения графика — *y* = *f*(*x*). В простейшем случае эта функция имеет вид

plot (*x*,*y*).

При этом важно отметить, что на первом месте необходимо указать одномерный массив аргументов (*x*), а на втором — функций (*y*).

В MATLAB существует стандартная функция “figure”, которая используется, когда необходимо разместить графическую информацию в нескольких графических окнах.

В таблице 2.8 приведены различные стандартные функции MATLAB для построения двумерных графиков.

Таблица 2.8

Стандартные функции MATLAB для построения двумерных графиков

Функция	Описание
plot (…)	график функции одной независимой переменной в декартовой системе координат

[1] Нижняя точка рядом с идентификатором переменной, как указывалось ранее, означает, что в степень возводятся все элементы массива *x*.

Продолжение табл. 2.8

Функция	Описание
line (…)	ломаная, соединяющая прямыми линиями точки, координаты которых заданы в декартовой системе координат (частный случай: если заданы координаты двух точек, строится прямая)
loglog (…)	график функции одной переменной в логарифмическом масштабе (по обеим осям)
bar (…)	столбцовая диаграмма (вертикальная)
barh (…)	столбцовая диаграмма (горизонтальная)
hist (…)	гистограмма
errorbar (…)	график функции одной переменной с зонами погрешности в каждой точке
stem (…)	дискретный график функции одной переменной (не соединяет точки)
pie (…)	круговая диаграмма по значениям одномерного массива

При построении графиков, в частности в декартовой системе координат с применением функции “plot”, обычно в кавычках можно указать три параметра графика (табл. 2.9): тип линии, тип маркера и цвет линии. Так, например, запись

plot (*x*, *y*,‘-.dg’)

означает, что тип линии — штрихпунктир, с ромбами в реперных точках и зеленого цвета.

Таблица 2.9

Способы задания типа и цвета линий, а также маркеров точек на графике в MATLAB

Тип линии		**Тип маркера**	
–	сплошная	.	точка
:	двойной пунктир	○	окружность
–.	штрихпунктир	x	крест
_	штриховая	+	плюс
Цвет линии		*	звездочка
y	желтый	s	квадрат
m	фиолетовый	d	ромб
c	голубой	v	треугольник (вниз)
r	красный	^	треугольник (вверх)
g	зеленый	<	треугольник (влево)
b	синий	>	треугольник (вправо)
w	белый	p	пятиугольник
k	черный	h	шестиугольник

При построении нескольких графиков в одной системе координат должны быть созданы одномерные массивы с разными идентификаторами (*x*, *y* и *x*1, *y*1), как на рисунках 2.17, 1 и 2.17, 2, а три параметра линий графиков могут задаваться в кавычках отличными друг от друга.

Следует отметить, что некоторые из трех параметров линий могут вообще не задаваться и будут в таком случае автоматически установлены программой.

Примеры построения двух графиков в одной системе координат приведены в программах на рисунках 2.17, 1 и 2.17, 2. В первом случае (рис. 2.17, 1) второй график ($x1$, $y1$) задан в виде диагонали (первый график x, y — в виде функции в табличном виде) (рис. 2.18):

```
x1 = 0 : 0.1 : 1;
y1 = x1.^1;
plot (x,y,'k-.',x1,y1,'k-'),
```

а во втором — второй график ($x1$, $y1$) задан в виде прямой на оси абсцисс, а первый график (x, y) — в виде многочлена шестой степени (рис. 2.19):

```
x1 =-2:0.1: 1;
y1 = 1 – x1.^0;
plot (x, y, 'k-.',x1,y1,'k-').
```

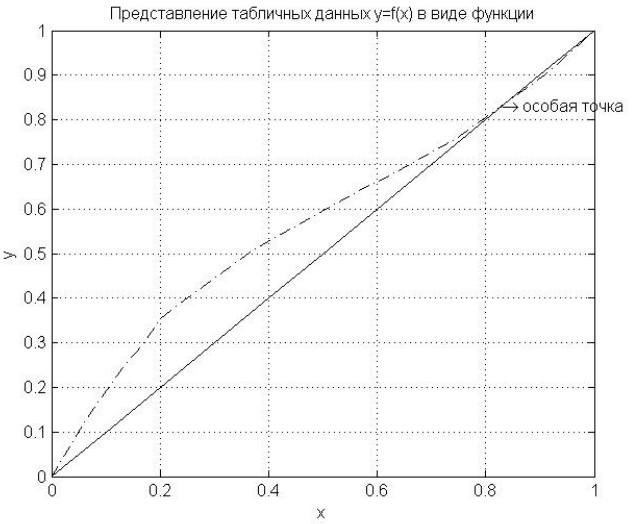


Рис. 2.18

График плоский с применением функции “plot” и фиксацией особой точки

С. Задание параметров графика.

Основные опции для задания параметров графиков приведены в таблице 2.10.

Таблица 2.10

Настройки графиков в MATLAB

Функция	Описание
title (‘надпись’)	позволяет задать титульную надпись
x label (‘надпись’)	позволяет задать надпись для оси X
y label (‘надпись’)	позволяет задать надпись для оси Y
text (x, y, ‘надпись’)	позволяет задать надпись в точке с координатами (x, y)

Продолжение табл. 2.10

Функция	Описание
gtext (‘надпись’)	позволяет задать надпись в позиции курсора мыши (навести курсор и нажать любую клавишу)
axis ([*xmin xmax ymin ymax*])	устанавливает диапазон координат по осям *X* и *Y*
axis auto	устанавливает параметры осей по умолчанию
axis off	убирает оси
axis on	восстанавливает оси
grid off	отключает сетку
grid on	добавляет сетку
legend (‘надпись1’, ‘надпись2’, …, Pos)	добавляет к текущему графику легенду из перечисленных надписей; Pos задает положение легенды (0 – выбирать автоматически; 1 — верхний правый угол; 2 — верхний левый угол; 3 — нижний левый угол; 4 — нижний правый угол)
legend off	удаляет легенду
hold on	включает режим продолжения вывода графиков в текущее окно
hold off	выключает режим продолжения вывода графиков в текущее окно
subplot (*n*, *m*, *p*)	разбивает графическое окно на подокна (*n* — количество подокон по горизонтали; *m* — количество подокон по вертикали; *p* — номер текущего подокна)
zoom (factor)	устанавливает коэффициент увеличения (уменьшения)

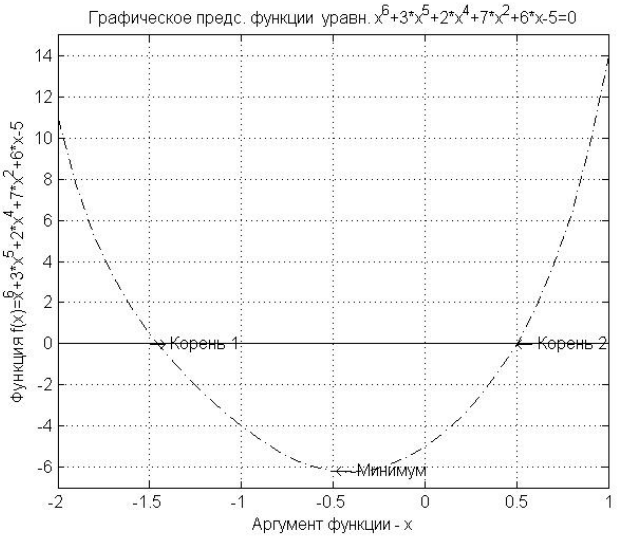


Рис. 2.19

График плоский с применением функции “plot” и фиксацией двух вещественных корней уравнения и минимума функции

Следует обратить внимание на функцию “axis”, которая позволяет на графике в явном виде установить диапазоны изменения переменных на осях абсцисс и ординат. Например, для более четкого отображения первого корня

(Корень 1) на рисунке 2.19 функция “axis” может быть записана в следующем виде:

axis ([–1.5 –1 –22]),

и второго корня (Корень 2):

axis ([0 1 –22]).

Как видно из программы на рисунке 2.17, 2, основными дополнительными настройками графиков являются (табл. 2.10):

title (‘надпись’) — название графика;

*x*label (‘надпись’) — надпись для оси абсцисс (*x*);

*y*label (‘надпись’) — надпись для оси ординат (*y*);

axis ([*xmin xmax ymin ymax*]) — диапазон координат по осям *x* и *y*.

Может быть важным задание надписи с использованием функции “text” в точке с координатами (*x*, *y*), например:

на рисунках 2.17, 1 и 2.18 — точка пересечения функции с диагональю в точке с координатами $x = 0.83$ и $y = 0.83$:

text (0.83,0.83,‘\rightarrow особая точка’)

(стрелка с надписью «особая точка» направлена вправо);

на рисунках 2.17, 2 и 2.19:

а) первая точка с координатами $x = 1.5$ и $y = 0$ — корень первого уравнения:

text (1.5,0,‘\rightarrow Корень 1’)

(стрелка с надписью «Корень 1» направлена вправо);

б) вторая точка с координатами $x = 0.5$ и $y = 0$ — корень второго уравнения:

text (0.5, 0, ‘\leftarrow Корень 2’)

(стрелка с надписью «Корень 2» направлена влево);

в) третья точка с координатами $x = –0.485$ и $y = –6.15$ — минимум функции:

text (–0.485,–6.15,‘\leftarrow Минимум’)

(стрелка с надписью «Минимум» направлена влево).

2.8.2. Построение нескольких графиков в разных окнах с использованием функции “figure”

В MATLAB для создания нескольких графических окон используется стандартная функция “figure”.

При использовании этой функции сначала следует продекларировать указатели на предполагаемые графические окна. Например, в программном коде (рис. 2.19, 1) для создания двух графических окон с указателями hfig(1) и hfig(2) следует записать:

hfig(1) = figure;

hfig(2) =figure.

Далее, перед построением графиков sin(t), cos(t), tg(t) в двух графических окнах с применением функции “plot”, следует обратиться к функции “figure” следующим образом (рис. 2.19, 1):

а) для первого графического окна —

figure (hfig(1)); plot (t, x, t, y);

б) для второго графического окна —

figure (hfig(2)); plot (t, z).

Результат работы программы, приведенный на рисунке 2.19, 1, представляет собой два разных графика: Figure 1 — с отображением графиков двух функций sin(t) и cos(t), и Figure 2 — с отображением функции tg(t).

2.8.3. Построение нескольких графиков в одном окне с использованием функции “subplot”

Как следует из таблицы 2.10, для этой цели используется функция “subplot(n, m, p)”, которая разбивает графическое окно на подокна:

n — число подокон по горизонтали;

m — число подокон по вертикали;

p — номер текущего подокна по порядку, начиная с 1.

Например, в соответствии с программой (рис. 2.10, 2) одно окно разбивается на два подокна по горизонтали и два подокна по вертикали. Для того чтобы разместить три графика в соответствующие подокна, перед каждой функцией построения графика “plot” необходимо записать функцию “subplot” в следующем виде (рис. 2.19, 2):

а) для первого графика

subplot(2, 2, 1)

plot(t,x)

б) для второго графика

subplot(2,2,2)

plot(t,y)

в) для третьего графика

subplot(2,2,3)

plot(t,z).

Четвертое подокно отсутствует, так как четвертая функция не строится, поэтому окно не задействовано.

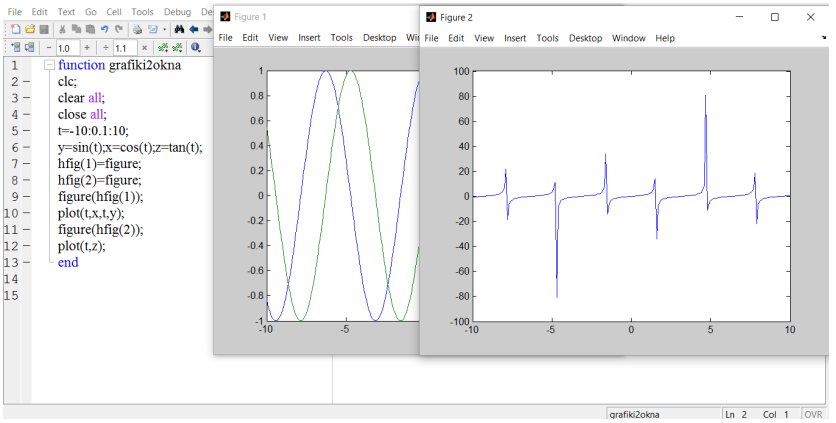


Рис. 2.19, 1

Программа создания двух графических окон для трех графиков (sin(t), cos(t), tg(t)) и результат ее работы: Figure 1 и Figure 2

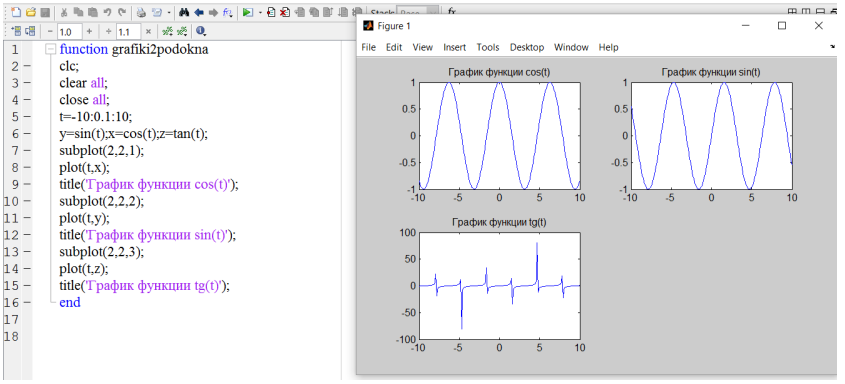


Рис. 2.19, 2

Программа создания трех подокон для трех графиков (sin(t), cos(t), tg(t)) в одном окне

2.9. Построение трехмерных (объемных) графиков с использованием функций “mesh”, “surf”, “meshc” и “surfc”

При построении графиков функций от двух независимых переменных *x* и *y* необходимо задать множество координат точек на плоскости *x*–*y* и рассчитать значения функции зависимой переменной *z* в каждой из них. Данные должны быть представлены в виде трех массивов, например *x*, *y*, *z*.

Для задания первых двух массивов обычно используется функция “mesh-grid” (рис. 2.20):

[*x*, *y*] = meshgrid (*xmin*:*xd*:*xmax*,*ymin*:*yd*:*ymax*),

которая позволяет сгенерировать дискретный набор значений *x* в диапазоне [*xmin*, *xmax*] с шагом *xd* и дискретный набор значений *y* в диапазоне [*ymin*, *ymax*] с шагом *yd*.

Для получения значений функции z — третьего массива данных — используются матричные операции, как, например, на рисунках 2.20, 1–2.20, 4:

$z = x.\hat{}2 + y.\hat{}2;$

или (рис. 2.20, 5 и 2.20, 6):

$z = 100*(x.\hat{}2 - y).\hat{}2 + (1 - x).\hat{}2;$

```
1. Трехмерный график функции с оператором "mesh" без линий постоянного уровня на плоскости;
>>[x,y]=meshgrid(-0.5:0.01:0.5,-0.5:0.01:0.5);...
z=x.^2+y.^2;...
mesh(x,y,z);...
title('График функции z=x^2+y^2');...
xlabel('x');ylabel('y');zlabel('z');

2. Трехмерный график функции с оператором "meshc" с линиями постоянного уровня на плоскости;
>>[x,y]=meshgrid(-0.5:0.01:0.5,-0.5:0.01:0.5);...
z=x.^2+y.^2;...
meshc(x,y,z);...
title('График функции z=x^2+y^2');...
xlabel('x');ylabel('y');zlabel('z');

3. Трехмерный график функции с оператором "surf" без линий постоянного уровня на плоскости;
>>[x,y]=meshgrid(-0.5:0.01:0.5,-0.5:0.01:0.5);...
z=x.^2+y.^2;...
surf(x,y,z);...
title('График функции z=x^2+y^2');...
xlabel('x');ylabel('y');zlabel('z');

4. Трехмерный график функции с оператором "surfc" с линиями постоянного уровня на плоскости;
>>[x,y]=meshgrid(-0.5:0.01:0.5,-0.5:0.01:0.5);...
z=x.^2+y.^2;...
surfc(x,y,z);...
title('График функции z=x^2+y^2');...
xlabel('x');ylabel('y');zlabel('z');

5. Трехмерный график функции с оператором "surf" без линий постоянного уровня на плоскости;
>>[x,y]=meshgrid(-2:0.1:2,-3:0.1:3);...
z=100*(x.^2-y).^2+(1-x).^2;...
surf(x,y,z);...
title('z=100*(x^2-y)^2+(1-x)^2');...
xlabel('x');ylabel('y');zlabel('z');

6. Трехмерный график функции с оператором "surfc" с линиями постоянного уровня на плоскости;
>>[x,y]=meshgrid(-2:0.1:2,-3:0.1:3);...
z=100*(x.^2-y).^2+(1-x).^2;...
surfc(x,y,z);...
title('z=100*(x^2-y)^2+(1-x)^2');...
xlabel('x');ylabel('y');zlabel('z');
```

Рис. 2.20

Изображение трехмерных (объемных) графиков с функциями “mesh”, “meshc”, “surf”, “surfc”

Наиболее удобно для построения трехмерных графиков использовать функции “mesh” и “surf” с тремя параметрами, например (рис. 2.20, 1, 2.20, 3, 2.20, 5):

mesh (x, y, z)

или

surf (x, y, z).

Результаты применения этих функций для построения графиков приведены на рисунках 2.21–2.23.

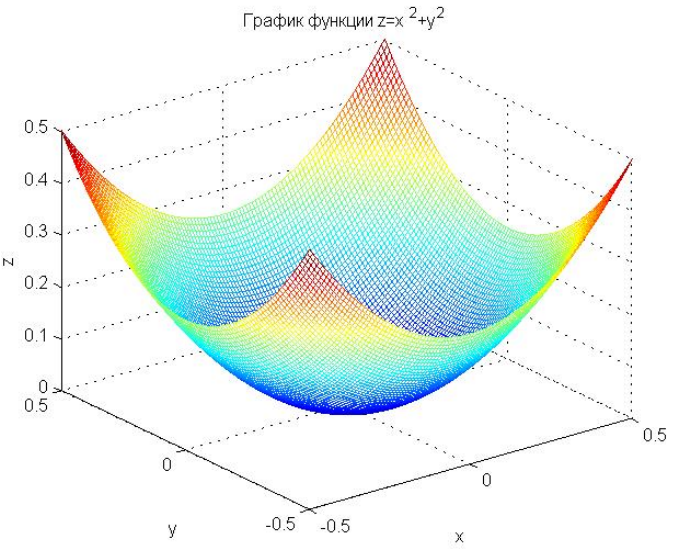


Рис. 2.21

График объемный с применением функции “mesh”

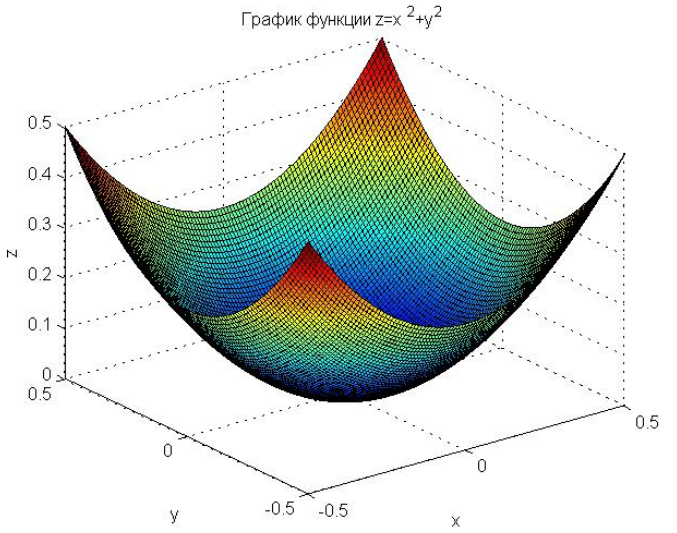


Рис. 2.22

График объемный с применением функции “surf”

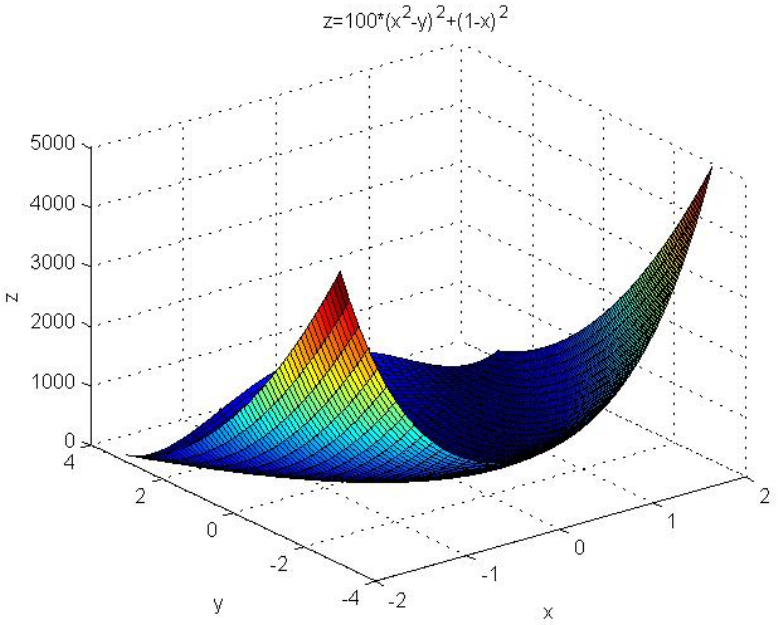


Рис. 2.23

График объемный с применением функции “surf”

При добавлении к названиям этих стандартных функций буквы “c”:

meshc (*x*, *y*, *z*)

или

surfc (*x*, *y*, *z*)

можно наблюдать на графиках проекции зависимой переменной (*z*) в виде линий равного уровня на плоскости независимых переменных *x* и *y* (рис. 2.24–2.26).

Краткая характеристика этих функций приведена в таблице 2.11.

Таблица 2.11

Стандартные функции MATLAB для построения трехмерных графиков

Функция	Описание
mesh (…)	график поверхности в виде сетки, цвет линий которой зависит от высоты
meshc (…)	график поверхности в виде сетки, цвет линий которой зависит от высоты, а также ее проекция в виде линий равного уровня
surf (…)	график поверхности в виде сетки, цвет ячеек которой зависит от высоты
surfc (…)	график поверхности в виде сетки, цвет ячеек которой зависит от высоты, а также ее проекция в виде линий равного уровня

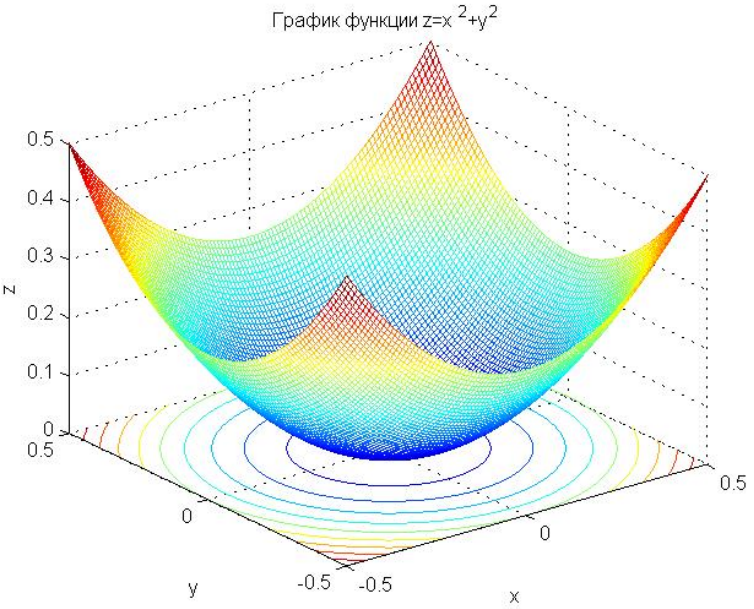


Рис. 2.24

График объемный с применением функции “meshc”

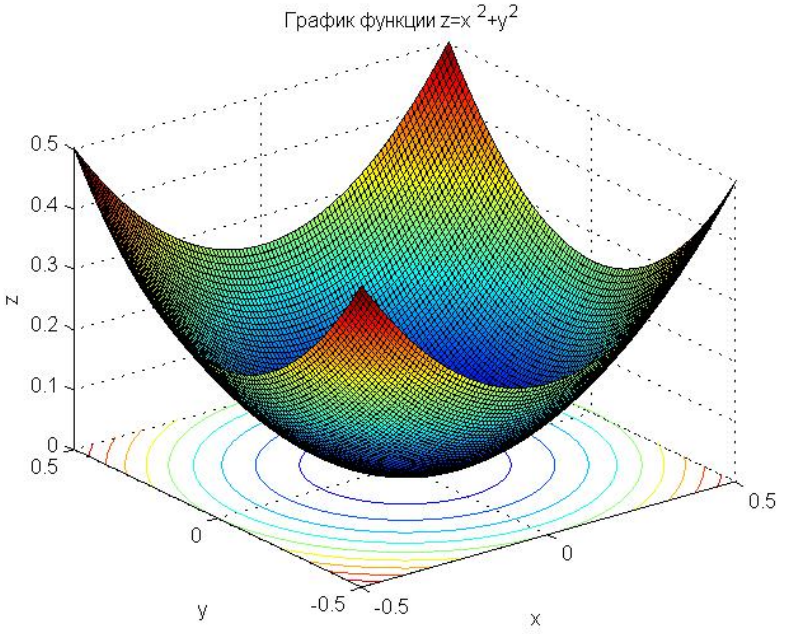


Рис. 2.25

График объемный с применением функции “surfc”

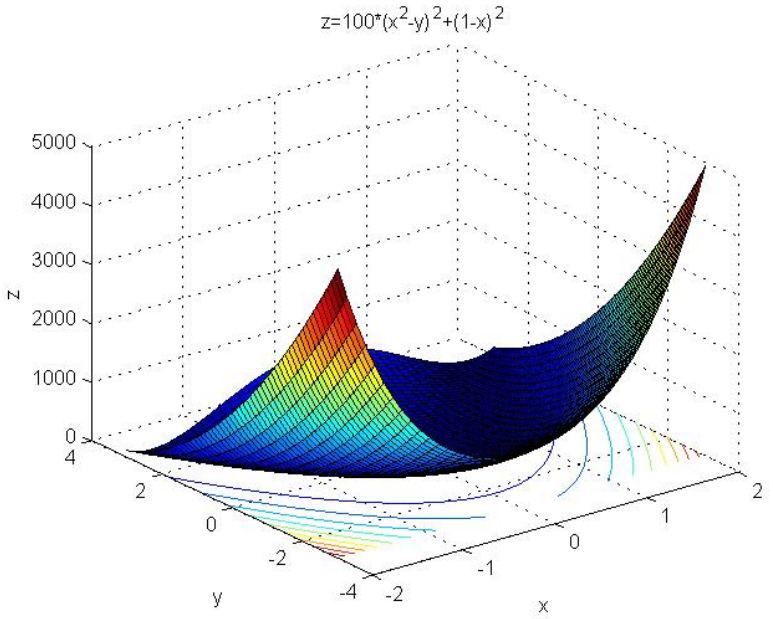


Рис. 2.26

График объемный с применением функции “surfc”

В заключение следует отметить, что вызов функций трехмерных графиков аналогичен вызову функций построения двумерных графиков (рис. 2.20). Но вместо двух массивов координат здесь следует указать три массива. Далее могут быть указаны параметры настройки графиков, аналогичные параметрам настроек, рассмотренных при построении двумерных графиков.

2.10. Разработка компьютерных программ в интегрированной среде MATLAB

Для реализации программ на *m*-языке MATLAB — строго определенной последовательности рассмотренных в предыдущих разделах операторов, существуют две возможности:

- запись программы в командном окне (Command Window) в виде последовательности операторов *m*-языка программирования и их выполнение;
- создание в текущей папке (Current Folder) интегрированной среды MATLAB определенного числа *m*-файлов, которые в соответствии с методологией структурного программирования связаны между собой и при запуске которых происходит выполнение *m*-файлов с включенными в них операторами.

Первый подход имеет ограниченное применение и может использоваться для решения простых задач.

Реализация второго подхода с использованием *m*-файлов позволяет разрабатывать достаточно сложные программные комплексы, которые могут быть конвертированы в другие алгоритмические языки высокого уровня, например

C/C++ и др., и, соответственно, интегрированы в сложные компьютерные программные и информационные системы.

2.10.1. Создание программ в Командном окне (Command Window)

Создание программы в Командном окне показано на примере решения квадратного уравнения. В командных строках Командного окна (см. рис. 2.1) для определения корней квадратного уравнения (рис. 2.27) в определенной последовательности через точку с запятой записываются операторы *m*-языка. При этом после нажатия клавиши <Enter> программа выполняется и значения всех переменных программы отражаются в рабочей памяти (Workspace) интегрированной среды MATLAB (рис. 2.1).

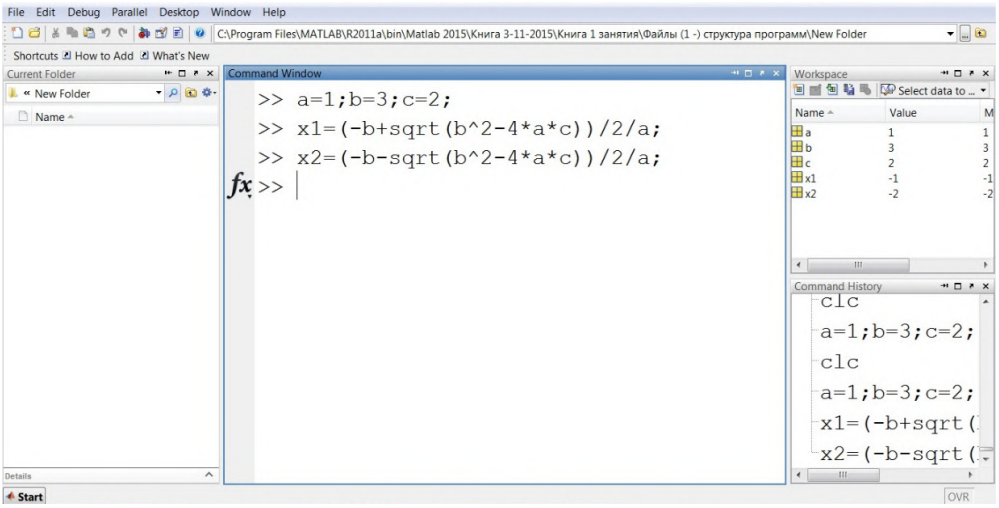


Рис. 2.27

Командное окно MATLAB с операторами *m*-языка и результатом решения квадратного уравнения ($x1$, $x2$) в Рабочей памяти (Workspace)

Для сохранения такой программы с возможностью ее дальнейшего использования необходимо зафиксировать все ее операторы (например, выделив комбинацией клавиш <Ctrl+A>, скопировать их комбинацией <Ctrl+C> и разместить, например, в файле Блокнота (Notebook) комбинацией клавиш <Ctrl+V>.

```
>> a=1;b=3;c=2;
>> x1=(-b+sqrt(b^2-4*a*c))/2/a;
>> x2=(-b-sqrt(b^2-4*a*c))/2/a;
```

Рис. 2.28

Копия программы из Командного окна (Command Window) в текстовом формате .txt

В результате получается файл с расширением .txt (рис. 2.28), содержимое которого всегда можно разместить в командную строку Командного окна

(Command Window) интегрированной среды MATLAB путем последовательной реализации для .txt файла комбинации клавиш <Ctrl+A>, <Ctrl+C>, <Ctrl+V> (рис. 2.29).

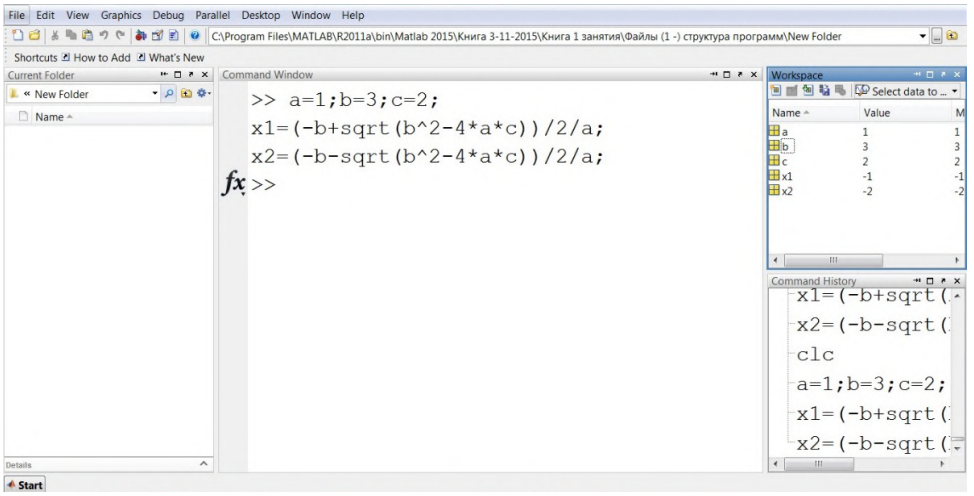


Рис. 2.29

Скопированное содержание *.txt файла в Командное окно и результат его выполнения — $x1$, $x2$ в Рабочей памяти (Workspace)

На примере решения квадратного уравнения с комплексными корнями (рис. 2.30) показано, что если в файле с расширением *.txt изменить значения его коэффициентов (a = 1; b = 3; c = 2) на a = 3; b = 2; c = 1, после перечисленных процедур выделения (<Ctrl+A>) и копирования (<Ctrl+C>) в текстовом файле и размещения содержимого (<Ctrl+V>) в командной строке Командного окна (Command Window) получается решение с комплексными корнями (рис. 2.30), которое будет отражаться в Рабочей памяти (Workspace) (рис. 2.1).

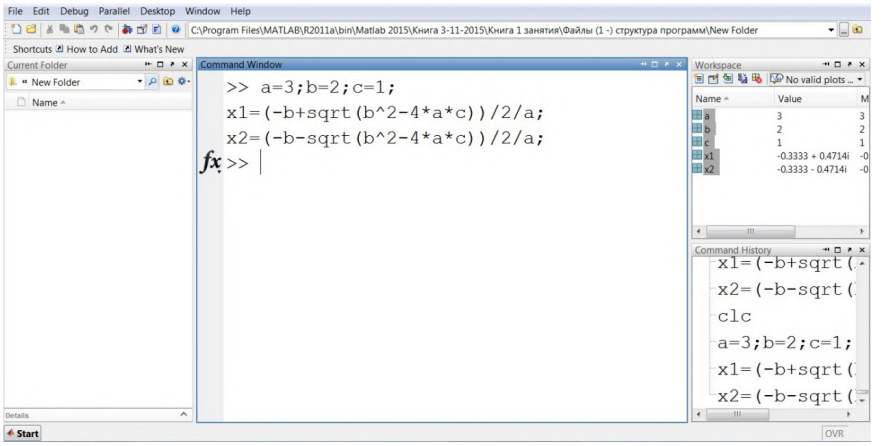


Рис. 2.30

Решение квадратного уравнения с комплексными корнями с программой в Командном окне (Command Window) и результатами решения в Рабочей памяти (Workspace)

Для сохранения переменных, размещенных в Рабочей памяти (Workspace), необходимо выполнить следующие команды: File → Save Workspace As, и получить файл с расширением *.mat. Вместо звездочки файлу с расширением .mat необходимо присвоить конкретное имя.

В любом случае при запуске MATLAB и при размещении файла с расширением .mat в Текущую папку (Current Folder) при нажатии клавишью мыши на имя этого файла в память загрузятся все переменные, сохраненные описанным выше способом.

Создание файла с расширением *.mat проиллюстрировано на примерах создания файла исходных данных для получения вещественных (рис. 2.31а–в) корней квадратного уравнения.

При определении вещественных корней для создания файла Коэф.вещ.уравн..mat сначала в командной строке Командного окна после курсора записываются три оператора, определяющие коэффициенты:

```
>> a = 1; b = 3; c = 2;
```

и после нажатия клавиши <Enter> в Рабочей памяти (Workspace) появляются их значения (рис. 2.31а).

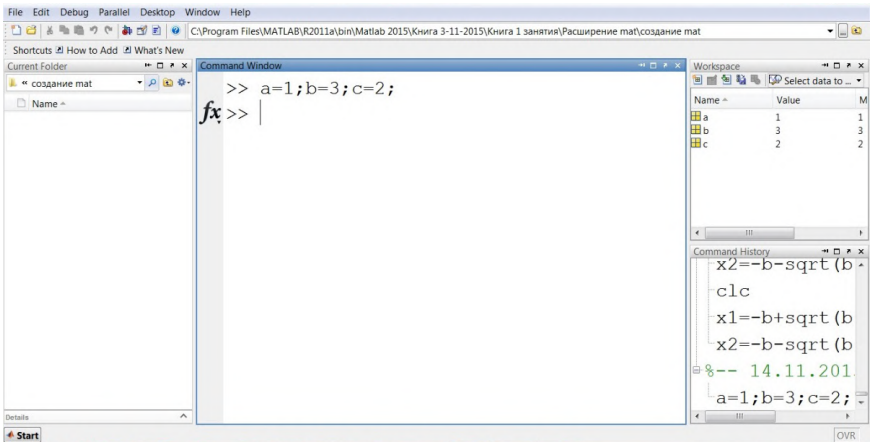


Рис. 2.31а

Создание файла исходных данных с расширением *.mat для квадратного уравнения с вещественными корнями

После выполнения команд File → Save Workspace As потребуется присвоить имя файлу с расширением *.mat — коэф.вещ.уравн..mat, после чего этот файл размещается в Текущей папке (Current Folder) с расширением .mat (рис. 2.31б).

В дальнейшем для выполнения программы из двух операторов:

```
>> x1=(-b+sqrt(b^2-4*a*c))/2/a;
>> x2=(-b-sqrt(b^2-4*a*c))/2/a;
```

с исходными данными для переменных a, b, c, заключенными в файле Коэф.вещ.уравн..mat, необходимо (рис. 2.31в):

- в Командное окно MATLAB поместить приведенные выше два оператора, определяющие корни квадратного уравнения;
- в Текущую папку поместить файл Коэф.вещ.уравн..mat;
- щелкнуть левой кнопкой мыши на название этого файла, в результате чего в Рабочей памяти (Workspace) появятся переменные *a*, *b* и *c* с их конкретными значениями;
- после второго оператора (x2=(-b-sqrt(b^2-4*a*c))/2/a) в командной строке Командного окна нажать на клавишу <Enter>.

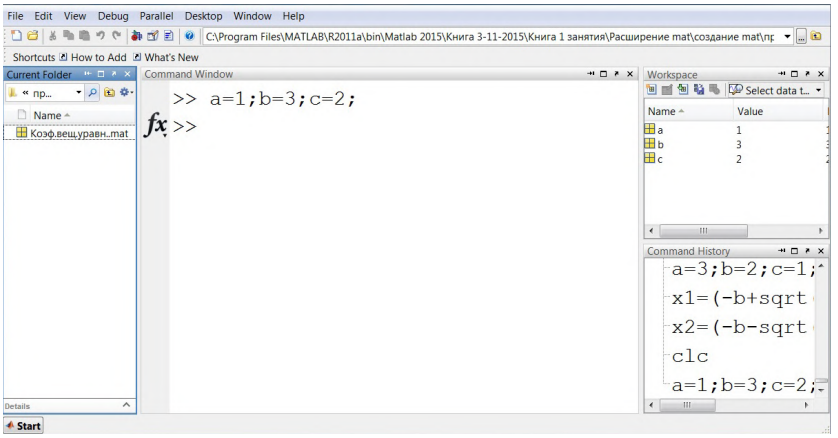


Рис. 2.31б

Получение файла коэффициентов уравнения с расширением *.mat

В результате получается:

- в Рабочей памяти (Workspace) появляются результирующие переменные (корни) квадратного уравнения;
- в Командном окне появляется x2 с конкретным значением корня, так как после второго оператора нет точки с запятой.

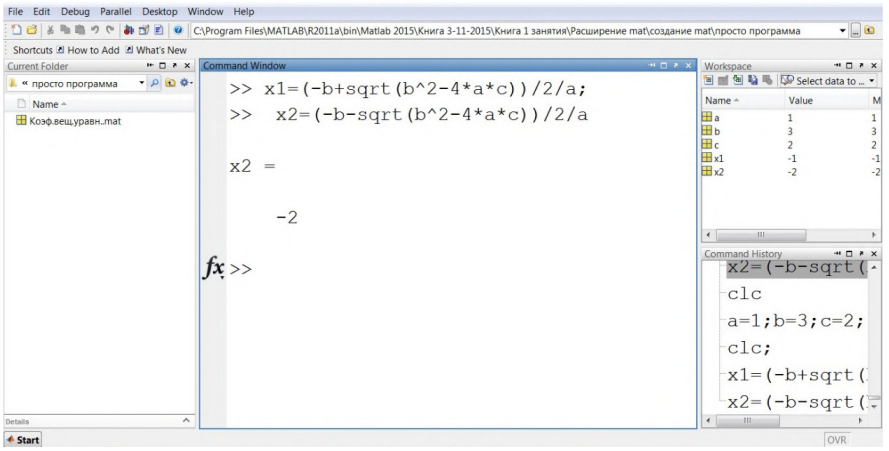


Рис. 2.31в

Применение файла коэффициентов уравнения с расширением .mat

Аналогичные действия с файлом исходной информации Коэф.вещ. уравн..mat проделаны и для решения квадратного уравнения с коэффициентами a = 3; b = 2; c = 1 (рис. 2.32а–в).

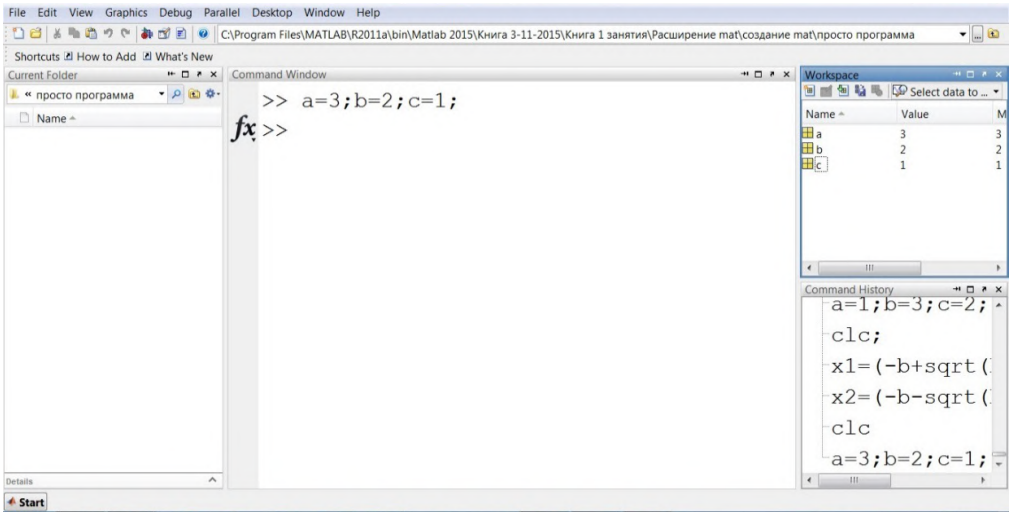


Рис. 2.32а

Создание файла исходных данных с расширением *.mat для квадратного уравнения с комплексными корнями

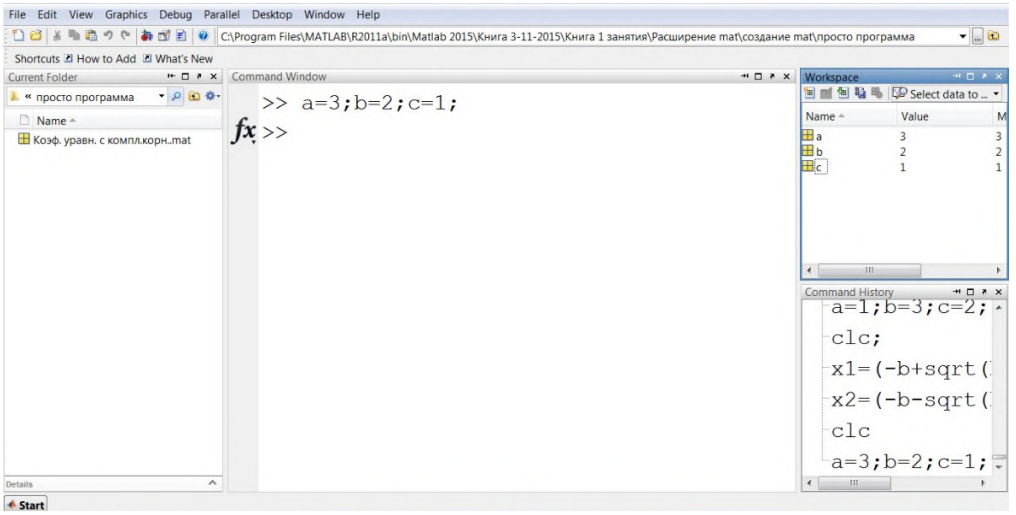


Рис. 2.32б

Получение файла коэффициентов уравнения с комплексными корнями с расширением *.mat

Следует отметить, что аналогичным образом файлы *.mat могут быть использованы и в программах с *m*-файлами.

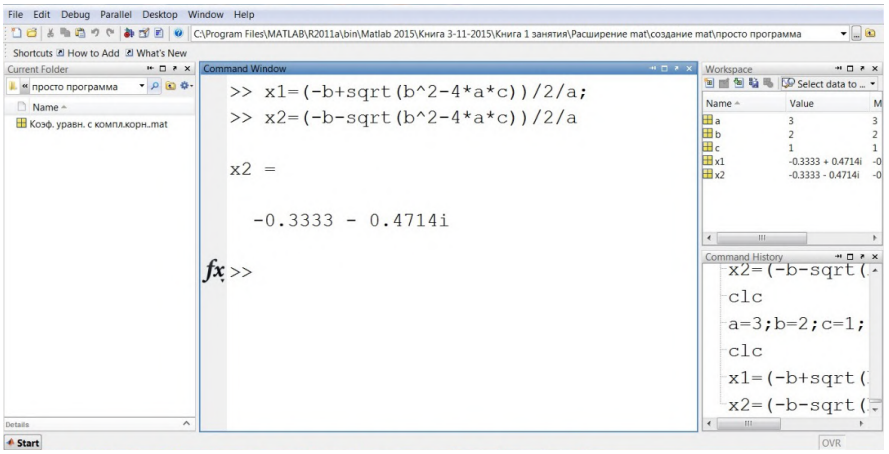


Рис. 2.32в

Применение файла коэффициентов уравнения с комплексными корнями с расширением *.mat

2.10.2. Создание программ с *m*-файлами

Для создания программ с *m*-файлами для каждой отдельной программы, обычно включающей в себя несколько *m*-файлов, необходимо открыть отдельную папку и все *m*-файлы, относящиеся к этой программе, помещать в эту папку.

Различают два типа *m*-файлов — файлов с расширением .m: скрипты и функции.

Запись программного кода *m*-файла осуществляется в Редакторе (Editor), для чего в среде MATLAB необходимо выбрать команды меню File → New → Script (если создается скрипт) или File → New → Function (если создается функция).

Скрипт — это программа, которая предназначена для интеграции (связывания) различных готовых программных модулей *m*-языка программирования — *m*-функций и других *m*-скриптов, создаваемых пользователем и служащих для реализации сценария решения поставленной задачи. Поэтому в виде скриптов оформляются *m*-файлы главных (основных) управляющих программ, которые также хранятся в списке файлов Текущей папки (Current Folder). Для запуска программы решения задачи из командной строки Командного окна (Command Window) целесообразно использовать **имя скрипта**.

Программный код *m*-скрипта представляет собой последовательность операторов на *m*-языке программирования, при сохранении которого по команде Save или Save As ее *m*-файлу обязательно надо присвоить имя, по которому можно обратиться для выполнения операторов *m*-скрипта.

Программный код функции имеет фиксированную структуру, описание которой начинается со слова **function** с именем в заголовке и заканчивается словом end при завершении описания функции. Имя в заголовке программного кода функции автоматически присваивается имени *m*-файла при его сохранении. Программный код функции хранится в виде самостоятельного *m*-файла в списке файлов в Текущей папке (Current Folder). Для работы функции и ее вы-

полнения к ней надо специальным образом обратиться по имени ее *m*-файла, либо из командной строки Командного окна (Command Window), либо из другой *m*-функции или *m*-скрипта.

Целесообразно рассмотреть следующие десять вариантов разработки программ с применением *m*-файлов — *m*-скриптов и *m*-функций.

1. Программы со скриптами и функциями без параметров.
2. Программа с внешней функцией с параметрами и функцией “global”.
3. Программа с внешней функцией с параметрами и функцией “varargin”.
4. Программа с внешней функцией с параметрами и функцией “feval”.
5. Программа с внутренней функцией.
6. Программа с функцией “inline”.
7. Программа стандартная со скриптом в качестве основной управляющей программы и *m*-файлами: DATA — для задания исходной информации, и REPORT — для отчета о результатах вычислений.
8. Программа со стандартной функцией MATLAB для решения вычислительной задачи определения корней уравнения многочлена произвольной степени “roots”.
9. Программа стандартная с отчетом в текстовом файле.
10. Программа с визуальным интерфейсом в среде GUIDE.

Для решения поставленных задач в Текущей папке (Current Folder) создаются соответствующие десять папок (рис. 2.33), в каждой из которых будут размещаться *m*-файлы, необходимые для выполнения расчетов.

Приведенные варианты разработки программ будут проиллюстрированы на примере определения вещественных и комплексных корней квадратного уравнения $ax^2 + bx + c = 0$.

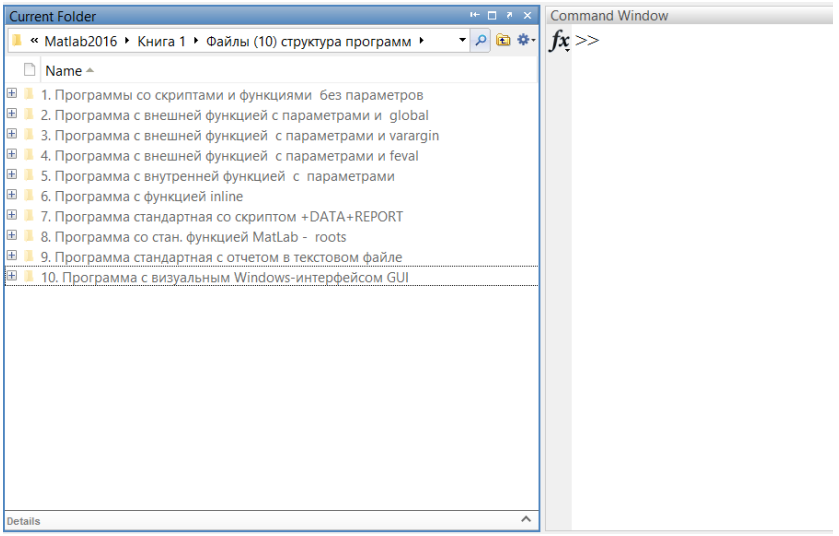


Рис. 2.33

Папки для создания различных вариантов программ с *v*-скриптами и *m*-функциями

2.10.3. 1. Программы со скриптами и функциями без параметров

Файлы папки со скриптами и функциями без параметров приведены на рисунке 2.34.

Для создания скрипта необходимо выбрать команды меню **File → New → Script**, в результате чего откроется окно редактора (**Editor**), куда следует записать последовательность операторов на *m*-языке для решения задачи.

Программный код *m*-скрипта представлен на рисунке 2.35, и при его сохранении с использованием команды **File → Save или Save As** ему присваивается имя kvurrs1_33_1. Этот файл хранится в **Текущей папке (Current Folder)** интегрированной среды MATLAB (рис. 2.1).

Все комментарии программы помечены в начале сроки знаком «%», в том числе и имя *m*-скрипта, с которым он сохранен в **Текущей папке (Current Folder)**.

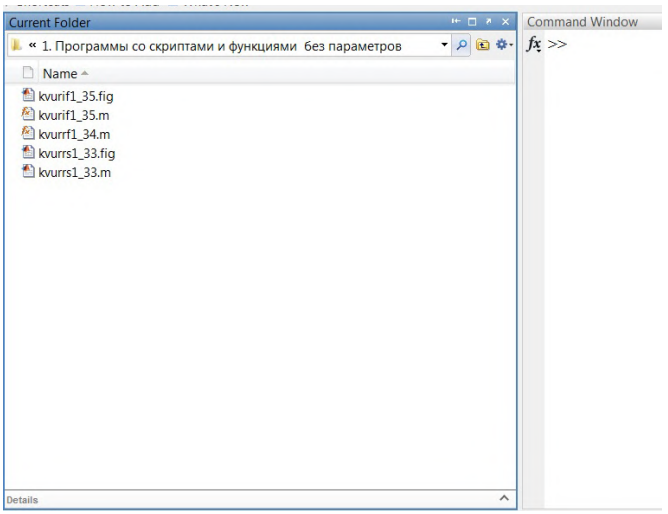


Рис. 2.34

Файлы папки, содержащие скрипты и функции без параметров

```
%Программный код файла kvurrs1_33.m — основная программа
%Программа включает следующие файлы: kvurrs1_33.m
%Программа определения корней квадратного уравнения
clc;
clear all;
close all;
disp('Программа определения корней квадратного уравнения')
disp('              a*x^2+b*x+c=0              ')
disp('Файлы программы: kvurrs1.m');
disp('ИСХОДНАЯ ИНФОРМАЦИЯ ДЛЯ РАСЧЕТОВ');
%Задание значений коэффициентов квадратного уравнения
%a*x^2+b*x+c=0
a=1
b=3
c=2
```

Рис. 2.35 (начало)

Программный код *m*-скрипта kvurrs1_33.m для определения вещественных корней квадратного уравнения

```
%ВЫПОЛНЕНИЕ ВЫЧИСЛЕНИЙ
det=b^2-4*a*c;
if det<0
  ss='Получены комплексные корни';
  else
    ss='Получены вещественные корни';
end
x1=(-b-sqrt(det))/2/a;
x2=(-b+sqrt(det))/2/a;
disp ('РЕЗУЛЬТАТЫ РАСЧЕТОВ')
disp(ss);disp('x1=');disp(x1);disp('x2=');disp(x2);
%ПОСТРОЕНИЕ ГРАФИКА ФУНКЦИИ УРАВНЕНИЯ f(x)=a*x^2+b*x+c
%Формирование  массива данных для оси абсцисс
x=[-5:0.1:0];
%Формирование  соотвествующего массива данных для оси ординат
y=(x.^2)*a+x.*b+c;
%Изображение графика
plot(x,y,'r.');
%Нанесение сетки на график
grid on;
%Заголовок графика
title('Графическое представление функции кв. уравнения a*x^2+b*x+c=0');
%Текст под осью абсцисс
xlabel('Аргумент функции — x');
%Текст рядом с осью ординат
ylabel('Функция  f(x)=a*x^2+b*x+c');
```

Рис. 2.35 (окончание)

Программный код *m*-скрипта kvurrs1_33.m для определения вещественных корней квадратного уравнения

Следует обратить внимание на то, что m-скрипт не заканчивается оператором "end".

Программа состоит из пяти частей.

1. Очистка Командного окна памяти и графических приложений от результатов работы предыдущих программ. Для этой цели используются соответственно операторы "clc", "clean all", "close all".

2. Ввод исходной информации — значений коэффициентов уравнения *a*, *b* и *c* с использованием операторов присвоения: a = 1; b = 3; c = 2.

3. Определение знака детерминанта квадратного уравнения — знака переменной "det" с выдачей информации в **Командное окно (Command Window)** с использованием символьной строковой переменной "ss" (рис. 2.35, 1).

4. Вычисление корней и вывод результатов расчетов в Командное окно с использованием переменных "x1" и "x2" (рис. 2.35, 1).

5. Построение графика функции квадратного уравнения $f(x) = ax^2 + bx + c$ с использованием оператора "plot (x, y, 'r.')" и фиксацией нулевых значений функции (корней квадратного уравнения) в точках –2 и –1 на оси абсцисс (рис. 2.35.2).

Для выполнения расчетов по рассматриваемой программе необходимо обратиться к ней по имени ее файла — kvurrs1_33.m из любого *m*-файла или, как в данном случае, из командной строки **Командного окна (Command Window)**.

Вся информация о выполняемой программе будет выводиться в **Командное окно (Command Window)**, так как в программе либо использован оператор "disp", либо после операторов нет точки с запятой (рис. 2.35, 1).

```
Программа определения корней квадратного уравнения
          a*x^2+b*x+c=0
Файлы программы: kvurrs1_33_1.m
ИСХОДНАЯ ИНФОРМАЦИЯ ДЛЯ РАСЧЕТОВ

a = 1
b = 3
c = 2

РЕЗУЛЬТАТЫ РАСЧЕТОВ
Получены вещественные корни

x1= -2
x2= -1
```

Рис. 2.35, 1

Результаты расчета вещественных корней квадратного уравнения

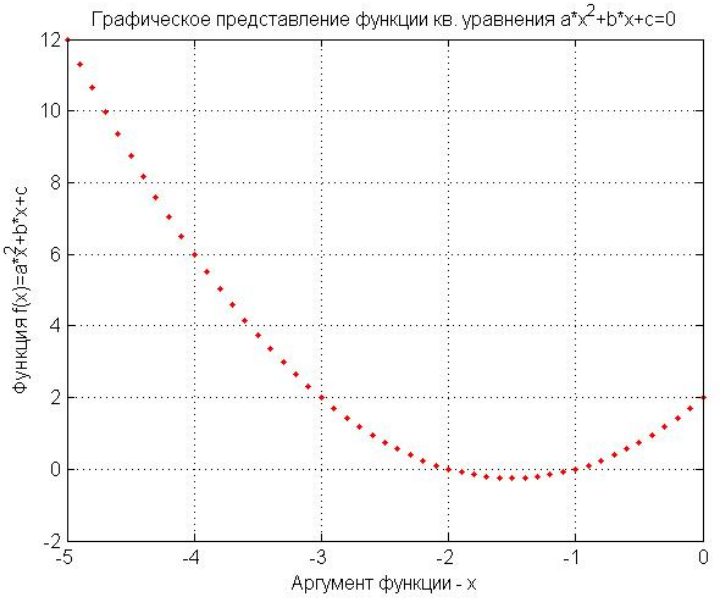


Рис. 2.35, 2

Графическое изображение функции квадратного уравнения с вещественными корнями

Для сохранения графика (рис. 2.35, 2) с расширением *.fig в **Текущей папке (Current Folder)** с именем kvurrs1_33.fig (рис. 2.34) необходимо воспользоваться командой **File → Save или Save As**.

Программу определения вещественных корней квадратного уравнения $ax^2 + bx + c = 0$ можно создать в виде *m*-функции (Function) без параметров с именем “kvurrf1_34”

Для создания функции необходимо выбрать команды меню **File → New → Function**, в результате чего откроется окно **Редактора (Editor)** со следующим текстом:

```
function [output args] = Untitled (input args)
% Untitled Summary of this function goes here
% Detailed explanation goes here
end
```

```
function kvurrf1_34
%Программный код файла kvurrf1_34.m — основная программа
%Программа включает следующие файлы: kvurrf1_34.m
%Программа определения корней квадратного уравнения
clc;
clear all;
close all;
disp('Программа определения корней квадратного уравнения')
disp('              a*x^2+b*x+c=0              ')
disp('Файлы программы: kvurrf1.m');
disp('ИСХОДНАЯ ИНФОРМАЦИЯ ДЛЯ РАСЧЕТОВ');
%Задание значений коэффициентов квадратного уравнения
%a*x^2+b*x+c=0
a=1
b=3
c=2
%ВЫПОЛНЕНИЕ ВЫЧИСЛЕНИЙ
det=b^2-4*a*c;
if det<0
 ss='Получены комплексные корни';
 else
   ss='Получены вещественные корни';
end
x1=(-b-sqrt(det))/2/a;
x2=(-b+sqrt(det))/2/a;
disp ('РЕЗУЛЬТАТЫ РАСЧЕТОВ')
disp(ss);disp('x1=');disp(x1);disp('x2=');disp(x2);
%ПОСТРОЕНИЕ ГРАФИКА ФУНКЦИИ УРАВНЕНИЯ f(x)=a*x^2+b*x+c
%Формирование массива данных для оси абсцисс
x=[-5:0.1:0];
%Формирование соответствующего массива данных для оси ординат
y=(x.^2)*a+x.*b+c;
%Изображение графика
plot(x,y,'r.');
%Нанесение сетки на график
grid on;
%Заголовок графика
title('Графическое представление функции кв. уравнения a*x^2+b*x+c=0');
%Текст под осью абсцисс
xlabel('Аргумент функции — x');
%Текст рядом с осью ординат
ylabel('Функция f(x)=a*x^2+b*x+c');
   end
```

Рис. 2.36

Программный код *m*-функции без параметров kvurrf1_34.m для определения вещественных корней квадратного уравнения

При создании функции без параметров необходимо удалить квадратные и круглые скобки в заголовке (первой строке) функции и вместо “Untitled” пользователь должен дать имя функции, например kvurrf1_34.m (рис. 2.36).

После двух строк комментариев, отмеченных слева знаками «%», следует записать все операторы программы до слова “end”. Комментарии можно изменить либо удалить, но фиксированная структура функции со словами “function имя” в первой строке и “end” в последней строке должна остаться.

При сохранении функции в **Текущей папке (Current Folder)** ее *m*-файл автоматически сохраняется с именем по умолчанию, которое было присвоено вместо “Untitled” в программном коде (рис. 2.34 и 2.36).

Следует отметить, что m-файлу функции может быть присвоено любое другое имя, не соответствующее имени по умолчанию.

При оформлении рассматриваемой программы в виде *m*-функции без параметров необходимо все операторы *m*-скрипта (рис. 2.35) скопировать и поместить в тело *m*-функции kvurrf1_34.m между первой и последней строками (рис. 2.36).

Для выполнения *m*-функции без параметров kvurrf1_34.m необходимо обратиться к ней по имени kvurrf1_34.m либо из любого *m*-файла, либо, как в данном случае, из командной строки **Командного окна (Command Window)**.

Результаты расчетов с использованием этой *m*-функции (рис. 2.36) полностью совпадают с результатами вычислений с применением *m*-скрипта — рисунки 2.35, 1 и 2.35, 2.

Программа определения комплексных корней квадратного уравнения $ax^2 + bx + c = 0$ в виде m-функции (function) без параметров с именем kvurif1_35

Программный код создаваемой *m*-функции без параметров с именем kvurif1_35.m представлен на рисунке 2.37.

```
function kvurif1_35
%Программный код файла kvurif1_35.m — основная программа
%Программа включает следующие файлы: kvurif1_35.m
%Программа определения корней квадратного уравнения
clc;
clear all;
close all;
disp('Программа определения корней квадратного уравнения')
disp('              a*x^2+b*x+c=0              ')
disp('Файлы программы: kvurif1.m');
disp('ИСХОДНАЯ ИНФОРМАЦИЯ ДЛЯ РАСЧЕТОВ');
%Задание значений коэффициентов квадратного уравнения
%a*x^2+b*x+c=0
a=1
b=2
c=3
%ВЫПОЛНЕНИЕ ВЫЧИСЛЕНИЙ
det=b^2-4*a*c;
if det<0
  ss='Получены комплексные корни';
  else
    ss='Получены вещественные корни';
end
x1=(-b-sqrt(det))/2/a;
x2=(-b+sqrt(det))/2/a;
disp ('РЕЗУЛЬТАТЫ РАСЧЕТОВ')
```

Рис. 2.37 (начало)

Программный код *m*-функции без параметров для определения комплексных корней квадратного уравнения

```
disp(ss);disp('x1=');disp(x1);disp('x2=');disp(x2);
%ПОСТРОЕНИЕ ГРАФИКА ФУНКЦИИ УРАВНЕНИЯ f(x)=a*x^2+b*x+c
%Формирование массива данных для оси абсцисс
x=[-5:0.1:0];
%Формирование соответствующего массива данных для оси ординат
y=(x.^2)*a+x.*b+c;
%Изображение графика
plot(x,y,'r.');
%Нанесение сетки на график
grid on;
%Заголовок графика
title('Графическое представление функции кв. уравнения a*x^2+b*x+c=0');
%Текст под осью абсцисс
xlabel('Аргумент функции — x');
%Текст рядом с осью ординат
ylabel('Функция f(x)=a*x^2+b*x+c');
%Задание диапазона изменения параметров осей абсцисс и ординат
axis([-5 0 0 20]);
   end
```

Рис. 2.37 (окончание)

Программный код *m*-функции без параметров для определения комплексных корней квадратного уравнения

В отличие от программного кода *m*-функции, определяющей вещественные корни уравнения (рис. 2.36), в этом случае задаются другие коэффициенты: a = 1; b = 2; c = 3. В связи с тем, что при таких коэффициентах знак детерминанта квадратного уравнения меньше нуля, в Командное окно при выполнении m-файла kvurif1_35.m выводится символьная (строковая) переменная — полученные комплексные корни и корни x1 и x2 представляют собой комплексные числа (рис. 2.37, 1).

```
Программа определения корней квадратного уравнения
            a*x^2+b*x+c=0
Файлы программы: kvurif1_35_1.m
ИСХОДНАЯ ИНФОРМАЦИЯ ДЛЯ РАСЧЕТОВ

a = 1

b = 2

c = 3

РЕЗУЛЬТАТЫ РАСЧЕТОВ
Получены комплексные корни

x1= -1.0000 – 1.4142i

x2= -1.0000 + 1.4142i
```

Рис. 2.37, 1

Результаты решения квадратного уравнения с комплексными корнями

Операторы построения двухмерного (плоского) графика необходимо корректировать путем изменения диапазона границ осей абсцисс и ординат в предпоследней строке *m*-функции (рис. 2.37):

axis ([–5 0 0 20]).

В результате при изображении графика функции $f(x) = x^2 + 2x + 3$ отчетливо видно (рис. 2.37, 2), что она не пересекает ось абсцисс с нулевым значением функции f(x), и у квадратного уравнения с коэффициентами a = 1; b = 2; c = 3; нет вещественных корней.

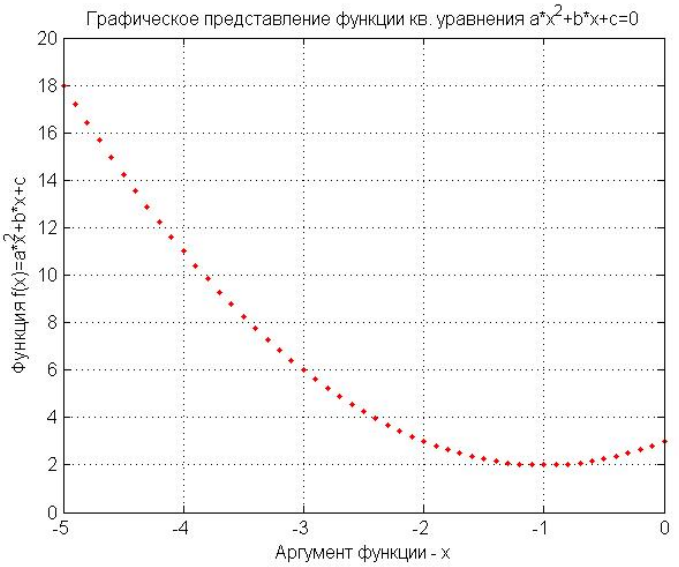


Рис. 2.37, 2

Графическое представление функции квадратного уравнения с комплексными корнями

Для сохранения графика (рис. 2.37, 2) с расширением *.fig в **Текущей папке (Current Folder)** с именем kvurif1_35.fig (рис. 2.34) необходимо воспользоваться командой **File → Save или Save As**.

2.10.4. 2. Программа с внешней функцией с параметрами и функцией “global”

Программа решения квадратного уравнения может включать в себя два *m*-файла: Glavkvurrf1vne_36.m и gfdet_36_1.m. Поэтому Текущая папка для решения задачи должна содержать два *m*-файла (рис. 2.38).

Программный модуль Glavkvurrf1vne_36.m представляет собой основную управляющую программу и является *m*-функцией, которая для расчета детерминанта квадратного уравнения ($det = b^2 – 4ac$) обращается к выполнению программного модуля gfdet_36_1.m, который является **внешней функцией с тремя входными параметрами — *a*, *b*, *c***:

det = gfdet_36_1(a, b, c).

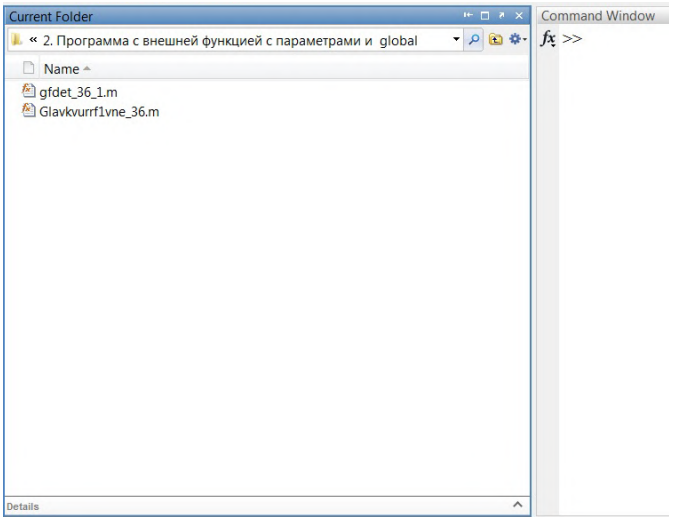


Рис. 2.38

Файлы папки с внешней функцией gfdet_36_1.m

Для вычисления конкретного значения функции gfdet_36_1(a, b, c) и присвоения результата выходной переменной det каждый из входных параметров (аргументов) функции должен получить фактическое числовое значение.

С этой целью в основной программе Glavkvurrf1vne_36.m коэффициенты квадратного уравнения получают следующие значения: a = 1, b = 3, c = 2 (рис. 2.39).

```
function Galvkvurrf1vne_36
%Программный код файла Glavkvurrf1vne_36.m  — основная программа
%Программа включает следующие файлы: Glavkvurrf1vne_36.m+gfdet_36_1.m
%Программа определения корней квадратного уравнения a*x^2+b*x+c=0
clc;
clear all;
close all;
global ss;
disp('Программа определения корней квадратного уравнения')
disp('              a*x^2+b*x+c=0              ')
disp('Файлы программы:Glavkvurrf1vne_36.m+gfdet_36_1.m  ');
disp('ИСХОДНАЯ  ИНФОРМАЦИЯ ДЛЯ РАСЧЕТОВ');
%Задание значений коэффициентов квадратного уравнения
%a*x^2+b*x+c=0
a=1
b=3
c=2
%ВЫПОЛНЕНИЕ  ВЫЧИСЛЕНИЙ
[det]=gfdet_36_1(a,b,c);
x1=(-b-sqrt(det))/2/a;
x2=(-b+sqrt(det))/2/a;
disp ('РЕЗУЛЬТАТЫ  РАСЧЕТОВ')
```

Рис. 2.39 (начало)

Программный код основной программы при решении квадратного уравнения с использованием внешней функции для расчета детерминанта

```
disp(ss);disp('x1=');disp(x1);disp('x2=');disp(x2);
%ПОСТРОЕНИЕ ГРАФИКА ФУНКЦИИ УРАВНЕНИЯ f(x)=a*x^2+b*x+c
%Формирование массива данных для оси абсцисс
x=[-5:0.1:0];
%Формирование соответствующего массива данных для оси ординат
y=(x.^2)*a+x.*b+c;
%Изображение графика
plot(x,y,'r.');
%Нанесение сетки на график
grid on;
%Заголовок графика
title('Графическое представление функции кв. уравнения a*x^2+b*x+c=0');
%Текст под осью абсцисс
xlabel('Аргумент функции — x');
%Текст рядом с осью ординат
ylabel('Функция f(x)=a*x^2+b*x+c');
  end
```

Рис. 2.39 (окончание)

Программный код основной программы при решении квадратного уравнения с использованием внешней функции для расчета детерминанта

```
function det = gfdet_36_1(a,b,c)
%Программный код файла gfdet_36_1.m — определение детерминанта квадратного
%уравнения
%Программа включает следующие файлы: Glavkvurrf1vne_36.m+gfdet_36_1.m
%Программа определения корней квадратного уравнения a*x^2+b*x+c=0
global ss;
det=b^2-4*a*c;
if det<0
 ss='Получены комплексные корни';
 else
  ss='Получены вещественные корни';
end
end
```

Рис. 2.40

Программный код внешней функции расчета детерминанта при решении квадратного уравнения

С этими значениями *a*, *b*, *c* происходит расчет детерминанта в модуле gfdet_36_1.m (рис. 2.40) при обращении (вызове) из основной программы *m*-файла gfdet_36_1.m с использованием оператора:

[det] = gfdet (a, b, c).

Переменная det в левой части оператора в соответствии с основополагающей концепцией *m*-языка программирования считается массивом с размером (1 * 1), и поэтому может быть представлена в квадратных скобках.

Как правило, когда результатом расчета функции является одно число (скалярная величина), квадратные скобки не используются.

Использование внешних функций с параметрами является эффективным способом обмена параметрами между различными модулями (*m*-скриптами и *m*-функциями) *m*-языка программирования.

Часто более целесообразно осуществлять обмен значениями параметров между программными модулями с применением функции “global”, так как перечисленные после нее через пробелы переменные и располагаемые в разных программных модулях (при условии, что все они расположены в одной **Текущей папке**) являются общими для соответствующих модулей и могут быть использованы ими для выполнения конкретных задач.

В рассматриваемом случае такой общей (global) переменной является символьная переменная ss, которая, в зависимости от знака детерминанта (рис. 2.40), может принимать различные значения строковых констант:

ss = ‘Полученные комплексные корни'

или

ss = ‘Полученные вещественные корни’.

Как следует из рисунков 2.39 и 2.40, конструкция global ss представлена в обоих программных кодах *m*-файлов — Glavkvurrf1vne_36.m и gfdet_36_1.m.

При создании программ с несколькими программными модулями в комментариях (знак «%» перед началом строки) следует в каждом из них перечислить все используемые *m*-файлы и обязательно указать назначение каждого *m*-файла. В рассматриваемом примере последнее обстоятельство указывается первым, а все используемые *m*-файлы следуют после этого.

При использовании функций, как уже отмечалось, важно не только помнить, что они имеют фиксированную структуру, начинающуюся со слова ‘function’ и заканчивающуюся словом ‘end’.

В общем случае при создании функций вида

```
function [resultat] = kvadratura (a, b, c, …, d)
end
```

необходимо обратить внимание на следующие обстоятельства:

1) тип и количество параметров *a*, *b*, *c*, … функции “kvadratura”. Тип и количество присваиваемых им при обращении к выполнению аргументов должны строго соответствовать друг другу: [resultat] = kvadratura (a, b, c, …).

2) результатом выполнения функции должно быть присвоение переменным в левой части оператора присваивания в квадратных скобках определенных значений — скалярных чисел, числовых массивов, символьных значений в виде строковых констант или строковых массивов.

3) одни и те же переменные для использования в разных программных модулях не могут быть в круглых скобках заголовка функции и конструкции “global”.

4) в функциях без параметров опускаются квадратные скобки и круглые скобки (рис. 2.36 и 2.37).

5) обмен переменными между *m*-функциями и *m*-скриптами осуществляется исключительно с использованием конструкции “global”.

В заключение следует отметить, что использование функций в строгом соответствии с принципами структурного программирования позволяет разра-

батывать наиболее эффективные компьютерные программы и дает возможность интегрировать их в другие компьютерно-информационные системы более высокого уровня.

2.10.5. 3. Программа с внешней функцией с параметрами и функцией “varargin”

В некоторых случаях могут создаваться функции с различным числом и типом параметров, и при обращении к этим функциям целесообразно проанализировать число параметров в круглых скобках, их тип и организовать работу с ними в зависимости от проведенного анализа.

Для этого при создании функции используется единственный параметр в круглых скобках после имени функции — “varargin”. С помощью функции “length(varargin)” можно вычислить количество поступивших в функцию входных параметров (предполагается, что их общее число равно N), а с помощью конструкции “varargin {i}” — обратиться к *i*-му из них.

С применением, например, условия с функцией “isnumeric”:

```
if isnumeric(varargin(i)) = = 1
```

можно определить, является ли *i*-й (i = 1,…,N) входной параметр числом.

Пример применения конструкции “varargin” представлен в соответствующей папке (рис. 2.33), содержащей два *m*-файла (рис. 2.41): Glavkvurrf1vne_36.m и gfdet_36_1.m.

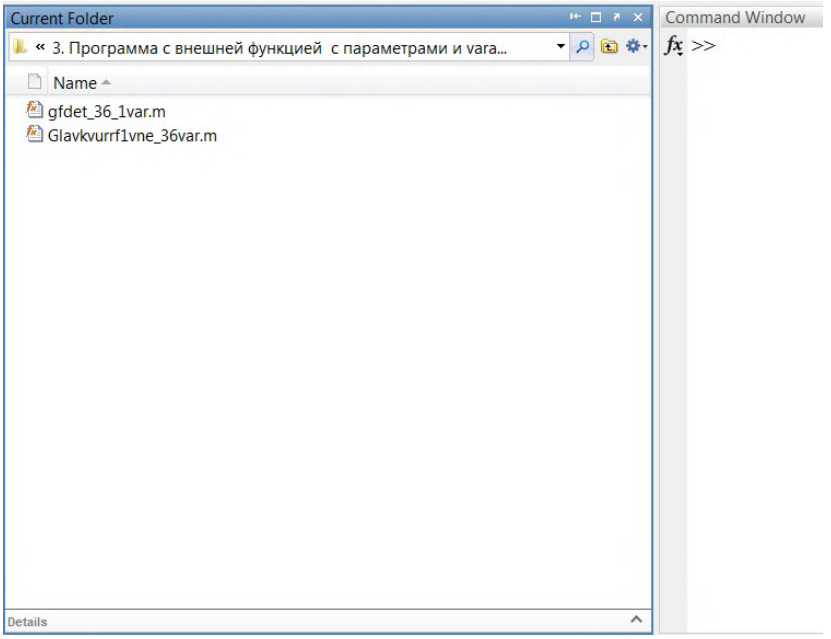


Рис. 2.41

Файлы папки с внешней функцией gfdet_36_1.m и конструкцией “varargin”

Из основной программы Glavkvurrf1vne_36.m (рис. 2.42) происходит обращение к функции вычисления детерминанта квадратного уравнения gfdet_36_1.m, включающей в себя четыре входных параметра, первые три из которых числовые, а четвертый, d — строковая константа:

d = ‘Определение детерминанта квадратного уравнения в функции gfdet_36_1.m’

Программный код функции gfdet_36_1 содержит единственный входной параметр — varargin (рис. 2.43).

```
function Glavkvurrf1vne_36var
%Программный код файла Glavkvurrf1vne_36.m — основная программа
%Программа включает следующие файлы: Glavkvurrf1vne_36.m+gfdet_36_1.m
%Программа определения корней квадратного уравнения a*x^2+b*x+c=0
clc;
clear all;
close all;
global ss;
disp('Программа определения корней квадратного уравнения')
disp('              a*x^2+b*x+c=0            ')
disp('Файлы программы:Glavkvurrf1vne_36var.m+gfdet_36_1var.m ');
disp('ИСХОДНАЯ ИНФОРМАЦИЯ ДЛЯ РАСЧЕТОВ');
%Задание значений коэффициентов квадратного уравнения
%a*x^2+b*x+c=0
a=1
b=3
c=2
d='Определение детерминанта квадратного уравнения в функции gfdet_36_1var';
%ВЫПОЛНЕНИЕ ВЫЧИСЛЕНИИ
[det]=gfdet_36_1var(a,b,c,d);
x1=(-b-sqrt(det))/2/a;
x2=(-b+sqrt(det))/2/a;
disp ('РЕЗУЛЬТАТЫ РАСЧЕТОВ')
disp(ss);disp('x1=');disp(x1);disp('x2=');disp(x2);
%ПОСТРОЕНИЕ ГРАФИКА ФУНКЦИИ УРАВНЕНИЯ f(x)=a*x^2+b*x+c
%Формирование  массива данных для оси абсцисс
x=[-5:0.1:0];
%Формирование  соответствующего  массива данных для оси ординат
y=(x.^2)*a+x.*b+c;
%Изображение графика
plot(x,y,'r.');
%Нанесение сетки на график
grid on;
%Заголовок графика
title('Графическое представление функции кв. уравнения a*x^2+b*x+c=0');
%Текст под осью абсцисс
xlabel('Аргумент функции — x');
%Текст рядом с осью ординат
ylabel('Функция  f(x)=a*x^2+b*x+c');
  end
```

Рис. 2.42

Программный код файла Glavkvurrf1vne_36.m с внешней функцией gfdet_36_1.m при использовании конструкции “varargin”

```
function det = gfdet_36_1(varargin)
%Программный код файла gfdet_36_1.m — определение детерминанта квадратно-
го
%уравнения
%Программа включает следующие файлы: Glavkvurrf1vne_36.m+gfdet_36_1.m
%Программа определения корней квадратного уравнения a*x^2+b*x+c=0
global ss;
a=varargin{1};
b=varargin{2};
c=varargin{3};
d=varargin{4};
if isnumeric(d)~=1
disp(d);
end
det=b^2-4*a*c;
if det<0
  ss='Получены комплексные корни';
  else
    ss='Получены вещественные корни';
end
end
```

Рис. 2.43

Программный код файла gfdet_36_1.m для расчета детерминанта квадратного уравнения с конструкцией varargin

```
Программа определения корней квадратного уравнения
            a*x^2+b*x+c=0
Файлы программы:Glavkvurrf1vne_36var.m+gfdet_36_1var.m
ИСХОДНАЯ ИНФОРМАЦИЯ ДЛЯ РАСЧЕТОВ

a = 1

b = 3

c = 2

Определение детерминанта квадратного уравнения в функции gfdet_36_1var
РЕЗУЛЬТАТЫ РАСЧЕТОВ
Получены вещественные корни
x1= -2

x2= -1
>>
```

Рис. 2.44

Результат расчета корней квадратного уравнения с внешней функцией и использованием конструкции “varargin”

При анализе содержимого конструкции “varargin” (рис. 2.43) первым трем параметрам присваиваются числовые значения коэффициентов квадратного уравнения, а переменной *d* — строковая константа, приведенная в основной программе (рис. 2.42), которая выводится в Командное окно с использованием оператора disp(d) при условии, что isnumeric(d) не равно 1 (рис. 2.43 и 2.44).

Следует обратить внимание на то, что в случае применения конструкции “varargin” при обращении к элементам его массива в соответствующей функции используются не круглые скобки, а фигурные (рис. 2.43).

2.10.6. 4. Программа с внешней функцией с параметрами и функцией “feval”

Функция “feval” применяется, когда необходимо передавать имя функции из одного программного модуля в другой и рассматривать как входной параметр функции *m*-языка наряду с другими входными параметрами функции. При этом имя функции является строковой константой (берется в кавычки) и может присваиваться символьной переменной.

В списке входных переменных функции “feval” при обращении к выполнению некоторой функции ее имя всегда стоит впереди числовых переменных. Например, при обычном способе вычисления математической функции cos(x), при x = 0.4468 в командной строке Командного окна MATLAB следует записать два оператора:

>> x = 0.4488; z = cos(x).

Тогда в Командном окне (из-за отсутствия точки с запятой после второго оператора) появится результат в следующем виде:

z =

0.9010

С применением оператора “feval” этот же результат можно получить, если в списке входных параметров сначала указать имя вызываемой функции в кавычках — в этом случае ‘cos’:

>> z = feval(‘cos’, x).

Для иллюстрации применения функции “feval” необходимо создать отдельную папку, например с именем «Программа с внешней функцией с параметрами и “feval”» (рис. 2.33) и разместить в ней четыре *m*-оператора (рис. 2.45).

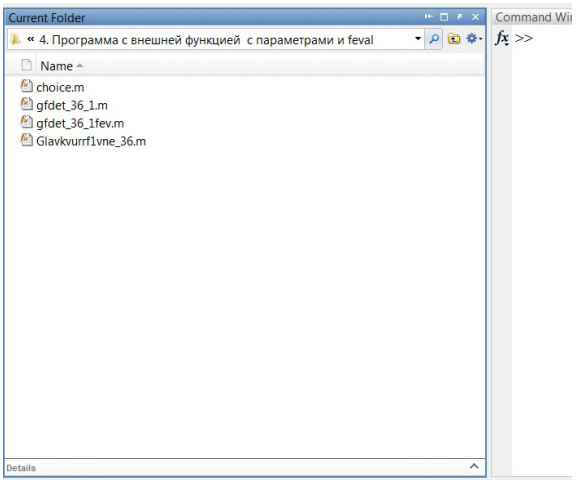


Рис. 2.45

Файлы папки, в которой размещены *m*-файлы, иллюстрирующие применение функции “feval”

В представленной программе с основной управляющей программой Glavkvurrf1vne_36.m (рис. 2.46) показано, как с использованием оператора “feval” в функции choice.m (рис. 2.47) можно обратиться к выполнению двух различных функций gfdet_36_1.m (рис. 2.48) или gfdet_36_1fev.m (рис. 2.49) с различными вариантами определения детерминанта квадратного уравнения.

Во второй функции для возведения в квадрат используется стандартная функция MATLAB pow2() вместо знака возведения в квадрат — “^”, как в первой функции.

Для решения этой задачи в основной программе (рис. 2.46) задается признак, и ему присваивается определенное значение — priznak = 0. Двум символьным переменным fname1 и fname2 присваиваются соответственно имена функций — 'gfdet_36_1’ и ‘gfdet_36_1fev’ с различными вариантами вычисления детерминанта квадратного уравнения.

```
function Glavkvurrf1vne_36
%Программный код файла Glavkvurrf1vne_36.m — основная программа
%Программа включает в себя следующие файлы:
%Glavkvurrf1vne_36.m+gfdet_36_1.m+gfdet_36_1fev.m+choice.m
%Программа определения корней квадратного уравнения a*x^2+b*x+c=0
clc;
clear all;
close all;
global ss priznak;
disp('Программа определения корней квадратного уравнения')
disp('              a*x^2+b*x+c=0            ')
disp('Файлы программы:Glavkvurrf1vne_36.m+gfdet_36_1.m ');
disp('ИСХОДНАЯ ИНФОРМАЦИЯ ДЛЯ РАСЧЕТОВ');
%Задание значений коэффициентов квадратного уравнения
%a*x^2+b*x+c=0
a=1
b=3
c=2
priznak=0
fname1='gfdet_36_1';
fname2= 'gfdet_36_1fev';
%ВЫПОЛНЕНИЕ ВЫЧИСЛЕНИЙ
det=choice(fname1,fname2,a,b,c);
x1=(-b-sqrt(det))/2/a;
x2=(-b+sqrt(det))/2/a;

disp ('РЕЗУЛЬТАТЫ РАСЧЕТОВ')
disp(ss);disp('x1=');disp(x1);disp('x2=');disp(x2);
%ПОСТРОЕНИЕ ГРАФИКА ФУНКЦИИ УРАВНЕНИЯ f(x)=a*x^2+b*x+c
%Формирование  массива данных для оси абсцисс
x=[-5:0.1:0];
%Формирование  соответствующего массива данных для оси ординат
y=(x.^2)*a+x.*b+c;
%Изображение графика
plot(x,y,'r.');
%Нанесение сетки на график
grid on;
%Заголовок графика
title('Графическое представление функции кв. уравнения a*x^2+b*x+c=0');
%Текст под осью абсцисс
xlabel('Аргумент функции — x');
%Текст рядом с осью ординат
ylabel('Функция  f(x)=a*x^2+b*x+c');
   end
```

Рис. 2.46

Программный код файла основной программы Glavkvurrf1vne_36.m для иллюстрации применения функции “feval”

```
function det= choice(fname1,fname2,a,b,c);
%Программный код файла choice.m — программа выбора функции с при-
% менением “feval”;
%Программа включает в себя следующие файлы:
%Glavkvurrf1vne_36.m+gfdet_36_1.m+gfdet_36_1fev.m+choice.m
%Программа определения корней квадратного уравнения a*x^2+b*x+c=0
global priznak;
if priznak==0
   fname2
   det=feval (fname2,a,b,c);
else
   fname1
  det=feval (fname1,a,b,c);
end
end
```

Рис. 2.47

Программный код файла функции choice.m для выбора варианта расчета детерминанта квадратного уравнения с применением функции “feval”

```
function det = gfdet_36_1(a,b,c)
%Программный код файла gfdet_36_1.m-расчет детерминанта квадратного...
уравнения с применением знака операции возведения в степень;
%Программа включает следующие файлы:
%Glavkvurrf1vne_36.m+gfdet_36_1.m+gfdet_36_1fev.m+choice.m
%Программа определения корней квадратного уравнения a*x^2+b*x+c=0
global ss;
det=b^2-4*a*c;
if det<0
  ss='Получены комплексные корни';
  else
    ss='Получены вещественные корни';
end
end
```

Рис. 2.48

Программный код файла функции gfdet_36_1.m для определения детерминанта квадратного уравнения с обычным знаком возведения в степень

```
function [det] = gfdet_36_1fev(a,b,c)
%Программный код файла gfdet_36_1fev.m — расчет детерминанта квадратного...
уравнения с применением стандартной функции MATLAB  — pow2;
%Программа включает следующие файлы:
Glavkvurrf1vne_36.m+gfdet_36_1.m+gfdet_36_1fev.m+choice.m
%Программа определения корней квадратного уравнения a*x^2+b*x+c=0
global ss;
det=pow2(b)-4*a*c;
if det<0
  ss='Получены комплексные корни';
  else
    ss='Получены вещественные корни';
end
end
```

Рис. 2.49

Программный код файла функции gfdet_36_1fev.m для определения детерминанта квадратного уравнения с возведением в квадрат с использованием стандартной функции MATLAB pow2()

В функции choice.m (рис. 2.47) в зависимости от значения переменной “priznak” (это значение передается из основной программы функцией “global”) выполняются различные варианты расчета детерминанта (*b* в квадрате, либо b^2 (рис. 2.48), либо pow2(b) (рис. 2.49) с использованием функции “feval” (рис. 2.46)):

det = feval(fname2, a, b, c)

или

det = feval(fname1, a, b, c)

Новые имена выполняемых *m*-функций — fname1='gfdet_36_1’ и fname2=‘gfdet_36_1fev’ (рис. 2.48, 2.49) — передаются в функцию ‘choice’ в списке ее входных параметров с использованием символьных переменных fname1 и fname2 в заголовке функции (рис. 2.47):

function det = choice(fname1, fname2, a, b, c);

2.10.7. 5. Программа с внутренней функцией с параметрами

Функции, создаваемые пользователем, можно размещать и внутри *m*-скриптов и *m*-функций. При этом все правила создания таких внутренних функций и обращения к ним для выполнения действий такие же, как и для рассмотренных выше внешних функциях.

Важное отличие внутренних функций от внешних состоит в том, что обратиться к их выполнению можно только из того *m*-файла, в котором они созданы.

Для иллюстрации работы с внутренней функцией целесообразно создать папку, например «5. Программа с внутренней функцией с параметрами» (рис. 2.33) и разместить в ней один *m*-файл с именем kvurrf1vnu_37, содержащий внутреннюю функцию (рис. 2.50).

Программный код *m*-функции без параметров kvurrf1vnu_37, содержащий внутреннюю функцию ‘vfdet’, представлен на рисунке 2.51.

Внутренняя функция для расчета детерминанта квадратного уравнения с заголовком:

function det = vfdet(a, b, c)

располагается в программе после оператора:

disp (‘Файлы программы: kvurrf1vnu_37.m’)

и заканчивается оператором ‘end’ перед оператором:

disp ('ИСХОДНАЯ ИНФОРМАЦИЯ ДЛЯ РАСЧЕТОВ’)

Обращение к выполнению внутренней функции в программе располагается после комментария:

%ВЫПОЛНЕНИЕ ВЫЧИСЛЕНИЙ

с конкретными значениями входных переменных (a = 1 b = 3 c = 2).

det = vfdet(a, b, c)

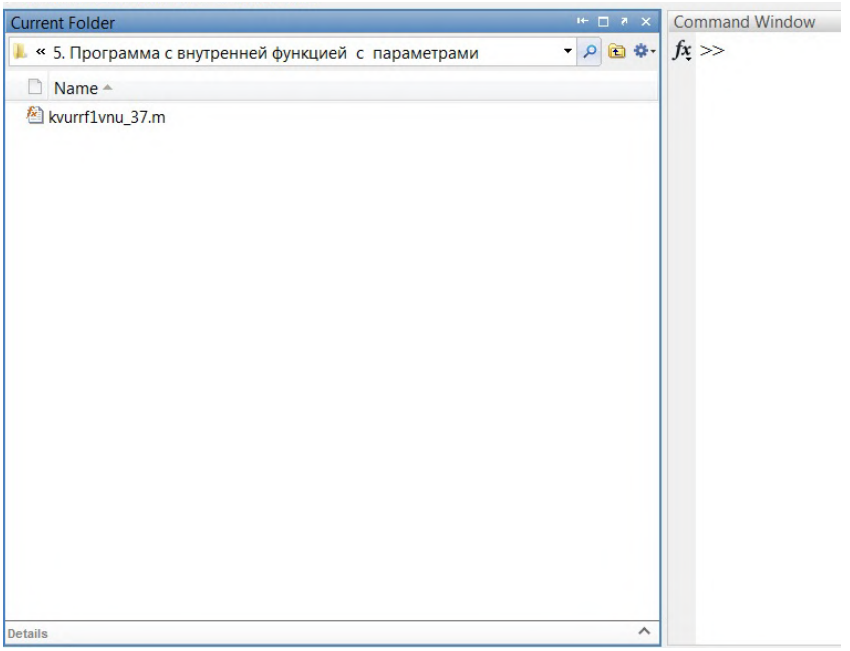


Рис. 2.50

Содержимое папки при создании внутренней функции с параметрами в файле kvurrf1vnu_37.m

```
function kvurrf1vnu_37
%Программный код файла kvurrf1vnu_37.m — основная программа
%Программа включает следующие файлы: kvurrf1vnu_37.m
%Программа определения корней квадратного уравнения
clc;
clear all;
close all;
global ss;
disp('Программа определения корней квадратного уравнения')
disp('              a*x^2+b*x+c=0              ')
disp('Файлы программы: kvurrf1vnu_37.m');
function det = vfdet(a,b,c)
%Внутренняя функция vfdet программы для определения детерминанта и типа корней
%квадратного уравнения a*x^2+b*x+c=0
det=b^2-4*a*c;
if det<0
  ss='Получены комплексные корни';
  else
    ss='Получены вещественные корни';
end
end
disp('ИСХОДНАЯ ИНФОРМАЦИЯ ДЛЯ РАСЧЕТОВ');
%Задание значений коэффициентов квадратного уравнения
%a*x^2+b*x+c=0
a=1
b=3
c=2
```

Рис. 2.51 (начало)

Программный код *m*-функции с внутренней функцией ‘vfdet’

```
%ВЫПОЛНЕНИЕ  ВЫЧИСЛЕНИЙ
det=vfdet(a,b,c);
x1=(-b-sqrt(det))/2/a;
x2=(-b+sqrt(det))/2/a;
disp ('РЕЗУЛЬТАТЫ  РАСЧЕТОВ')
disp(ss);disp('x1=');disp(x1);disp('x2=');disp(x2);
%ПОСТРОЕНИЕ  ГРАФИКА  ФУНКЦИИ  УРАВНЕНИЯ  f(x)=a*x^2+b*x+c
%Формирование  массива  данных  для  оси  абсцисс
x=[-5:0.1:0];
%Формирование  соответствующего  массива  данных  для  оси  ординат
y=(x.^2)*a+x.*b+c;
%Изображение  графика
plot(x,y,'r.');
%Нанесение  сетки  на  график
grid on;
%Заголовок  графика
title('Графическое  представление  функции  кв.  уравнения  a*x^2+b*x+c=0');
%Текст  под  осью  абсцисс
xlabel('Аргумент  функции — x');
%Текст  рядом  с  осью  ординат
ylabel('Функция  f(x)=a*x^2+b*x+c');
   end
```

Рис. 2.51 (окончание)

Программный код *m*-функции с внутренней функцией ‘vfdet’

2.10.8. 6. Программа с функцией “inline”

Собственные функции, включающие в себя одно выражение с различным числом параметров, могут быть созданы с использованием стандартной функции MATLAB “inline”.

Так, например, функция расчета детерминанта квадратного уравнения (fdet) с применением функции “inline” может быть записана

fdet = inline(‘b^2 – 4*a*c’);

Обращение к выполнению этой функции происходит обычным способом с предварительным заданием значений *a*, *b* и *c*:

det = fdet(a, b, c);

Для иллюстрации функции с “inline” целесообразно создать отдельную папку, например «6. Программа с функцией “inline”» (рис. 2.33) и разместить в ней файл kvurrf1inl_38 (рис. 2.52).

Создаваемая функция “inline” располагается в программе (рис. 2.53) после оператора

close all,

а сама функция записывается после слова “inline” в круглых скобках и кавычках.

Обращение к выполнению функции “inline” размещается в программе (рис. 2.53) после комментария:

%ВЫПОЛНЕНИЕ ВЫЧИСЛЕНИЙ.

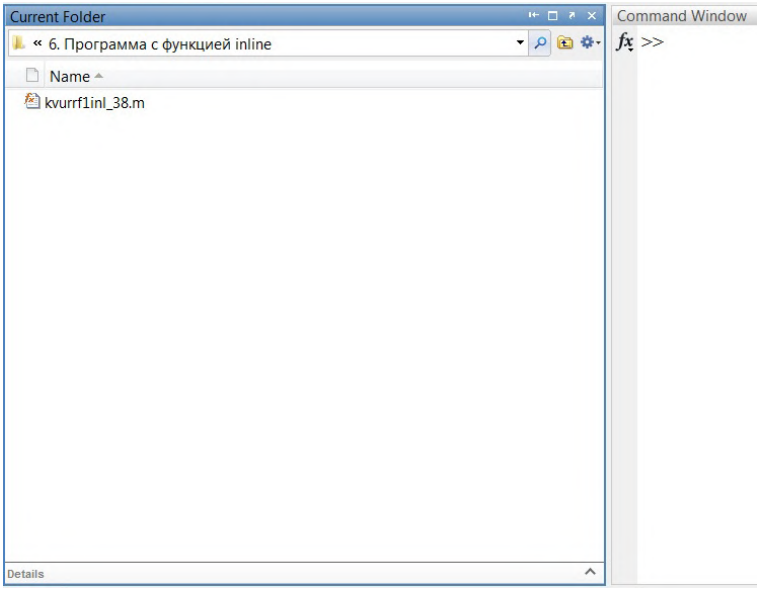


Рис. 2.52

Содержимое папки при создании в файле kvurrf1inl_38.m функции “inline”

```
function kvurrf1inl_38
%Программный код файла kvurrf1inl.m — основная программа
%Программа включает следующие файлы: kvurrf1inl.m
%Программа определения корней квадратного уравнения
clc;
clear all;
close all;
fdet=inline('b^2-4*a*c');
disp('Программа определения корней квадратного уравнения')
disp('                a*x^2+b*x+c=0              ')
disp('Файлы программы: kvurrf1inl.m');
disp('ИСХОДНАЯ ИНФОРМАЦИЯ ДЛЯ РАСЧЕТОВ');
%Задание значений коэффициентов квадратного уравнения
%a*x^2+b*x+c=0
a=1
b=3
c=2
%ВЫПОЛНЕНИЕ ВЫЧИСЛЕНИЙ
det=fdet(a,b,c);
if det<0
  ss='Получены комплексные корни';
  else
    ss='Получены вещественные корни';
end
x1=(-b-sqrt(det))/2/a;
x2=(-b+sqrt(det))/2/a;
disp ('РЕЗУЛЬТАТЫ РАСЧЕТОВ')
disp(ss);disp('x1=');disp(x1);disp('x2=');disp(x2);
%ПОСТРОЕНИЕ ГРАФИКА ФУНКЦИИ УРАВНЕНИЯ f(x)=a*x^2+b*x+c
%Формирование массива данных для оси абсцисс
x=[-5:0.1:0];
```

Рис. 2.53 (начало)

Программный код файла kvurrf1inl_38.m с функцией “inline”

```
%Формирование соответствующего массива данных для оси ординат
y=(x.^2)*a+x.*b+c;
%Изображение графика
plot(x,y,'r.');
%Нанесение сетки на график
grid on;
%Заголовок графика
title('Графическое представление функции кв. уравнения a*x^2+b*x+c=0');
%Текст под осью абсцисс
xlabel('Аргумент функции — x');
%Текст рядом с осью ординат
ylabel('Функция f(x)=a*x^2+b*x+c');
   end
```

Рис. 2.53 (окончание)

Программный код файла kvurrf1inl_38.m с функцией “inline”

2.10.9. 7. Стандартное оформление программы с *m*-скриптом в качестве основной управляющей программы и отдельными файлами для ввода информации (DATA.m) и отчета о результатах вычислений (REPORT.m)

В соответствии с методологией структурного программирования целесообразно отдельные фрагменты программы оформлять в виде блоков — различных программных модулей (*m*-скриптов и *m*-функций), имеющих самостоятельное смысловое и практическое назначение.

В данном случае рассматриваемая программа определения корней квадратного уравнения должна состоять из пяти *m*-файлов, размещаемых в отдельной папке «7. Программа стандартная со скриптом +DATA +REPORT» (рис. 2.33), которые представлены на рис. 2.54[2]:

Glavkvurscript_39.m
DATA_39_1.m
kvurcalc_39_2.m
gfdet_39_3.m
REPORT_39_4.m

Программные коды перечисленных *m*-файлов приведены соответственно на рисунках 2.55–2.59.

В соответствии с требованиями стандартного оформления программ на *m*-языке, в отдельном файле (*m*-функция без параметров DATA_39_1.m) вводится исходная информация и в отдельном файле (*m*-функция без параметров REPORT_39_4.m) оформляются результаты вычислений, т. е. формируется отчет о выполненных действиях, включая построение графиков.

Основная управляющая программа Glavkvurscript_39.m оформлена в виде *m*-скрипта и вызывает для выполнения последовательно следующие *m*-файлы (рис. 2.55):

[2] Файл Glavkvurscript.fig получается в результате решения и представляет собой графическую интерпретацию расчетных результатов.

DATA_39_1 — задание исходной информации

[x1, x2] = kvurcalc_39_2(a, b, c) — определение корней квадратного уравнения

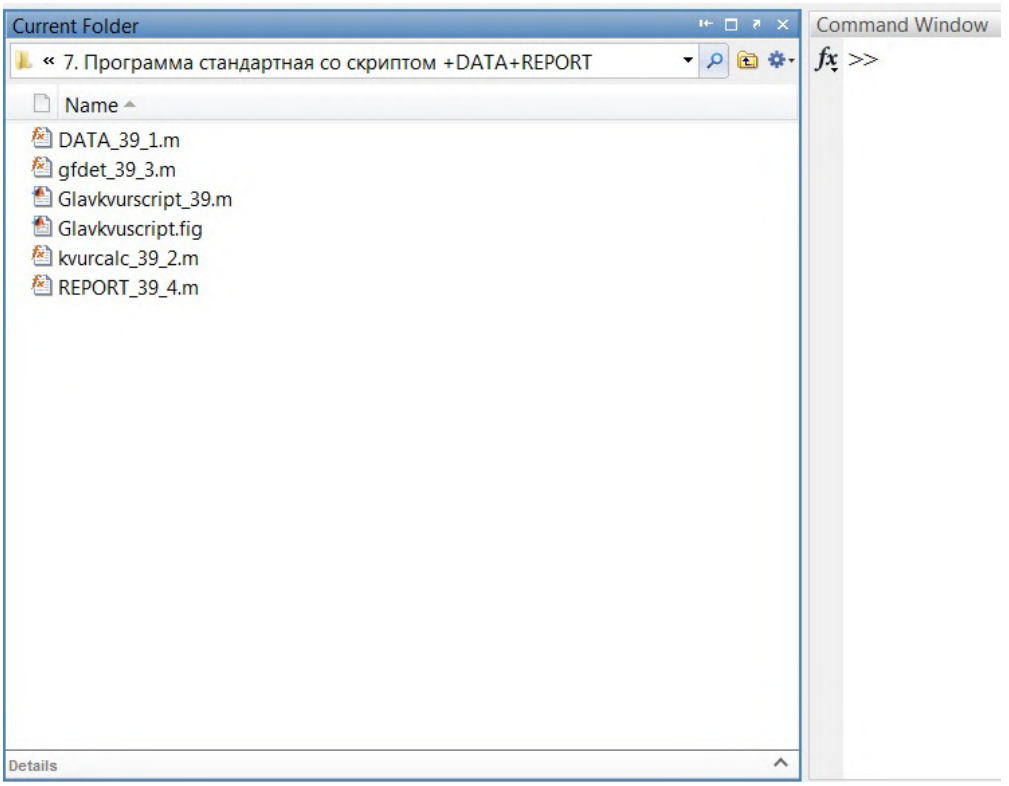


Рис. 2.54

Содержимое папки при стандартном оформлении решения квадратного уравнения

```
%Программный код файла Glavkvurscript_39.m — основная управляющая программа
%Программа включает следующие  файлы:
%:Glavkvurscript_39.m+DATA_39_1.m+kvurcalc_39_2.m+gfdet_39_3.m+REPORT_39_4.m
%Программа определения корней квадратного уравнения a*x^2+b*x+c=0
clc;
clear all;
close all;
global a b c x1 x2;
DATA_39_1;
[x1,x2]=kvurcalc_39_2(a,b,c);
REPORT_39_4;
```

Рис. 2.55

Программный код скрипта основной управляющей программы Glavkvurscript_39.m при стандартном оформлении программы решения квадратного уравнения

```
function DATA_39_1
%Программный код файла DATA_39_1.m — задание информации для расчетов
%Программа включает следующие файлы:
%Glavkvurscript_39.m+DATA_39_1.m+kvurcalc_39_2.m+gfdet_39_3.m+REPORT_39_4.m
%Программа определения корней квадратного уравнения a*x^2+b*x+c=0
%global a b c;
%Задание коэффициентов квадратного уравнения a*x^2+b*x+c=0
a=1;b=3;c=2;
end
```

Рис. 2.56

Программный код функции задания исходной информации DATA_39_1.m при стандартном оформлении программы решения квадратного уравнения

```
function [ x1,x2] = kvurcalc_39_2(a,b,c)
%Программный код файла kvurcalc_39_2.m — вычисление корней квадратного уравне-
ния
%Программа включает следующие файлы:
%Glavkvurscript_39.m+DATA_39_1.m+kvurcalc_39_2.m+gfdet_39_3.m+REPORT_39_4.m
%Программа определения корней квадратного уравнения a*x^2+b*x+c=0
%ВЫПОЛНЕНИЕ ВЫЧИСЛЕНИЙ
det=gfdet_39_3(a,b,c);
x1=(-b-sqrt(det))/2/a;
x2=(-b+sqrt(det))/2/a;
end
```

Рис. 2.57

Программный код функции вычисления корней квадратного уравнения kvurcalc_39_2.m при стандартном оформлении программы решения квадратного уравнения

```
function det = gfdet_39_3(a,b,c)
%Программный код файла gfdet_39_3.m — определение детерминанта квадратного
%уравнения a*x^2+b*x+c=0
%Программа включает следующие файлы:...
Glavkvurscript_39.m+DATA_39_1.m+kvurcalc_39_2.m+gfdet_39_3.m+REPORT_39_4.m
%Программа определения корней квадратного уравнения a*x^2+b*x+c=0
global ss;
det=b^2-4*a*c;
if det<0
  ss='Получены комплексные корни';
  else
    ss='Получены вещественные корни';
end
end
```

Рис. 2.58

Программный код функции определения детерминанта квадратного уравнения gfdet_39_3.m при стандартном оформлении программы решения квадратного уравнения

```
function REPORT_39_4
%Программный код файла REPORT_39_4.m — отчет о работе программы
%Программа включает следующие %фай-
лы:...Glavkvurscript_39.m+DATA_39_1.m+kvurcalc_39_2.m+gfdet_39_3.m+REPORT_39_4.m
%Программа определения корней квадратного уравнения a*x^2+b*x+c=0
global ss a b c x1 x2;
disp('Программа определения корней квадратного уравнения')
disp('              a*x^2+b*x+c=0              ')
disp('Файлы
программы:Glavkvurscript_39.m+DATA_39_1.m+kvurcalc_39_2.m+gfdet_39_3.m+REPORT_39_4.m
```

Рис. 2.59 (начало)

Программный код функции отчета о работе программы REPORT_39_4.m при стандартном оформлении программы решения квадратного уравнения

```
');
disp('ИСХОДНАЯ ИНФОРМАЦИЯ ДЛЯ РАСЧЕТОВ');
disp('Коэффициенты квадратного уравнения a*x^2+b*x+c=0.:');
disp('a=');disp(a);
disp('b=');disp(b);
disp('c=');disp(c);
disp ('РЕЗУЛЬТАТЫ РАСЧЕТОВ')
disp(ss);disp('x1=');disp(x1);disp('x2=');disp(x2);
%ПОСТРОЕНИЕ ГРАФИКА ФУНКЦИИ УРАВНЕНИЯ f(x)=a*x^2+b*x+c
%Формирование массива данных для оси абсцисс
x=[-5:0.1:0];
%Формирование соответствующего массива данных для оси ординат
y=(x.^2)*a+x.*b+c;
%Изображение графика
plot(x,y,'r.');
%Нанесение сетки на график
grid on;
%Заголовок графика
title('Графическое представление функции кв. уравнения a*x^2+b*x+c=0');
%Текст под осью абсцисс
xlabel('Аргумент функции — x');
%Текст рядом с осью ординат
ylabel('Функция f(x)=a*x^2+b*x+c');

end
```

Рис. 2.59 (окончание)

Программный код функции отчета о работе программы REPORT_39_4.m при стандартном оформлении программы решения квадратного уравнения

В свою очередь функция с параметрами kvurcalc_39_2(a, b, c) (рис. 2.57) вызывает для выполнения функцию gfdet_39_3(a, b, c) (рис. 2.58) определения детерминанта квадратного уравнения.

В результате выполнения функции отчета о работе программы REPORT_39_4.m (рис. 2.59) в командном окне интегрированной среды MATLAB появляется текстовый отчет о работе программы (рис. 2.60) и графический отчет (рис. 2.61).

```
Программа определения корней квадратного уравнения
          a*x^2+b*x+c=0
Файлы программы:
Glavkvurscript_39.m+DATA_39_1.m+kvurcalc_39_2.m+gfdet_39_3.m+REPORT_39_4.m
ИСХОДНАЯ ИНФОРМАЦИЯ ДЛЯ РАСЧЕТОВ
Коэффициенты квадратного уравнения a*x^2+b*x+c=0.:
a=
    1

b=
    3

c=
    2

РЕЗУЛЬТАТЫ РАСЧЕТОВ
Получены вещественные корни
x1=
   -2

x2=
   -1
```

Рис. 2.60

Текстовый отчет о работе программы определения корней квадратного уравнения

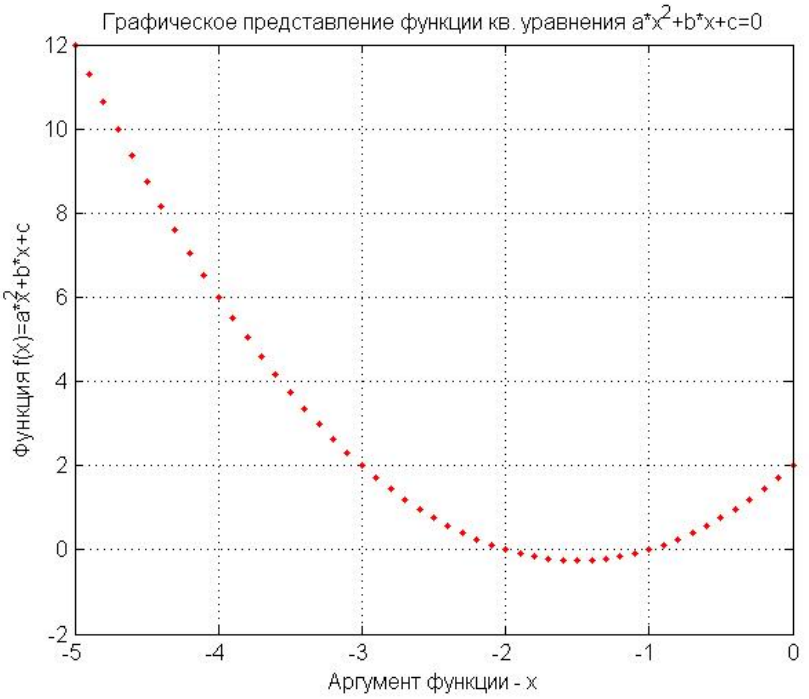


Рис. 2.61

Графический отчет о работе программы определения корней квадратного уравнения

Для сохранения текстового отчета (рис. 2.60), полученного в Командном окне, необходимо скопировать его комбинацией клавиш <Ctrl+A>, а затем <Ctrl+C> и после этого разместить отчет, например, в Блокноте путем комбинации клавиш <Ctrl+V> с соответствующим присвоением имени *.txt файлу.

Сохранение графического отчета (рис. 2.61) осуществляется после его открытия командой меню MATLAB **File → Save As**.

По умолчанию графики сохраняются с расширением *.fig в Текущей папке, где располагаются *m*-файлы программы. При этом файлу графика требуется присвоить имя. Для использования графиков вне интегрированной среды MATLAB они должны быть сохранены с расширением, например, *.jpeg также с присвоением имени файлу.

2.10.10. 8. Программа с решателем MATLAB “roots”

Для решения вычислительных задач MATLAB допускает применение широкого набора решателей (solver), которые с применением стандартных алгоритмов позволяют наиболее удобно получать корректные решения, как правило, численными методами.

К таким решателям относится и решатель “roots”, с применением которого можно определять вещественные и комплексные корни уравнений многочлена произвольной степени.

Для этой цели, например в случае уравнения многочлена второй степени $x^2 + 3x + 2 = 0$, необходимо в командной строке Командного окна MATLAB в квадратных скобках записать коэффициенты уравнения в порядке убывания степени неизвестной x:

>> zz = roots([1,3,2]).

Понятно, что для уравнения многочлена второй степени это три коэффициента, а для уравнения многочлена, например, шестой степени — семь коэффициентов.

В результате выполнения решателя “roots” с коэффициентами многочлена в квадратных скобках переменная zz получит все значения корней многочлена, как вещественные, так и комплексные.

Так как числа в квадратных скобках в терминологии *m*-языка программирования представляют собой одномерный массив, то предыдущий оператор для определения корней квадратного уравнения может быть записан с использованием двух операторов в командной строке командного окна:

>> pp = [1, 2 3];

>> zz = roots(pp);

Применение функции “roots” в *m*-файлах показано на примере решения уравнений многочленов второй и шестой степени, для чего целесообразно создать папку, например, с именем «Программа со стандартной функцией MATLAB roots» (рис. 2.33) и в ней еще две папки с именами (рис. 2.62):

- «Квадратное уравнение»;
- «Уравнение 6-й степени».

В первой папке размещены три *m*-файла для решения приведенного выше квадратного уравнения (рис. 2.63). Программные коды этих файлов приведены на рисунках 2.64–2.66.

В основной управляющей программе Glavkvurroots.m (рис. 2.64) происходит обращение к вычислению корней квадратного уравнения с использованием операторов

```
pp = [a,b,c]
results = roots(pp).
```

В результате первый корень равен x1, а второй — x2:

x1 = results(1); x2 = results(2).

Содержимое папки для решения уравнения многочлена шестой степени показано на рисунке 2.67 и включает в себя один *m*-файл — Grkv6.m, представляющий собой *m*-скрипт, в котором уравнение решается с применением решателя “roots” (рис. 2.68):

[result1] = roots([1,3,2,0,7,6,-5]).

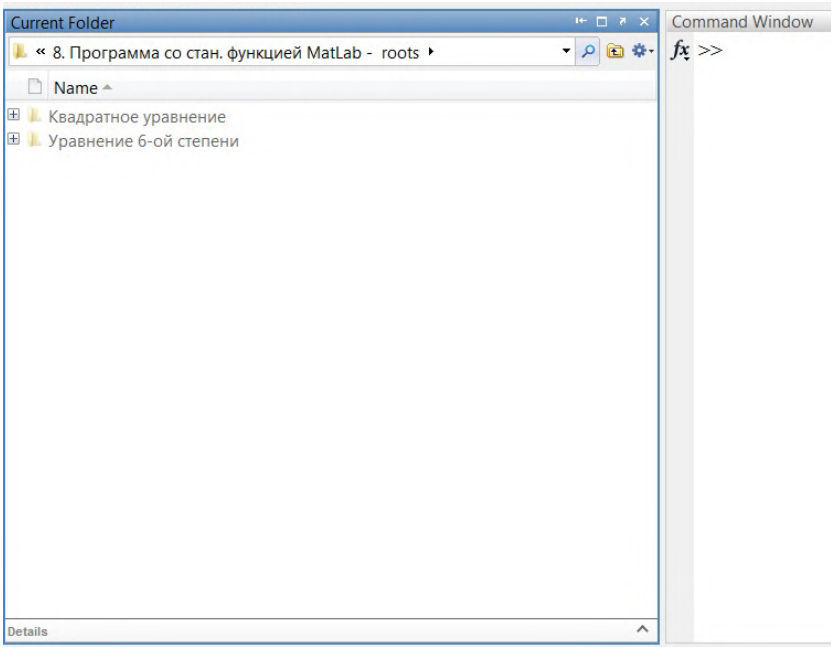


Рис. 2.62

Две папки для решения квадратного уравнения и уравнения шестой степени с применением решателя “roots”

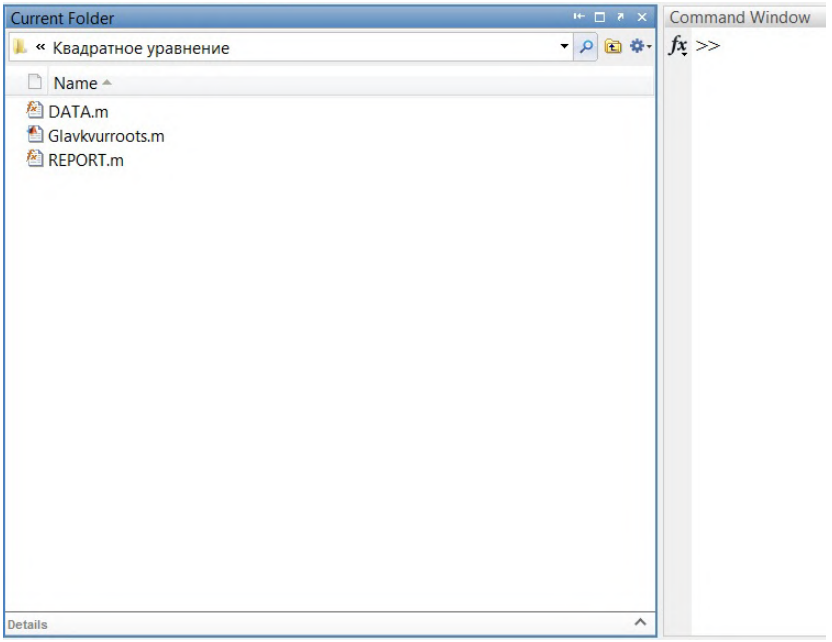


Рис. 2.63

Содержание папки для решения квадратного уравнения с применением решателя “roots”

```
%Программный код файла Glavkvurroots.m — основная управляющая программа
%Программа включает следующие файлы: Glavkvurroots.m+DATA.m+REPORT.m
%Программа определения корней квадратного уравнения a*x^2+b*x+c=0
clc;
clear all;
close all;
global a b c x1 x2;
DATA;
pp=[a,b,c];
results=roots(pp);
x1=results(1);x2=results(2);
REPORT;
```

Рис. 2.64

Программный код *m*-скрипта Glavkvurroots.m основной управляющей программы для решения квадратного уравнения с применением стандартной функции “roots”

```
function DATA
%Программный код файла DATA.m — задание информации для расчетов
%Программа включает следующие файлы: Glavkvurroots.m+DATA.m+REPORT.m
%Программа определения корней квадратного уравнения a*x^2+b*x+c=0
global a b c;
%Задание коэффициентов квадратного уравнения a*x^2+b*x+c=0
a=1;b=3;c=2;
end
```

Рис. 2.65

Программный код *m*-функции для задания исходной информации при решении квадратного уравнения с применением решателя “roots”

```
function REPORT
%Программный код файла REPORT.m — отчет о работе программы
%Программа включает следующие файлы: Glavkvurroots.m+DATA.m+REPORT.m
%Программа определения корней квадратного уравнения a*x^2+b*x+c=0
global a  b c x1 x2;
disp('Программа определения корней квадратного уравнения')
disp('              a*x^2+b*x+c=0            ')
disp('Файлы программы:Glavkvurscript.m+DATA.m+kvurcalc.m+gfdet.m+REPORT.m ');
disp('ИСХОДНАЯ ИНФОРМАЦИЯ ДЛЯ РАСЧЕТОВ');
disp('Коэффициенты квадратного уравнения a*x^2+b*x+c=0 ;');
disp('a=');disp(a);
disp('b=');disp(b);
disp('c=');disp(c);
disp ('РЕЗУЛЬТАТЫ РАСЧЕТОВ')
disp('x1=');disp(x1);disp('x2=');disp(x2);
%ПОСТРОЕНИЕ ГРАФИКА ФУНКЦИИ УРАВНЕНИЯ f(x)=a*x^2+b*x+c
%Формирование  массива данных для оси абсцисс
x=[-5:0.1:0];
%Формирование  соответствующего массива данных для оси ординат
y=(x.^2)*a+x.*b+c;
%Изображение графика
plot(x,y,'r.');
%Нанесение сетки на график
grid on;
%Заголовок графика
title('Графическое представление функции кв. уравнения a*x^2+b*x+c=0');
%Текст под осью абсцисс
xlabel('Аргумент функции — x');
%Текст рядом с осью ординат
ylabel('Функция  f(x)=a*x^2+b*x+c');

end
```

Рис. 2.66

Программный код *m*-функции для формирования отчета при решении квадратного уравнения с применением решателя “roots”

Как следует из этого оператора и применяемого решателя “roots”, в этом случае решается следующее уравнение многочлена шестой степени:

$x^6 + 3x^5 + 2x^4 + 7x^2 + 6x - 5 = 0$.

Результаты решения, которые появляются в Командном окне, приведены на рисунке 2.69 и включают в себя два вещественных корня, а остальные — комплексные.

Графическая интерпретация решения (рис. 2.70) подтверждает достоверность рассчитанных значений вещественных корней.

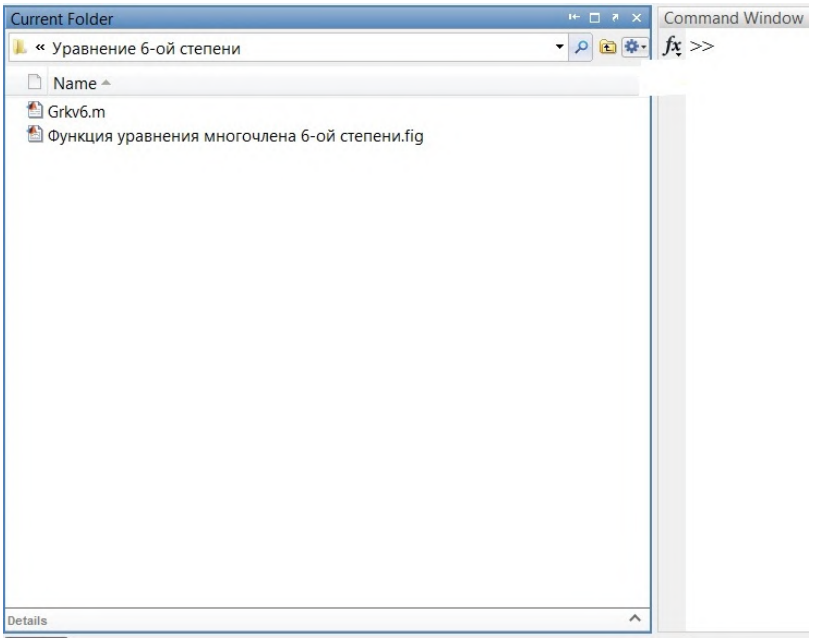


Рис. 2.67

Содержание папки для решения уравнения многочлена шестой степени с применением решателя “roots”

```
%Программный код m-скрипта Grkv6.m
clc;
clear all;
close all;
disp('Решение уравнения многочлена 6-ой степени с применением m-скрипта Gvrkv6.m');
% Решение уравнения многочлена 6-ой степени с применением стандартной функции roots
disp('Результаты решения уравнения многочлена 6-ой степени вида:');
disp('x^6+3*x^5+2*x^4+7*x^2+x*6-5=0 ');
disp('с применением стандартной функции roots(...)');
[result1]=roots([1,3,2,0,7,6,-5])
% Графическая интерпретация результатов решения
%Построение графика функции уравнения многочлена 6-ой степени
%f(x)=x^6+3*x^5+2*x^4+7*x^2+6*x-5;
x=[-3:0.1:3];
```

Рис. 2.68 (начало)

Программный код файла Grkv6.m *m*-скрипта для решения уравнения многочлена шестой степени

```
y=x.^6+3*x.^5+2*x.^4+7*x.^2+x.*6-5;
plot(x,y,'r-');
grid on;
title('Графическое предс. функции уравн. x^6+3*x^5+2*x^4+7*x^2+6*x-5=0');
xlabel('Аргумент функции — x');
ylabel('Функция f(x)=x^6+3*x^5+2*x^4+7*x^2+6*x-5');
axis([-3 3 -20 20]);
```

Рис. 2.68 (окончание)

Программный код файла Grkv6.m *m*-скрипта для решения уравнения многочлена шестой степени

```
Решение уравнения многочлена 6-ой степени с применением m-скрипта Gvrkv6.m
Результаты решения уравнения многочлена 6-ой степени вида:
x^6+3*x^5+2*x^4+7*x^2+x*6-5=0
с применением стандартной функции roots(...)

result1 =

  -1.7248 + 0.9310i
  -1.7248 — 0.9310i
  -1.4570
   0.7028 + 1.1353i
   0.7028 — 1.1353i
   0.5010

>>
```

Рис. 2.69

Результат решения уравнения многочлена шестой степени с применением *m*-скрипта Grkv6.m

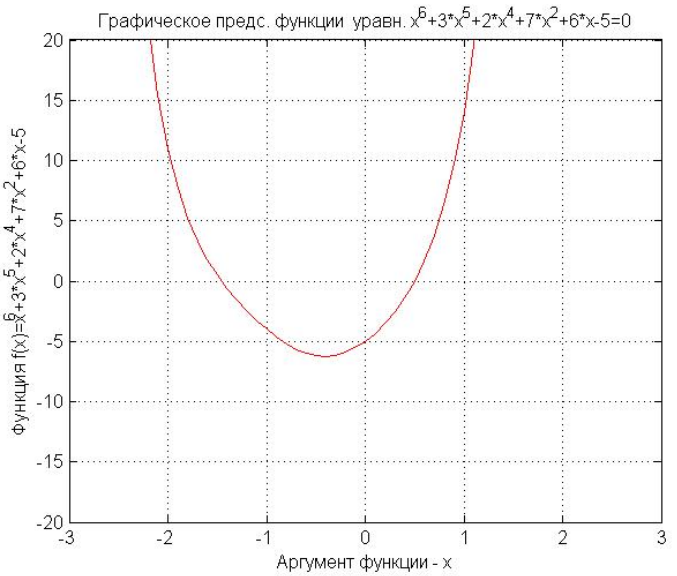


Рис. 2.70

Графическая интерпретация полученных вещественных корней уравнения многочлена шестой степени с применением *m*-скрипта Grkv6.m

2.10.11. 9. Программа стандартная с отчетом в текстовом файле

Представление отчета о результатах работы программы в Командном окне интегрированной среды MATLAB с использованием функции “disp” не всегда удобно, так как, как было указано в разделе 7, приходится предпринимать определенные действия для перевода текстового отчета в соответствующие текстовые редакторы типа Notepad (блокнот), WordPad.

На *m*-языке программирования возможно создание текстовых файлов с расширением *.txt. Текстовый файл может содержать любые символы, организуется по строкам и заканчивается символом «конец файла» — “fclose”. Для создания текстового файла на *m*-языке программирования сначала записывается функция “fopen” следующего вида:

ff = fopen (‘имя файла’, ‘режим работы файла’),

где ff — идентификатор файла; ‘имя файла’ — это название текстового файла, который имеет расширение *.txt, например ‘rachet.txt’, ‘pusk.txt’, ‘ocenka.txt’ и т. п.; ‘режим работы файла’ — в простейшем случае может быть: ‘rt’ — открываемый для чтения, или ‘wt’ — создаваемый текстовый файл предназначен только для записи информации. Кроме этих двух режимов, существуют режимы: ‘rt+’ — открыть файл для чтения и записи, ‘at’ — открываемый текстовый файл будет использоваться для добавления данных в конец файла (если файла нет, он будет создан).

Программирование текстового файла на *m*-языке программирования заканчивается функцией (последний оператор текстового файла):

fclose (ff),

где ff — идентификатор закрываемого файла, декларируемый в начальном операторе текстового файла с функцией “fopen”.

Между первым и последним операторами при программировании текстового файла записываются различные известные операторы *m*-языка, но вместо функции “disp”, используемой для вывода данных в Командное окно, записывается оператор “fprintf” — для вывода данных в текстовый файл.

В общем случае функция “fprint” имеет следующий вид:

fprintf (<идентификатор файла>, <формат переменных>, <переменные>),

где <идентификатор файла>, например ff в рассматриваемом случае, — это файл, значение которого задается функцией “fopen” при открытии текстового файла; <формат переменных> — указывается, к какому типу нужно преобразовать данные о переменных, перечисленных в следующей позиции функции “fprintf” — <переменные>, и как расположить их в текстовом файле — сколько значений в каждой строке; <переменные> — выводимые переменные, отделяемые друг от друга пробелами или запятыми.

Так, например, при создании текстового файла с именем Sagolovok.txt в *m*-файле, в котором он создается, следует записать следующее:

```
ff1 = fopen ('Sagolovok.txt', 'wt');
fprintf (ff1, '% s\n', 'Решение уравнения');
fclose (ff1).
```

В этом случае формат 's\n' означает, что в текстовый файл выводится символьная переменная 'Решение уравнения' — формат 's', и после нее выполняется переход на следующую строку — формат '\n'.

Таблица 2.12

Символы управления форматированием

Параметр	Назначение
Флаги	
–	Выравнивание числа влево. Правая сторона дополняется пробелами. По умолчанию — выравнивание вправо
+	Перед числом выводится знак «+» или «–»
0	Заполнение. Незаполненные позиции дополняются нулями
Ширина	
n	Ширина поля вывода. Если *n* позиций недостаточно, то поле вывода расширяется до минимально необходимого. Незаполненные позиции дополняются пробелами
Точность	
нет	Точность по умолчанию
m	Для типов *e*, *E*, *f* выводить *m* знаков после десятичной точки
Тип	
e	При вводе — символьный тип char, при выводе — один байт
d	Десятичное целое со знаком
i	Десятичное целое со знаком
o	Восьмеричное целое без знака
u	Десятичное целое без знака
x, X	Шестнадцатеричное целое без знака, при *x* используются символы a–f, при *X* — A–F.
f	Значение со знаком вида [-]dddd.dddd
g	Значение со знаком вида [-]d.dddd e [+|-]ddd
E	Значение со знаком вида [-]d.dddd E [+|-]ddd
q	Значение со знаком типа *e* или *f* в зависимости от значения и точности
G	Значение со знаком типа *E* или *F* в зависимости от значения и точности
s	Строка символов

Таблица 2.13

Специальные символы

Символ	Назначение
\b	Сдвиг текущей позиции влево
\n	Перевод строки
\F	Перевод в начало строки, не переходя на новую строку
\t	Горизонтальная табуляция
\'	Символ одинарной кавычки
\"	Символ двойной кавычки

Продолжение табл. 2.13

Символ	Назначение
\?	Символ «?»

Если необходимо выполнить перевод строки с применением специального символа (\n) до строки со строчной константой ‘Решение уравнения’, то до описанной функции “fprintf” необходимо записать:

fprintf (ff1, ‘\n’),

в случае пропуска двух строк:

fprintf (ff1, ‘\n\n’).

В случае пропуска двух строк до строковой константы ‘Решение уравнения’ и двух строк после нее допустима запись

fprintf (ff1, ‘\n\n Решение уравнения \n\n’).

Программный код файла REPORT_39_4.m, с использованием которого создается текстовый файл отчета о решении рассматриваемого квадратного уравнения, приведен на рисунке 2.71, а результаты расчетов в текстовом файле «Отчет о решении квадратного уравнения.txt» представлены на рисунках 2.72 и 2.73.

```
function REPORT_39_4
%Программный код файла REPORT.m — отчет о работе программы
%Программа включает следующие %фай-
лы:...Glavkvurscript.m+DATA.m+kvurcalc.m+gfdet.m+REPORT.m
%Программа определения корней квадратного уравнения a*x^2+b*x+c=0
global ss a b;global c x1 x2 det;
ff=fopen('Отчет о решении квадратного уравнения.txt','wt');
fprintf(ff,'\n-----------------------------------------------------------------------------\n');
fprintf(ff,'Программа определения корней квадратного уравнения');
fprintf(ff,'\n-----------------------------------------------------------------------------\n');
fprintf(ff,'              a*x^2+b*x+c=0             ');
fprintf(ff,'\n-----------------------------------------------------------------------------\n');
fprintf(ff,'Файлы программы:Glavkvurscript.m+DATA.m+kvurcalc.m+gfdet.m+REPORT.m ');
fprintf(ff,'\n-----------------------------------------------------------------------------\n');
fprintf(ff,'ИСХОДНАЯ ИНФОРМАЦИЯ ДЛЯ РАСЧЕТОВ');
fprintf(ff,'\n-----------------------------------------------------------------------------\n');
fprintf(ff,'Коэффициенты  квадратного уравнения a*x^2+b*x+c=0 ;');
fprintf(ff,'\n-----------------------------------------------------------------------------\n');
fprintf(ff,'a=');fprintf(ff,'%g\n',a);fprintf(ff,'b=');fprintf(ff,'%g\n',b);fprintf(ff,'c=');fprintf(ff,'%g\n',c);
fprintf(ff,'\n-----------------------------------------------------------------------------\n');
fprintf(ff,'РЕЗУЛЬТАТЫ РАСЧЕТОВ');
fprintf(ff,'\n-----------------------------------------------------------------------------\n');
fprintf(ff,ss);fprintf(ff,'\n-----------------------------------------------------------------------------\n');
if det>=0
fprintf(ff,'x1=');fprintf(ff,'%g\n',x1);fprintf(ff,'x2=');fprintf(ff,'%g\n',x2);
else
fprintf(ff,'x1=');fprintf(ff,'%g %g',real(x1),imag(x1));fprintf(ff,'i');fprintf(ff,'\n');
fprintf(ff,'x2=');fprintf(ff,'%g',real(x2));fprintf(ff,'+');fprintf(ff,'%g',imag(x2));fprintf(ff,'i');
end
%ПОСТРОЕНИЕ ГРАФИКА ФУНКЦИИ УРАВНЕНИЯ f(x)=a*x^2+b*x+c
%Формирование  массива данных для оси абсцисс
x=[-5:0.1:0];
%Формирование  соответствующего массива данных для оси ординат
y=(x.^2)*a+x.*b+c;
%Изображение графика
plot(x,y,'r.');
%Нанесение сетки на график
```

Рис. 2.71 (начало)

Программный код файла отчета о решении квадратного уравнения, с использованием которого создается текстовый файл ‘Отчет о решении квадратного уравнения.txt’

```
grid on;
%Заголовок графика
title('Графическое представление функции кв. уравнения a*x^2+b*x+c=0');
%Текст под осью абсцисс
xlabel('Аргумент функции — x');
%Текст рядом с осью ординат
ylabel('Функция f(x)=a*x^2+b*x+c');
axis([-5 0 -1 18]);
fclose(ff);
end
```

Рис. 2.71 (окончание)

Программный код файла отчета о решении квадратного уравнения, с использованием которого создается текстовый файл ‘Отчет о решении квадратного уравнения.txt’

```
---------------------------------------------------------------------------
Программа определения корней квадратного уравнения
---------------------------------------------------------------------------
            a*x^2+b*x+c=0
---------------------------------------------------------------------------
Файлы программы: Glavkvurscript.m+DATA.m+kvurcalc.m+gfdet.m+REPORT.m
---------------------------------------------------------------------------
ИСХОДНАЯ ИНФОРМАЦИЯ ДЛЯ РАСЧЕТОВ
---------------------------------------------------------------------------
Коэффициенты квадратного уравнения a*x^2+b*x+c=0 :
---------------------------------------------------------------------------
a=1
b=3
c=2

---------------------------------------------------------------------------
РЕЗУЛЬТАТЫ РАСЧЕТОВ
---------------------------------------------------------------------------
Получены вещественные корни
---------------------------------------------------------------------------
x1=-2
x2=-1
```

Рис. 2.72

Отчет о решении квадратного уравнения с вещественными корнями в соответствующем текстовом файле

```
---------------------------------------------------------------------------
Программа определения корней квадратного уравнения
---------------------------------------------------------------------------
            a*x^2+b*x+c=0
---------------------------------------------------------------------------
Файлы программы: Glavkvurscript.m+DATA.m+kvurcalc.m+gfdet.m+REPORT.m
---------------------------------------------------------------------------
ИСХОДНАЯ ИНФОРМАЦИЯ ДЛЯ РАСЧЕТОВ
---------------------------------------------------------------------------
Коэффициенты квадратного уравнения a*x^2+b*x+c=0 :
---------------------------------------------------------------------------
a=1
b=2
c=3

---------------------------------------------------------------------------
РЕЗУЛЬТАТЫ РАСЧЕТОВ
---------------------------------------------------------------------------
Получены комплексные корни
---------------------------------------------------------------------------
x1=-1 -1.41421i
x2=-1+1.41421i
```

Рис. 2.73

Отчет о решении квадратного уравнения с комплексными корнями в соответствующем текстовом файле

Дополнительные возможности работы с текстовыми файлами

I. Запись информации.

При создании текстовых файлов из элементов матриц иногда целесообразно организовать ввод данных из Командного окна. В этом случае для ввода числа строк (M) и столбцов (N) матрицы A используется оператор “input” в следующем виде (рис. 2.74):

M = input (‘Число строк матрицы M =’)

и

N = input (‘Число столбцов матрицы N =’).

Поэлементный ввод матрицы A выполняется из отдельной строки для каждого ее компонента в виде:

A (i, j) =

где i и j изменяются соответственно от 1 до M и от 1 до N.

Для этой цели в код программы включается двойной итерационный цикл следующего вида (рис. 2.74):

```
for i=1:M
   for j=1:N
      A(i,j)=input(strcat('A(',int2str(i),',',int2str(j),')='));
   end
end
```

При этом в функции “input” для ввода данных используются функции работы со строками — “strcat” и “int2str”.

Функция “strcat” предназначена для объединения различных строковых переменных в одну строку, а функция “int2str” преобразует целые числовые переменные в символьные, как это требуется в функции “input”.

В создаваемом текстовом файле Matrix_txt_sozdanie.txt (рис. 2.74) в первой строке располагается информация о числе строк и столбцов матрицы A, а во второй и третьей строках — элементы первой и второй строки матрицы, так как задавалась матрица с двумя строками (рис. 2.75).

В программном коде файла Matrix_txt_sozdanie (рис. 2.73) для перехода к следующей строке при создании текстового файла в функции “fprintf” используется специальный символ (\w) после записи в файл первой строки:

fprintf(f,'%d\t%d\n',M,N);

и после того, как все элементы каждой отдельной строки записаны в файл (рис. 2.75):

if i<M fprintf(f,'\n').

```
function Matrix_txt_sozdanie
%ввод размеров матрицы
M=input(' Число строк матрицы M=');
N=input('Число столбцов матрицы N=');
```

Рис. 2.74 (начало)

Программный код файла создания текстового файла ‘Matrix_txt_sozdanie.txt’

```
disp('ПОЭЛЕМЕНТНЫЙ  ВВОД МАТРИЦЫ A');
for i=1:M
   for j=1:N
      A(i,j)=input(strcat('A(',int2str(i),',',int2str(j),')='));
   end
end
%открыть файл для записи — создать новый файл
f=fopen('Matrix_txt_sozdanie.txt','wt');
%записать в файл f.txt  с M и N, а) разделив их символом табуляции и
%б)после чего перейти на новую строку в этом файле
fprintf(f,'%d\t%d\n',M,N);
%цикл для построчной записи элементов матрицы в файл
for i=1:M
 %цикл для поэлементной записи в файл i-ой строки матрицы
 for j=1:N
  %запись очередного элемента A(i,j) и символа табуляции в файл f.txt
  fprintf(f,'%g\t',A(i,j));
 end
 %после записи очередной строки переход к следующей строке файла f.txt
if i<M fprintf(f,'\n');end
end
 %закрыть файл
 fclose(f);
end
```

Рис. 2.74 (окончание)

Программный код файла создания текстового файла ‘Matrix_txt_sozdanie.txt’

```
2       4
7.3     2.1     7.8     2.3
9.1     2.7     7.7     13.7
```

Рис. 2.75

Текстовый файл ‘Matrix_txt_sozdanie.txt’

II. Чтение информации.

Чтение информации из текстовых файлов осуществляется с использованием функций “fscanf” и “dlmread”.

При **полном считывании** всей информации из текстового файла функции “fscanf” и “dlmread” присваиваются некоторой переменной, представляющей собой массив (например, xx в программных кодах на рис. 2.75 и 2.75б):

xx = fscanf (f, ‘%g’)

и

xx = dlmread ('Matrix_txt_sozdanie_m.txt').

Из записи понятно, что считывание информации происходит из текстового файла 'Matrix_txt_sozdanie_m.txt', который представляет собой матрицу размером 2×4 (рис. 2.76в).

Как следует из рисунка 2.76б, если для формирования массива xx при использовании функции “dlmread” достаточно указать имя текстового файла, то для функции “fscanf” (рис. 2.76а) необходимо открыть текстовый файл с использованием функции “fopen” и при формировании массива xx задать формат считываемой информации — в данном случае это (‘%g’).

Важное отличие функций “fscanf” и “dlmread” состоит в том, что в случае функции “fscanf” считываемая информация размещается в одномерном массиве xx в виде вектора-столбца, а в случае функции “dlmread” — в двумерном массиве xx (матрице).

```
function Matrix_txt_chtenie_m1
f=fopen('Matrix_txt_sozdanie_m.txt','rt');
xx=fscanf(f,'%g')
fclose(f);
end
```

Рис. 2.76а

Полное считывание всей информации из текстового файла Matrix_txt_sozdanie_m.txt с использованием функции “fscanf”

```
function Matrix_txt_chtenie_m2
xx=dlmread('Matrix_txt_sozdanie_m.txt')
end
```

Рис. 2.76б

Полное считывание всей информации из текстового файла Matrix_txt_sozdanie_m.txt с использованием функции “dlmread”

```
7.3    2.1    7.8    2.3
9.1    2.7    7.7    13.7
```

Рис. 2.76в

Содержание считываемого текстового файла Matrix_txt_sozdanie_m.txt

Варианты поэлементного и строчного считывания информации из текстового файла (рис. 2.75) приведены в программных кодах на рисунках 2.77 и 2.78. В этом случае в функции “fopen” при открытии текстового файла указывается его название — 'Matrix_txt_sozdanie_m.txt', и режим чтения из него — ‘rt’. Важно, что при поэлементном считывании (рис. 2.77), кроме форматов ‘%d’ и ‘%g’, в функции fscanf указано число 1, а при построчном считывании информации (рис. 2.78) указано число N, соответствующее числу элементов в строках матрицы (рис. 2.75).

```
function matrix_txt_chtenie_1
%открываем файл для чтения
f=fopen('Matrix_txt_sozdanie.txt','rt');
M=fscanf(f,'%d',1);
N=fscanf(f,'%d',1);
%считываем в двойном цикле по одному элементу в матрицу A
for i=1:M
   for j=1:N
      A(i,j)=fscanf(f,'%g',1);
   end;
end;
```

Рис. 2.77 (начало)

Программный код файла для поэлементного считывания информации из текстового файла

```
fclose(f);
disp('Число строк матрицы A');
disp(M);
disp('Число столбцов матрицы A');
disp(N);
disp(' Элементы матрицы A');
disp(A);
end
```

Рис. 2.77 (окончание)

Программный код файла для поэлементного считывания информации из текстового файла

```
function Matrix_txt_chtenie_2
%открываем файл для чтения
f=fopen('Matrix_txt_sozdanie.txt','rt');
M=fscanf(f,'%d',1);
N=fscanf(f,'%d',1);
%считываем  в 1 цикле по строкам M элементов  в матрицу A
for i=1:M
        A(i,:)=fscanf(f,'%g',N);
end;
fclose(f);
disp('Число строк матрицы A');
disp(M);
disp('Число столбцов матрицы A');
disp(N);
disp(' Элементы матрицы A');
disp(A);
end
```

Рис. 2.78

Программный код файла для построчного считывания информации из текстового файла

2.10.12. 10. Программа с визуальным Windows-интерфейсом GUI

Интегрированная среда MATLAB предоставляет средства разработки программ с Windows-интерфейсом GUI — Graphical User Interface. Создаваемые с применением GUI программы называются Windows-приложениями или GUI-приложениями и используются при разработке прикладных программ.

Для создания GUI-приложений необходимо в основном окне среды MATLAB (рис. 2.1) выбрать позицию **New** и в раскрывающемся меню — GUI (рис. 2.78). В результате сначала появится запрос на выбор типа создаваемого программного приложения (рис. 2.80) — по умолчанию рекомендуется выбрать **Blank GUI (Default)**: пустое графическое окно, а затем откроется окно конструктора графического интерфейса **GUIDE — GUI DESIGNER** (рис. 2.81).

Такого же результата можно достичь командой **guide** в Командной строке Командного окна среды MATLAB (рис. 2.1). При этом, когда происходит выбор типа GUI-приложения (рис. 2.79), могут не только создаваться новые GUI-приложения — **Create New GUI**, но и открываться существующие GUI-приложения — **Open Existing GUI**.

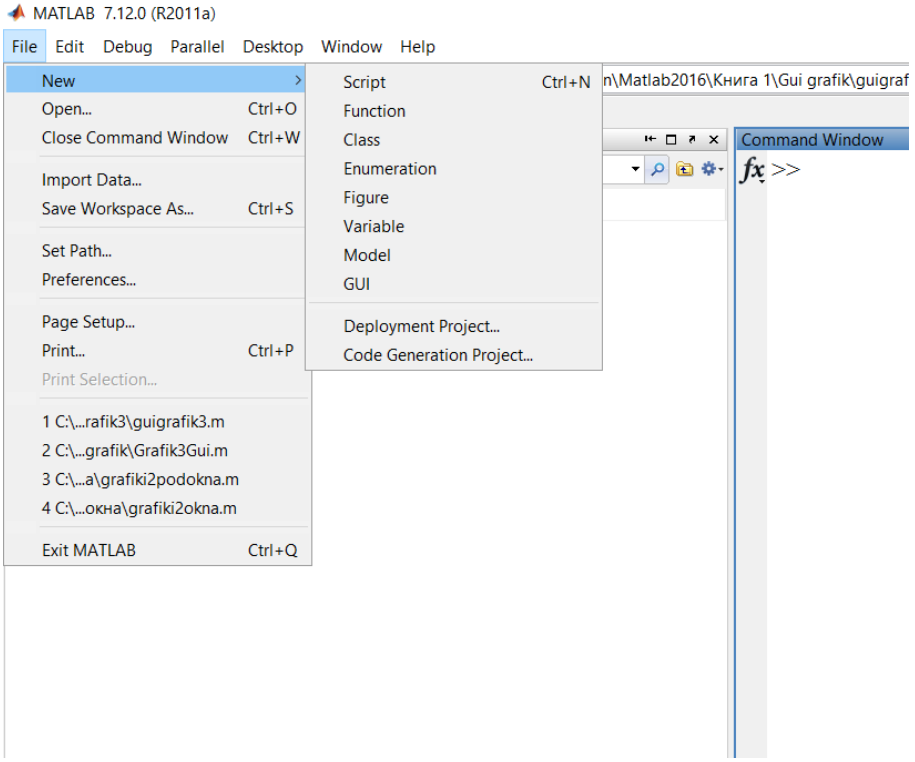


Рис. 2.79

Создание GUI-приложения из основного окна MATLAB

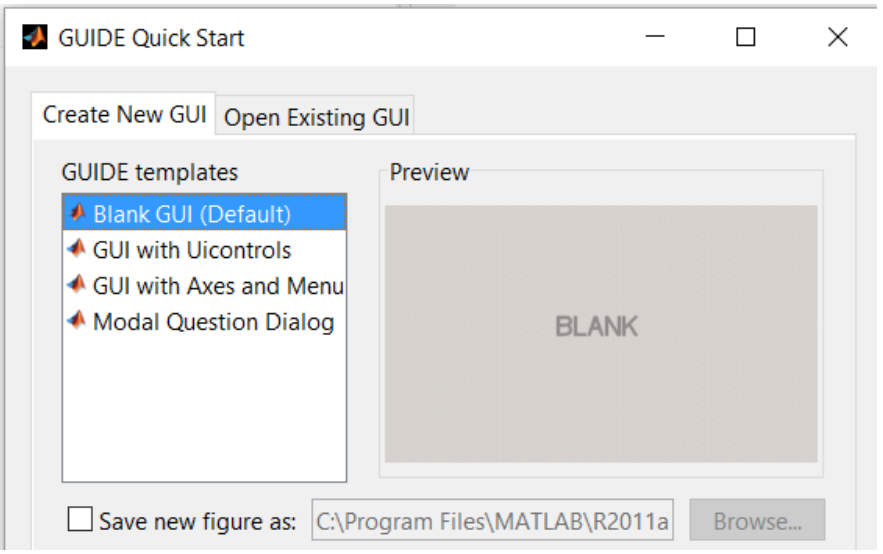


Рис. 2.80

Выбор типа создаваемого GUI-приложения

Перед разработкой GUI-приложений рекомендуется и в Текущей папке среды MATLAB открыть новую папку, которой будут храниться файлы с расширениями *.m и *.fig, создаваемые конструктором GUIDE. Так, например, после появления окна конструктора и файла с именем untitled.fig (рис. 2.81) следует командой меню Save As присвоить имя guigrafik3m, в результате чего во вновь созданной папке будут располагаться два файла (рис. 2.82):

guigrafik3m.fig

guigrafik3m.m

При разработке GUI-приложений файлы guigrafik3m.fig и guigrafik3m.m создаются автоматически, и любые изменения, производимые конструктором графического интерфейса, отражаются в программном коде *m*-файла guigrafik3m.m.

С применением конструктора графического интерфейса GUI разрабатываются и корректируются прикладные программы, для чего все необходимые действия выполняются в окне конструктора GUIDE (рис. 2.82). В этом окне следует обеспечить:

1) ввод исходной информации для работы программы;

2) вывод результатов работы программы, как в виде тестовой информации, так и в виде графиков;

3) в соответствующем *m*-файле, например guigrafik3m.m, следует поместить программный код, соответствующий выполнению необходимых действий, которые приводят к получению требуемых результатов.

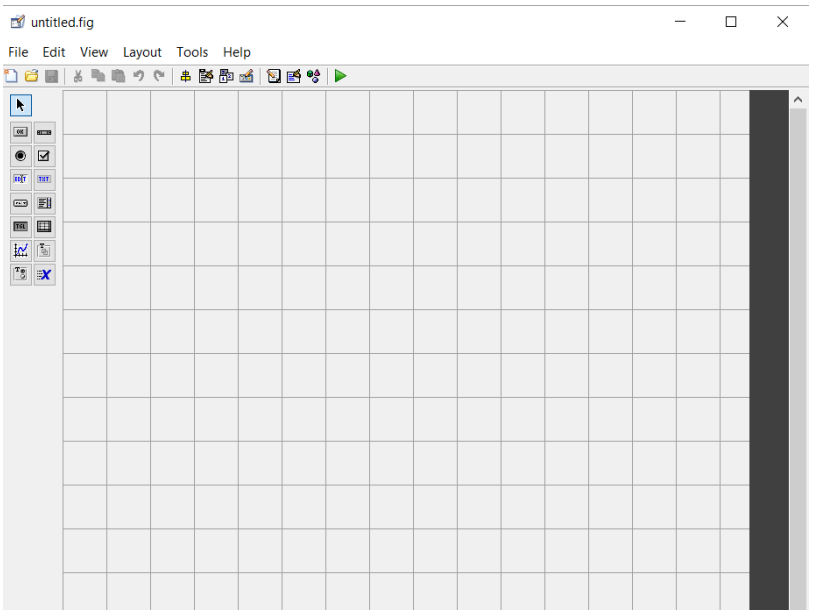


Рис. 2.81

Окно конструктора графического интерфейса с заготовкой приложения типа Blank GUI (Default)

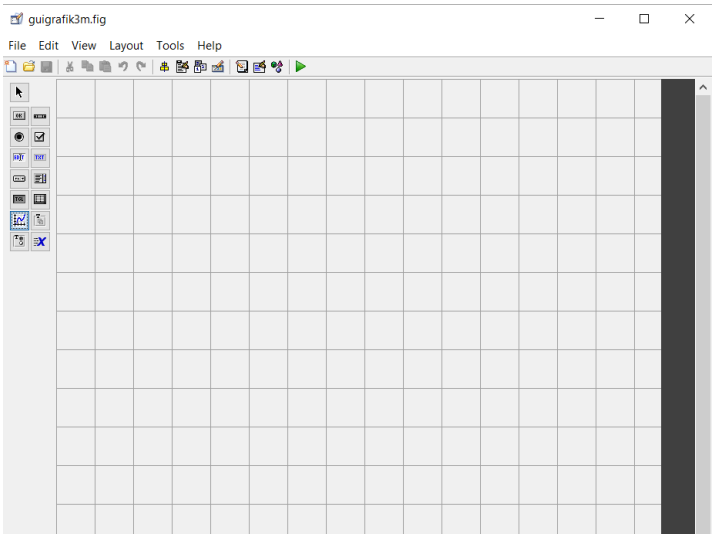


Рис. 2.82

Файлы guigrafik3m.fig и guigrafik3m.m в Текущей папке среды MATLAB при работе с конструктором графического интерфейса GUI

Конструктор графического интерфейса

Для работы с конструктором графического интерфейса (рис. 2.81) необходимо либо в меню выбора типа GUI-приложения выбрать тип приложения — в случае создаваемого файла **Blank GUI (Default)** при установленном приложении **Create New GUI** (рис. 2.80), либо выделить в Текущей папке файл с расширением *.fig, например guigrafik3m.fig, и правой кнопкой мыши щелкнуть в выпадающем меню позицию **Open in GUIDE** (рис. 2.83).

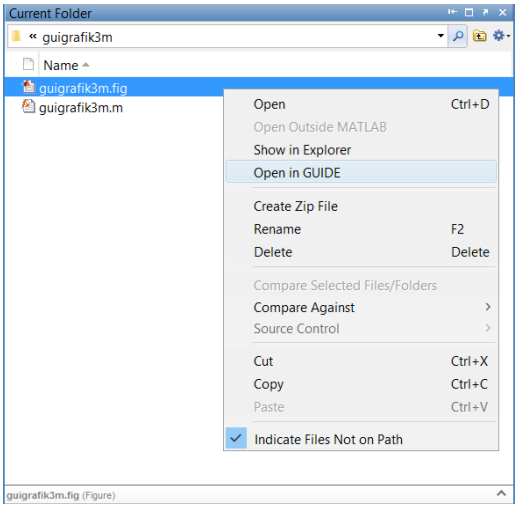


Рис. 2.83

Открывающееся меню с Open in GUIDE для работы с файлом guigrafik3m.fig в конструкторе графического интерфейса (рис. 2.82)

Окно конструктора GUIDE (рис. 2.81) состоит из меню двух панелей инструментов — вертикальной и горизонтальной, а также рабочей области, в которой и происходит создание интерфейса GUI-приложений.

В рабочей области размещаются отдельные **объекты** для ввода-вывода текстовой информации и графиков.

Горизонтальная панель инструментов содержит следующие пункты меню:

- File — для выполнения стандартных операций с файлами (New, Open, Close, Save, Print);
- Edit — позволяет выделять, копировать, дублировать, вырезать, удалять выделенные объекты, размещенные в графическом окне;
- View — позволяет управлять отображением в окне панелей инструментов, а также осуществлять переход к инспектору свойств (Property Inspector), обзору объектов (Object Browser), редактированию программного кода (Editor) и обработки событий (View Callbacks);
- Layout — позволяет управлять отображением вспомогательной сетки в графическом окне (Snap to Grid), а также перемещением отдельных объектов на задний (Back) или передний (Front) план;
- Tools — содержит вспомогательные инструменты настройки объектов, а также настройки конструктора GUIDE;
- Help — позволяет получать справочную информацию по использованию конструктора GUIDE для создания приложений.

Горизонтальная панель инструментов конструктора GUIDE (рис. 2.81) содержит следующие кнопки (слева направо):

- New Figure — позволяет создать новое приложение (графическое окно);
- Open Figure — позволяет открыть файл сохраненного ранее приложения;
- Save Figure — позволяет сохранить приложение;
- Cut — позволяет вырезать объект;
- Copy — позволяет копировать объект;
- Paste — позволяет вставить объект;
- Undo — позволяет отменить последнюю операцию;
- Redo — позволяет восстановить отмененную операцию;
- Align Objects — выполняет выравнивание объектов;
- Menu Editor — Редактор меню для приложения;
- Tab Order Editor — Редактор последовательности перехода между объектами;
- Toolbar Editor — Редактор панели инструментов;
- Editor — Редактор программного кода;
- Property Inspector — Инспектор свойств;

- Object Browser — Обзор объектов;
- Run — Запуск приложения на выполнение.

Некоторые из этих инструментов можно вызывать также с помощью пункта Tools в меню конструктора GUIDE.

Добавление объектов

Вертикальная панель инструментов конструктора GUIDE (рис. 2.81) позволяет добавлять к разрабатываемому приложению следующие объекты (сверху вниз):

- Select — Выделение объектов;
- Push Button — обычная Кнопка;
- Radio Button — Радиокнопка;
- Edit Text — поле для ввода и редактирования текста;
- Pop-up Menu — Всплывающее меню;
- Toggle Button — Кнопка-переключатель;
- Axes — Область построения графика;
- Button Group — Область размещения кнопок;
- Slider — Полоса прокрутки;
- Check Box — Флажок;
- Static Text — нередактируемый текст;
- Listbox — Список;
- Table — Таблица;
- Panel — Область для размещения объектов;
- ActiveX Control — Управление ActiveX.

Для добавления объекта к разрабатываемому приложению необходимо выбрать соответствующую кнопку на вертикальной панели инструментов конструктора GUIDE, затем нажать левой кнопкой мыши в том месте графического окна, где требуется разместить объект. Выбранный объект появится на поверхности окна с выделением квадратными маркерами. Наличие маркеров вокруг объекта (выделение объекта) означает, что доступные в данный момент операции будут применяться к данному выделенному объекту.

Изменение расположения объектов в графическом окне осуществляется с помощью мыши путем перетаскивания. Изменение размеров объектов производится перетаскиванием в нужном направлении маркеров, выделяющих объект.

Например, при создании трехмерного графика guigrafik3m на вертикальной панели инструментов выбирается кнопка с изображением графика (рис. 2.84) и однократным щелчком левой кнопки мыши график размещается в рабочей области конструктора GUIDE. После растаскивания объекта мышью (в данном случае пустого графика) и увеличения его размеров на всю рабочую область получается окно конструктора GUIDE в виде, изображенном на рисунке 2.85.

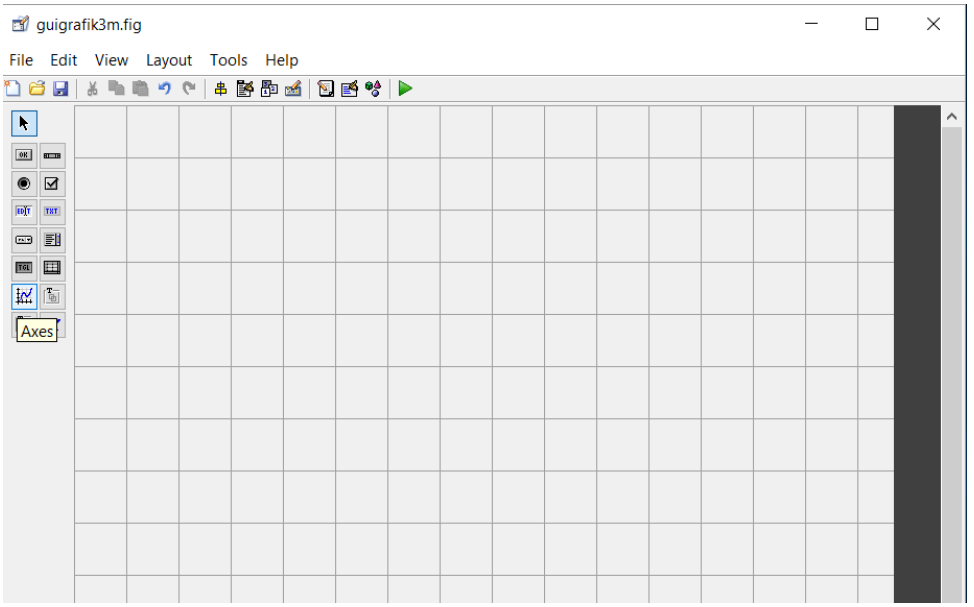


Рис. 2.84

Выбор кнопки Графика (Axes) на вертикальной панели инструментов конструктора GUIDE

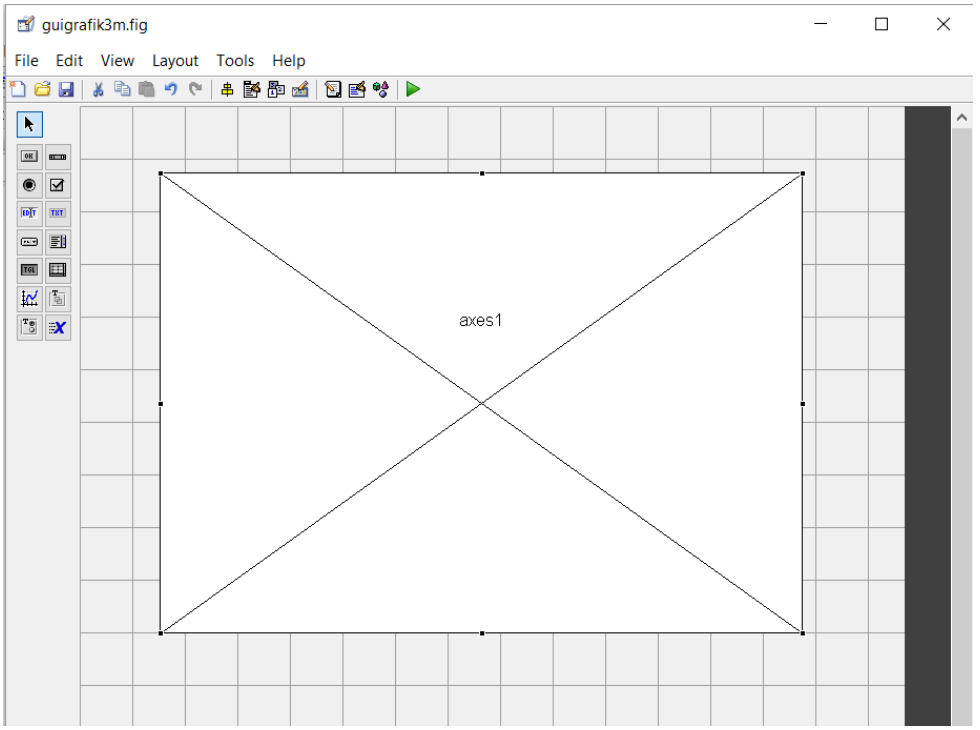


Рис. 2.85

Размещение кнопки Графика в рабочей области конструктора GUIDE

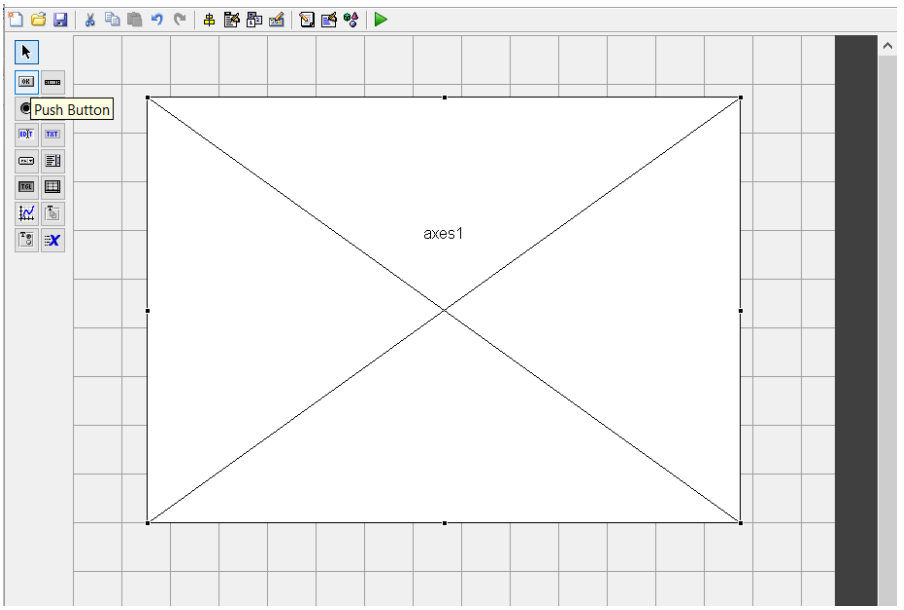


Рис. 2.86

Выбор кнопки Push Button в вертикальной панели инструментов конструктора GUIDE

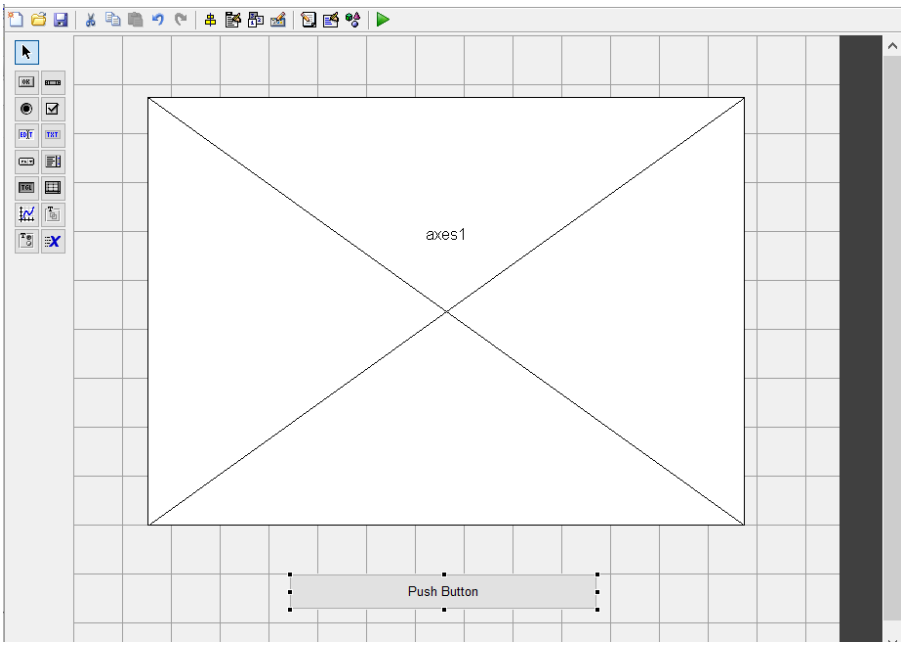


Рис. 2.87

Размещение кнопки Push Button в вертикальной панели инструментов конструктора GUIDE

Следующим объектом, размещаемым в рабочей области и необходимым для пуска программы построения графика, должна быть кнопка Push Button на вертикальной панели инструментов (рис. 2.86). После ее размещения в рабочей

области и увеличения размеров конструктор GUIDE принимает вид, представленный на рисунке 2.87.

В результате рабочая область на рисунке 2.87 содержит два объекта:

- Push Button — для запуска программы построения трехмерного графика;
- Axes1 — для отображения результатов работы в графическом виде.

Процедура построения графика на *m*-языке должна быть специальным образом размещена в сгенерированном файле guigrafik3m.m.

Редактирование свойств объектов

Для получения доступа ко всем свойствам выбранного объекта используется Инспектор свойств (Property Inspector) — рисунки 2.88 и 2.89, открываемый одним из перечисленных ниже способов.

1. Выполнением двойного щелчка левой клавишей мыши на изображении объекта в окне редактируемого приложения.
2. Выделением изображения объекта правой кнопкой мыши и выбором в открывшемся меню пункта Property Inspector (рис. 2.88).
3. Выбором в меню конструктора GUIDE пункта View → Property Inspector.
4. Нажатием на кнопку Property Inspector на горизонтальной панели инструментов конструктора GUIDE.

В результате любого из этих действий появится окно (рис. 2.88), в котором в виде таблицы будут перечислены все свойства и их текущие значения для выбранного объекта (в левом столбце отображаются свойства, а в правом столбце — их значения). Значения свойств можно редактировать, соблюдая допустимый формат данных.

Различные объекты имеют разные наборы свойств. Некоторые свойства присущи всем объектам:

- Tag — Идентификатор объекта (Имя переменной);
- Style — Тип объекта;
- Position — Координаты и размеры объекта;
- String — Текст на объекте;
- BackgroundColor — Цвет фона объекта;
- FontName — Название шрифта;
- FontSize — Размер шрифта;
- Value — Текущее значение;
- Min — Минимальное значение;
- Max — Максимальное значение.

Например, текст для кнопки Push Button — «Построить график» (рис. 2.89) хранится в ее свойстве String. Положение и размер кнопки можно изменять как с помощью мыши, так и изменяя значения свойства Position. Кроме того, свойства можно изменять в программном коде при обработке событий с различными объектами.

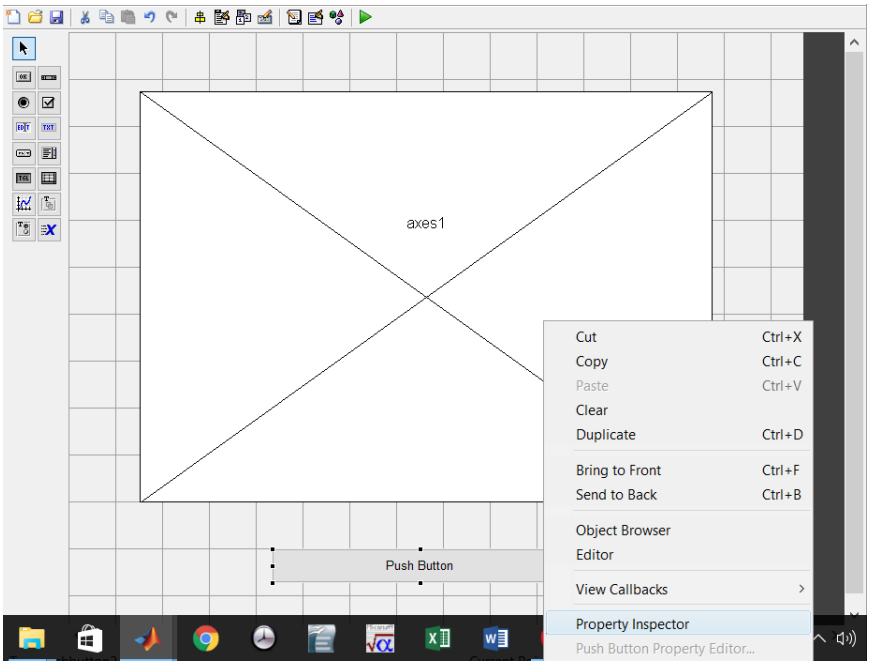


Рис. 2.88

Выбор Редактора свойств — Property Inspector при нажатии на объект в окне GUIDE правой кнопкой мыши

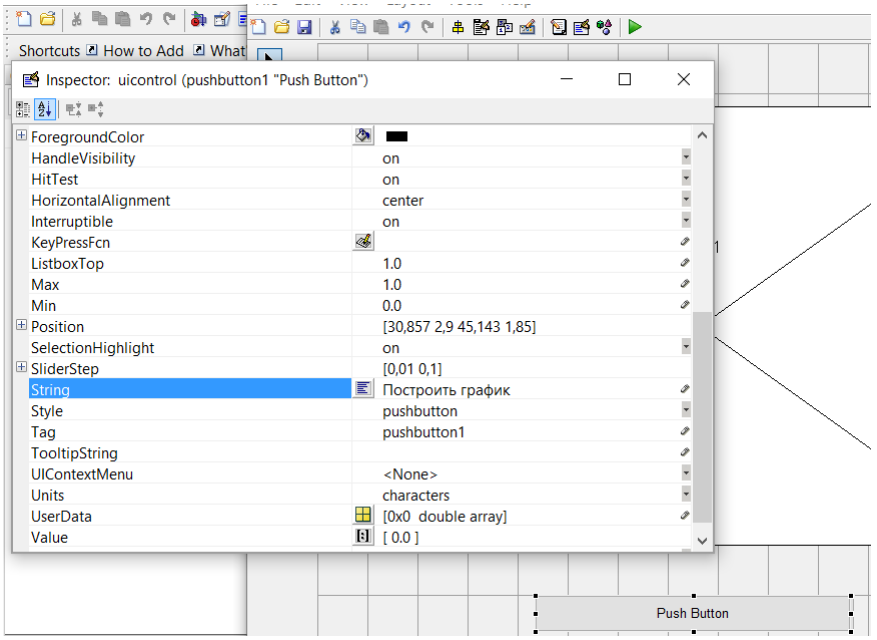


Рис. 2.89

Изменение названия объекта в строке String (Property Inspector) — построить график.

Программная обработка событий с объектами

Каждому объекту в GUI-приложении соответствует не только определенный набор свойств, но и определенный набор событий, которые могут происходить с этим объектом. Например, для объекта Push Button (в рассматриваемом примере — Построить график) таким событием и выполняемым действием является построение трехмерного графика.

Список доступных объектов для выделенного в конструкторе GUIDE объекта можно увидеть, выбрав в меню конструктора пункт View → View Callback или пункт View Callback в контекстном меню объекта, которое появляется после однократного щелчка правой клавишей мыши на объекте.

Основные события, доступные для большинства объектов в конструкторе GUIDE:

- Callback — основная функция объекта (зависит от типа объекта);
- KeyPressFcn — нажата клавиша на клавиатуре;
- CreateFcn — создание объекта;
- DeleteFcn — удаление объекта;
- OpenFcn — открытие графического окна (только для объекта Figure).

При первом сохранении разрабатываемого GUI-приложения автоматически генерируется программный код, описывающий создание и расположение добавленных к приложению в конструкторе объектов. Программный код обновляется при каждом сохранении приложения. Информация хранится в виде двух файлов: *m*-файл содержит программный код, fig-файл — визуальное представление графического окна вместе со всеми расположенными в нем объектами. Однако действия, которые должна выполнять программа в случае наступления того или иного события с объектами, пишутся разработчиком приложения.

Для перехода к написанию или редактированию программного кода, который должен выполняться при наступлении определенных событий, необходимо выбрать соответствующее событие в Инспекторе свойств или в раскрывающемся меню View Callbacks → Callback. При этом среда MATLAB автоматически находит в связанном с приложением *m*-файле подфункцию, описывающую данное событие. Если такая подфункция отсутствует, то ее заготовка автоматически создается в *m*-файле.

Например, для события «нажатие на кнопку», соответствующего основной функции (Callback) объекта Push Button, имеющего в качестве идентификатора (Tag) значение имени pushbutton1 (рис. 2.89), создается заготовка подфункции в соответствующем *m*-файле — guigrafik3m.m:

```
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback (hObject, eventdata, handles)
% hObject      handle to pushbutton1   (see GCBO)
% eventdata reserved — to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)
```

Заготовка содержит только заголовок подфункции со списком исходных параметров. Остальной текст — комментарии, поясняющие назначение подфункции (button press — нажатие на кнопку) и смысл исходных параметров. В данной подфункции идентификатором hObject обозначена сама кнопка pushbutton1, с которой связано рассматриваемое событие. Параметр handles содержит информацию о структуре всех объектов в разрабатываемом приложении и позволяет обращаться к свойствам других объектов приложения. Параметр eventdata зарезервирован для использования в дальнейших версиях MATLAB.

Заготовки подфункций для других событий имеют аналогичную структуру, но могут отличаться друг от друга в зависимости от типа объекта.

Заготовки для основной функции “Callback” каждого из объектов, используемых в приложении, создаются в *m*-файле автоматически.

2.10.12.1. Построение трехмерного графика

Для построения трехмерного графика для конкретной функции:

$$z = \left(\frac{x-3}{100}\right)^2 - (y-x) + e^{20(y-x)}$$

в заготовку подфункции pushbutton1_Callback (hObject, eventdata, handles) необходимо включить программный код следующих операторов:

```
[x,y]=meshgrid(2.9:0.01:3.1,2.8:0.01:2.9);
z=(1/100*(x-3)).^2-(y-x)+exp((y-x)*20);
mesh(x,y,z);
title('z=(1/100*(x-3))^2-(y-x)+exp((y-x)*20)');
xlabel('x');
ylabel('y');
zlabel('z');
```

С этой целью выделяется кнопка «Построить график» и при нажатии на правую кнопку мыши выбираются пункты меню View Callbacks → Callback (рис. 2.90). В результате на дисплее появится программный код *m*-файла guigrafik3m.m (рис. 2.91), в конец которого в подфункцию pushbutton1_Callback (hObject, eventdata, handles) следует записать или скопировать семь операторов для построения трехмерного графика. Программный код файла guigrafik3m.m построения графика с применением GUI-интерфейса представлен на рисунке 2.91.

Запуск программы построения графика может быть осуществлен:

— из конструктора GUIDE путем нажатия на кнопку Run (Пуск, рис. 2.88);

— из командной строки командного окна путем записи в нее названия файла guigrafik3m.m;

— после того как в текущей папке выделен файл guigrafik3m.m и нажатия на правую кнопку мыши в выпадающем меню на Run.

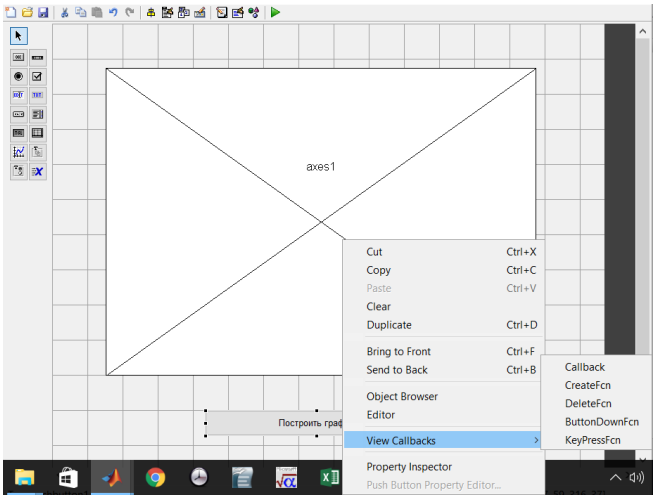


Рис. 2.90

Выбор пунктов меню View Callbacks → Callback для отображения программного кода *m*-файла guigrafik3m.m

```
function varargout = guigrafik3m(varargin)
% GUIGRAFIK3M MATLAB code for guigrafik3m.fig
%      GUIGRAFIK3M, by itself, creates a new GUIGRAFIK3M or raises the existing
%      singleton*.
%
%      H = GUIGRAFIK3M returns the handle to a new GUIGRAFIK3M or the handle to
%      the existing singleton*.
%
%      GUIGRAFIK3M('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in GUIGRAFIK3M.M with the given input arguments.
%
%      GUIGRAFIK3M('Property','Value',...) creates a new GUIGRAFIK3M or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before guigrafik3m_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to guigrafik3m_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help guigrafik3m

% Last Modified by GUIDE v2.5 04-Feb-2016 20:59:44

% --- Executes just before guigrafik3m is made visible.
function guigrafik3m_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to guigrafik3m (see VARARGIN)

% Choose default command line output for guigrafik3m
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes guigrafik3m wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = guigrafik3m_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

Рис. 2.91 (начало)

Программный код файла guigrafik3m.m для построения трехмерного графика с применением GUI-интерфейса

```
% Begin initialization code — DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @guigrafik3m_OpeningFcn, ...
                   'gui_OutputFcn',  @guigrafik3m_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code — DO NOT EDIT
```

```
% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved — to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[x,y]=meshgrid(2.9:0.01:3.1,2.8:0.01:2.9);
z=(1/100*(x-3)).^2-(y-x)+exp((y-x)*20);
mesh(x,y,z);
title('z=(1/100*(x-3))^2-(y-x)+exp((y-x)*20)');
xlabel('x');
ylabel('y');
zlabel('z');
```

Рис. 2.91 (окончание)

Программный код файла guigrafik3m.m для построения трехмерного графика с применением GUI-интерфейса

Результат запуска программы, получаемый на дисплее компьютера, представлен на рисунке 2.92. Для выполнения программы необходимо щелкнуть левой кнопкой мыши на кнопке «Построить график», после чего на дисплей выводится трехмерный график, изображенный на рисунке 2.93.

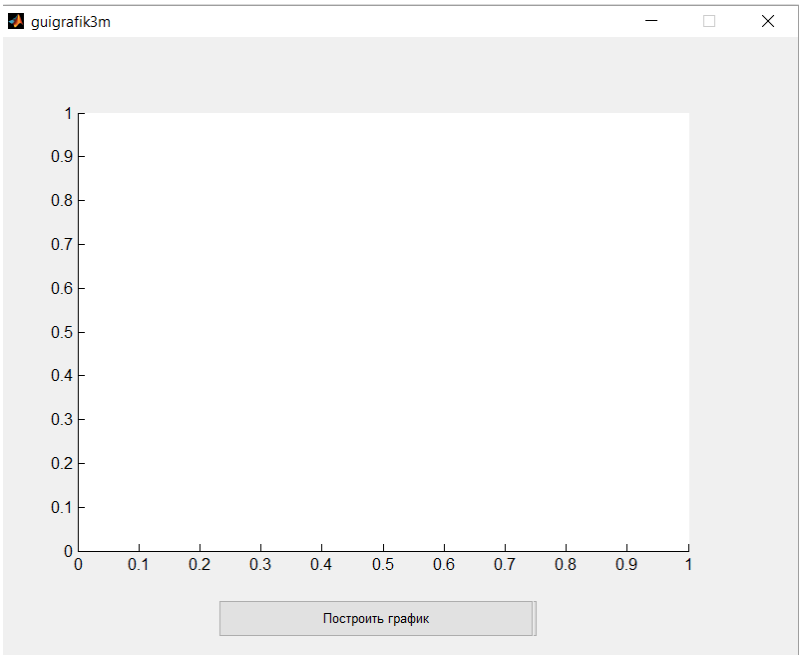


Рис. 2.92

Результат запуска программы построения трехмерного графика, для выполнения программы следует щелкнуть левой кнопкой мыши на кнопке «Построить график»

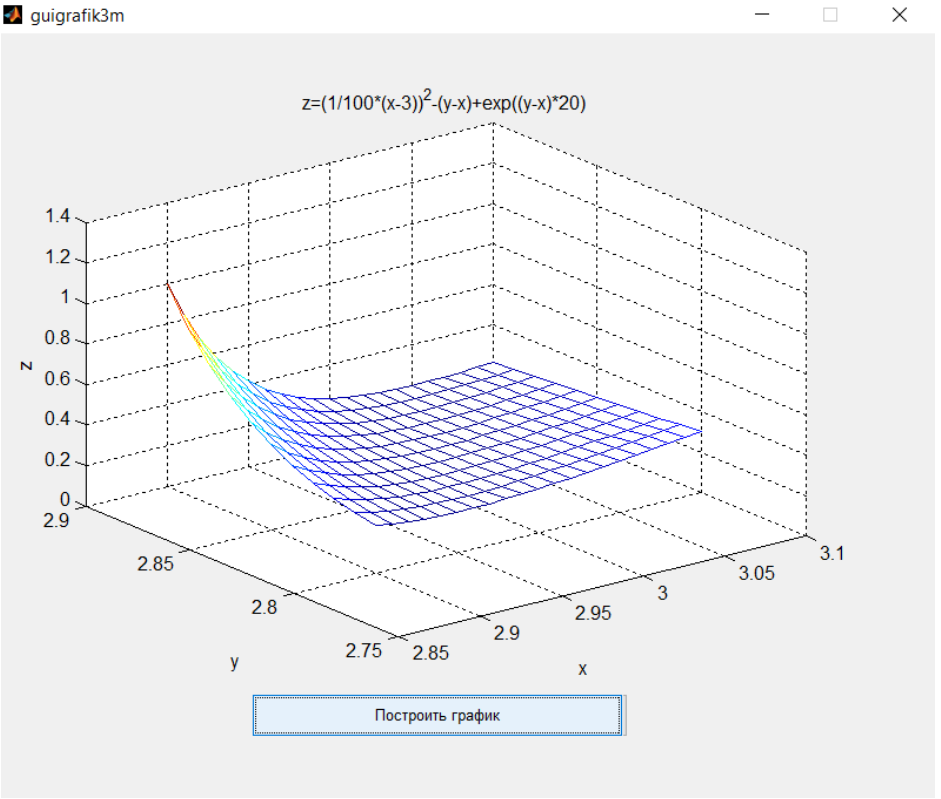


Рис. 2.93

Результат выполнения программы построения трехмерного графика после нажатия на кнопку «Построить график»

2.10.12.2. Внесение изменений в программы, созданные с использованием конструктора GUIDE

Программы, разработанные с применением Windows-интерфейса (Graphical User Interface — GUI-интерфейс), могут изменяться с использованием конструктора GUIDE.

При создании таких программ генерируются два взаимосвязанных файла с одинаковым именем: *m*-файл и файл с расширением fig. В рассматриваемом случае при построении трехмерного графика это файлы guigrafik3m.m и guigrafik3m.fig (рис. 2.83).

Для внесения изменений в графическом окне конструктора GUIDE и/или в программном коде *m*-файла необходимо выделить файл с расширением fig (guigrafik3m.fig) и правой кнопкой мыши в выпадающем меню выбрать пункт Open in GUIDE (рис. 2.83).

В результате на дисплее появится графическое окно конструктора GUIDE (рис. 2.88 — выпадающего контекстного меню не будет), которое содержит два объекта: верхний (axis) — это пространство для изображения графика, и ниж-

ний (Push Button1) — кнопка для запуска программы действий, описанной в соответствующем *m*-файле после заголовка подфункции:

function pushbutton1_Callback (hObject, eventdata, handles).

В общем случае при проведении расчетов следует обратить внимание на три типа объектов (четвертый тип объекта — график — рассмотрен выше):

1) статический текст с именами text1, text2 для задания текстовой информации;

2) текстовое поле с именами edit1, edit2 для ввода и вывода числовой информации;

3) кнопка с именами pushbutton1, pushbutton2 для запуска программы.

С объектами в окне конструктора GUIDE, после того как они выделены левой кнопкой мыши, можно выполнять следующие манипуляции.

1. Передвигать и изменять их размеры.

2. Нажав на правую кнопку мыши, выбрать пункт меню Property Inspector (Инспектор свойств) и просмотреть свойства объекта, а при желании и изменить их. Прежде всего это относится к Имени (Tag) и к свойству Строка (String). Например, в рассматриваемом примере (рис. 2.89) Строка (String) в свойстве Кнопка (Push Button) заменена на строку «Построить график». Цифры в именах (text, edit и pushbutton) можно при желании заменить на другие. При этом все изменения автоматически отразятся в программном коде *m*-файла программы — guigrafik3m.m.

3. Нажав на правую кнопку мыши и выбрав пункт контекстного меню View Callbacks → Callback, на дисплее можно будет увидеть фрагмент программного кода *m*-файла, относящийся к данному объекту. Тут же могут быть произведены коррекции программного кода, которые автоматически отобразятся в программном коде *m*-файла программы.

Особенно важна такая манипуляция для объекта Кнопка (Push Button), так как программные коды реализуемого вычислительного процесса, как правило, записываются в *m*-файле после заголовка соответствующей подфункции, например:

function pushbutton1_Callback (hObject, eventdata, handles).

2.10.12.2.1. Внесение изменений в программу guigrafik3m

В программу guigrafik3m.m, созданную GUIDE-конструктором, необходимо внести два изменения:

1) установить еще одну кнопку в окне программы с визуальным Windows-интерфейсом для трехмерного графика с целью обеспечения возможности очистки области построения графика;

2) в визуальном Windows-интерфейсе программы написать заголовок «Построение трехмерного графика».

Для решения первой задачи в графическом окне конструктора GUIDE программы guigrafik3m.m (рис. 2.88) размещается вторая кнопка Push Button, с помощью Инспектора свойств (Property Inspector) изменяется ее имя на «Очистить область построения графика» (рис. 2.94).

Чтобы произвести изменения в *m*-файле программы для выполнения соответствующего действия, необходимо выделить кнопку «Очистить область построения графика» в графическом окне и нажатием на правую кнопку мыши выделить пункты меню View Callbacks → Callback (рис. 2.95), после чего станет доступен для редактирования *m*-файл программы (рис. 2.91) с новой подфункцией:

function pushbutton2_Callback (hObject, eventdata, handles)

(рис. 2.96), образовавшейся при включении второй кнопки (pushbutton2) в визуальный Windows-интерфейс.

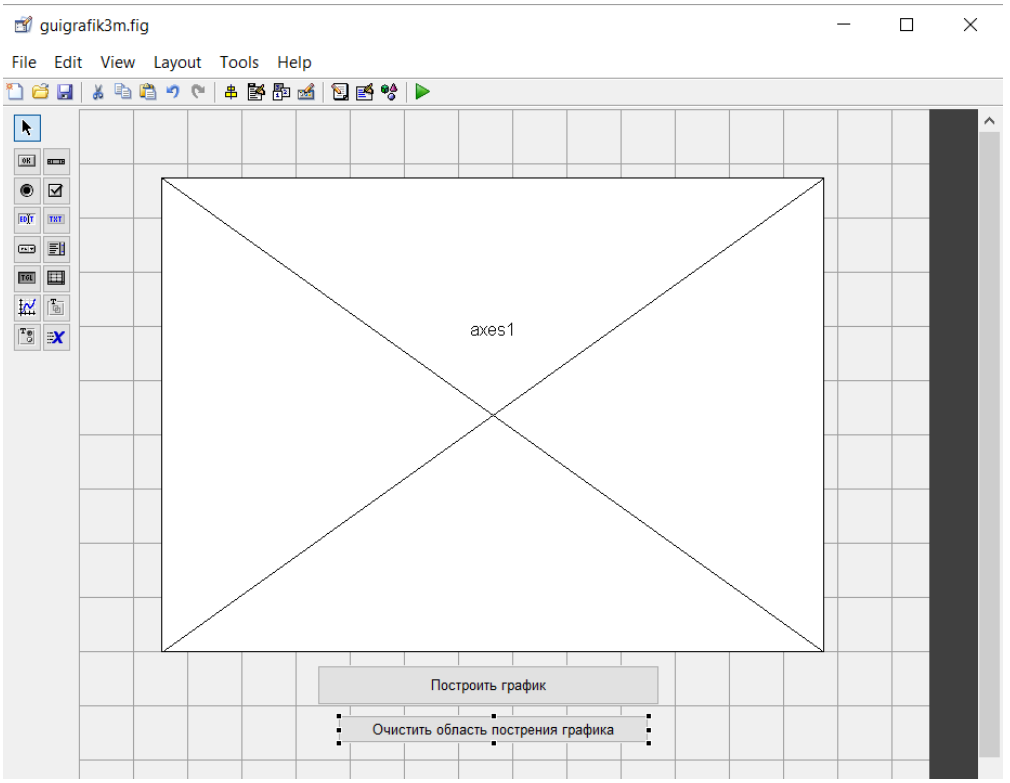


Рис. 2.94

Размещение дополнительной кнопки в окне Windows-интерфейса
с именем «Очистить область построения графика»

Для очистки области построения графика в подфункцию pushbutton2_Callback включаются два оператора (рис. 2.95):

cla — очистка области от графика;

grid off — очистка области от сетки на графике.

Для решения второй задачи — создание заголовка визуального Windows-интерфейса программы guigrafik3m.m — в конструкторе GUIDE на вертикаль-

ной панели инструментов (рис. 2.97) выбирается кнопка Статический текст (Static Text) (рис. 2.97) и перетаскивается на место заголовка.

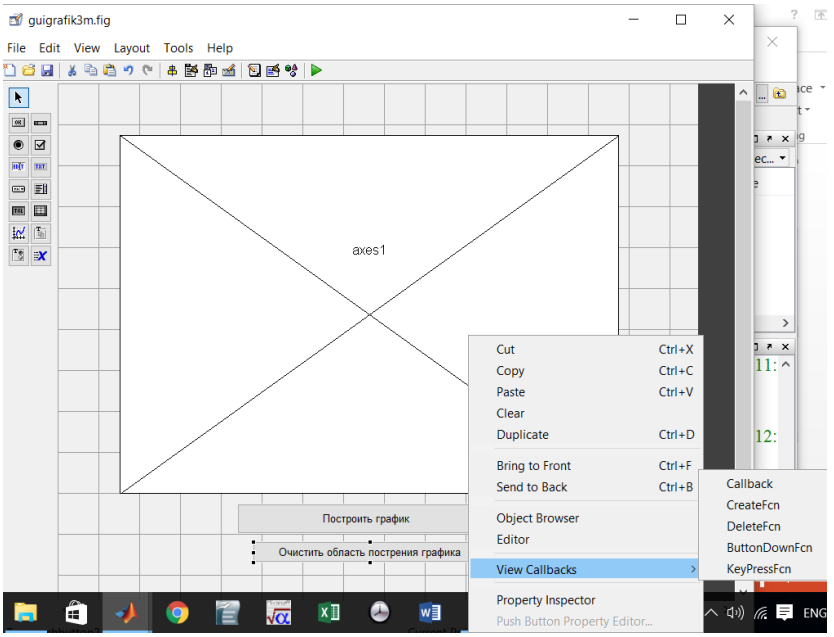


Рис. 2.95

Процедура вызова фрагмента программного кода *m*-файла для коррекции View Callbacks → Callback с целью выполнения действий в соответствии с командами программы при нажатии кнопки «Очистить область построения графика»

```
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[x,y]=meshgrid(2.9:0.01:3.1,2.8:0.01:2.9);
z=(1/100*(x-3)).^2-(y-x)+exp((y-x)*20);
mesh(x,y,z);
title('z=(1/100*(x-3))^2-(y-x)+exp((y-x)*20)');
xlabel('x');
ylabel('y');
zlabel('z');
% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
cla;
grid off;
```

Рис. 2.96

Откорректированный *m*-файл с командами ‘cla’, ‘grid off’
(после function pushbutton2_Callback (hObject, eventdata, handles))
для очистки области построения графика

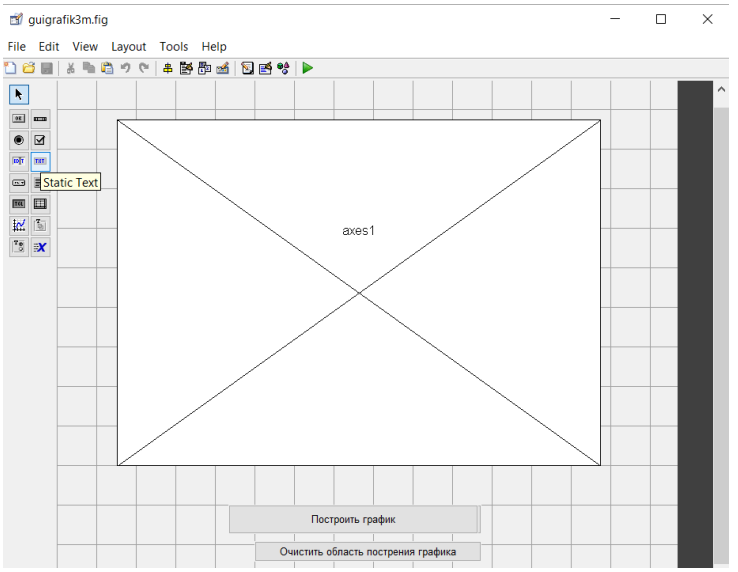


Рис. 2.97

Выбор кнопки Статический текст (Static Text) записи заголовка визуального Windows-интерфейса

Далее, после ее выделения, с использованием Инспектора свойств (Property Inspector) в Строке (String) изменяется имя (вместо Static Text — Построение трехмерного графика) (рис. 2.98).

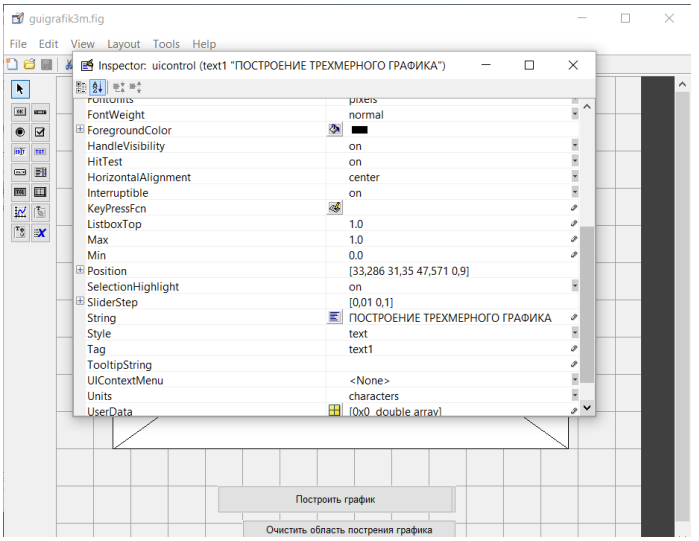


Рис. 2.98

Редактирование Строки (String) Статического текста (Static Text) с использованием Инспектора свойств (Property Inspector)

Результат работы программы guigrafik3m.m после нажатия кнопки «Построить график» представлен на рисунке 2.99.

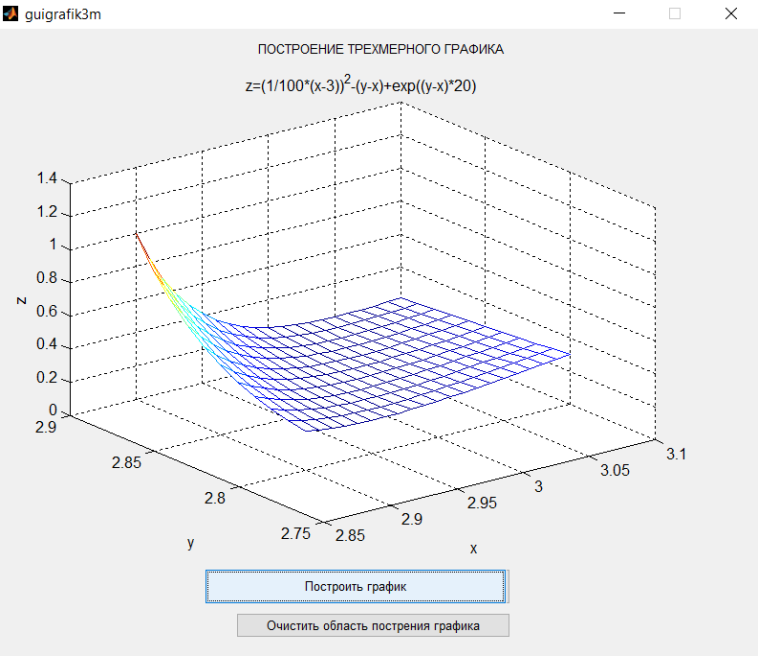


Рис. 2.99

Результат запуска программы guigrafik3m.m после нажатия на кнопку «Построить график»

При очистке области построения графика с помощью соответствующей кнопки (рис. 2.99) целесообразно изменить имя заголовка визуального Windows-интерфейса: вместо «ПОСТРОЕНИЕ ТРЕХМЕРНОГО ГРАФИКА» (рис. 2.99) записать «ОБЛАСТЬ ТРЕХМЕРНОГО ГРАФИКА ОЧИЩЕНА».

Так как запись «ПОСТРОЕНИЕ ТРЕХМЕРНОГО ГРАФИКА» соответствует Статическому тексту (Static Text) с Именем (Tag) “text1” (рис. 2.98), то в подфункцию pushbutton2_Callback следует записать оператор (рис. 2.100):

set (hadles.text1, ‘string’, ‘ОБЛАСТЬ ТРЕХМЕРНОГО ГРАФИКА ОЧИЩЕНА’)

Этот оператор всегда будет выполняться при нажатии второй кнопки, действующей в соответствии с операторами, описанными в подфункции pushbutton2_Callback.

Для того чтобы при нажатии на первую кнопку в заголовке визуального интерфейса всегда появлялась надпись «ПОСТРОЕНИЕ ТРЕХМЕРНОГО ГРАФИКА», в подфункцию pushbutton1_Callback следует записать оператор (рис. 2.100):

set (hadles.text1, ‘string’, ‘ПОСТРОЕНИЕ ТРЕХМЕРНОГО ГРАФИКА’).

На рисунке 2.101 изображен результат работы программы guigrafik3m.m, когда после построения графика (рис. 2.99) была нажата кнопка «Очистить область построения графика». Как видно из рисунка 2.101, заголовок Windows-интерфейса сменился на «ОБЛАСТЬ ТРЕХМЕРНОГО ГРАФИКА ОЧИЩЕНА».

```
74      % --- Executes on button press in pushbutton1.
75      function pushbutton1_Callback(hObject, eventdata, handles)
76      % hObject    handle to pushbutton1 (see GCBO)
77      % eventdata  reserved - to be defined in a future version of MATLAB
78      % handles    structure with handles and user data (see GUIDATA)
79 -    set(handles.text1,'String','ПОСТРОЕНИЕ ТРЕХМЕРНОГО ГРАФИКА');
80 -    [x,y]=meshgrid(2.9:0.01:3.1,2.8:0.01:2.9);
81 -    z=(1/100*(x-3)).^2-(y-x)+exp((y-x)*20);
82 -    mesh(x,y,z);
83 -    title('z=(1/100*(x-3))^2-(y-x)+exp((y-x)*20)');
84 -    xlabel('x');
85 -    ylabel('y');
86 -    zlabel('z');
87      % --- Executes on button press in pushbutton2.
88      function pushbutton2_Callback(hObject, eventdata, handles)
89      % hObject    handle to pushbutton2 (see GCBO)
90      % eventdata  reserved - to be defined in a future version of MATLAB
91      % handles    structure with handles and user data (see GUIDATA)
92 -    cla;
93 -    grid off;
94 -    set(handles.text1,'String','ОБЛАСТЬ ТРЕХМЕРНОГО ГРАФИКА ОЧИЩЕНА');
```

Рис. 2.100

Коррекции, вносимые в программный код *m*-файла guigrafik3m.m
для получения возможности изменения заголовка визуального Windows-интерфейса

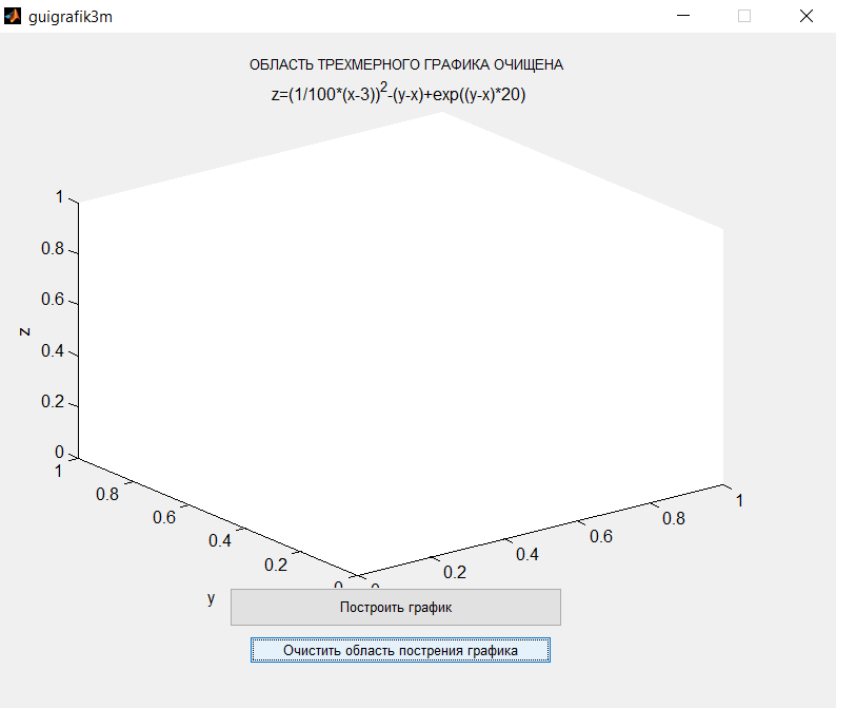


Рис. 2.101

Результат работы программы guigrafik3m.m после нажатия кнопки
«Очистить область построения графика»

2.10.12.3. Выполнение элементарных вычислений

Для решения этой задачи в визуальном Windows-интерфейсе следует разместить:

1) окна редактирования с именами edit1, edit2, … editN для ввода числовой исходной информации для расчетов и вывода результатов;

2) текстовые окна с именами text1, text2, … textN для ввода и вывода текстовой информации, в том числе и о результатах расчетов;

3) окно построения графика с именем axis1 для графического изображения функции квадратного уравнения;

4) окно для запуска процесса решения квадратного уравнения с именем pushbutton1.

Для ввода числовой исходной информации требуется три окна: для коэффициентов *a*, *b* и *c*, для размещения которых конструктором GUIDE в поле визуального Windows-интерфейса необходимо перетащить кнопку Edit Text (Редактировать текст) из вертикальной панели инструментов, разместив ее в желаемом месте Windows-интерфейса (рис. 2.102 и 2.103).

Текстовое поле Edit Text (Редактировать текст) позволяет пользователю вводить и изменять текстовую информацию. Введенная информация хранится в свойстве String объекта. Чтобы получить эту информацию, используется функция ‘get’:

<переменная> = get (<идентификатор объекта>, <свойство>).

<Идентификатор объекта> указывает, к свойству какого объекта обращается пользователь. Название <свойства> записывается в строковом виде, например ‘string’. Полученное в результате применения функции ‘get’ значение свойства объекта помещается в <переменную>, что обеспечивает возможность его дальнейшего использования.

Если обращение к свойству объекта выполняется в подфункции, связанной с событием этого объекта, то в качестве <идентификатора объекта> используют hObject. Обращение к свойству объекта из подфункции, связанной с событием в другом объекте, выполняется через handles. Например, чтобы в подфункции нажатия на кнопку получить значение свойства ‘String’ текстового поля Edit Text (Редактировать текст), имеющего идентификатор edit1, необходимо записать:

s = get (handles.edit1, ‘String’).

В результате переменной s будет присвоено значение, хранящееся в свойстве String объекта edit1. Полученное значение может быть использовано для изменения свойств других объектов, расположенных в графическом окне. Это позволяет многократно вводить данные и получать результаты, не прибегая к перезапуску приложения.

После коррекции левой кнопкой мыши размеров окна Редактирование текста (Edit Text), нажав на правую кнопку мыши, следует выбрать из выпадающего меню пункт Инспектор свойств (Property Inspector) и очистить поле

String от текста edit1 (рис. 2.103), и поле ввода/вывода чисел станет пустым (рис. 2.105 и 2.106).

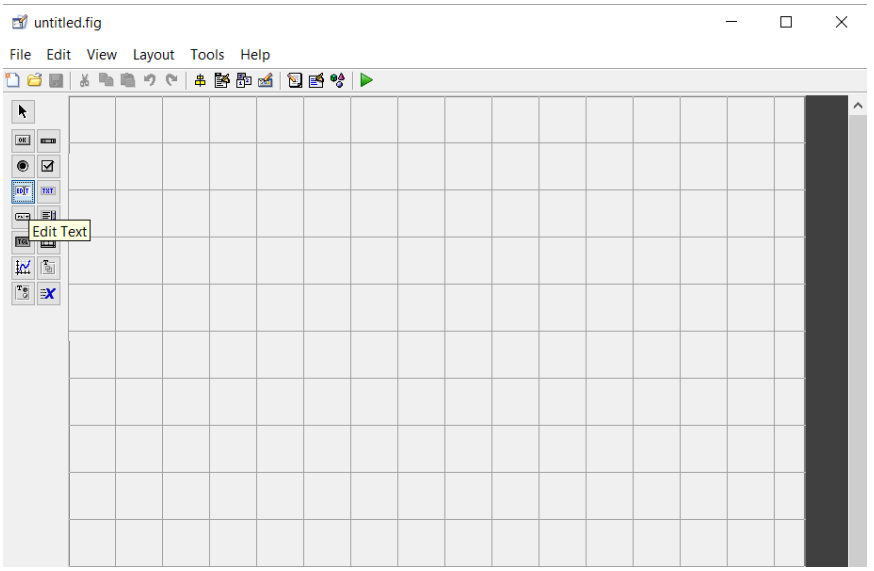


Рис. 2.102

Выбор кнопки Редактирование текста (Edit Text) на вертикальной панели инструментов конструктора GUIDE

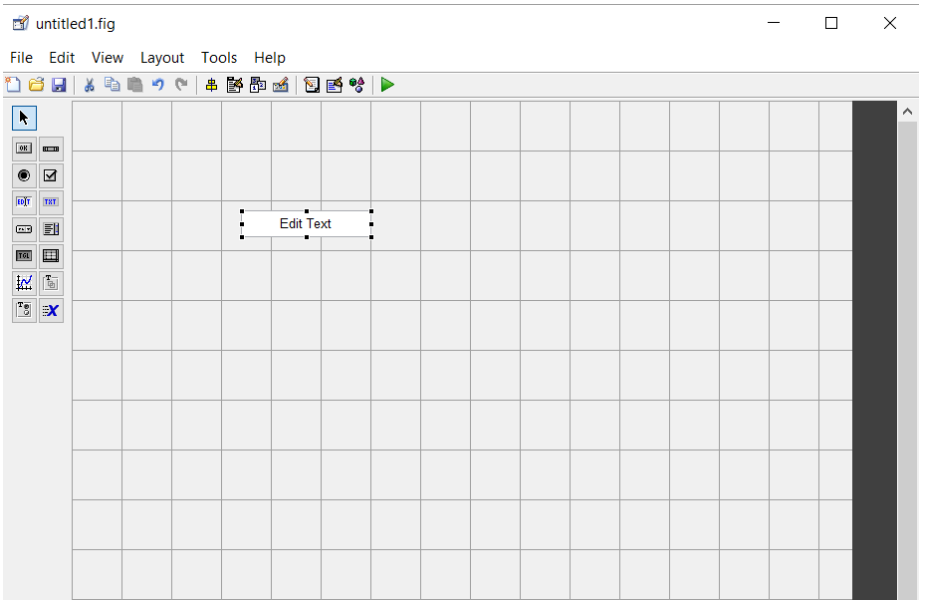


Рис. 2.103

Размещение окна Редактирование текста (Edit Text)
в графическом окне Windows-интерфейса

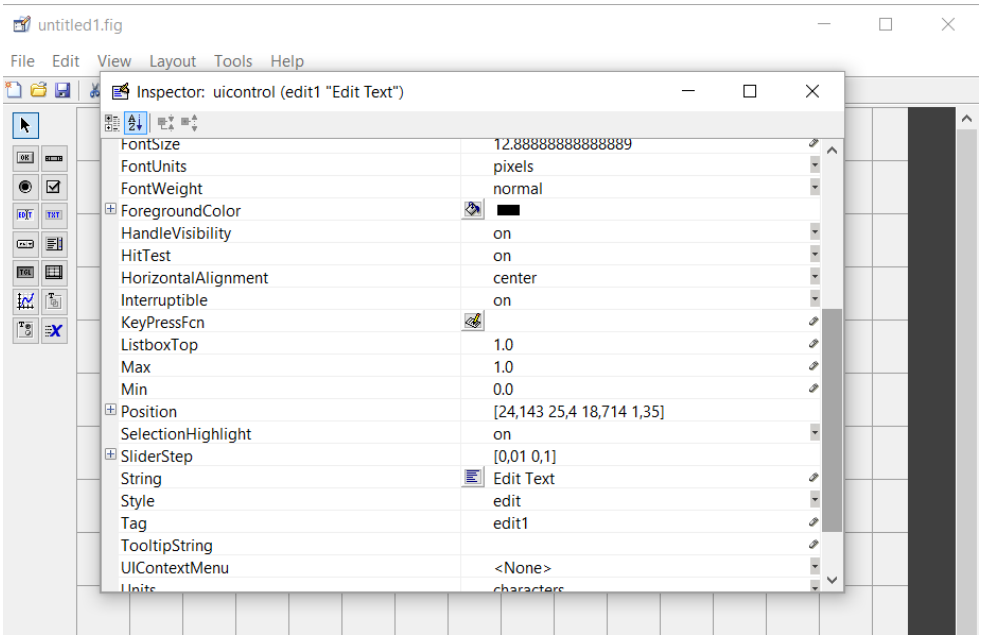


Рис. 2.104

Изменение текста (Edit Text) в окне редактируемого текста с помощью Инспектора свойств (Property Inspector) в строке String

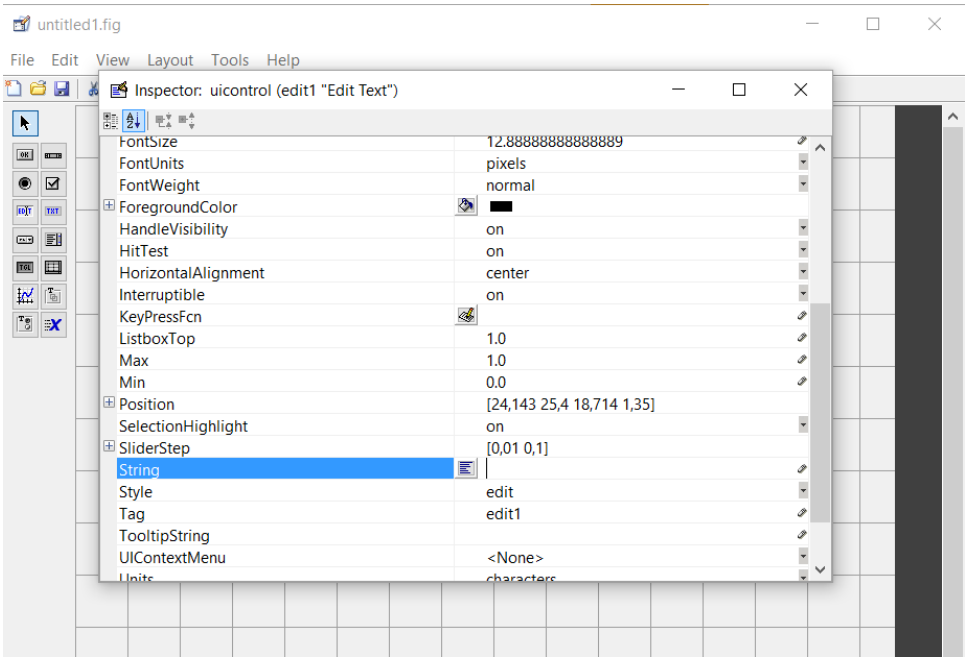


Рис. 2.105

Очистка строк Инспектором свойств (Property Inspector) в строке String

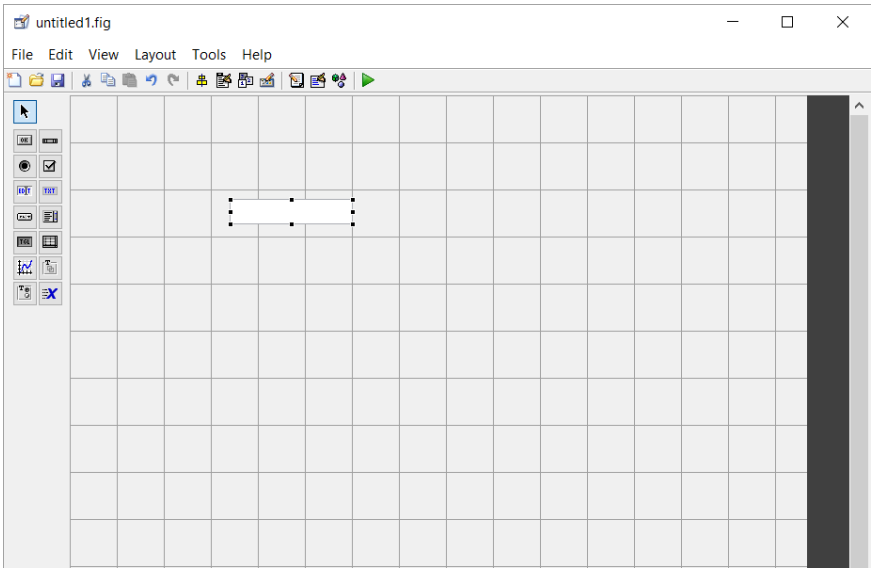


Рис. 2.106

Пустое окно редактируемого текста (Edit Text) в графическом окне Windows-интерфейса

Чтобы использовать число, введенное в пустое поле, например с именем edit1, необходимо его «получить» (get) и присвоить какой-либо переменной следующим образом:

as = get (handles.edit1, ‘String’);

Переменная as является текстовой. Для того чтобы превратить ее в числовую, следует воспользоваться известной функцией “str2double”:

a = str2double(as);

или, объединяя два последних оператора:

a = str2double (get (handles.edit1, ‘String’));

Следует обратить внимание на то, что после точки в конструкции handles.edit1 записано имя окна редактирования, откуда получено число.

В программе *m*-файла выполняются команды:

aa = a^2;

sss = 2*a;

Если ставится задача разработки программы Test1 с визуальным Windows-интерфейсом, в котором будет два окна со Статическим текстом (Static Text): «ИСХОДНАЯ ИНФОРМАЦИЯ ДЛЯ РАСЧЕТОВ» и «РЕЗУЛЬТАТЫ РАСЧЕТОВ», одно окно запуска программы Push Button и три окна с Редактируемым текстом (Edit Text): одно для задания чисел и два для записи результатов расчетов, то Windows-интерфейс будет выглядеть, как показано на рисунке 2.107. На рисунке 2.108 приведен фрагмент программного кода с элементарными вычислениями для иллюстрации работы Редактора текста (Edit Text), а ре-

зультат работы программы представлен на рисунке 2.109 с соответствующим программным кодом для программы Test1 — на рисунке 2.110.

Для записи чисел в *m*-файле программы в окно редактирования используется оператор “set” с указанием имени окна редактирования (edit2 и edit3) в конструкции, например handles.edit2 и handles.edit3 (рис. 2.108):

set (handles.edit2, ‘String’, aa);

set (handles.edit3, ‘String’, sss).

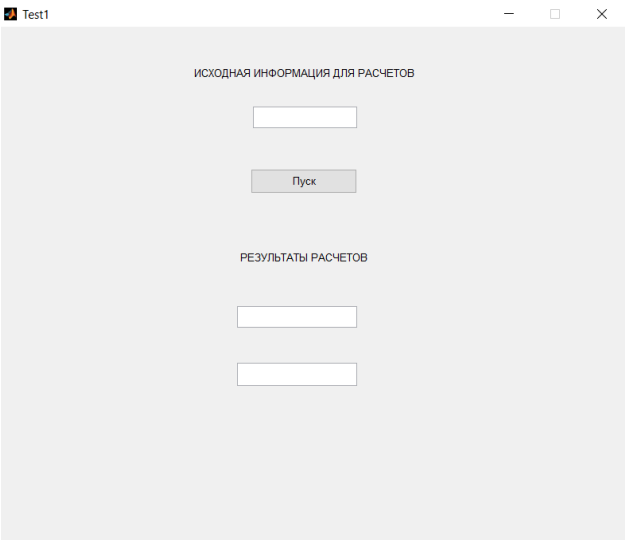


Рис. 2.107

Визуальный Windows-интерфейс программы Test1.m с двумя окнами
со Статическим текстом (Static Text) и тремя окнами с Редактируемым текстом (Edit Text)

```
119 -   end
120     % --- Executes on button press in pushbutton1.
121     function pushbutton1_Callback(hObject, eventdata, handles)
122     % hObject    handle to pushbutton1 (see GCBO)
123     % eventdata  reserved - to be defined in a future version of MATLAB
124     % handles    structure with handles and user data (see GUIDATA)
125     %Присвоение числа из окна с редактируемым текстом и с именем edit1 переменной as
126 -   as=get(handles.edit1,'String');
127     %Преобразование строковой константы в числовую
128 -   a=str2double(as);
129     %Действие над числовой константой 'a'
130 -   aa=a^2;
131     %Размещение результата в  окно с редактируемым текстом и именем edit2
132 -   set(handles.edit2,'String',aa);
133     %Действие над числовой константой 'aa'
134 -   sss=2*aa;
135     %Размещение результата в  окно с редактируемым текстом и именем edit3
136 -   set(handles.edit3,'String',sss);
137     function edit3_Callback(hObject, eventdata, handles)
138     % hObject    handle to edit3 (see GCBO)
139     % eventdata  reserved - to be defined in a future version of MATLAB
```

Рис. 2.108

Фрагмент выполняемого программного кода программы Test1.m
с элементарными вычислениями

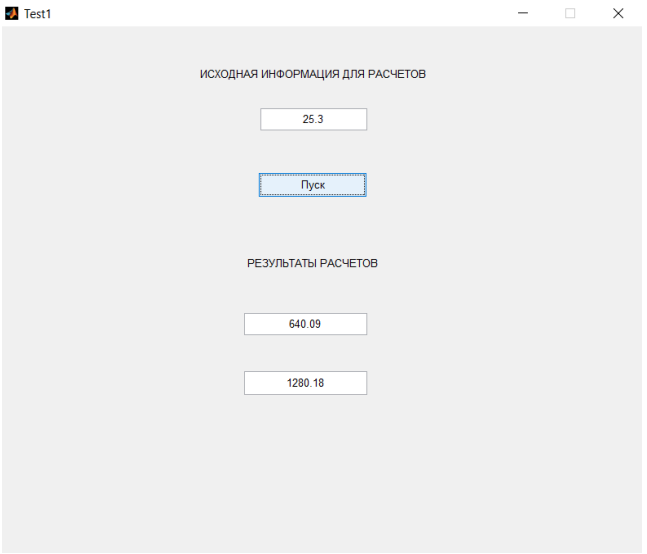


Рис. 2.109

Результат работы программы Test1.m с элементарными вычислениями

```
function varargout = gui_test1(varargin)
% GUI_TEST1 MATLAB code for gui_test1.fig
%      GUI_TEST1, by itself, creates a new GUI_TEST1 or raises the existing
%      singleton*.
%
%      H = GUI_TEST1 returns the handle to a new GUI_TEST1 or the handle to
%      the existing singleton*.
%
%      GUI_TEST1('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in GUI_TEST1.M with the given input arguments.
%
%      GUI_TEST1('Property','Value',...) creates a new GUI_TEST1 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before gui_test1_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to gui_test1_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help gui_test1

% Last Modified by GUIDE v2.5 17-Nov-2018 20:24:00

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @gui_test1_OpeningFcn, ...
                   'gui_OutputFcn',  @gui_test1_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
```

Рис. 2.110 (начало)

Программный код *m*-файла Test1.m с визуальным Windows-интерфейсом для выполнения элементарных вычислений

```
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before gui_test1 is made visible.
function gui_test1_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to gui_test1 (see VARARGIN)

% Choose default command line output for gui_test1
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes gui_test1 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = gui_test1_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a double


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

Рис. 2.110 (продолжение)
Программный код *m*-файла Test1.m с визуальным Windows-интерфейсом для выполнения элементарных вычислений

```
% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a double


% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a double


% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%Присвоение числа из окна с редактируемым текстом и именем edit1 переменной
%as
as=get(handles.edit1,'String');
%Преобразование строковой константы в числовую
a=str2double(as);
%Действие над числовой константой 'a'
aa=a^2;
%Размещение результатав в окно с редактируемым текстом и именем edit2
set(handles.edit2,'String',aa);
%Действие над числовой константой 'aa'
sss=aa*2;
%Размещение результатав в окно с редактируемым текстом и именем edit3
set(handles.edit3,'String',sss);
```

Рис. 2.110 (окончание)

Программный код *m*-файла Test1.m с визуальным Windows-интерфейсом для выполнения элементарных вычислений

2.10.12.4. Решение квадратного уравнения

Процедура решения квадратного уравнения с использованием визуального Windows-интерфейса включает в себя следующие этапы.

1. Размещение в графическом окне конструктора GUIDE окон для редактируемого текста — Edit Text.

2. Размещение в графическом окне конструктора GUIDE окон для статического текста — Static Text.

3. Размещение окна для построения графика функции квадратного уравнения — Axis.

4. Размещение окна для запуска программы решения — Push Button.

5. Коррекция сгенерированного программного кода *m*-функции программы в подфункции pushbutton путем включения в нее необходимой программы решения квадратного уравнения с чтением исходной информации с визуального Windows-интерфейса и выводом на него результатов расчета в числовом и графическом виде.

Прежде чем записать нужный программный код для решения квадратного уравнения (пункт 5), необходимо разместить в конструкторе GUIDE на графическом окне (рис. 2.111):

- семь пустых окон для редактирования текста (Edit Text), первые три из которых с именами edit1, edit2, edit3 предназначены для задания коэффициентов квадратного уравнения, а остальные четыре окна — для вывода значений вещественных корней с именами edit5, edit6 и комплексных корней: для первого корня edit5 и edit7 — соответственно вещественная и мнимая часть; для второго корня edit6 и edit8 — соответственно вещественная и мнимая часть;
- пять окон статических переменных (Static Text) для обозначения переменных, которые задаются в исходной информации (a = , b = , c =) и выводятся в качестве результатов расчета ($x1$ = , $x2$ =);
- два окна статических переменных, заполненных буквой “i” после мнимой части комплексного числа;
- одно окно статической переменной в виде строки для вывода корней квадратного уравнения — вещественных и комплексных (после вывода результатов с редактируемым текстом);
- четыре окна с фиксированным статическим текстом в виде заголовков: «Решение квадратного уравнения», «$ax^2 + bx + c = 0$», «Исходная информация», «Результаты расчетов»;
- одно окно с переменным статическим текстом: либо «Вещественные корни», либо «Комплексные корни»;
- одно окно для запуска программы (Push Button), которое переименовано из «Push Button» в «Расчет»;
- одно окно для построения графика функции квадратного уравнения (Axis).

Сохранение созданного графического окна в конструкторе GUIDE приводит к автоматической генерации заголовка программного кода в виде *m*-файла, т. е. в рассматриваемом случае наряду с файлом SQEQ_m.fig генерируется *m*-файл SQEQ_m.m (рис. 2.112).

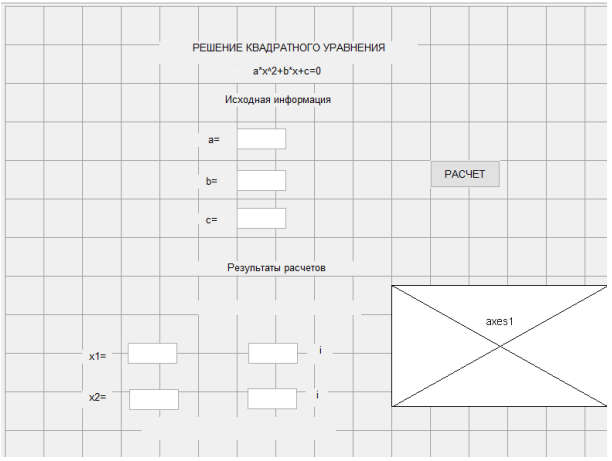


Рис. 2.111

Размещение пустых окон на графическом интерфейсе с использованием конструктора GUIDE для ввода исходной информации и вывода результатов при решении квадратного уравнения

```
function varargout = SQEQ_m(varargin)
% SQEQ_M MATLAB code for SQEQ_m.fig
%      SQEQ_M, by itself, creates a new SQEQ_M or raises the existing
%      singleton*.
%
%      H = SQEQ_M returns the handle to a new SQEQ_M or the handle to
%      the existing singleton*.
%
%      SQEQ_M('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in SQEQ_M.M with the given input arguments.
%
%      SQEQ_M('Property','Value',...) creates a new SQEQ_M or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before SQEQ_m_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to SQEQ_m_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help SQEQ_m

% Last Modified by GUIDE v2.5 23-Jan-2016 23:05:58

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @SQEQ_m_OpeningFcn, ...
                   'gui_OutputFcn',  @SQEQ_m_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
```

Рис. 2.112 (начало)

Программный код файла SQEQ_m.m решения квадратного уравнения с использованием конструктора GUIDE

```
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before SQEQ_m is made visible.
function SQEQ_m_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to SQEQ_m (see VARARGIN)

% Choose default command line output for SQEQ_m
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes SQEQ_m wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = SQEQ_m_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a double


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a double


% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

Рис. 2.112 (продолжение)
Программный код файла SQEQ_m.m решения квадратного уравнения
с использованием конструктора GUIDE

```
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a double

% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved — to be defined in a future version of MATLAB
% handles    empty — handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved — to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4 as a double

% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved — to be defined in a future version of MATLAB
% handles    empty — handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

function edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved — to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit5 as text
%        str2double(get(hObject,'String')) returns contents of edit5 as a double

% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved — to be defined in a future version of MATLAB
% handles    empty — handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved — to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.edit6,'String',' ');set(handles.edit7,'String',' ');
a=str2double(get(handles.edit1,'String'));
b=str2double(get(handles.edit2,'String'));
c=str2double(get(handles.edit3,'String'));
det=b^2-4*a*c;
 x1=(-b+sqrt(det))/2/a;
 x2=(-b-sqrt(det))/2/a;

 if det<0
   ss='комплексные корни';
  s=sprintf('x1=%g +%gi;\t x2=%g %gi',real(x1),imag(x1),real(x2),imag(x2));
  set(handles.edit4,'String',real(x1));set(handles.edit6,'String',imag(x1));
set(handles.edit5,'String',real(x2));set(handles.edit7,'String',imag(x2));
else
ss='вещественные корни';
s=sprintf('x1=%g;\t x2=%g',x1,x2);
```

Рис. 2.112 (продолжение)
Программный код файла SQEQ_m.m решения квадратного уравнения
с использованием конструктора GUIDE

```
set(handles.edit4,'String',real(x1));
set(handles.edit5,'String',real(x2));
end
 set(handles.text55,'String',ss);
 set(handles.text6,'String',s);
  x=-3:0.1:0;
 y=a*x.^2+b*x+c;
 plot(x,y,'k-');
 title('График функции кв. уравнения a*x^2+b*x+c=0');
 xlabel('x');ylabel('y');
axis([-3 0 -5 5]);
 grid on;

function edit6_Callback(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved — to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit6 as text
%        str2double(get(hObject,'String')) returns contents of edit6 as a double

% --- Executes during object creation, after setting all properties.
function edit6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved — to be defined in a future version of MATLAB
% handles    empty — handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

function edit7_Callback(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved — to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit7 as text
%        str2double(get(hObject,'String')) returns contents of edit7 as a double

% --- Executes during object creation, after setting all properties.
function edit7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved — to be defined in a future version of MATLAB
% handles    empty — handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
```

Рис. 2.112 (окончание)

Программный код файла SQEQ_m.m решения квадратного уравнения с использованием конструктора GUIDE

Каждому объекту графического окна конструктора GUIDE (рис. 2.111) — Редактируемого текста (Edit Text), Статического текста (Static Text), графика (Axis) и кнопке пуска (Push Button) — соответствует фрагмент программы SQEQ_m.

Чтобы посмотреть фрагмент программного кода, соответствующего окну в конструкторе GUIDE, его надо выделить левой кнопкой мыши, затем, нажав на правую кнопку мыши, выбрать пункты выпадающего меню View Callbacks — Callback (рис. 2.113), после чего на дисплее можно будет увидеть программный код окна, соответствующего Редактируемому тексту (Edit Text) с именем edit1 (рис. 2.114).

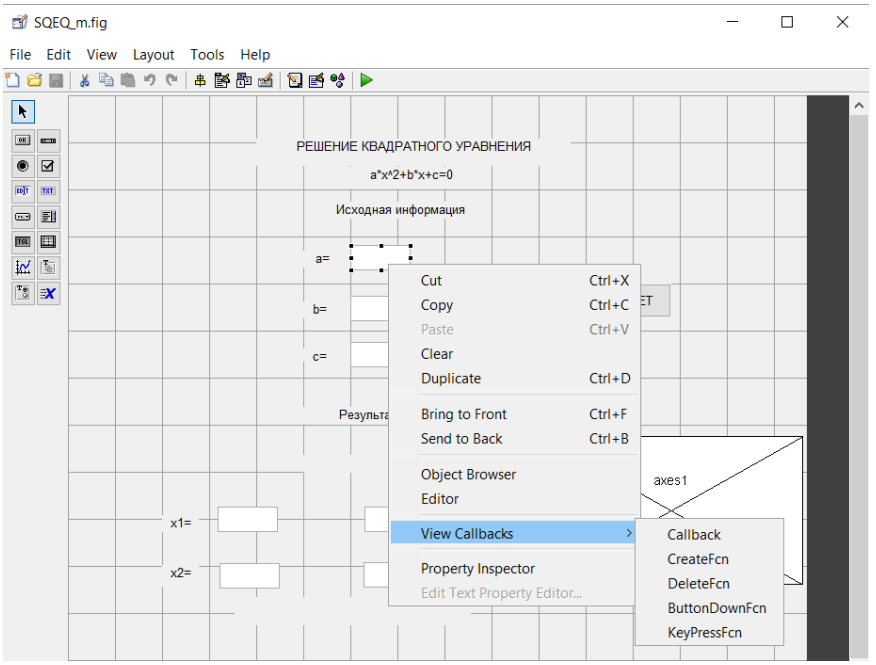


Рис. 2.113

Процедура обнаружения программного кода *m*-файла, соответствующего окну Редактируемого текста (Edit Text)

```
74
75    function edit1_Callback(hObject, eventdata, handles)
76    % hObject    handle to edit1 (see GCBO)
77    % eventdata  reserved - to be defined in a future version of MATLAB
78    % handles    structure with handles and user data (see GUIDATA)
79
80    % Hints: get(hObject,'String') returns contents of edit1 as text
81    %        str2double(get(hObject,'String')) returns contents of edit1 as a double
82
83    % --- Executes during object creation, after setting all properties.
84    function edit1_CreateFcn(hObject, eventdata, handles)
85    % hObject    handle to edit1 (see GCBO)
86    % eventdata  reserved - to be defined in a future version of MATLAB
87    % handles    empty - handles not created until after all CreateFcns called
88
89    % Hint: edit controls usually have a white background on Windows.
90    %       See ISPC and COMPUTER.
91    if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
92        set(hObject,'BackgroundColor','white');
93    end
94
```

Рис. 2.114

Фрагмент программного кода *m*-файла SQEQ_m.m, соответствующего окну Редактируемого текста (Edit Text) с именем edit1

На рисунке 2.115 показано, что Имя (Tag) edit1 объекта (в рассматриваемом случае — окна редактируемого текста) может быть изменено с помощью Инспектора свойств (Property Inspector) (рис. 2.115).

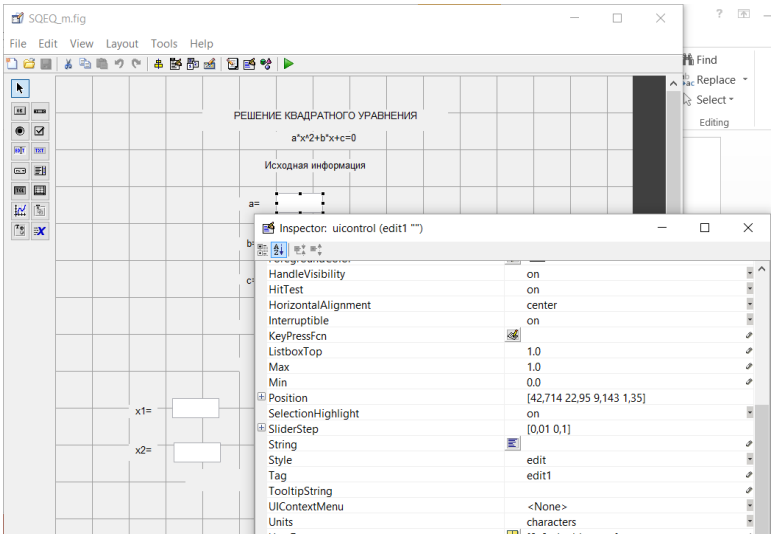


Рис. 2.115

Возможность изменения свойств объекта (в данном случае edit1) при использовании Инспектора свойств (Property Inspector)

Фрагмент программы *m*-файла SQEQ_m.m, соответствующего окну Статического текста (Static Text), можно увидеть, если выделить объект и правой кнопкой мыши в выпадающем меню отметить View Callbacks — Callback (рис. 2.116), в результате чего на дисплее появится программный код подфункции text55 (рис. 2.117).

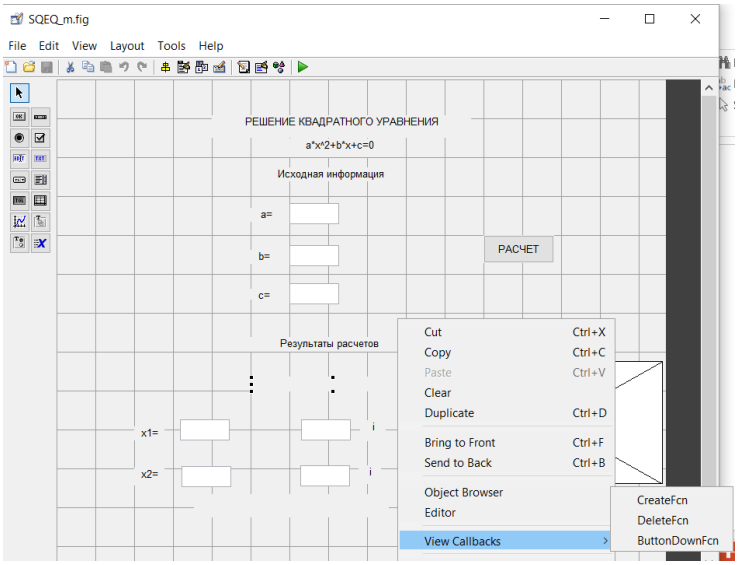


Рис. 2.116

Процедура вывода на дисплей программного кода подфункции, соответствующей окну Статического текста (Static Text) с использованием позиции выпадающего меню View Callbacks — Callback

```
260     % handles    empty - handles not created until after all CreateFcns called
261
262     % Hint: edit controls usually have a white background on Windows.
263     %       See ISPC and COMPUTER.
264 -   if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
265 -       set(hObject,'BackgroundColor','white');
266 -   end
267
268
269     % --- Executes during object creation, after setting all properties.
270     function text55_CreateFcn(hObject, eventdata, handles)
271     % hObject    handle to text55 (see GCBO)
272     % eventdata  reserved - to be defined in a future version of MATLAB
273     % handles    empty - handles not created until after all CreateFcns called
274
275
276     % --- Executes during object creation, after setting all properties.
277     function axes1_CreateFcn(hObject, eventdata, handles)
278     % hObject    handle to axes1 (see GCBO)
279     % eventdata  reserved - to be defined in a future version of MATLAB
280     % handles    empty - handles not created until after all CreateFcns called
```

Рис. 2.117

Фрагмент программного кода *m*-файла SQEQ_m.m, соответствующего окну Статического текста (Static Text) с именем text55

По аналогии можно увидеть фрагменты *m*-файлов, относящихся к двум объектам — графику (Axis) с именем axis1 (рис. 2.117) и кнопке запуска (Push Button) с именем pushbutton1 (рис. 2.118).

```
263     %       See ISPC and COMPUTER.
264 -   if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
265 -       set(hObject,'BackgroundColor','white');
266 -   end
267
268
269     % --- Executes during object creation, after setting all properties.
270     function text55_CreateFcn(hObject, eventdata, handles)
271     % hObject    handle to text55 (see GCBO)
272     % eventdata  reserved - to be defined in a future version of MATLAB
273     % handles    empty - handles not created until after all CreateFcns called
274
275
276     % --- Executes during object creation, after setting all properties.
277     function axes1_CreateFcn(hObject, eventdata, handles)
278     % hObject    handle to axes1 (see GCBO)
279     % eventdata  reserved - to be defined in a future version of MATLAB
280     % handles    empty - handles not created until after all CreateFcns called
281
282     % Hint: place code in OpeningFcn to populate axes1
283
```

Рис. 2.118

Фрагмент программного кода *m*-файла SQEQ_m.m, соответствующего окну Графика (Axis) с именем axis1

```
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.edit6,'String',' ');set(handles.edit7,'String',' ');
a=str2double(get(handles.edit1,'String'));
b=str2double(get(handles.edit2,'String'));
c=str2double(get(handles.edit3,'String'));
det=b^2-4*a*c;
 x1=(-b+sqrt(det))/2/a;
 x2=(-b-sqrt(det))/2/a;

 if det<0
    ss='комплексные корни';
   s=sprintf('x1=%g +%gi;\t x2=%g %gi',real(x1),imag(x1),real(x2),imag(x2));
   set(handles.edit4,'String',real(x1));set(handles.edit6,'String',imag(x1));
set(handles.edit5,'String',real(x2));set(handles.edit7,'String',imag(x2));
else
ss='вещественные корни';
s=sprintf('x1=%g;\t x2=%g',x1,x2);
set(handles.edit4,'String',real(x1));
```

Рис. 2.119

Фрагмент программного кода *m*-файла SQEQ_m.m, соответствующего кнопке запуска (Push Button) с именем pushbutton1

Программный код алгоритма решения квадратного уравнения записан в подфункции pushbutton1.Callback *m*-файла SQEQ_m.m (рис. 2.112) и имеет вид, представленный на рисунке 2.120.

```
Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved — to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.edit6,'String',' ');set(handles.edit7,'String',' ');
a=str2double(get(handles.edit1,'String'));
b=str2double(get(handles.edit2,'String'));
c=str2double(get(handles.edit3,'String'));
det=b^2-4*a*c;
 x1=(-b+sqrt(det))/2/a;
 x2=(-b-sqrt(det))/2/a;

 if det<0
    ss='комплексные корни';
   s=sprintf('x1=%g +%gi;\t x2=%g %gi',real(x1),imag(x1),real(x2),imag(x2));
set(handles.edit4,'String',real(x1));set(handles.edit6,'String',imag(x1));
set(handles.edit5,'String',real(x2));set(handles.edit7,'String',imag(x2));
else
ss='вещественные корни';
s=sprintf('x1=%g;\t x2=%g',x1,x2);
set(handles.edit4,'String',real(x1));
set(handles.edit5,'String',real(x2));
end
 set(handles.text55,'String',ss);
 set(handles.text6,'String',s);
  x=-3:0.1:0;
 y=a*x.^2+b*x+c;
 plot(x,y,'k-');
 title('График функции кв. уравнения a*x^2+b*x+c=0');
 xlabel('x');ylabel('y');
axis([-3 0 -5 5]);
 grid on;
```

Рис. 2.120

Программный код решения квадратного уравнения при использовании визуального Windows-интерфейса

Исходная информация о коэффициентах уравнения считывается с использованием функции ‘get’ из текстовых полей Редактируемого текста (Edit Text) с именами edit1, edit2 и edit3, а результаты записываются в текстовые поля Редактируемого текста (Edit Text) с именами edit4 и edit5 — в случае вещественных корней, и edit4 и edit6, а также edit5 и edit7 — в случае комплексных корней с использованием функции ‘set’.

Операторы для задания коэффициентов *a*, *b*, *c* с учетом того, что строковые данные должны быть преобразованы в числовые с использованием функции ‘str2double’, имеют вид (рис. 2.119):

```
a=str2double(get(handles.edit1,'String'));

b=str2double(get(handles.edit2,'String'));

c=str2double(get(handles.edit3,'String')).
```

Вывод результатов (корней уравнения) осуществляется в текстовые поля Редактируемого текста (Edit Text) следующим образом (рис. 2.119):

— в случае вещественных корней (функция ‘real’)

```
set(handles.edit4,'String',real(x1));

set(handles.edit5,'String',real(x2));
```

— в случае комплексных корней (функция ‘imag’)

```
set(handles.edit4,'String',real(x1));set(handles.edit6,'String',imag(x1));

set(handles.edit5,'String',real(x2));set(handles.edit7,'String',imag(x2)).
```

Для очистки текстовых полей мнимых частей корней в полях Редактируемого текста (Edit Text) при запуске программы используется функция ‘set’ в начале подфункции pushbutton1:

```
set(handles.edit6,'String',' ');set(handles.edit7,'String',' ').
```

Результаты выводятся также в текстовые поля Статического текста (Static Text). При этом символьной переменной “s” сначала присваиваются значения строковых констант с форматом “%g” (рис. 2.119):

— для вещественных корней

```
s=sprintf('x1=%g;\t x2=%g',x1,x2);
```

— для комплексных корней

```
s=sprintf('x1=%g +%gi;\t x2=%g %gi',real(x1),imag(x1),real(x2),imag(x2)).
```

Затем переменная “s” размещается в текстовое поле Статического текста (Static Text) с именем text6 (рис. 2.120):

```
set(handles.text6,'String',s),
```

а также в текстовое поле Статического текста (Static Text) с именем text55 (рис. 2.120):

```
set(handles.text55,'String',ss),
```

где ss — строковая константа, принимающая следующие значения в зависимости от знака детерминанта квадратного уравнения (det):

ss='комплексные корни', если det < 0;

ss='вещественные корни', если det ≥ 0.

В завершающей части программного кода (рис. 2.120) записаны операторы для построения графика функции квадратного уравнения, который будет размещен в окне графика (axis1) визуального Windows-интерфейса.

Запуск программы SQEQ_m можно выполнить либо из окна конструктора GUIDE, нажав кнопку «Пуск» (►) на верхней горизонтальной панели инструментов, либо из основного окна интегрированной среды MATLAB, записав в Командную строку Командного окна имя *m*-файла SQEQ_m. При этом в Текущей папке (Current Folder) должны расположиться два сгенерированных и скорректированных файла — SQEQ_m.m и SQEQ_m.fig.

Программа SQEQ_m может быть также запущена, если открыть программный код *m*-файла в Редакторе (Editor) и нажать кнопку «Пуск» (►) на верхней горизонтальной панели инструментов.

После запуска программы на дисплее появляется окно визуального Windows-интерфейса (рис. 2.121) с пустыми текстовыми полями и пустым графиком. Для выполнения расчетов в пустые поля для ввода исходной информации необходимо ввести коэффициенты, например $a = 1$; $b = 3$; $c = 2$, и щелкнуть левой кнопкой мыши по клавише «Расчет», соответствующей кнопке «Пуск» (Push Button).

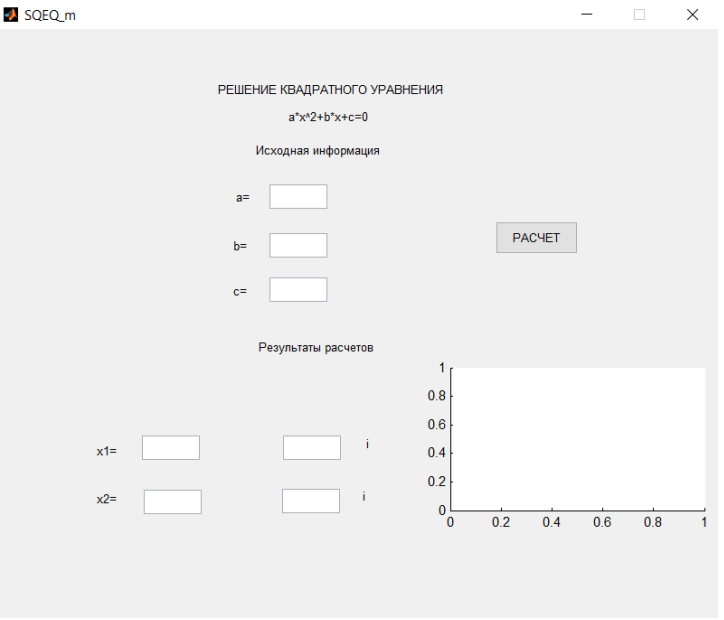


Рис. 2.121

Визуальный Windows-интерфейс программы решения квадратного уравнения после запуска программы SQEQ_m до ввода исходных данных

В результате будут выполнены все действия в соответствии с программой решения квадратного уравнения (рис. 2.120), и визуальный Windows-интерфейс примет вид, изображенный на рисунке 2.122.

Для уравнения с комплексными корнями (коэффициенты $a = 1$; $b = 3$; $c = 2$) вид визуального интерфейса представлен на рисунке 2.123.

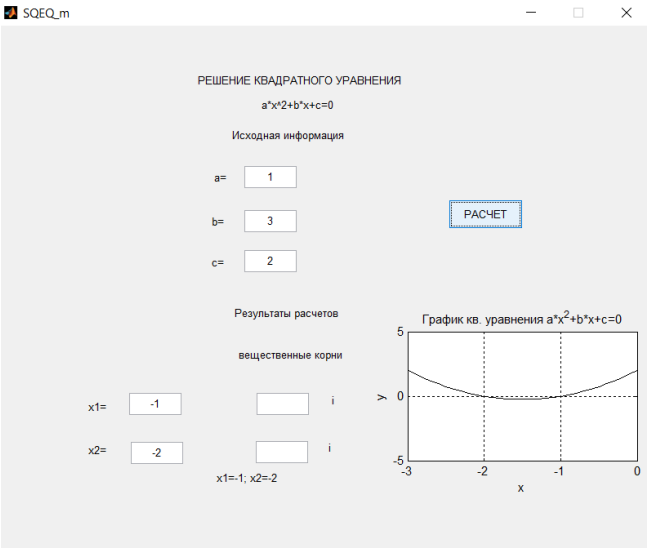


Рис. 2.122

Визуальный Windows-интерфейс программы решения квадратного уравнения с вещественными корнями после запуска программы SQEQ_m после ввода исходных данных

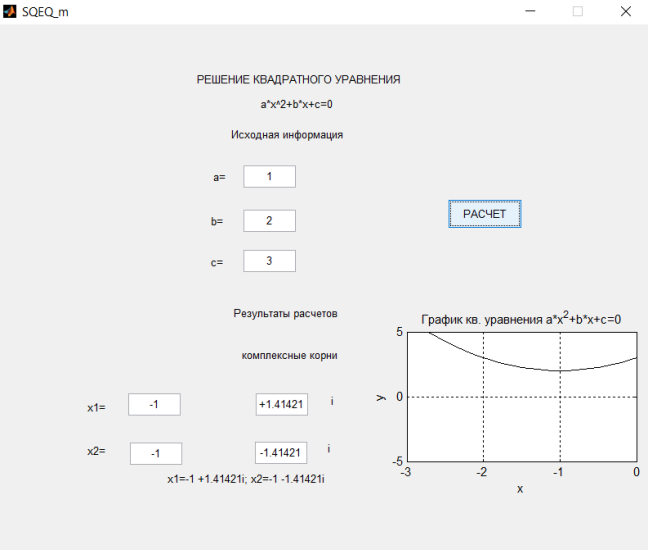


Рис. 2.123

Визуальный Windows-интерфейс программы решения квадратного уравнения с комплексными корнями после запуска программы SQEQ_m после ввода исходных данных

2.10.12.5. Задание элементов массивов с использованием объекта Listbox в окне графического интерфейса GUIDE

Для задания элементов одномерного массива из объектов графического интерфейса GUIDE выбирается объект Listbox (рис. 2.124) и размещается в поле графического интерфейса с названием Listbox. После выделения этого объекта левой кнопкой мыши и выбора правой кнопкой позиции Инспектора свойств (Property Inspector) в выпадающем меню (рис. 2.126) раскрывается окно инспектора свойств (рис. 2.127). В этом окне в строке String необходимо левой кнопкой мыши щелкнуть по значку, стоящему после слова String, и в поле графического интерфейса (рис. 2.127) появляется окно с заголовком String (рис. 2.128), куда друг под другом следует записать элементы одномерного массива, предварительно удалив из первой строки слово “Listbox”.

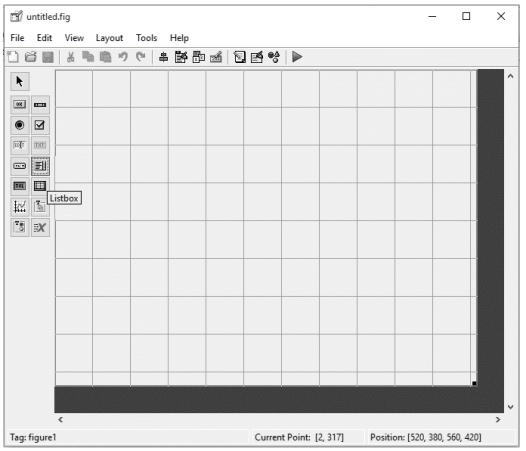


Рис. 2.124

Выбор объекта Listbox для задания элементов одномерного массива

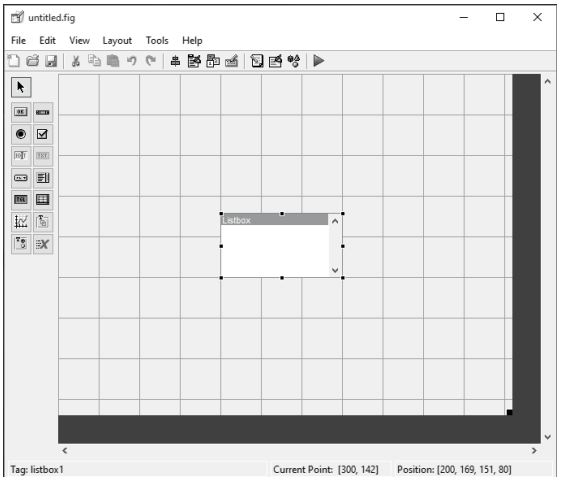


Рис. 2.125

Выделение объекта Listbox левой кнопкой мыши

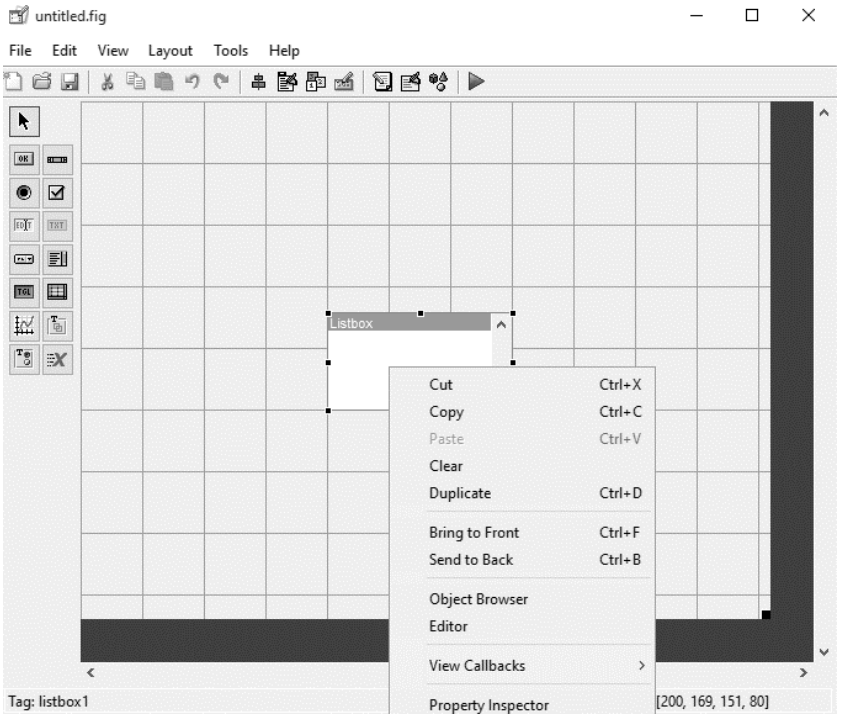


Рис. 2.126

Выбор из падающего меню Инспектора свойств (Property Inspector) объекта Listbox правой кнопкой мыши

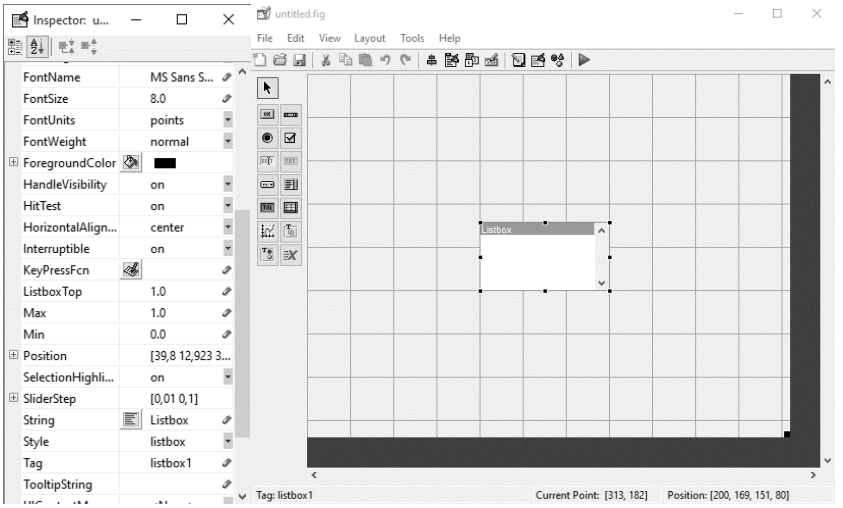


Рис. 2.127

Выбор пиктограммы String (строка) объекта Listbox в меню Инспектора свойств (Property Inspector)

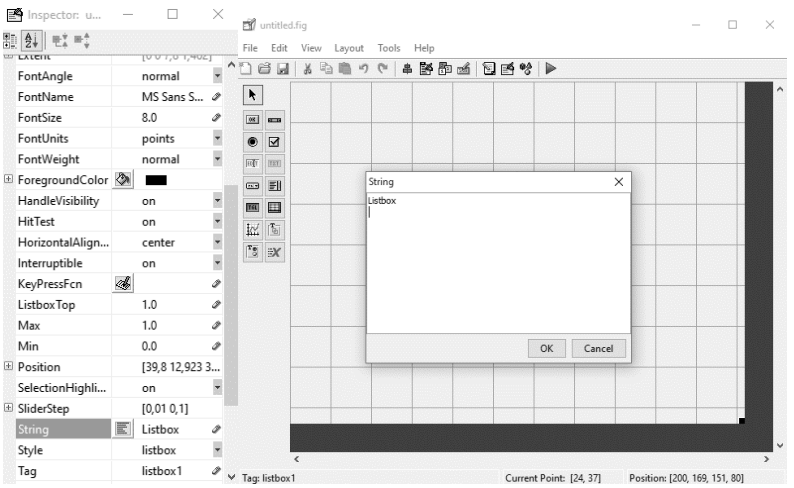
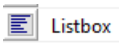


Рис. 2.128

Окно для задания элементов одномерного массива для использования в программе

В результате можно сформировать одномерный массив, как показано на рисунке 2.129, из шести элементов. Для формирования вектор-столбца нужно левой кнопкой мыщи щелкнуть по значку в Property Inspector слева от Listbox — 

. Все указанные действия в графическом интерфейсе GUIDE отразятся в соответствующем *m*-файле, автоматически создаваемом наряду с файлом с расширением .fig при сохранении GUI-приложения (рис. 2.124).

Необходимо отметить, что для числовой обработки элементов созданного массива (рис. 2.129) он должен быть обязательно преобразован из символьной формы (String) в числовую (double) с использованием функции str2double.

Ниже рассматриваются два примера, когда задаются два вектора с использованием объекта Listbox с метками (Tag) — listbox1 и listbox2.

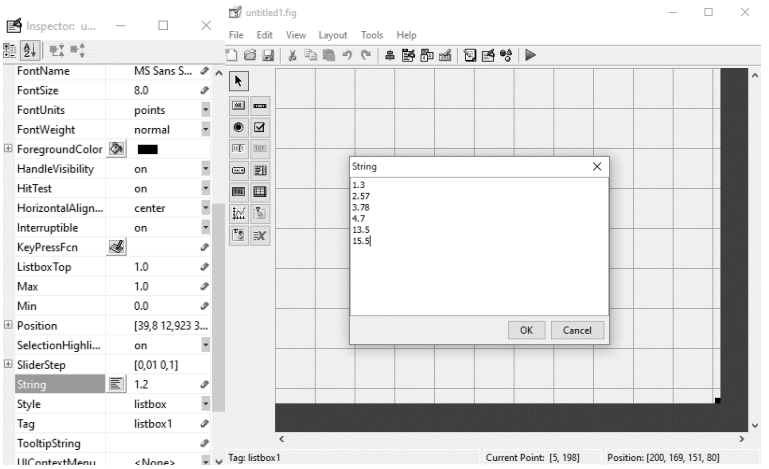


Рис. 2.129

Задание в графическом интерфейсе GUIDE элемента одномерного массива

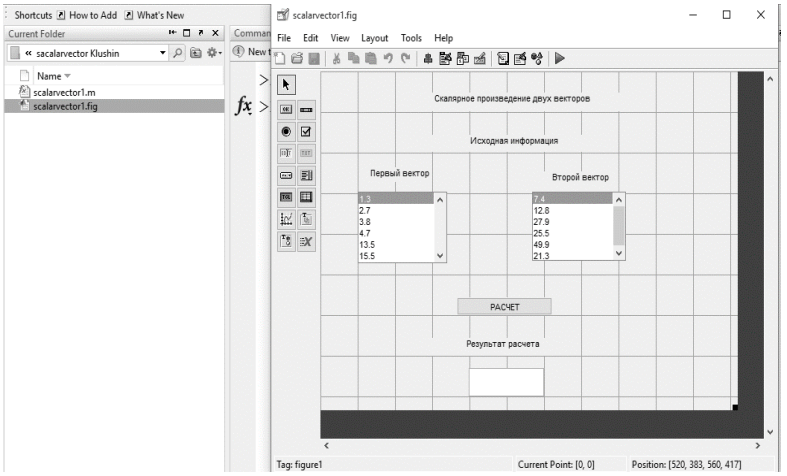


Рис. 2.130

Задание элементов двух векторов с использованием объекта Listbox графического интерфейса GUIDE для вывода результата скалярного произведения двух векторов

```
function varargout = scalarvector1(varargin)
% SCALARVECTOR1 MATLAB code for scalarvector1.fig
%      SCALARVECTOR1, by itself, creates a new SCALAR-
VECTOR1 or raises the existing
%      singleton*.
%
%      H = SCALARVECTOR1 returns the handle to a new SCA-
LARVECTOR1 or the handle to
%      the existing singleton*.
%
%      SCALARVEC-
TOR1('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in SCALARVECTOR1.M
with the given input arguments.
%
%      SCALARVECTOR1('Property','Value',...) creates a new
SCALARVECTOR1 or raises the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before scalarvector1_OpeningFcn gets
called.  An
%      unrecognized property name or invalid value makes proper-
ty application
%      stop.  All inputs are passed to scalarvector1_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI
allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help scalarvector1

% Last Modified by GUIDE v2.5 14-Mar-2016 00:00:57

% Begin initialization code — DO NOT EDIT
% handles    structure with handles and user data (see GUIDA-
TA)

% Hints: contents = cellstr(get(hObject,'String')) returns
listbox1 contents as cell array
%        contents{get(hObject,'Value')} returns selected item from
listbox1


% --- Executes during object creation, after setting all proper-
ties.
function listbox1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listbox1 (see GCBO)
% eventdata  reserved — to be defined in a future version of
MATLAB
% handles    empty — handles not created until after all Cre-
ateFcns called

% Hint: listbox controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in listbox2.
function listbox2_Callback(hObject, eventdata, handles)
% hObject    handle to listbox2 (see GCBO)
% eventdata  reserved — to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDA-
TA)

% Hints: contents = cellstr(get(hObject,'String')) returns
listbox2 contents as cell array
```

Рис. 2.131 (начало)

Программный код *m*-файла scalarvector1.m для расчета скалярного произведения двух векторов с использованием графического интерфейса GUIDE

```
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @scalarvector1_OpeningFcn, ...
                   'gui_OutputFcn',  @scalarvector1_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before scalarvector1 is made visible.
function scalarvector1_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to scalarvector1 (see
VARARGIN)

% Choose default command line output for scalarvector1
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes scalarvector1 wait for user response (see
UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command
line.
function varargout = scalarvector1_OutputFcn(hObject, event-
data, handles)
% varargout  cell array for returning output args (see
VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on selection change in listbox1.
function listbox1_Callback(hObject, eventdata, handles)
% hObject    handle to listbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
%        contents{get(hObject,'Value')} returns selected item from
listbox2


% --- Executes during object creation, after setting all proper-
ties.
function listbox2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listbox2 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

% Hint: listbox controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDA-
TA)
s1=get(handles.listbox1,'String');
s2=get(handles.listbox2,'String');
ss1=str2double(s1);
ss2=str2double(s2);
scalar=ss1'*ss2;
scalar=num2str(scalar);
set(handles.edit1,'String',scalar);
function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDA-
TA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1
as a double


% --- Executes during object creation, after setting all proper-
ties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

% Hint: edit controls usually have a white background on Win-
dows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

Рис. 2.131 (окончание)

Программный код *m*-файла scalarvector1.m для расчета скалярного произведения двух векторов с использованием графического интерфейса GUIDE

В первом примере (рис. 2.130) с программным кодом *m*-файла scalarvector1.m (рис. 2.131) реализована программа скалярного произведения двух векторов, каждый из которых состоит из шести элементов.

С этой целью сначала в окне конструктора графического интерфейса GUIDE (рис. 2.130) размещаются следующие объекты:

а) текстовые (Static Text)

- скалярное произведение двух векторов;
- исходная информация;
- первый вектор;
- второй вектор;
- результат расчета;

б) редактируемые (Edit Text);

- пустое поле после текста «Результат расчета»;

в) два одномерных массива (listbox1 и listbox2) со значениями элементов первого и второго векторов;

г) кнопка запуска (Push Button), обозначенная словом «РАСЧЕТ».

После сохранения созданного окна конструктором GUIDE в Текущей папке MATLAB появляются два файла:

- scalarvector1.m;
- scalarvector1.fig.

Следует отметить, что изменения созданного приложения рекомендуется проводить в окне конструктора GUIDE, что будет автоматически отражаться в соответствующем *m*-файле — scalarvector1.m.

Выполнение действий, связанных с записью программного кода для определения скалярного произведения двух векторов, размещенных в одномерных массивах с метками (Tag) — listbox1 и listbox2, можно выполнить путем редактирования фрагмента программы scalarvector1.m в функции (рис. 2.130)

function pushbutton1_Callback.

Для решения поставленной задачи в эту функцию необходимо записать семь операторов.

Первые два оператора с функцией “get”:

```
s1 = get (handles, listbox1, ‘String’);
```

```
s2 = get (handles, listbox2, ‘String’)
```

считывают данные об элементах двух векторов из заданных значений одномерных массивов с метками (Tag) — listbox1 и listbox2.

Следующие два оператора с функцией “str2double” (рис. 2.130):

```
ss1 = str2double (s1);
```

```
ss2 = str2double (s2)
```

преобразуют элементы векторов s1 и s2 в числовые данные.

Оператор

```
scalar = ss1’ * ss2
```

позволяет вычислить произведение транспонированного вектора (ss1’) на вектор (ss2), т. е. скалярного произведения двух векторов ss1 и ss2.

Предпоследний оператор с функцией “num2str”:

```
scalar = num2str (scalar);
```

преобразует числовую константу в строковую (str).

Последний оператор (рис. 2.131) set позволяет разместить результат (scalar) в пустую ячейку (рис. 2.130) с редактируемым текстом и меткой “edit1”:

```
set (hadles.edit1,’String’,scalar).
```

Программу scalarvector1.m можно запустить из командной строки основного окна MATLAB, набрав scalarvector1, после чего появится окно, показанное на рисунке 2.131.

Для выполнения расчетов по программе необходимо щелкнуть мышью по кнопке «РАСЧЕТ».

Результат расчета появится внизу окна (рис. 2.133), которое до запуска расчетов было пустым (рис. 2.132).

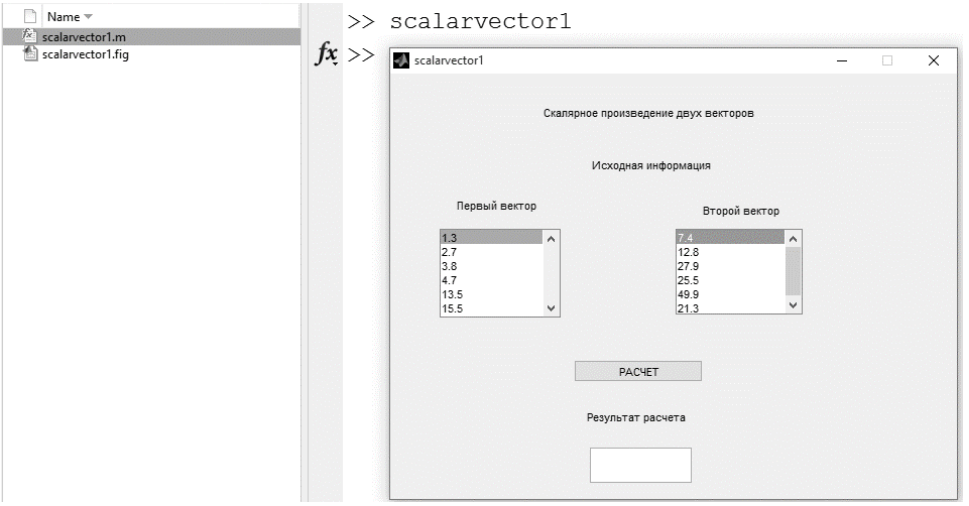


Рис. 2.132

Командное окно MATLAB после запуска программы расчета скалярного произведения двух векторов, когда используется объект Listbox графического интерфейса GUIDE

Во втором примере с использованием графического интерфейса GUIDE реализуется программа расчета векторного произведения двух векторов, состоящих из трех элементов, в результате чего получается квадратная матрица размером 3×3:

$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} * \begin{bmatrix} a_1 & a_2 & a_3 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}.$$

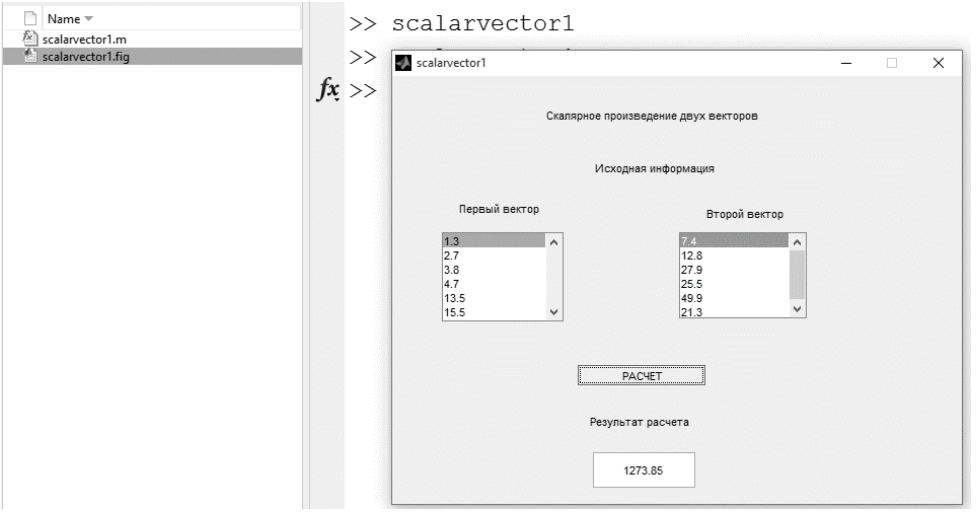


Рис. 2.133

Командное окно MATLAB после выполнения расчетов при вычислении скалярного произведения двух векторов, для задания элементов которых используется объект Listbox графического интерфейса GUIDE

Вид создаваемого в этом случае графического интерфейса представлен на рисунке 2.134, а результаты вычислений по программе (рис. 2.135) приведены на рисунке 2.137.

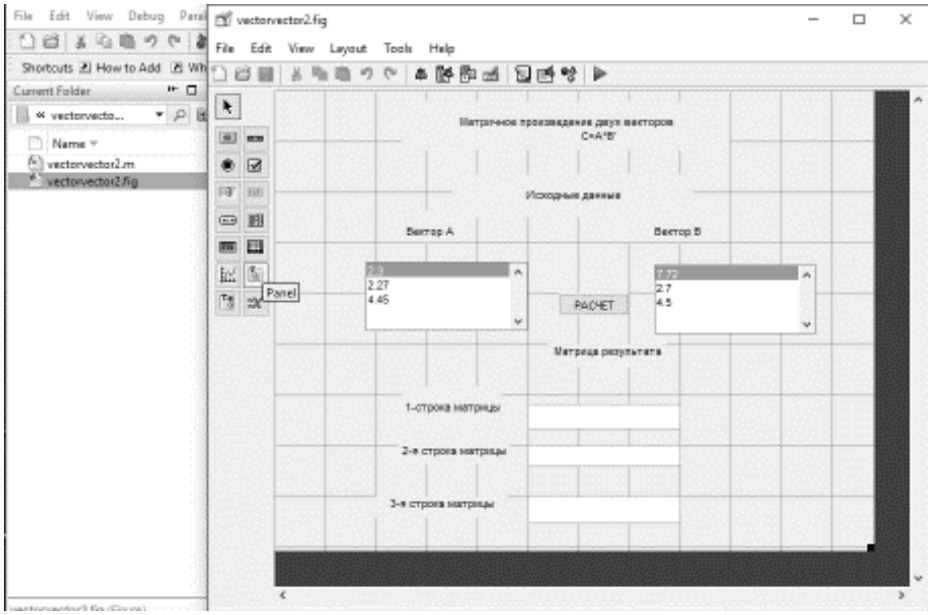


Рис. 2.134

Задание элементов двух векторов с использованием объекта Listbox графического интерфейса GUIDE и трех объектов Edit для вывода строк результирующей матрицы с тремя строками векторного произведения

```
function varargout = vectorvector2(varargin)
% VECTORVECTOR2 MATLAB code for vectorvector2.fig
%      VECTORVECTOR2, by itself, creates a new VEC-
TORVECTOR2 or raises the existing
%      singleton*.
%
%      H = VECTORVECTOR2 returns the handle to a new VEC-
TORVECTOR2 or the handle to
%      the existing singleton*.
%
%      VECTORVEC-
TOR2('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in VECTORVECTOR2.M
with the given input arguments.
%
%      VECTORVECTOR2('Property','Value',...) creates a new
VECTORVECTOR2 or raises the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before vectorvector2_OpeningFcn gets
called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to vectorvector2_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI
allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help vectorvector2

% Last Modified by GUIDE v2.5 14-Mar-2016 01:48:32

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @vectorvector2_OpeningFcn, ...
                   'gui_OutputFcn',  @vectorvector2_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before vectorvector2 is made visible.
function vectorvector2_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to vectorvector2 (see
VARARGIN)

% --- Executes on selection change in listbox3.
function listbox3_Callback(hObject, eventdata, handles)
% hObject    handle to listbox3 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDA-
TA)

% Hints: contents = cellstr(get(hObject,'String')) returns
listbox3 contents as cell array
%        contents{get(hObject,'Value')} returns selected item from
listbox3


% --- Executes during object creation, after setting all proper-
ties.
function listbox3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listbox3 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

% Hint: listbox controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in listbox4.
function listbox4_Callback(hObject, eventdata, handles)
% hObject    handle to listbox4 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDA-
TA)

% Hints: contents = cellstr(get(hObject,'String')) returns
listbox4 contents as cell array
%        contents{get(hObject,'Value')} returns selected item from
listbox4


% --- Executes during object creation, after setting all proper-
ties.
function listbox4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listbox4 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

% Hint: listbox controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in listbox5.
function listbox5_Callback(hObject, eventdata, handles)
% hObject    handle to listbox5 (see GCBO)
```

Рис. 2.135 (начало)

Программный код *m*-файла vectorvector2.m вычисления векторного произведения двух векторов с использованием графического интерфейса GUIDE

```
% Choose default command line output for vectorvector2
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes vectorvector2 wait for user response (see
UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command
line.
function varargout = vectorvector2_OutputFcn(hObject, event-
data, handles)
% varargout  cell array for returning output args (see
VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on selection change in listbox1.
function listbox1_Callback(hObject, eventdata, handles)
% hObject    handle to listbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns listbox1
contents as cell array
%        contents{get(hObject,'Value')} returns selected item from
listbox1


% --- Executes during object creation, after setting all properties.
function listbox1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

% Hint: listbox controls usually have a white background on Win-
dows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in listbox2.
function listbox2_Callback(hObject, eventdata, handles)
% hObject    handle to listbox2 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns listbox2
contents as cell array
%        contents{get(hObject,'Value')} returns selected item from
listbox2
```

```
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDA-
TA)

% Hints: contents = cellstr(get(hObject,'String')) returns
listbox5 contents as cell array
%        contents{get(hObject,'Value')} returns selected item from
listbox5


% --- Executes during object creation, after setting all proper-
ties.
function listbox5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listbox5 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

% Hint: listbox controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDA-
TA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1
as a double


% --- Executes during object creation, after setting all proper-
ties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all Cre-
ateFcns called

% Hint: edit controls usually have a white background on Win-
dows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDA-
TA)
```

Рис. 2.135 (продолжение)
Программный код *m*-файла vectorvector2.m вычисления векторного произведения двух векторов с использованием графического интерфейса GUIDE

```
% --- Executes during object creation, after setting all properties.
function listbox2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listbox2 (see GCBO)
% eventdata  reserved — to be defined in a future version of
MATLAB
% handles    empty — handles not created until after all Cre-
ateFcns called

% Hint: listbox controls usually have a white background on Win-
dows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved — to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
clc;
matrix1='';matrix2='';matrix3='';
s1=get(handles.listbox1,'String');
s2=get(handles.listbox2,'String');
ss1=str2double(s1);
ss2=str2double(s2);
matrix=ss1*ss2';
matrix1=[matrix(1,1) matrix(1,2) matrix(1,3)];
matrix1=num2str(matrix1);
set(handles.edit1,'String',matrix1);
matrix2=[matrix(2,1) matrix(2,2) matrix(2,3)];
matrix2=num2str(matrix2);
set(handles.edit2,'String',matrix2);
matrix3=[matrix(3,1) matrix(3,2) matrix(3,3)];
matrix3=num2str(matrix3);
set(handles.edit3,'String',matrix3);
```

```
% Hints: get(hObject,'String') returns contents of edit2 as text
%       str2double(get(hObject,'String')) returns contents of edit2
as a double


% --- Executes during object creation, after setting all proper-
ties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved — to be defined in a future version of
MATLAB
% handles    empty — handles not created until after all Cre-
ateFcns called

% Hint: edit controls usually have a white background on Win-
dows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved — to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDA-
TA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%       str2double(get(hObject,'String')) returns contents of edit3
as a double


% --- Executes during object creation, after setting all proper-
ties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved — to be defined in a future version of
MATLAB
% handles    empty — handles not created until after all Cre-
ateFcns called

% Hint: edit controls usually have a white background on Win-
dows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

Рис. 2.135 (окончание)

Программный код *m*-файла vectorvector2.m вычисления векторного произведения двух векторов с использованием графического интерфейса GUIDE

В соответствии с программным кодом (рис. 2.135 — функция pushbutton1Callback), результирующая квадратная матрица представляется графическим интерфейсом построчно (рис. 2.134, 2.136, 2.137) с помощью трех векторов: matrix1, matrix2, matrix3 (соответственно первая, вторая и третья строки результирующей матрицы). При этом сначала элементы векторов «очищаются»:

matrix1 = ‘ ’; matrix2 = ‘ ’; matrix3 = ‘ ’,

а затем, после вычислений, формируются одномерные массивы:

matrix = ss1 * ss2’;

matrix1 = [matrix(1,1) matrix(1,2) matrix(1,3)];

matrix2 = [matrix(2,1) matrix(2,2) matrix(2,3)];

matrix3 = [matrix(3,1) matrix(3,2) matrix(3,3)],

после чего они преобразуются в текстовый формат:

matrix1 = num2str (matrix1);

matrix2 = num2str (matrix2);

matrix3 = num2str (matrix3),

и выводятся с использованием следующих операторов:

set (handles.edit1, ‘string’, matrix1);

set (handles.edit2, ‘string’, matrix2);

set (handles.edit3, ‘string’, matrix3).

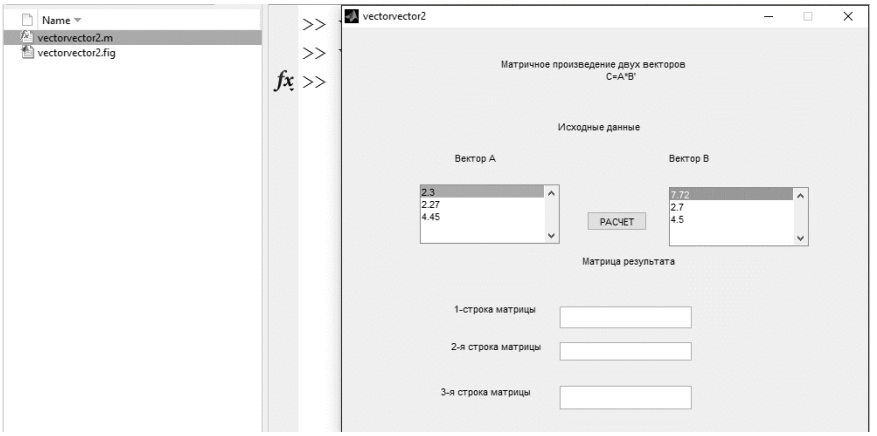


Рис. 2.136

Командное окно MATLAB после запуска программы расчета произведения двух векторов до выполнения расчетов, когда для задания векторов используется объект Listbox графического интерфейса GUIDE

2.10.12.6. Переключатели для выбора различных вариантов вычислений, использующие объект Radio Button в окне графического интерфейса GUIDE

В окне графического интерфейса GUIDE три раза среди объектов, расположенных слева на дисплее, выбирается объект Push Button, который в виде «пустой» кнопки три раза размещается в поле окна. Кнопки размещаются друг под другом (рис. 2.138), и при запуске программы будет работать тот вариант

вычислительного или другого процесса, который будет соответствовать «кнопке», отмеченной левой кнопкой мыши, и внутри которой появится точка.

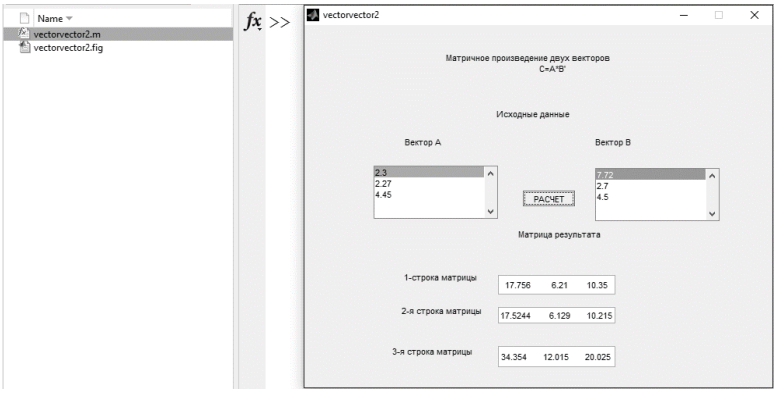


Рис. 2.137

Командное окно MATLAB после выполнения расчетов при вычислении векторного произведения двух векторов, для задания элементов которых используется объект Listbox графического интерфейса GUIDE

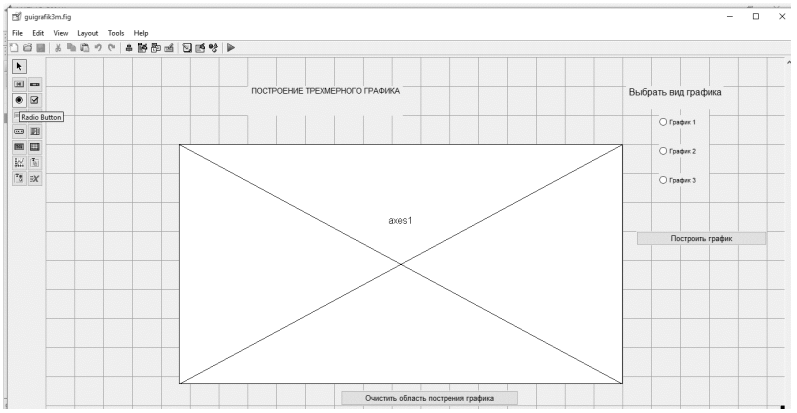


Рис. 2.138

Создание окна графического интерфейса GUIDE при использовании переключателя Push Button для выбора одного из трех свойств графиков с двумя способами запуска программы — «Построить график» и «Очистить область построения графика»

Рядом с каждой кнопкой в поле интерфейса появится слово “Push Button” (переключатель), а для задания характеристики каждого из трех переключателей необходимо выделить его и нажать правую кнопку мыши, выбрав из выпадающего меню опцию Property Inspector (Инспектор свойств) (рис. 2.139), в результате чего слева появится список свойств с тремя важнейшими свойствами (рис. 2.140):

- Tag — имя объекта, задаваемое по умолчанию, например pushbutton1, pushbutton2 и т. п., которое использует *m*-файл создаваемой программы для идентификации переключателя. В рассматриваемом примере переключателям присвоены имена (Tag) knp1, knp2 и knp3.

- String — строка, выводимая рядом с переключателем. В рассматриваемом примере рядом с переключателями соответственно с именами knp1, knp2 и knp3 выводятся названия (String): График1, График2 и График3;
- Value — число, позволяющее определить, какой из переключателей выделен, у выделенного переключателя значение Value совпадает со значением свойства Max, которое по умолчанию равно 1.

В коде программы для трех переключателей необходимо записать:

if get (handles.knp1, ‘Value’) = = 1; if get (handles.knp2, ‘Value’) = = 1;

if get (handles.knp3, ‘Value’) = = 1.

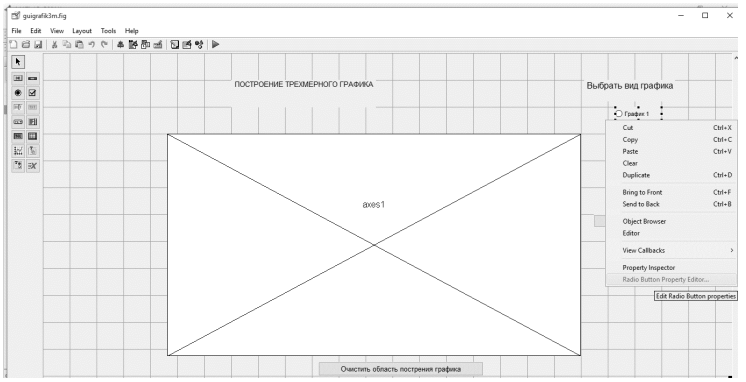


Рис. 2.139

Выбор Property Inspector (Инспектора свойств) для переключателя График1 путем выделения его левой кнопкой мыши и нажатия на правую кнопку мыши

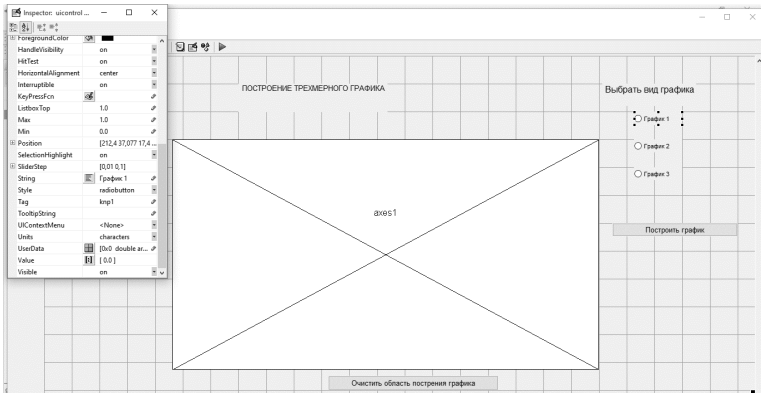


Рис. 2.140

Изменение свойства String (строка) на График1 и Tag (имя объекта) на knp1 в Property Inspector (Инспектор свойств)

В рассматриваемом примере необходимо выбрать один из трех видов графиков:

$$z = \left(\frac{x-3}{100}\right)^2 - (y-x) + \exp(20*(y-x)) \text{ — График1;}$$

$$z = 100(x^2 - y^2) + (1 - x)^2 \text{ — График2;}$$

$$z = (x - y)^2 + \left(\frac{x + y - 10}{3}\right)^2 \text{ — График3.}$$

С этой целью в окне графического интерфейса размещаются (рис. 2.137):

- текстовый объект для обозначения заголовка: «Построение трехмерного графика»;
- текстовый объект для выбора вида заголовка: «Выбрать вид графика»;
- объект Push Button для запуска программы построения графика с обозначением «Построить график»;
- объект Push Button с именем pushbutton2 для очистки области построения графика с обозначением «Очистить область построения графика».

В *m*-файле программы (рис. 2.141) в функции pushbutton1 выбор графика, который должен быть построен, осуществляется с использованием условного оператора следующего вида:

if get (handles.knp1, ‘Value’) =1

для кнопки с именем (Tag) knp1.

Соответственно при выборе графиков с помощью кнопки с именем (tag) knp2 или knp3 в условном операторе последние записываются вместо knp1.

Для запуска программы с помощью объекта Push Button (Очистка области построения графика) с именем (Tag) pushbutton2 в соответствующую функцию *m*-файла программы (рис. 2.141) включена функция “cla” — очистка текущих осей графика, а также очистка графика от сетки grid off.

При этом создается надпись «Область трехмерного графика очищена» с использованием оператора:

set (handles.text1, ‘String’, «ОБЛАСТЬ ТРЕХМЕРНОГО ГРАФИКА ОЧИЩЕНА»).

Результаты работы программы (рис. 2.141) при запуске с помощью функции pushbutton1 (Построить график) представлены:

на рисунке 2.142 — до выполнения программы;

на рисунке 2.143 — при выборе переключателя с именем knp1 (График1);

на рисунке 2.144 — при выборе переключателя с именем knp2 (График2);

на рисунке 2.145 — при выборе переключателя с именем knp3 (График3).

```
function varargout = guigrafik3m(varargin)
% GUIGRAFIK3M MATLAB code for guigrafik3m.fig
%      GUIGRAFIK3M, by itself, creates a new GUIGRAFIK3M or raises the existing
%      singleton*.
%
%      H = GUIGRAFIK3M returns the handle to a new GUIGRAF-
```

```
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.text1,'String','ПОСТРОЕНИЕ ТРЕХМЕРНОГО
```

Рис. 2.141 (начало)

Программный код *m*-файла программы построения трехмерного графика с использованием переключателя — объекта Push Button графического интерфейса GUIDE

```
IK3M or the handle to
%     the existing singleton*.
%
%     GUIGRAFIK3M('CALLBACK',hObject,eventData,handles,...)
calls the local
%     function named CALLBACK in GUIGRAFIK3M.M with the
given input arguments.
%
%     GUIGRAFIK3M('Property','Value',...) creates a new GUI-
GRAFIK3M or raises the
%     existing singleton*.  Starting from the left, property value pairs
are
%     applied to the GUI before guigrafik3m_OpeningFcn gets called.
An
%     unrecognized property name or invalid value makes property
application
%     stop.  All inputs are passed to guigrafik3m_OpeningFcn via
varargin.
%
%     *See GUI Options on GUIDE's Tools menu.  Choose "GUI al-
lows only one
%     instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help guigrafik3m

% Last Modified by GUIDE v2.5 17-Mar-2016 19:33:30

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @guigrafik3m_OpeningFcn, ...
                   'gui_OutputFcn',  @guigrafik3m_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before guigrafik3m is made visible.
function guigrafik3m_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to guigrafik3m (see
VARARGIN)

% Choose default command line output for guigrafik3m
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes guigrafik3m wait for user response (see
UIRESUME)
% uiwait(handles.figure1);
ГРАФИКА');
if get(handles.knp1,'Value')==1
[x,y]=meshgrid(2.9:0.01:3.1,2.8:0.01:2.9);
z=(1/100*(x-3)).^2-(y-x)+exp((y-x)*20);
mesh(x,y,z);
title('z=(1/100*(x-3))^2-(y-x)+exp((y-x)*20)');
xlabel('x');
ylabel('y');
zlabel('z');
elseif get(handles.knp2,'Value')==1
[x,y]=meshgrid(-2:0.1:2,-3:0.1:3);
z=100*(x.^2-y).^2+(1-x).^2;
mesh(x,y,z);
title('z=100*(x^2-y)^2+(1-x)^2');
xlabel('x');
ylabel('y');
zlabel('z');
elseif get(handles.knp3,'Value')==1
[x,y]=meshgrid(4:0.1:6,4:0.1:6);
z=(x-y).^2+((x+y-10)*(1/3)).^2;
mesh(x,y,z);
title('z=(x-y)^2+((x+y-10)*(1/3))^2');
xlabel('x');
ylabel('y');
zlabel('z');
end


% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
cla;
grid off;
set(handles.text1,'String','ОБЛАСТЬ ТРЕХМЕРНОГО ГРА-
ФИКА ОЧИЩЕНА');
%if get(handles.knp1,'Value')==1

% --- Executes during object creation, after setting all properties.
function text1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all CreateFcns
called


% --- Executes on button press in knp1.
function knp1_Callback(hObject, eventdata, handles)
% hObject    handle to knp1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of knp1


% --- Executes on button press in knp2.
function knp2_Callback(hObject, eventdata, handles)
% hObject    handle to knp2 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of knp2
```

Рис. 2.141 (продолжение)

Программный код *m*-файла программы построения трехмерного графика с использованием переключателя — объекта Push Button графического интерфейса GUIDE

```
% --- Outputs from this function are returned to the command line.
function varargout = guigrafik3m_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see
VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
```

```
% --- Executes on button press in knp3.
function knp3_Callback(hObject, eventdata, handles)
% hObject    handle to knp3 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of knp3
```

Рис. 2.141 (окончание)

Программный код *m*-файла программы построения трехмерного графика с использованием переключателя — объекта Push Button графического интерфейса GUIDE

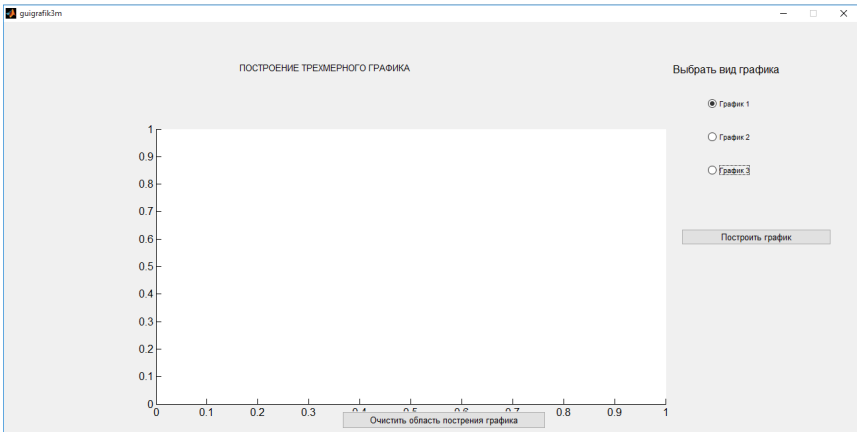


Рис. 2.142

Запуск программы guigrafik3m (рис. 2.141) с выбором переключателя в позиции График1 до нажатия на клавишу «Построить график» или «Очистить область построения графика»

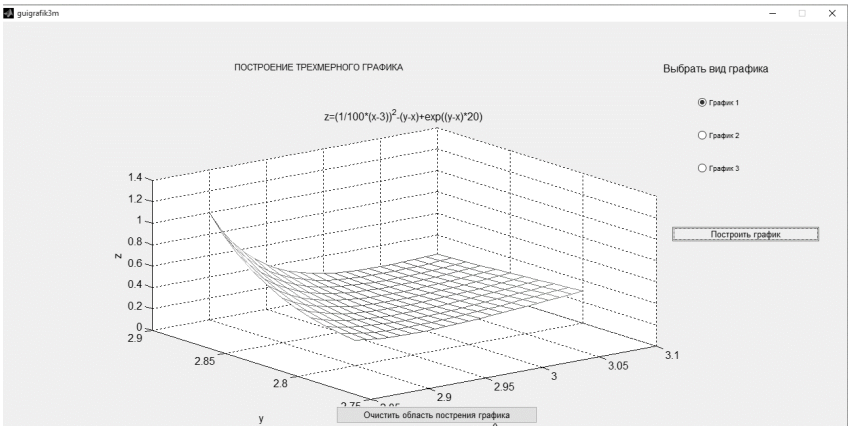


Рис. 2.143

Результат работы программы guigrafik3m (рис. 2.141) с выбором позиции переключателя График1 после нажатия на клавишу «Построить график» левой кнопкой мыши

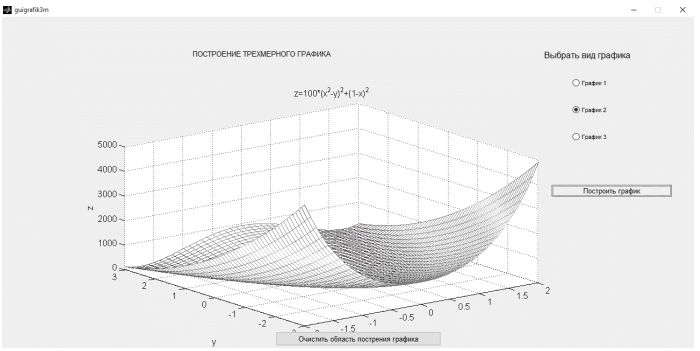


Рис. 2.144

Результат работы программы guigrafik3m (рис. 2.141) с выбором позиции переключателя График2 после нажатия на клавишу «Построить график» левой кнопкой мыши

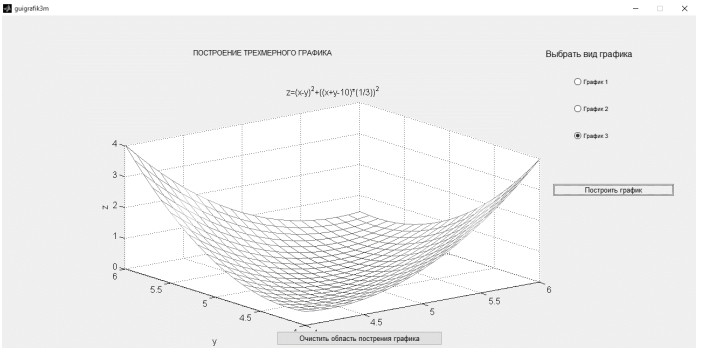


Рис. 2.145

Результат работы программы guigrafik3m (рис. 2.141) с выбором позиции переключателя График3 после нажатия на клавишу «Построить график» левой кнопкой мыши

Результат работы программы при запуске с помощью функции pushbutton2 (Очистка области построения графика) после построения График3 представлен на рисунке 2.145.

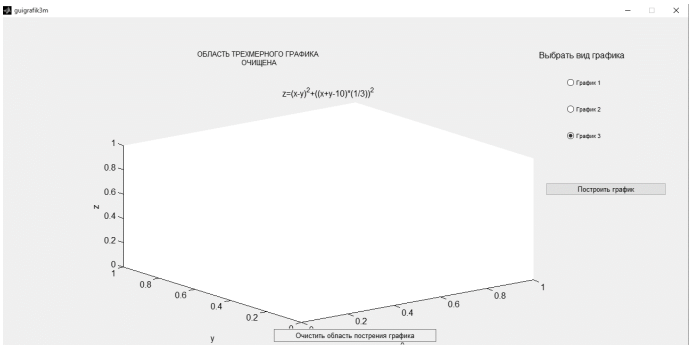


Рис. 2.146

Результат работы программы guigrafik3m (рис. 2.141) с выбором позиции переключателя График3 после нажатия на клавишу «Очистить область построения графика» левой кнопкой мыши

ГЛАВА 3. ПРИМЕНЕНИЕ РЕШАТЕЛЕЙ ПАКЕТА MATLAB ДЛЯ РЕАЛИЗАЦИИ ЧИСЛЕННЫХ МЕТОДОВ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ ПРИ МОДЕЛИРОВАНИИ ХИМИКО-ТЕХНОЛОГИЧЕСКИХ ПРОЦЕССОВ

В общем случае для решения расчетных задач химической термодинамики, физикохимии и химической технологии используются численные методы вычислительной математики. Применение аналитических методов возможно только для решения простейших задач или для решения фрагментов (частей) сложных задач. Поэтому в системах компьютерной математики (СКМ) типа MATLAB представлены, как правило, численные методы в виде решателей, позволяющие находить решения в основном итерационными методами последовательных приближений.

Название решателя, как правило, соответствует смыслу решаемой задачи. Например, при поиске корня нелинейного уравнения вида $f(x) = 0$ (где $f(x)$ — нелинейная функция аргумента x), т. е. нуля функции $f(x)$, решатель носит название «fzero». При запуске решателя в программе на *m*-языке MATLAB выполняется сложный алгоритм или набор алгоритмов, которые позволяют найти решение задачи — определить корень уравнения в рассматриваемом случае. Следует отметить, что функция $f(x)$ может быть произвольной и должна быть представлена в программе в соответствии с синтаксисом *m*-языка программирования, используемого в среде MATLAB.

В общем виде обращение к выполнению (запуску) решателя имеет вид:

[РЕЗУЛЬТАТ] = НАЗВАНИЕ РЕШАТЕЛЯ (‘название функции’, начальные приближения (условия) или их интервал, **optimset**)

Справа от знака равенства записывается название стандартного решателя MATLAB (например, fzero, а в скобках после названия в кавычках указывается название конкретного вида функции (можно также перед именем функции поставить знак @, например @furavn, что соответствует ‘furavn’), а после запятой — стартовые значения (начальные приближения/начальные условия или их интервал) переменных, инициализирующие процесс вычислений, затем следует специальная функция optimset, и скобки закрываются.

Слева от знака равенства в квадратных скобках выводятся («возвращаются») следующие конкретные результаты вычислений:

- первое значение — решение уравнения (скалярное значение — x^*) или уравнений (векторное значение — $\overline{x}$); в случае дифференциальных уравнений выводится матрица, так как одновременно с указанными выше значениями возвращаются и значения независимой переменной;
- второе значение — финальное значение функции $f(x^*)$ или функций $\overline{f}(\overline{x}^*)$ — для систем уравнений, полученных в результате решения;

- третье значение — признак корректного окончания вычислительного процесса. Это одно число, при равенстве которого единице вычисление считается успешным и к результатам нет претензий;
- четвертое значение — краткая характеристика параметров вычислительного процесса.

Важным атрибутом решателей, реализующих итерационные процессы, является специальная функция optimset, которая позволяет отслеживать процесс изменения искомых переменных при последовательных приближениях к решению, а также дает возможность устанавливать требуемую точность вычислений и максимальное число итераций. В отдельных случаях функция optimset позволяет установить конкретный алгоритм численного метода для проведения расчетов.

Следует отметить, что характерной особенностью практических расчетных задач в химии и химической технологии является их высокая вычислительная сложность, что связано с большой размерностью пространства поиска решений, сложной топологией области допустимых решений, а также нелинейностью, недифференцируемостью, многоэкстремальностью (мультимодальностью), овражностью, плохой формализуемостью функций (в том числе и оптимизируемых), используемых при реализации численных методов.

С самой общей точки зрения, именно указанные особенности объясняют отсутствие универсальных алгоритмов решения отдельных задач вычислительной математики и, напротив, наличие чрезвычайно большого числа алгоритмов, их модификаций и гибридизаций.

Поэтому в среде MATLAB часто представлено большое число решателей для решения одной и той же задачи вычислительной математики, вследствие чего приходится производить оценку применимости того или иного (тех или иных) решателя-алгоритма для решения конкретной задачи и в итоге выбирать наиболее эффективный метод решения или тип решателя.

3.1. Вычисление производных и интегралов

При определении значений производных и интегралов различных функций $f(x)$ их необходимо представлять в дискретном (табличном виде) с определенным, по возможности наименьшим, шагом дискретизации h по аргументу x. Чем меньше шаг дискретизации h, тем точнее результат расчета численным методом, и одновременно возрастает объем вычислений. Поэтому для достижения требуемой точности расчетов на практике приходится варьировать шаг дискретизации h, постоянно уменьшая его до тех пор, пока последующее уменьшение шага h не приводит к несущественному изменению результатов.

3.1.1. Вычисление производных с применением метода разделенных разностей

Одним из методов определения производных различных порядков является метод разделенных разностей. Он связан с существенно меньшими вычислительными затратами, чем, например, использование аппроксимирующих

функций, позволяющих описать функции, представленные в табличном виде, аналитически, в простейшем случае — в полиномиальном виде. Несмотря на то, что дифференцирование таких функций является наиболее удобным, определение вида аппроксимирующей функции и ее коэффициентов часто связано с определенными трудностями.

Другой способ определения производных связан с применением метода разделенных разностей, при этом функция $y = f(x)$ должна быть представлена в табличном виде $y(i)= f(x(i))$, $i = 1, ..., n – 1$, по возможности с наименьшим шагом дискретизации $h(i) = x(i) – x(i +1)$, где $(n – 1)$ — число шагов. Для приближенного определения производных первого, второго, третьего и т. д. порядков в каждой дискретной точке аргумента $x(i)$ (так называемом узле сетки) используются разделенные разности соответственно первого, второго, третьего и т. д. порядков, которые для произвольных точек сетки i, j, k вычисляются по следующим формулам.

1. Разделенная разность первого порядка — $F(x(i), x(j))$:

$$F(x(i), x(j)) = [f(x(i)) – f(x(j)]/[x(i) – x(j)]. \quad (3.1.1)$$

Для ее определения необходимы два значения аргументов $x(i)$ и $x(j)$ и соответственно два значения функций $f(x(i))$ и $f(x(j))$.

2. Разделенная разность второго порядка — $F(x(i), x(j), x(k))$:

$$F(x(i), x(j), x(k)) = [F(x(i), x(j)) – F(x(j), x(k))]/[x(i) – x(k)]. \quad (3.1.2)$$

Для ее определения необходимы три значения аргументов $x(i)$, $x(j)$ и $x(k)$ и соответственно два значения разделенных разностей первого порядка $F(x(i), x(j))$ и $F(x(j), x(k))$.

3. Разделенная разность третьего порядка — $F(x(i), x(j), x(k), x(l))$:

$$F(x(i), x(j), x(k), x(l)) = [F(x(i), x(j), x(k)) – F(x(j), x(k), x(l))]/[x(i) – x(l)]. \quad (3.1.3)$$

Для ее определения необходимы четыре значения аргументов $x(i)$, $x(j)$, $x(k))$ и $x(l)$ и соответственно два значения разделенных разностей второго порядка $F(x(i), x(j), x(k))$ и $F(x(j), x(k), x(l)$.

Аналогично определяются разделенные разности более высоких порядков.

Формула для определения производной произвольного k-го порядка на основе разделенных разностей в общем случае имеет вид:

$$d(k)y/dx = k!*F(x(1), x(2), ..., x(k + 1)). \quad (3.2)$$

Отсюда следует, что для определения производной k-го порядка наряду со значением k-факториала (k!) необходимо определить разделенную разность k-го порядка по $(k + 1)$ дискретным точкам сетки на оси абсцисс (ось x).

3.1. Формула для определения первой производной по двум узлам сетки $x(1)$ и $x(2)$, а также соответственно $f(x(1))$ и $f(x(2))$ имеет вид

$$dy/dx = 1!*F(x(1), x(2)) = [f(x(1)) – f(x(2)]/[x(1) – x(2)]. \quad (3.3.1)$$

3.2. Формула для определения второй производной по трем узлам сетки $x(1)$, $x(2)$ и $x(3)$, а также соответственно $f(x(1))$, $f(x(2))$ и $f(x(3))$ имеет вид

$$d(2)y/dx(2) = 2!*F\left(x(1),x(2),x(3)\right) =$$
$$= 2*\left[F\left(x(1),x(2)\right) - F\left(x(2),x(3)\right)\right] / \left[x(1) - x(3)\right] = \quad (3.3.2)$$
$$= 2*\left[f\left(x(1)\right) - 2*f\left(x(2)\right) + f\left(x(3)\right)\right] / \left[x(1) - x(3)\right].$$

3.3. Формула для определения третьей производной по четырем узлам сетки $x(1)$, $x(2)$, $x(3)$ и $x(4)$, а также соответственно $f(x(1))$, $f(x(2))$, $f(x(3))$ и $f(x(4))$ имеет вид

$$d(3)y/dx(3) = 3!*F\left(x(1),x(2),x(3),x(4)\right) =$$
$$= 6*\left[F\left(x(1),x(2),x(3)\right) - F\left(x(2),x(3),x(4)\right)\right] / \quad (3.3.3)$$
$$/\left[x(1) - x(4)\right] = 6*\left[f\left(x(1)\right) - 2*f\left(x(2)\right) + f\left(x(3)\right)\right] / \left[x(1) - x(4)\right].$$

Как правило, для узлов сетки справедливо, что $h = x(1) - x(2) = x(2) - x(3) = x(3) - x(4)$, и тогда формулы для определения производных могут быть записаны:

$$dy/dx = \left[f\left(x(1)\right) - f\left(x(2)\right)\right] / h$$
$$d(2)y/dx(2) = [f(x(1)) - 2*f(x(2)) + f(x(3))] / h^2 \quad (3.3)$$
$$d(3)y/dx(3) = [2*f(x(1)) - 6*f(x(2)) + 6*f(x(3)) - 2*f(x(4))] / h^3.$$

Очевидно, что с уменьшением шага сетки h точность вычисления производных должна возрастать.

Приближенное определение первой и второй производных показано на простейшем примере функции $y = \sin(x)$, для которой точное значение первой и второй производных соответственно определяются как $dy1 = y' = \cos(x)$ и $dy2 = y'' = -\sin(x)$, а код программы вычисления точных и приближенных значений приведен на рисунке 3.1.

Значения производных вычисляются в интервале изменения аргумента x [0 – 3.14] с постоянным шагом дискретизации $h = 0.1$.

```
function PROIZV
clc;
clear all;
close all;
%
%-----------------------------------------------------------------
%I.Сравнение результатов расчета первых и вторых производных с их точным значением
%      в узлах        сетки для простейшей функции y=f(x)=sin(x)
%-----------------------------------------------------------------
%II.1. Задается шаг сетки для аргумента x:
h=0.1;
%II.3. Определяется диапазон изменения аргумента [0-3.14]:
x=0:h:3.14;
%II.3. Определяется число узлов сетки [0-3.14]:
n=length(x);
%II.4. Точное значение первой и второй производной равно:
dsinx1=cos(x);
dsinx2=-sin(x);
%II.4. Определение приближенных значений первой производной в узлах сетки,
```

Рис. 3.1 (начало)
Код программы определения первой и второй производных для функции $\sin(x)$ методом разделенных разностей

```
% сравнение с точным и вычисление абсолютной погрешности:
for i=1:n-2
dy1(i)=(sin(x(i+1))-sin(x(i)))/h;
er1(i)=abs(dsinx1(i)-dy1(i));
dy2(i)=(sin(x(i+2))-2*sin(x(i+1))+sin(x(i)))/h^2;
er2(i)=abs(dsinx2(i)-dy2(i));
end
for is=1:3
   hfig(is)=figure;
end
figure(hfig(1));
plot(x([1:(n-2)]),dy1([1:(n-2)]),'--',x([1:(n-2)]),dsinx1([1:(n-2)]),'-');
title('Сравнение расч. 1-ой произв. ф-ции y=sin(x) прибл. и точн. методом');
xlabel('x-независимая переменная;h=0.1 — шаг дискретизации');
ylabel('1- ая произ.: методы — точн.(-) и прибл. (-.) ');
figure(hfig(2));
plot(x([1:(n-2)]),dy2([1:(n-2)]),'--',x([1:(n-2)]),dsinx2([1:(n-2)]),'-');
title('Сравнение расч. 2-ой произв. ф-ции y=sin(x) прибл. и точн. методом');
xlabel('x-независимая переменная;h=0.1 — шаг дискретизации');
ylabel('2- ая произ.: методы — точн.(-) и прибл. (- -) ');
%figure(hfig(3));
%plot(x([1:(n-3)]),dy3([1:(n-3)]),'--',x([1:(n-3)]),dsinx3([1:(n-3)]),'-');
figure(hfig(3));
plot(x([1:(n-2)]),er1([1:(n-2)]),'-.',x([1:(n-2)]),er2([1:(n-2)]),'--');
title('Сравнение погрешностей при прибл. расч. 1-ой и 2-ой произв. ф-ции y=sin(x) ');
xlabel('x-независимая переменная;h=0.1 — шаг дискретизации');
ylabel('Погрешности 1- ой произ.(-.) и 2-ой произв. (- -) ');
end
```

Рис. 3.1 (окончание)

Код программы определения первой и второй производных для функции sin(x) методом разделенных разностей

Результаты сравнения точного и приближенного расчета указанных производных в диапазоне изменения аргумента x от 0 до 3.14 приведены на рисунке 3.2, 1 — для первой производной, и на рисунке 3.2, 2 — для второй производной, а графическое представление абсолютных погрешностей этих же производных — на рисунке 3.2, 3.

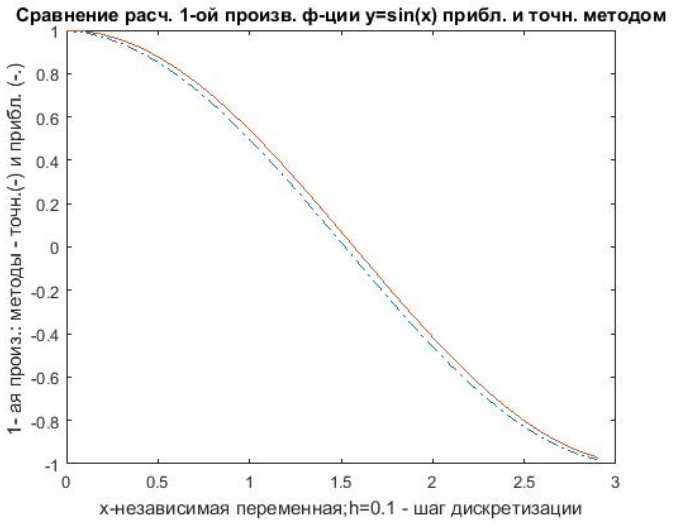


Рис. 3.2.1

Сравнение результатов определения первой производной функции sin(x) точным и численным методами

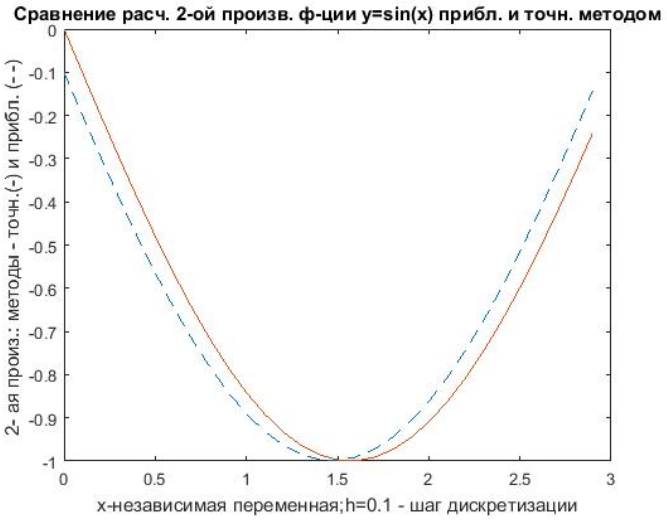


Рис. 3.2, 2

Сравнение результатов определения второй производных функции sin(*x*) точным и приближенным методами

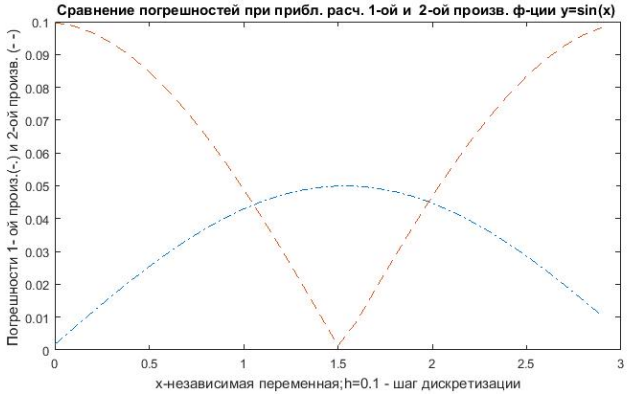


Рис. 3.2, 3

Сравнение погрешностей определения первой и второй производной численным методом разделенных разностей

3.1.2. Вычисление интегралов с применением решателей “trapz” и “quad”

В общем случае решатели “trapz” и “quad” реализуют алгоритм расчета интегралов по методу Ньютона — Котеса соответственно первого и второго порядков. Весь интервал интегрирования разбивается на *n* подынтервалов, в каждом из которых подынтегральная функция интерполируется соответственно многочленами первого порядка (решатель “trapz”) и второго порядка (решатель “quad”).

В первом случае интерполяция подынтегральной функции осуществляется по двум дискретным точкам для каждого шага интегрирования (дискретиза-

ции) h, а во втором случае — по трем точкам, включая, кроме крайних точек, и среднюю ее точку.

Решатель “trapz” реализует метод трапеций вычисления определенного интеграла и записывается в программе в виде:

$$\text{Integr1} = \text{trapz}(x, y), \tag{3.4}$$

где x — значение аргумента в дискретном виде $x(i)$; y — соответствующие значения функции в дискретных точках $y(i)$; $i = 1, ..., n$; n — число дискретных точек; Integr1 — результат расчета интеграла.

Очевидно, чем больше дискретных точек и меньше шаг интегрирования в интервале интегрирования, тем точнее расчет.

В примере на рисунке 3.3 приведен код программы для вычисления определенного интеграла с постоянным и переменным шагом, значение которого по формуле Ньютона — Лейбница равно 33.6667:

$$S = \int_{5}^{13} \sqrt{2x-1}. \tag{3.5}$$

При уменьшении постоянного шага интегрирования и увеличении точности расчета ($h = 1$; $h = 0.5$; $h = 0.1$), как и следовало ожидать, приближенные значения интеграла приближаются к точному значению (3.5) — 33.6556; 33.6639; 33.6666 (результаты соответственно $S1$, $S2$, $S3$ — на рис. 3.4).

Переменный шаг интегрирования может быть задан с использованием генератора равномерно распределенных случайных величин “rand”, как, например, для интеграла, точное значение которого равно 2:

$$S4 = \int_{0}^{3.14} \sin(x). \tag{3.6}$$

В этом случае, как следует из программы (рис. 3.3), полученные случайные величины должны быть упорядочены в возрастающей последовательности с применением функции “sort”. Дискретные точки в интервале изменения аргумента x[0; 3.14] задаются следующим образом:

$$x = \text{sort}(\text{rand}(1,101))*3.14), \tag{3.7}$$

а соответствующие значения y определяются как $y = \sin(x)$ (результат $S4$ приведен на рис. 3.4).

```
function INTEGRAL_1
clc;
clear all;
close all;
%
disp('-----------------------------------------------------------------');
disp('1. Интегрирование с применением решателя "trapz" с постоянным шагом интегрирования');
%       Подынтегральное выражение: (2*x-1)^(1/2)
%-----------------------------------------------------------------
```

Рис. 3.3 (начало)

Код программы вычисления определенного интеграла с использованием решателя “trapz” с постоянным и переменным шагом интегрирования

```
disp('-----------------------------------------------------------------');
disp('1.1.Подынтегральное выражение: sqrt(2*x-1)');
disp('-----------------------------------------------------------------');
%Задание интервала интегрирования
a=5;b=13;
%a) шаг дискретизации равен 1:
h=1
x=a:b;
y=sqrt(2*x-1);
S1=trapz(x,y)
%b) шаг дискретизации равен 0.5:
h=0.5
x=a:h:b;
y=sqrt(2*x-1);
S2=trapz(x,y)
%c) шаг дискретизации равен 0.1:
h=0.1
x=a:h:b;
y=sqrt(2*x-1);
S3=trapz(x,y)
%-----------------------------------------------------------------
disp('-----------------------------------------------------------------');
disp('3. Интегрирование с применением решателя "trapz" со случайным шагом интегрирования');
 %Подынтегральное выражение: sin(x)
disp('-----------------------------------------------------------------');
 %-----------------------------------------------------------------
disp('3.1. Подынтегральное выражение: sin(x)');
disp('-----------------------------------------------------------------');
%Задание интервала интегрирования
a=0;b=3.14;
x=sort(rand(1,101)*3.14);
y=sin(x);
S4=trapz(x,y)

end
```

Рис. 3.3 (окончание)

Код программы вычисления определенного интеграла с использованием решателя “trapz” с постоянным и переменным шагом интегрирования

```
-----------------------------------------
1. Интегрирование с применением решателя "trapz" с постоянным шагом интегрирования
-----------------------------------------
1.1. Подынтегральное выражение: sqrt(2*x-1)
-----------------------------------------

h = 1

S1 = 33.6556

h = 0.5000

S2 = 33.6639

h = 0.1000

S3 = 33.6666

-----------------------------------------
3. Интегрирование с применением решателя "trapz" со случайным шагом интегрирования
-----------------------------------------
3.1. Подынтегральное выражение: sin(x)
-----------------------------------------

S4 = 1.9979

>>
```

Рис. 3.4

Результаты расчета определенных интегралов с применением решателя “trapz” с постоянным и переменным шагом интегрирования

Решатель “quad” реализует метод Симпсона с автоматическим выбором шага и точностью по умолчанию 10^{-6}.

Он записывается (код программы представлен на рис. 3.5) с указанием вида функции, например sin(*x*), и пределов интегрирования, например, [0; 3.14], в следующем виде:

$$S1 = \text{quad('sin}(x)\text{'}, 0, 3.14). \quad (3.8)$$

Если необходимо уменьшить точность вычислений вместо 10^{-6} на 10^{-3}, то выражение следует записать (рис. 3.5):

$$S2 = \text{quad('sin}(x)\text{'}, 0, 3.14, 1e\text{-}3). \quad (3.9)$$

Автоматический выбор шага реализуется следующим образом: сначала интервал интегрирования разбивается на некоторое произвольное число отрезков и определяется значение интеграла. Если полученное значение интеграла не удовлетворяет предварительно заданной точности, то число отрезков удваивается и вновь вычисляется значение интеграла, и так происходит до тех пор, пока не будет достигнута требуемая точность.

Результаты расчетов с разной точностью по методу Симпсона приведены на рисунке 3.6.

Для интегрирования нестандартных подынтегральных функций, как показано в программе (рис. 3.5), для интеграла вида

$$S3 = \int_0^1 \sqrt{1+2x} \quad (3.10)$$

нужно создать функцию, *F*(*x*) в виде файла F.m (рис. 3.5, 1).

Результаты расчетов при варьировании точности интегрирования tol = 10^{-6} и tol = 10^{-9} приведены на рисунке 3.6.

```
function INTEGRAL_2
clc;
clear all;
close all;
%
disp('-------------------------------------------------------------------');
disp('1. Интегрирование с применением решателя "quad" с подынтегральной стандартной функций');
disp('-------------------------------------------------------------------');
    disp('Подынтегральное выражение: sin(x)');
disp('-------------------------------------------------------------------');
%-------------------------------------------------------------------
disp('1. Подынтегральное выражение: sin(x)');
format long;
%Задание интервала интегрирования
a=0;b=3.14;
s1=quad('sin(x)',a,b)
s2=quad('sin(x)',a,b,1e-3)
%-------------------------------------------------------------------
disp('-------------------------------------------------------------------');
disp('3. Интегрирование с применением решателя "quad" с подынтегральной нестандартной функций');
disp('-------------------------------------------------------------------');
disp('       Подынтегральное выражение: F(x)');
disp('-------------------------------------------------------------------');
disp('3.Подынтегральное выражение: F(x)=sqrt(1+2*x)');
disp('-------------------------------------------------------------------');
```

Рис. 3.5 (начало)
Код программы вычисления определенного интеграла с применением решателя “quad” со стандартной и нестандартной функциями

```
tol=1e-6
s3=quad('F',0,1)
tol=1e-9
s4=quad('F',0,1,tol)
end
```

Рис. 3.5 (окончание)

Код программы вычисления определенного интеграла с применением решателя “quad” со стандартной и нестандартной функциями

```
function y=F(x)
y=(1+2*x).^(1/2);
end
```

Рис. 3.5, 1

Код программы расчета нестандартной подынтегральной функции

```
------------------------------------------------------------------
1. Интегрирование с применением решателя "quad" с подынтегральной стандартной функций
------------------------------------------------------------------
1.Подынтегральное выражение: sin(x)

s1 = 1.999998728135827

s2 = 1.999992251012134

------------------------------------------------------------------
3. Интегрирование с применением решателя "quad" с подынтегральной нестандартной функций
------------------------------------------------------------------
3.Подынтегральное выражение: F(x)=sqrt(1+2*x)
------------------------------------------------------------------

tol = 1.000000000000000e-06

s3 = 1.398717411042350

tol = 1.000000000000000e-09

s4 = 1.398717474220476
>>
```

Рис. 3.6

Результаты расчета определенных интегралов с применением решателя “quad” со стандартной и нестандартной функциями

3.2. Решение нелинейных уравнений

Целый ряд закономерностей в химии и химической технологии могут быть описаны уравнениями с одной неизвестной. В простейших случаях путем алгебраических преобразований уравнения можно выразить искомую неизвестную и получить аналитическую формулу для определения корня (решения) уравнения. Сложнее обстоит дело с уравнениями, функции которых представляют собой многочлены произвольной степени, так называемые полиномиальные алгебраические уравнения, а также функции которых включают в себя, например, логарифмические, экспоненциальные, степенные, тригонометрические и т. п. функции, так называемые трансцендентные уравнения.

В общем случае уравнения, функции $f(x)$ которых нелинейны относительно определяемых переменных (неизвестных), к числу которых относятся поли-

номиальные и трансцендентные уравнения, принято называть нелинейными уравнениями.

Для решения полиномиальных уравнений разработаны строгие методы, которые позволяют определить все корни (решения) уравнения, не только вещественные, но и комплексные. Для определения корней полиномиального уравнения используется решатель “roots”.

Обычно алгоритмы решения нелинейных, в том числе трансцендентных уравнений позволяют определить только один вещественный корень из многих теоретически возможных и, как правило, являются приближенными и итерационными. Поэтому процесс непосредственного решения нелинейного уравнения обычно предваряет этап выбора интервала, в котором располагается физически обоснованный корень уравнения, так называемый этап отделения корня. Выбор интервала поиска корня может осуществляться исходя из физического смысла решаемой задачи, а также путем анализа нелинейной функции решения. Так как к искомому решению применяются итерационные методы последовательных приближений, задаваться может либо начальное приближение, располагаемое в интервале отделенного корня, либо сам интервал, в котором располагается искомый корень. Такой алгоритм решения нелинейного уравнения представлен в системе решателем “fzero”.

Для определения нескольких или всех корней нелинейного уравнения решатель “fzero” следует применять с разными начальными приближениями к решению уравнения, причем столько раз, сколько необходимо найти корней.

Решение полиномиального уравнения будет проиллюстрировано на примере определения мольных объемов жидкости и пара по кубическому полуэмпирическому уравнению состояния индивидуального вещества.

В качестве примера решения трансцендентного уравнения будет решена задача определения температур кипения индивидуального вещества в заданном диапазоне давлений по известному трансцендентному уравнению, описывающему зависимость давления насыщенного пара от температуры.

Постановка задачи

Пусть дано нелинейное уравнение:

$$f(x) = 0, \tag{3.11}$$

где $f(x)$ — функция уравнения.

Функция $f(x)$ может быть задана в виде:

а) алгебраического многочлена (полинома)

$$f(x) = a_1 x^m + a_2 x^{m-1} + \ldots + a_m x + a_{m+1}, \tag{3.12}$$

где a_1, …, a_m, a_{m+1} — коэффициенты многочлена; m — степень многочлена — натуральное число, и алгебраическое уравнение (3.12.1) имеет вид

$$a_1 x^m + a_2 x^{m-1} + \ldots + a_m x + a_{m+1} = 0; \tag{3.12.1}$$

б) трансцендентной функции, заданной в некотором интервале, например

$$f(x) = x^2 - e^{-x},$$

и ей соответствует трансцендентное уравнение вида

$$x^2 - e^{-x} = 0. \qquad (3.12.2)$$

Для решения уравнений (3.12.1) и (3.12.2) требуется найти их корни, т. е. числа

$$\bar{x}^* = [x_1^*, x_2^*, ...],$$

которые путем подстановки их в (3.12.1) и (3.12.2) превращают уравнения в верные числовые равенства.

При этом числа $x_1^*, x_2^*, ...$ — элементы вектора $\bar{x}^*$, называются также решениями или корнями уравнения $f(x)=0$.

3.2.1. Решение алгебраических полиномиальных уравнений с применением решателя “roots”

Для решения алгебраического уравнения (3.12.2) с полиномиальной функцией $f(x)$ используется решатель “roots” в виде

$$x = roots(a_1, a_2, ..., a_{m+1}), \qquad (3.13)$$

где $a_1, a_2, ..., a_{m+1}$ — коэффициенты полинома, заданного в виде (3.12).

Данный решатель позволяет определять все корни уравнения, и вещественные, и комплексные.

Коэффициенты многочлена в функции уравнения могут задаваться в виде массива — вектора-строки, например при $m = 5$ (см. код программы на рис. 3.7):

$$koef\_poly = [1,2,–5, 8,–7,–3], \qquad (3.14)$$

который справедлив для уравнения (3.12, 1) при $m = 5$; $a_1 = 1$; $a_2 = 2$; $a_3 = –5$; $a_4 = 8$; $a_5 = –7$; $a_6 = –3$.

```
function Eq_2_1
%ОПРЕДЕЛЕНИЕ КОРНЕИ АЛГЕБРАИЧЕСКОГО УРАВНЕНИЯ С ПРИМЕНЕНИЕМ РЕШАТЕЛЯ "roots"
%УРАВНЕНИЕ f(x)=0 С ПОЛИНОМИАЛЬНОИ ФУНКЦИЕИ ВИДА f(x)=x^5+2x^4-5x^3+8x^2-7x-3
clc;
clear all;
close all;
disp('ОПРЕДЕЛЕНИЕ КОРНЕИ АЛГЕБРАИЧЕСКОГО УРАВНЕНИЯ x^5+2x^4-5x^3+8x^2-7x-3=0');
%1. Задание коэффициентов многочлена
koef_poly=[1 2 -5 8 -7 -3];
%3. Построение графика функции f(x)
x=-4:0.1:4;
nx=length(x);
for i=1:nx
y0(i)=0;
end
%Расчет значений функции в диапазоне изменения аргументов x
f=polyval(koef_poly,x);
%График и задание его атрибутов — заголовка и диапазона изменения осей
plot(x,f,x,y0);
title ('Изображение функции f(x) уравнения x^5+2x^4-5x^3+8x^2-7x-3=0');
xlabel('x — ось абсцисс');ylabel('f(x) — ось ординат');
axis([-4 4 -50 450]);
grid on;
```

Рис. 3.7 (начало)
Код программы решения алгебраического уравнения $x^5 + 2x^4 – 5x^3 + 8x^2 – 7x – 3 = 0$ с применением решателя “roots”

```
%3. Определение вещественных и комплексных корней уравнения
x_resultat=roots(koef_poly);
disp('Все корни');
x_resultat
nres=length(x_resultat);
%Формирование массива вещественных и комплексных корней
j=0;jj=0;
for i=1:nres

   if imag(x_resultat(i))==0
      j=j+1;
   xr_resultat(j)=real(x_resultat(i));
   else
      jj=jj+1;
      xk_resultat(jj)=x_resultat(i);
   end
end
disp('Вещественные корни');
xr_resultat
disp('Комплексные корни');
xk_resultat
end
```

Рис. 3.7 (окончание)

Код программы решения алгебраического уравнения $x^5 + 2x^4 - 5x^3 + 8x^2 - 7x - 3 = 0$ с применением решателя “roots”

Тогда решение уравнения и определение всех его корней может быть записано (рис. 3.7):

$$x\_resultat = roots(koef\_poly). \tag{3.15}$$

Массив коэффициентов используется также для графического представления функции уравнения с применением функции MATLAB “polyval” (рис. 3.7):

$$f = polyval(koef\_poly,x), \tag{3.16}$$

которая позволяет сформировать массив значений функции f, элементы которого рассчитаны в заданном интервале массива x[–4, 4] с шагом 0.1 (рис. 3.7) по ее формуле (3.10).

После чего, как видно в коде программы (рис. 3.7), строится график (рис. 3.8) функции $f(x)$ с помощью функции “plot”:

$$plot(x, y, x, y0). \tag{3.17}$$

Анализ этого графика позволяет установить, что рассматриваемое уравнение имеет три вещественных корня.

Все вещественные и комплексные корни уравнения (3.12, 1) при m = 5, полученные при применении решателя “roots”, представлены на рисунке 3.9.

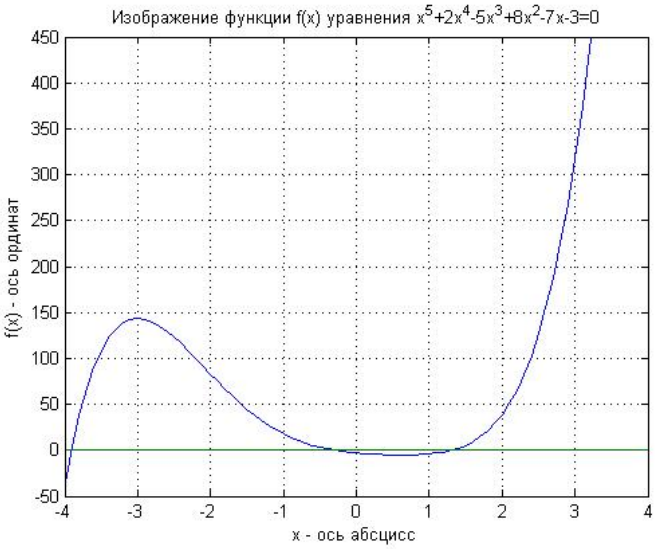


Рис. 3.8

Графическая интерпретация алгебраического уравнения функции, которая представляет собой многочлен 5-й степени, $x^5 + 2x^4 – 5x^2 – 7x – 3 = 0$

```
ОПРЕДЕЛЕНИЕ КОРНЕЙ АЛГЕБРАИЧЕСКОГО УРАВНЕНИЯ x^5+2x^4-5x^3+8x^2-7x-3=0
Все корни
x_resultat =
  -3.9078
   1.3068
   0.4517 + 1.3187i
   0.4517 — 1.3187i
  -0.3023
Вещественные корни
xr_resultat =

   -3.9078    1.3068   -0.3023
Комплексные корни
xk_resultat =  0.4517 + 1.3187i   0.4517 — 1.3187i
>>
```

Рис. 3.9

Результаты решения алгебраического уравнения $x^5 + 2x^4 – 5x^3 + 8x^2 – 7x – 3 = 0$, полученные с применением программы, приведенной на рисунке 3.7

3.2.2. Решение трансцендентного уравнения с применением решателя “fzero”

Для решения трансцендентного уравнения (3.13, 1) используется решатель “fzero” в виде (см. код программы на рис. 3.10):

$$[x\_resultat, func, pr] = fzero('f', x0). \qquad (3.18)$$

В круглых скобках справа от решателя “fzero” задаются исходные данные для решения:

f — вид функции уравнения без кавычек, так как в данном случае он задан в виде inline-функции:

$$f = inline('x \wedge 2 - \exp(-x)'), \tag{3.19}$$

если вид функции задается другой функцией, то ее название в этой позиции решателя "fzero" берется в кавычки — "*f*" или перед названием ставится знак «@» — @*f*; *x*0 — начальное приближение искомого решения; может также задаваться интервал, в котором ведется поиск, в квадратных скобках — [*xmin*, *xmax*], однако в этом случае значения функции уравнения *f*(*x*) на границах интервала [*xmin*, *xmax*] должны иметь разные знаки, т. е. искомый корень должен быть отделен в интервале, который наиболее удобно определить путем построения графика функции (рис. 3.11).

В квадратных скобках слева от решателя (3.18) приводятся переменные, характеризующие результаты решения задачи: *x_resultat* — корень уравнения; *func* — значение функции уравнения в точке решения (близкое к нулю); *pr* — признак корректности вычислительного процесса (1 — все верно; 0 — достигнуто предельное число итераций; числа со знаком «–» — решение либо не найдено, либо требует коррекции).

Решение уравнения $x^2 - e^{-x} = 0$ (3.18) может быть также представлено в программе в виде одной строки: [*x_resultat*, *func*, *pr*] = *fzero*('*x* ^ 2 – exp(–*x*)', *x*0).

```
function Eq_2_4

%ОПРЕДЕЛЕНИЕ КОРНЕИ ТРАНСЦЕДЕНТНОГО УРАВНЕНИЯ С ПРИМЕНЕНИЕМ РЕШАТЕЛЯ "fzero"
%УРАВНЕНИЕ f(x)=0 С ТРАНСЦЕНДЕНТНОИ ФУНКЦИЕИ ВИДА f(x)=x^2-exp(-x)
clc;
clear all;
close all;
disp('-----------------------------------------');
disp('ОПРЕДЕЛЕНИЕ КОРНЕИ УРАВНЕНИЯ x^2-exp(-x)=0');
disp('-----------------------------------------');
%1. Задание вида функции
f=inline('x^2-exp(-x)');
%3. Построение графика функции f(x)
X=-1:0.1:1;
%Расчет значений функции в диапазоне изменения аргументов x
nX=length(X);
for i=1:nX
x=X(i);
F(i)=f(x);
%F(i)=f(x);
y0(i)=0;
end
%График и задание его атрибутов — заголовка и диапазона изменения осей
plot(X,F,'k',X,y0,'k');
title ('Изображение функции f(x) уравнения x^2-exp(-x)=0');
xlabel('x — ось абсцисс');ylabel('f(x) — ось ординат');
grid on;
%3. Определение вещественных и комплексных корней уравнения
option=optimset('Display','iter','TolX',1e-9);
disp('-----------------------------------------');
disp('ИНФОРМАЦИЯ О ПРОМЕЖУТОЧНЫХ ИТЕРАЦИЯХ ВЫЧИСЛИТЕЛЬНОГО ПРОЦЕССА');
disp('-----------------------------------------');
```

Рис. 3.10 (начало)
Код программы решения трансцендентного уравнения $x^2 - e^{-x} = 0$ с применением решателя "fzero"

```
[x_resultat,func,pr]=fzero(f,-3,option);
disp('Корень уравнения');
x_resultat
disp('Значение функции в точке решения');
func
disp('Признак корректного завершения вычислительного процесса');
pr
end
```

Рис. 3.10 (окончание)

Код программы решения трансцендентного уравнения $x^2 - e^{-x} = 0$ с применением решателя “fzero”

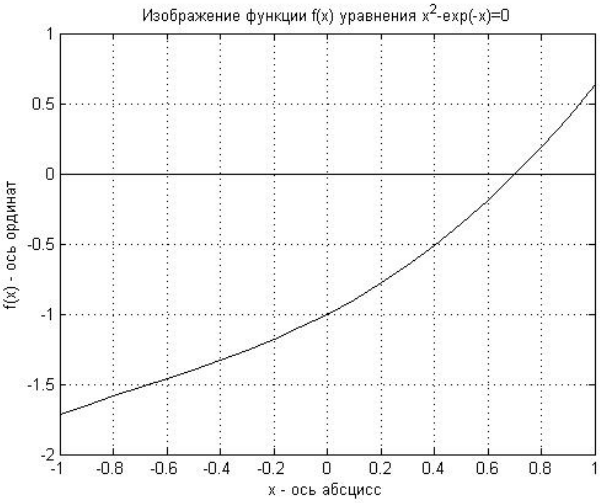


Рис. 3.11

Графическая интерпретация функции трансцендентного уравнения $x^2 - e^{-x} = 0$

Решатель “fzero” (3.18) может быть записан и с отдельной функцией — для отображения ее вида (рис. 3.12).

```
function fun= f(x)
fun=x^2-exp(-x);
end
```

Рис. 3.12

Программный код отдельно оформленной функции $f(x)$ для решения трансцендентного уравнения $x^2 - e^{-x} = 0$ с применением решателя “fzero”

При этом решатель записывается в виде

$$x\_resultat = fzero('f', -3). \qquad (3.20)$$

В этом случае название функции берется в кавычки (может быть записано также *@f*) и начальное приближение задается в виде конкретного числа (–3). Результат представляется скалярной величиной *x_resultat* без подробностей о корректности вычислительного процесса и значения функции уравнения в точке решения.

Для отслеживания хода вычислительного процесса и варьирования точности вычислений в правую часть решателя может включаться параметр, например option, задаваемый специальной функцией “optimset” (рис. 3.10):

$$[x\_resultat, func, pr] = fzero('f', x0, option). \quad (3.21)$$

При этом параметр option должен задаваться предварительно, например для определения точности вычислений, в виде

$$option = optimset('display', 'iter', 'Tolx', 1e - 3). \quad (3.22)$$

Это означает, что точность определения корня уравнения по аргументу (x) будет не больше 10^{-3}, а данные о ходе итерационного процесса будут выводиться на дисплей в командное окно MATLAB в виде (рис. 3.13): номера всех итераций, значения аргументов и соответствующие значения функции уравнения (рис. 3.13, 2).

В заключение следует отметить, что при решении уравнений с применением решателя “fzero” определяется только один корень и часто требуется проведение специальных исследований по инициализации начальных приближений решения и определения интервалов, в которых локализован корень уравнения.

```
-----------------------------------------------
ОПРЕДЕЛЕНИЕ КОРНЕЙ УРАВНЕНИЯ x^2-exp(-x)=0
-----------------------------------------------
ИНФОРМАЦИЯ О ПРОМЕЖУТОЧНЫХ ИТЕРАЦИЯХ ВЫЧИСЛИТЕЛЬНОГО ПРОЦЕССА
-----------------------------------------------
 Search for an interval around -3 containing a sign change:
 Func-count    a         f(a)         b         f(b)        Procedure
      1         -3     -11.0855         -3     -11.0855  initial interval
      3    -3.91515     -9.95344    -3.08485     -13.3479   search
      5      -3.88     -9.51987      -3.12      -13.912   search
      7    -3.83029     -8.93988    -3.16971     -13.7534   search
      9      -3.76     -8.18224      -3.24     -15.0361   search
     11    -3.66059     -7.22598    -3.33941     -17.0509   search
     13      -3.52     -6.0782       -3.48     -20.3493   search
     15    -3.32118      -4.7998    -3.67882      -26.066   search
     17      -3.04     -3.52901      -3.96     -36.7757   search
     19    -1.64235     -3.46999    -4.35765      -59.084   search
     21      -1.08     -1.77828      -4.92     -113.796   search
     23    -0.28471     -1.24832    -5.71529     -270.808   search
     24        0.84     0.273889    -5.71529     -270.808   search

 Search for a zero in the interval [0.84, -5.71529]:
 Func-count    x          f(x)          Procedure
     24        0.84     0.273889       initial
     25    0.833377     0.259938       interpolation
     26    0.710098     0.012644       interpolation
     27    0.710098     0.012644       bisection
     28    0.703287  -0.000342818       interpolation
     29    0.703467 -8.97954e-007       interpolation
     30    0.703467   3.8888e-013       interpolation
     31    0.703467   3.8888e-013       interpolation
```

Рис. 3.13 (начало)

Результат решения трансцендентного уравнения $x^2 - e^{-x} = 0$ с выводом данных о параметрах вычислительного процесса на промежуточных итерациях, полученный с применением функции “optimset” по программе, представленной на рисунке 3.10

```
Zero found in the interval [0.84, -5.71529]
Корень уравнения

x_resultat = 0.7035

Значение функции в точке решения

func = 3.8888e-013

Признак корректного завершения вычислительного процесса

pr = 1
>>
```

Рис. 3.13 (окончание)

Результат решения трансцендентного уравнения $x^2 - e^{-x} = 0$ с выводом данных о параметрах вычислительного процесса на промежуточных итерациях, полученный с применением функции “optimset” по программе, представленной на рисунке 3.10

3.2.3. Определение температуры кипения и мольных объемов жидкой и паровой фаз индивидуального вещества при различных давлениях в условиях парожидкостного равновесия

Для определения температуры кипения (T) используется известная из литературы пятикоэффициентная аппроксимирующая зависимость давления (P в Па) насыщенного пара индивидуального вещества от температуры (T в К) в виде уравнения Риделя:

$$P = \exp(A + \frac{B}{T} + C\ln T + DT^E). \tag{3.23}$$

Пять коэффициентов этой зависимости (A, B, C, D, E), справедливые в заданном диапазоне изменения температур и давлений, заимствуются из базы данных симулятора CHEMCAD.

Трансцендентное уравнение, определяющее при этом температуру при заданном давлении и известных коэффициентах A, B, C, D, E, имеет вид

$$P - \exp(A + \frac{B}{T} + C\ln T + DT^E) = 0. \tag{3.24}$$

Решением (корнем) этого уравнения является равновесная температура, соответствующая условиям парожидкостного равновесия.

Уравнение состояния, описывающее равновесие «жидкость — пар» индивидуального вещества и позволяющее определять мольные объемы жидкой и паровой (газовой) фаз, имеет вид

$$\frac{Pv}{RT} = z, \tag{3.25}$$

где v — мольный объем жидкости (пара/газа); z — фактор сжимаемости; R — универсальная газовая постоянная.

При этом когда $z = 1$, речь идет об идеальных системах, а для неидеальных систем $z \neq 1$.

Определение фактора сжимаемости z для неидеальных систем осуществляется с использованием полуэмпирического кубического уравнения состояния SRK (Soave-Redlich-Kwong) следующего вида:

$$P = \frac{RT}{v-b} - \frac{a\alpha(T)}{v(v+b)}, \tag{3.26}$$

где $a = 0.42748\frac{R^2T_{кр}^2}{P_{кр}}$, $b = 0.08664\frac{R\,T_{кр}}{P_{кр}}$, $\alpha(T) = [1 + (0.48 + 1.574w - 0.176w^2)(1 - -\sqrt{T/T_{кр}})]^2$.

В этом случае: $T_{кр}$, в К, — критическая температура; $P_{кр}$, в барах, — критическое давление; w — ацентрический фактор; T, в К, — значение равновесной температуры, определяемой при заданном давлении по уравнению (3.24), $R \approx 0.0831433\left[\frac{\text{дм}^3\text{бар}}{\text{мол}\cdot K}\right]$ — универсальная газовая постоянная.

Из последнего уравнения (3.26) можно выразить мольный объем (v), в результате чего получается уравнение в виде многочлена третьей степени:

$$v^3 - \frac{RT}{P}v^2 + \frac{1}{P}(a\alpha(T) - bRT - Pb^2)v - \frac{a\alpha(T)b}{P} = 0 \tag{3.27}$$

или с учетом выражения (3.25) для фактора сжимаемости z оно имеет вид

$$z^3 - z^2 + (A - B - B^2)z - A \cdot B = 0, \tag{3.28}$$

где $A = \frac{a\alpha P}{R^2T^2}$, $B = \frac{bP}{R\,T}$.

Три корня уравнения (3.28) могут быть определены с использованием решателя “roots”, причем в областях докритических значений параметров все они являются вещественными числами.

Определение мольных объемов (v) выполняется с помощью уравнения (3.27) при заданном значении давления P и рассчитанной по уравнению (3.24) величине равновесной температуры T. Величина фактора сжимаемости z определяется из полуэмпирического кубического уравнения состояния SRK (3.28) с учетом коэффициентов из (3.26). При этом наименьшему значению мольного объема соответствует мольный объем жидкости v^L, а наибольшему — мольный объем пара v^V.

Третье значение объема v^S, соответствующее третьему значению корня z, в расчет не принимается.

Программа решения задачи состоит из пяти файлов, коды которых приведены на рисунках 3.14–3.18.

В основной управляющей программе Glav_model_SRK_P (рис. 3.14) осуществляется считывание исходных данных для расчета из файла DATA.m, вызов функций $EqP(T)$ (рис. 3.16) с использованием решателя “fzero” для опреде-

ления температур в заданном диапазоне изменения давлений. Здесь же выполняется расчет мольных объемов жидкой и паровой фаз путем решения уравнения состояния SRK решателем “roots” с обращением к функции *ABSRK*0 (*T*, *P*) (рис. 3.17) для определения коэффициентов кубического уравнения. В конце файла управляющей программы (рис. 3.17) с использованием специального алгоритма осуществляется отделение вещественных корней от комплексно-сопряженных, если они появляются, что маловероятно.

В файле DATA.m (рис. 3.15) задаются исходные данные для расчетов.

В файле EqP.m (рис. 3.18) задается функция уравнения (3.21), определяющая температуру кипения при известном давлении.

```
%-----------------------------------------------------------------
%Программный код файла Glav_model_SRK_P.m — основная управляющая программа
%-----------------------------------------------------------------
% Программа включает следующие файлы: Glav_model_SRK_P.m+DATA.m +EgP.m+ABSRK0.m+REPORT.m
%-----------------------------------------------------------------
%Программа определения температуры кипения индивидуального вещества
%при заданном давлении по 5-коэффициентному аппроксимирующему уравнению Риделя для
%зависимости давления насыщенного пара от температуры и мольных объемов жидкости и пара
%в заданном диапазоне давлений по уравнению состояния Soave-Redlich-Kwong (SRK)
%-----------------------------------------------------------------
clc;
clear all;
close all;
global T0 Tres Pmin Pmax Pstep Pdata P_Pa P_Hg T_K Patm PPR TTR R;
global i VVres VLres Vres VSres Vtotal;
DATA;
Tres=T0+T_K;
i=0;
for P=Pmin:Pstep:Pmax
      i=i+1;
   Vres(i)=0;
   T0=Tres;
   Pdata=P*P_Pa/P_Hg;
%option=optimset('Display','iter');
[Tres,FuncT,pr]=fzero(@EqP,T0);
PPR(i)=P;TTR(i)=Tres-T_K;
Patm(i)=P/760;
T=Tres-T_K;TT=Tres;
%Преобразование давления в бары:
P=P/P_Hg*1.01325;
[A1 B1 C1 D1]=ABSRK0(T,P);
z=roots([A1 B1 C1 D1]);
XR=z;
j=0;jj=0;
for ii=1:3

   if imag(XR(ii))==0
         j=j+1;
   xr_resultat(j)=real(XR(ii));
      else
      jj=jj+1;
      xk_resultat(jj)=XR(ii);
   end
end
%disp('Вещественные корни');
if j==3
   vl=min(xr_resultat)*R*TT/P;
vs=xr_resultat(2)*R*TT/P;
   vv=max(xr_resultat)*R*TT/P;
   vvmas=[vl,vv];
   VLres(i)=vl;
   VVres(i)=vv;
VSres(i)=vs;
```

Рис. 3.14 (начало)

Код основной программы определения равновесной температуры с использованием пятикоэффициентного уравнения, аппроксимирующего давление насыщенного пара этилового спирта от температуры, и мольных объемов паровой и жидкой фаз по уравнению состояния SRK

```
end
%disp('Комплексные корни');

if j==1
   v=xr_resultat(1)*R*TT/P;
   VSres(i)=v;
   Vres(i)=v;
  end
end

for ii=1:i

 if Vres(ii)==0
      VVV=[VLres(ii),VVres(ii)];
       Vtotal(ii,1)=VVV(1);
       Vtotal(ii,2)=VVV(2);

 else
         %vkrit=mean(vvmas);
          Vtotal(ii,1)=Vres(i);
      Vtotal(ii,2)=Vres(i);
 end
end
REPORT;
```

Рис. 3.14 (окончание)

Код основной программы определения равновесной температуры с использованием пятикоэффициентного уравнения, аппроксимирующего давление насыщенного пара этилового спирта от температуры, и мольных объемов паровой и жидкой фаз по уравнению состояния SRK

```
function DATA
%-----------------------------------------------------------------
%Программный код файла DATA.m — задание исходных данных для расчетов
%-----------------------------------------------------------------
% Программа включает следующие файлы: Glav_model_SRK_P.m+DATA.m +EgP.m+ABSRK0.m+REPORT.m
%-----------------------------------------------------------------
%Программа определения температуры кипения индивидуального вещества
%при заданном давлении по 5-коэффициентному аппроксимирующему уравнению Риделя для
%зависимости давления насыщенного пара от температуры и мольных объемов жидкости и пара
%в заданном диапазоне давлений по уравнению состояния Soave-Redlich-Kwong (SRK)
%-----------------------------------------------------------------
global A B C D E T0 Pmin Pmax Pstep P_Pa P_Hg T_K comp priz_print;
global R TKR PKR P_HgZ omega;
%---------------------------------------
% 1. Название индивидуального вещества — comp:
comp='Этиловый спирт';
%---------------------------------------
% 3. A,B,C,D,E  — Параметры 5-коэффициентного аппроксимирующего уравнения
% зависимости давления насыщенного пара индивидуального вещества вида:
%              P[Па]=exp(A+B/T[K]+C*log(T[K])+D*T[K]^E)
A=74.475;B=-7164.3;C=-7.327;D=3.134e-6;E=2;
%с применением которого из трансцендентного уравнения:
%f(T)=P-exp(A+B/T+C*log(T)+D*T^E)=0 при известных P,A,B,C,D,E
% определяются температуры кипения Ткип при различных давлениях
%---------------------------------------
% 3. T0[C] — начальное приближение по температуре итерационного процесса
%расчета температур кипения:
T0=0;
%---------------------------------------
% 4. P_HgZ[мм.рт.ст.]- задаваемое давление, равное одному из представленных в таблице
%результатов, при котором приводится графическая интерпретация функции
%уравнения f(T)=0, используемого для определения температуры кипения при
%конкретном давлении:
P_HgZ=1540;
%P_HgZ=760;
%---------------------------------------
```

Рис. 3.15 (начало)

Программный код файла DATA.m задания исходных данных для расчетов

```
% 5. Параметры для перевода из одной системы единиц в другую:
P_Pa=101325;P_Hg=760;T_K=273.15;
%-----------------------------------------
% 6. Данные для расчета мольных объемов жидкой и паровой фаз по кубическому
% уравнению состояния Soave-Redlich-Kwong (SRK):
%-----------------------------------------
%6.1. Универсальная газовая постоянная (R) в дм^3*бар/мольK
R=0.0831433;
%6.3. Критическая температура (TKR) в K
TKR=513.92;
%6.3. Критическое давление (PKR)в бар
PKR=60.68;
%PKR=[63.83 220.48];
%6.4. Ацентрический фактор (omega)
omega=0.635;
%-----------------------------------------------------
%7. Интервал изменения давлений в [мм.рт.ст.] — [Pmin — Pmax]:
Pmin=640;
Pmax=4000;
%Pmax=43360;
%-----------------------------------------------------
%8. Шаг изменения температур Pstep[мм.рт.ст.]:
Pstep=60;
%-----------------------------------------------------
%9. Форма вывода отчета о работе программы
%-----------------------------------------
%9.1. Если priz_print=0 (задается в файле DATA.m) — отчет выводится в командное окно MATLAB;
%9.3. Если priz_print=1 (задается в файле DATA.m) — отчет выводится в текстовый файл,
%   который размещается в той же папке MATLAB, в которой находятся все m-файлы программы;
priz_print=1;
end
```

Рис. 3.15 (окончание)

Программный код файла DATA.m задания исходных данных для расчетов

```
function fdeltaP=EqP(T)
%-----------------------------------------------------
%Программный код файла EqP.m — функция уравнения f(T)=0, определяющего
%температуру кипения при заданном давлении по 5-коэффициентному аппроксимирующему
% уравнению зависимость давления насыщенного пара от температуры (ChemCad)
%-----------------------------------------------------
% Программа включает следующие файлы: Glav_model_SRK_P.m+DATA.m +EgP.m+ABSRK0.m+REPORT.m
%-----------------------------------------------------
%Программа определения температуры кипения индивидуального вещества
%при заданном давлении по 5-коэффициентному аппроксимирующему уравнению Риделя для
%зависимости давления насыщенного пара от температуры и мольных объемов жидкости и пара
%в заданном диапазоне давлений по уравнению состояния Soave-Redlich-Kwong (SRK)
%-----------------------------------------------------
global Pdata A B C D E;
fdeltaP=Pdata-exp(A+B/T+C*log(T)+D*T^E);
end
```

Рис. 3.16

Программный код файла, в котором задается функция нелинейного уравнения для определения температур кипения при различных давлениях

```
function [A1 B1 C1 D1]=ABSRK0(T,P)
%-----------------------------------------------------
%Программный код файла ABSRK0.m — определение констант уравнения состояния
%SRK
%-----------------------------------------------------
% Программа включает следующие файлы: Glav_model_SRK_P.m+DATA.m +EgP.m+ABSRK0.m+REPORT.m
%-----------------------------------------------------
%Программа определения температуры кипения индивидуального вещества
%при заданном давлении по 5-коэффициентному аппроксимирующему уравнению Риделя для
%зависимости давления насыщенного пара от температуры и мольных объемов жидкости и пара
%в заданном диапазоне давлений по уравнению состояния Soave-Redlich-Kwong (SRK)
%-----------------------------------------------------
```

Рис. 3.17 (начало)

Программный код файла определения коэффициентов кубического уравнения состояния SRK, записанного относительно фактора сжимаемости *z* (3.28)

```
global TKR PKR R omega
TT=T+273.15;
alfa=(1+(0.48+1.574*omega-0.176*omega^2)*(1-sqrt(TT/TKR)))^2;
A=0.42748*R^2*TKR^2/PKR*alfa;
B=0.08664*R*TKR/PKR;
AA =  A*P /(R^2*TT^2);
BB = B*P/(R*TT);
A1=1;B1=-1;C1=AA-BB-BB^2;D1=-AA*BB;
end
```

Рис. 3.17 (окончание)

Программный код файла определения коэффициентов кубического уравнения состояния SRK, записанного относительно фактора сжимаемости *z* (3.28)

В файле ABSRK0 (T, P).m (рис. 3.17) из критических параметров индивидуального вещества определяются коэффициенты кубического уравнения состояния SRK (3.28), записанного относительно фактора сжимаемости *z*.

Файл отчета о результатах работы программы REPORT.m приведен на рисунке 3.18, но пользователь может сформировать его и самостоятельно.

```
function REPORT
%-------------------------------------------------------------------
%Программный код файла REPORT.m — отчет о работе программы
%-------------------------------------------------------------------
% Программа включает следующие файлы: Glav_model_SRK_P.m+DATA.m +EgP.m+ABSRK0.m+REPORT.m
%-------------------------------------------------------------------
%Программа определения температуры кипения индивидуального вещества
%при заданном давлении по 5-коэффициентному аппроксимирующему уравнению Риделя для
%зависимости давления насыщенного пара от температуры и мольных объемов жидкости и пара
%в заданном диапазоне давлений по уравнению состояния Soave-Redlich-Kwong (SRK)
%-------------------------------------------------------------------
global comp PPR TTR i fff P_HgZ P_Hg T_K Pdata P_Pa priz_print T0;
global VSres Vtotal Patm TKR PKR omega A B C D E Pmin Pmax Pstep;
if priz_print==1
fff=fopen('Glav_model_SRK_P(REPORT).txt','wt');
else
fff=1;
end
fprintf(fff,'%s\n','-------------------------------------------------');
fprintf(fff,'%s\n','Программа определения температуры кипения индивидуального вещества ');
fprintf(fff,'%s\n',' по уравнению 5-коэффициентному аппроксимирующему Риделя');
fprintf(fff,'%s\n',' и мольных объемов жидкости и диапазоне пара в заданном диапазоне давлений');
fprintf(fff,'%s\n','по уравнению состояния Soave-Redlich-Kwong (SRK)' );
fprintf(fff,'%s\n','-------------------------------------------------');
fprintf(fff,'%s\n','Программа включает следующие файлы:
Glav_model_SRK_P.m+DATA.m+EqP.m+ABSRK0.m+REPORT.m');
fprintf(fff,'%s\n','_____ ');
fprintf(fff,'%s\n','ИСХОДНЫЕ ДАННЫЕ ');
fprintf(fff,'%s\n','-------------------------------------------------');
fprintf(fff,'%s\n',['1.Название индивидуального вещества( comp) = ' comp]);
fprintf(fff,'%s\n','-------------------------------------------------');
fprintf(fff,'%s\n',['3.Критическая температура( TKR ) = ' num2str(TKR,'%10.2f') ' K ']);
fprintf(fff,'%s\n',['3.Критическое давление( PKR ) = ' num2str(PKR,'%10.2f') ' атм']);
fprintf(fff,'%s\n',['4.Ацентрический фактор Питцера( omega ) = ' num2str(omega,'%10.4f') ' ']);
fprintf(fff,'%s\n','-------------------------------------------------');
fprintf(fff,'%s\n','5.Коэффициенты 5-коэффициентного уравнения  для определения давления нас. пара вещества
(ChemCad): ');
fprintf(fff,'%s\n','              P[Па]=exp(A+B/T[K]+C*log(T[K])+D*T[K]^E)');
fprintf(fff,'%s\n',['5.1. Первый коэффициент( A ) = ' num2str(A,'%10.4f') ' ']);
fprintf(fff,'%s\n',['5.3. Второй коэффициент( B ) = ' num2str(B,'%10.4f') ' ']);
fprintf(fff,'%s\n',['5.3. Третий коэффициент( C ) = ' num2str(C,'%10.4f') ' ']);
fprintf(fff,'%s\n',['5.4. Четвертый коэффициент( D ) = ' num2str(D,'%10.4g') ' ']);
fprintf(fff,'%s\n',['5.5. Пятый коэффициент( E ) = ' num2str(E,'%10.4f') ' ']);
fprintf(fff,'%s\n','_____ ');
fprintf(fff,'%s\n',['6.1.Левая граница интервала изменения давления( Pmin ) = ' num2str(Pmin,'%10.2f') ' мм.рт.ст. ']);
fprintf(fff,'%s\n','-------------------------------------------------');
fprintf(fff,'%s\n',['6.3. Правая граница интервала изменения давления( Pmax ) = ' num2str(Pmax,'%10.2f') ' мм.рт.ст.
```

Рис. 3.18 (начало)

Программный код файла отчета о работе программы, приведенной на рисунке 3.14

```
']);
fprintf(fff,'%s\n','-------------------------------------------------------');
fprintf(fff,'%s\n',['6.3. Шаг изменения давления ( Pstep ) = ' num2str(Pstep,'%10.2f') ' мм.рт.ст. ']);
fprintf(fff,'%s\n','-------------------------------------------------------');
fprintf(fff,'%s\n',['7. Оценка приближения по температуре при определении Т-кипения( T0 ) = ' num2str(T0,'%10.2f') '
K ']);
fprintf(fff,'%s\n','-------------------------------------------------------------------');
fprintf(fff,'%s\n','-------------------------------------------------------------------');
fprintf(fff,'%s\n','                    РЕЗУЛЬТАТЫ РАСЧЕТОВ                    ');

fprintf(fff,'%s\n','-------------------------------------------------------------------');
fprintf(fff,'%s\n','1.Результаты расчета температур кипения (Ткип) при различных давлениях');
fprintf(fff,'%s\n','путем решения трансцендентного уравнения вида P[Па]-exp(A+B/T[K]+Cln(T[K])+DT[K]^E=0');
fprintf(fff,'%s\n','использующим 5- коэффициентное аппроксимирующее уравнение Риделя для расчета
давл.нас.пара');
fprintf(fff,'%s\n','-----------------------------------------------------------------------------------------');
fprintf(fff,'%-6s %-15s %-16s %-16s %-16s %-16s \n','№','P(  мм.рт.ст.  )','Ткип(  C
)','VV(дм^3/мол.)','VL(дм^3/мол.)','VS(дм^3/мол.)');
fprintf(fff,'%s\n','-----------------------------------------------------------------------------------------');

for j=1:i
fprintf(fff,'%-6d %-15.1f %-16.3f %-16.5f %-16.5f %-16.5f \n',j,PPR(j),TTR(j),Vtotal(j,2),Vtotal(j,1),VSres(j));
end
fprintf(fff,'%s\n','-----------------------------------------------------------------------------------------');
%-------------------------------------------------------------------------;
for is=1:5
   hfig(is)=figure;
end
%1. Графическое представление зависимости температуры кипения от давления
%для индивидуального вещества, получаемое при решении уравнения:
%      P[Па]{i}-exp(A+B/T[K]{i}+Cln(T[K]{i})+DT[K]{i}^E=0, i=1,...N,
%где N — число задаваемых давлений, при которых определяются температуры кипения
% а коэффициенты для этанола равны: A=74.475;B=-7164.3;C=-7.327;D=3.134e-6;E=2;
%при определении температуры кипения при нормальных условиях (P=760 мм.рт.ст.)
%3. Графическое представление функции трансцендентного уравнения
%                    f(T)=0,
% где          f(T)=101325-exp(A+B/T[K]+Cln(T[K])+DT[K]^E
% и          A=74.475;B=-7164.3;C=-7.327;D=3.134e-6;E=2;
%при определении температуры кипения при нормальных условиях (P=760 мм.рт.ст.)
%-------------------------------------------------------------------------
for j=1:i
if PPR(j)==P_HgZ;
   TB=TTR(j);
Pdata=P_HgZ/P_Hg*P_Pa;
end
end
TB=0;
if TB==0
   TB=TTR(1);
   Pdata=PPR(1)/P_Hg*P_Pa;
end
fprintf(fff,'%s\n',['3. Заданное давление, при граф. интерпр. корня уравнения, опред. Т-кипения( Pdata ) = '
num2str(Pdata/P_Pa*P_Hg,'%10.2f') ' мм.рт.ст. ']);
fprintf(fff,'%s\n','_____________________________________________________________________ ');
iss=0;
for TT=TB-50:1:TB+50
   iss=iss+1;
   TB_K=TT+T_K;
   Tr(iss)=TB_K;
   FdeltaPr(iss)=EqP(TB_K);
   y0(iss)=0;

end
figure(hfig(1));
plot(Tr,FdeltaPr,':',Tr,y0);
title('f(T) при расчете Ткип для 1-ой точки диап.изм.P');
xlabel('Температура — T в K');ylabel('Функция уравнения f(T)=0 при P=Pdata (1)');
text(TB+T_K,0,'X — КОРЕНЬ');
legend('Функция f(T)=Pdata[Па]-exp(A+B/T[K]+Cln(T[K])+DT[K]^E)','нулевая ось абсцисс');
axis([290 400 -4*10^5 1*10^5]);
grid;
%-------------------------------------------------------------------------;
```

Рис. 3.18 (продолжение)

Программный код файла отчета о работе программы, приведенной на рисунке 3.14

```
for j=1:i
VV(j)=Vtotal(j,2);
VL(j)=Vtotal(j,1);
end
VLG=sort(VL)*10;VVG=sort(VV);VSG=sort(VSres)*10;
for jj=1:i
PatmG(jj)=1/Patm(jj);
end
PPatm=sort(PatmG);
for jj=1:i
PatmG(jj)=1/PPatm(jj);
end
for iss=1:i
x0(iss)=iss;
y00(iss)=1;
end
figure(hfig(2));
plot(VLG,PatmG,'--',VVG,PatmG,':',VSG,PatmG,'-.',x0,y00);
title('Корни кубического уравнения состояния SRK (VL,VS,VV)при P=1атм');
xlabel('VL — "--": дм^3/мол.*10,VV — " * ": дм^3/мол.,VS — "-.-": дм^3/мол.*10');
ylabel('P — атм');
text(0.1,0.8,'VL');
text(5,0.8,'VS');
text(30,0.8,'VV');
legend('VL — миним. корень ур-я : мол.об.жидк.фазы','VS — средний корень ур-я',...
'VV — максим. корень ур-я : мол.об.пар.фазы');
grid on;
figure(hfig(3));
plot(PPR,TTR);
title('График зависим. температуры кипения этилового спирта от давления ');
xlabel('Давление — P в мм. рт. ст.');ylabel('Температура кипения — T в C');
legend(' Ткип. опред. из ур-я : P[Па](i)-exp(A+B/T[K](i)+Cln(T[K](i))+DT[K](i)^E)=0,i=1,...N');
grid on;
figure(hfig(4));
plot(Patm,VV);
title('VV=VV(Patm) : Мольн объем пара, выч. по уравнению состояния SRK');
xlabel('Patm-  атм');
ylabel('VV — дм^3/мол.');
grid on;
figure(hfig(5));
plot(Patm,VL,Patm,VSres,'--');
title('Мольн объем жидк.(VL — "-") и доп. корня(VS - "- -"), выч. по уравн.SRK');
xlabel('Patm- атм');
ylabel('VL — дм^3/мол.,VS — дм^3/мол.');
grid on;
fprintf(fff,'%s\n','                    Выведено 5 графиков                    ');
disp('-------------------------------------------------------------------------');
disp('Расчет по программе Glav_model_SRK_P.m завершен, результаты представлены:');
%disp('\n');
disp('в виде текстового файла Glav_model_SRK_P(REPORT).txt и 5 отдельных графиков');
disp('-------------------------------------------------------------------------');
fprintf(fff,'%s\n','-------------------------------------------------------------------------');
fprintf(fff,'%s\n','Расчет по программе завершен, результаты также представлены в виде 5 отдельных графиков');
fprintf(fff,'%s\n','-------------------------------------------------------------------------');
if priz_print==1
fclose(fff);
end
end
```

Рис. 3.18 (окончание)

Программный код файла отчета о работе программы, приведенной на рисунке 3.14

Исходные данные и результаты расчетов по описанной методике для этилового спирта в диапазоне давлений от 640 до 4000 мм рт. ст. представлены соответственно в файле DATA.m (рис. 3.15).

Результаты расчетов также проиллюстрированы на пяти графиках:

- рисунок 3.19 — определение температуры кипения в первой точке диапазона изменения давления (P = 640 мм рт. ст.);

- рисунок 3.20 — изменение равновесных температур (температур кипения) в зависимости от величины давления;
- рисунок 3.21 — зависимость значений трех корней уравнения состояния SRK от давления, в частности величины корней при давлении $P = 1$ атм;
- рисунок 3.22 — изменение мольного объема паровой фазы в зависимости от величины давления;
- рисунок 3.23 — изменение мольного объема жидкой фазы и третьего, не принимаемого во внимание, корня уравнения SRK в зависимости от величины давления.

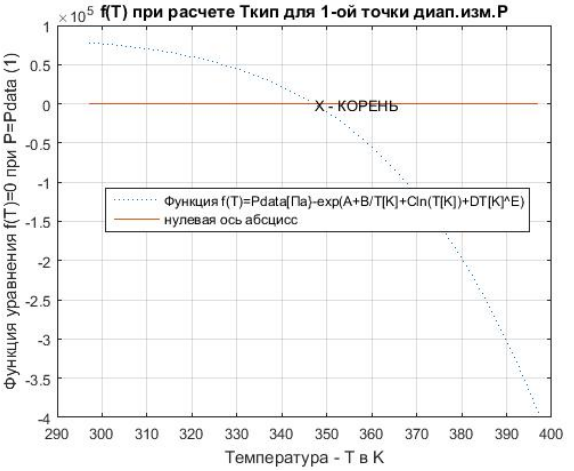


Рис. 3.19

Графическая интерпретация корня уравнения, по которому определяется температура кипения индивидуального вещества

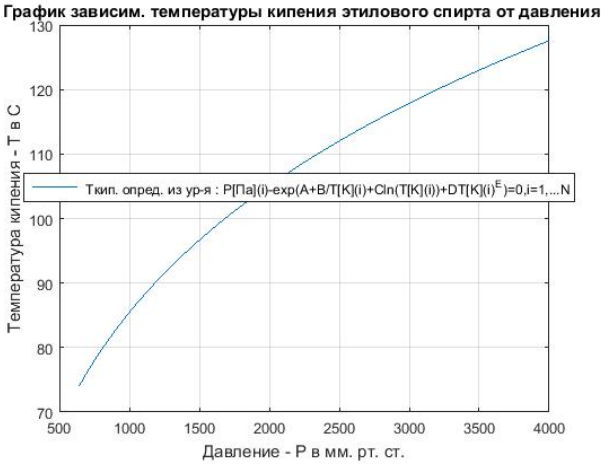


Рис. 3.20

Зависимость температур кипения этилового спирта от давления, полученная с применением пяти коэффициентов аппроксимирующего уравнения зависимости давления от температуры

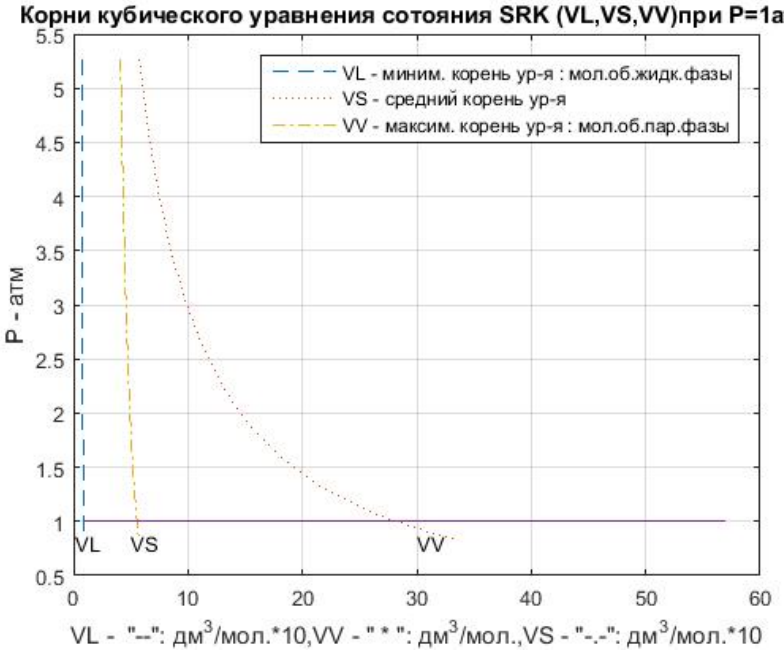


Рис. 3.21

Корни (v^L, v^S, v^V) кубического уравнения состояния SRK для этилового спирта при различных давлениях и их фиксация при P = 1 атм

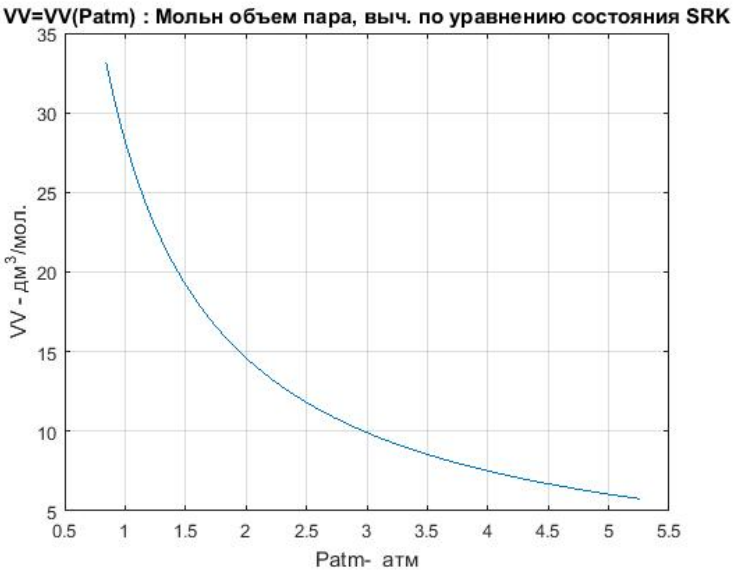


Рис. 3.22

Зависимость мольного объема паровой фазы этилового спирта от давления, полученная путем решения кубического уравнения состояния SRK

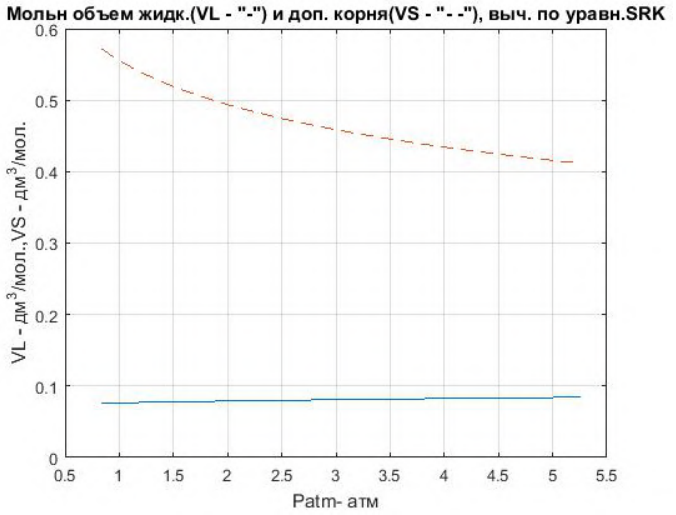


Рис. 3.23

Зависимость мольного объема жидкой фазы и дополнительного корня кубического уравнения состояния SRK от давления

Результаты работы программы в табличном виде с использованием сформированного авторами файла REPORT.m (рис. 3.18), представленные ранее на рисунках 3.19–3.23, приведены также в Приложении (табл. П.1).

3.3. Решение систем уравнений

Математическое описание широкого класса процессов и явлений термодинамики, химической кинетики, физической химии, а также процессов, протекающих в аппаратах химической технологии, представляет собой системы линейных и нелинейных уравнений. К системам уравнений, чаще всего нелинейных, относятся также уравнения, описывающие процессы в совокупности аппаратов — технологических схемах химических производств.

При этом достаточно часто приходится сталкиваться с системами уравнений большой размерности, функции в которых характеризуются существенными нелинейностями. Такие системы могут иметь множество решений, и определение достоверного, физически обоснованного решения является непростой задачей. Для сокращения размерности решаемых задач обычно используются информационные матрицы, позволяющие существенно сократить число одновременно решаемых уравнений. В результате удается не только сократить размерности решаемых задач, но и можно улучшить сходимость итерационных методов решения и упростить анализ получаемых результатов.

Следует отметить, что в общем случае для решения линейных систем уравнений могут применяться безытерационные алгоритмы, а нелинейных — исключительно итерационные.

Постановка задачи

Дана система уравнений с n неизвестными:

$$\begin{aligned} f_1(x_1, ..., x_n) &= 0 \\ f_2(x_1, ..., x_n) &= 0 \\ &\vdots \\ f_n(x_1, ..., x_n) &= 0, \end{aligned} \tag{3.29}$$

где $f_i(x_1, \ldots, x_n)$, $i = 1, \ldots, n$, — линейные и/или нелинейные функции, определенные в некоторой области.

Определить решение системы уравнений (3.29) означает, что требуется найти такой вектор $\bar{x} = (x_1, \ldots, x_n)^T$, который при подстановке в (3.29) превращает каждое уравнение в верное числовое равенство.

Если все функции $f_i(x_1, ..., x_n)$ линейны относительно искомых переменных $\bar{x} = [x_1, ..., x_n]$, то система уравнений (3.29) называется системой *линейных алгебраических уравнений* (СЛАУ), а в случае невыполнения этих условий — системой *нелинейных уравнений* (СНУ).

Простейший пример системы двух линейных уравнений (СЛАУ) имеет вид

$$\begin{aligned} 2x_1 + x_2 &= 4 \\ x_1 - x_2 &= -1 \end{aligned} \tag{3.30}$$

или в матричном виде:

$$\begin{bmatrix} 2 & 1 \\ 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 4 \\ -1 \end{bmatrix},$$

$$\overline{\overline{A}}\bar{x} = \bar{b}$$

где матрица $\overline{\overline{A}} = \begin{bmatrix} 2 & 1 \\ 1 & -1 \end{bmatrix}$ называется матрицей коэффициентов СЛАУ, а вектор $\bar{b} = \begin{bmatrix} 4 \\ -1 \end{bmatrix}$ — вектором свободных членов, а $\bar{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ — определяемое решение.

Пример системы двух нелинейных уравнений (СНУ) имеет вид

$$\begin{aligned} 2x_1^2 - x_1x_2 - 5x_1 + 1 &= 0 \\ x_1 + 3\lg x_1 - x_2^2 &= 0. \end{aligned} \tag{3.31}$$

3.3.1. Решение СЛАУ с применением функций “det”, “inv” и решателя “linsolve”

В общем случае СЛАУ для решения целесообразно представить в матричном виде для n уравнений:

$$\overline{\overline{A}} \cdot \overline{x} = \overline{b}\,, \tag{3.32}$$

где $\overline{\overline{A}}$ — квадратная матрица коэффициентов, которая имеет вид

$$\overline{\overline{A}} = \begin{bmatrix} a_{11} & a_{21} & \cdots & q_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}, \tag{3.32а}$$

а векторы свободных членов ($\overline{b}$) и искомых переменных — решений ($\overline{x}$) — имеют вид:

$$\overline{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}, \ \overline{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}. \tag{3.32б}$$

В соответствии с правилами матричного перемножения функции в исходной системе уравнений (3.29) в этом случае могут быть записаны:

$$f_i(x_1, \ldots, x_n) = a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n - \mathrm{b_i} \tag{3.33}$$

или

$$f_i(x_1, ..., x_n) = \sum_{j=1}^{n} a_{ij}x_j - b_i,$$

$$i = 1, ..., n.$$

3.3.1.1. Решение хорошо обусловленных СЛАУ

С использованием модельного примера (3.30) в программе, представленной на рисунке 3.24, реализованы три способа решения СЛАУ:

— по правилу Крамера с применением функции “det” для вычисления всех определителей

$$x_1 = \frac{\begin{vmatrix} b_1 & a_{12} \\ b_2 & a_{22} \end{vmatrix}}{\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix}}; \ x_2 = \frac{\begin{vmatrix} a_{11} & b_1 \\ a_{21} & b_2 \end{vmatrix}}{\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix}}, \tag{3.34}$$

или в программе:

$$x1\_res = \det(A1b)/\det(A),$$

$$x2\_res = \det(A2b)/\det(A);$$

— методом обратной матрицы с применением функции обращения матриц “inv”

$$\bar{x} = \bar{\bar{A}}^{-1} * \bar{b}, \tag{3.35}$$

или в программе:

$$x\_res = inv(A) * \bar{b};$$

— с применением решателя “linsolve” (3):

$$x\_res = linsolve(A, b). \tag{3.36}$$

```
function System_Eq_2_8
%РЕШЕНИЕ СИСТЕМ ЛИНЕИНЫХ УРАВНЕИИ 3-МЯ СПОСОБАМИ"
%СИСТЕМА УРАВНЕНИИ ВИДА 2x(1)+x(2)=4;x(1)-X(2)=-1;
clc;
clear all;
close all;
disp('-----------------------------------------------------');
disp('СИСТЕМА ЛИНЕИНЫХ УРАВНЕНИИ ВИДА: 2x(1)+x(2)=4;x(1)-X(2)=-1;');
disp('-----------------------------------------------------');
%1. Задание матрицы коэффициентов СЛАУ:
A=[2,1;1,-1];
%1. Задание вектора свободных членов СЛАУ:
b=[4;-1];
%3. Графическое представление уравнений в плоскости искомых переменных
i=0;
for x1=0:0.5:2
   i=i+1;
   x11(i)=x1;
x21(i)=4-2*x1;
x22(i)=x1+1;
end
plot(x11,x21,x11,x22,'-.');
title ('Зависимость x(2)=f(x(1)) для системы ур.:2x(1)+x(2)=4[I] и x(1)-x(2)=-1[II] ');
xlabel('x(1)');ylabel('x(2)[I] — "-" ,x(2)[II] — "_._"');
text(1,2,'O-РЕШЕНИЕ');text(0.4,3.5,'I-УРАВНЕНИЕ');text(0.2,1,'II-УРАВНЕНИЕ');
grid on;
disp('0.ОПРЕДЕЛЕНИЕ ЧИСЛА ОБУСЛОВЛЕННОСТИ СЛАУ "COND_A": ');
COND_A=cond(A);
COND_A
disp('1.ОПРЕДЕЛЕНИЕ РЕШЕНИЯ ПО ПРАВИЛУ КРАМЕРА С ПРИМЕНЕНИЕМ ФУНКЦИИ "det": ');
A1b=[4,1;-1,-1];A2b=[2,4;1,-1];
x_res(1)=det(A1b)/det(A);
x_res(2)=det(A2b)/det(A);
x_res'
disp('3.ОПРЕДЕЛЕНИЕ РЕШЕНИЯ МЕТОДОМ ОБРАТНОИ МАТРИЦЫ С ПРИМЕНЕНИЕМ ФУНКЦИИ "inv": ');
x_res=inv(A)*b;
x_res
disp('3.ОПРЕДЕЛЕНИЕ РЕШЕНИЯ С ПРИМЕНЕНИЕМ РЕШАТЕЛЯ "linsolve": ');
x_res=linsolve(A,b);
x_res

end
```

Рис. 3.24

Код программы решения системы двух линейных хорошо обусловленных уравнений (СЛАУ) тремя способами

Как и следовало ожидать, при решении СЛАУ перечисленными тремя способами получаются одинаковые результаты (рис. 3.25).

```
-------------------------------------------------
СИСТЕМА ЛИНЕИНЫХ УРАВНЕНИИ ВИДА: 2x(1)+x(2)=4;x(1)-X(2)=-1;
-------------------------------------------------
0.ОПРЕДЕЛЕНИЕ ЧИСЛА ОБУСЛОВЛЕННОСТИ СЛАУ "COND_A":

COND_A =

   1.7676

1.ОПРЕДЕЛЕНИЕ РЕШЕНИЯ ПО ПРАВИЛУ КРАМЕРА С ПРИМЕНЕНИЕМ ФУНКЦИИ "det":

ans =

   1
   2

3.ОПРЕДЕЛЕНИЕ РЕШЕНИЯ МЕТОДОМ ОБРАТНОИ МАТРИЦЫ С ПРИМЕНЕНИЕМ ФУНКЦИИ "inv":

x_res =

   1.0000
   3.0000

3.ОПРЕДЕЛЕНИЕ РЕШЕНИЯ С ПРИМЕНЕНИЕМ РЕШАТЕЛЯ "linsolve":

x_res =

   1
   2

>>
```

Рис. 3.25

Результат решения системы двух линейных уравнений (СЛАУ) тремя способами по программе, представленной на рисунке 3.24

Однако для плохо обусловленных СЛАУ, как будет показано ниже, решения могут существенно различаться при незначительных возмущениях (колебаниях) элементов матрицы коэффициентов $\overline{\overline{A}}$ (3.32а) и/или вектора свободных членов $\overline{b}$ (3.32б).

Плохая обусловленность может определяться числом обусловленности cond_A:

$$cond\_A = \left\| \overline{\overline{A}} \right\| \cdot \left\| \overline{\overline{A}}^{-1} \right\|, \tag{3.37}$$

где $\| \ \|$ — матричная норма, а число обусловленности в программе (рис. 3.24) определяется функцией cond(A).

Чем cond_A больше (в данном случае cond_A = 1.7676 (рис. 3.25)), тем хуже система обусловлена. Система считается плохо обусловленной уже при cond_A $\approx 10^3 – 10^4$.

На рисунке 3.26 представлена графическая интерпретация (см. программу на рис. 3.24) хорошо обусловленной СЛАУ двух уравнений, которая четко характеризует решение, и точки пересечения прямых, соответствующих этим уравнениям.

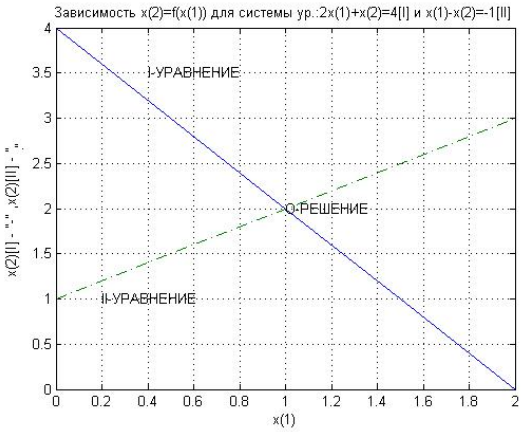


Рис. 3.26

Графическая интерпретация хорошо обусловленной системы двух линейных уравнений

3.3.1.2. Решение несовместных, неопределенных и плохо обусловленных СЛАУ

В программе (рис. 3.27) приведены примеры решения трех типов простейших систем двух линейных уравнений: несовместных (ранг расширенной матрицы не равен рангу матрицы коэффициентов системы), неопределенных (определитель матрицы коэффициентов системы равен нулю) и плохо обусловленных (число обусловленности $\geq 10^3$–10^4).

```
function System_Eq_2_11
%ПРОБЛЕМНЫЕ СЛАУ ПРИ РЕАЛИЗАЦИИ АЛГОРИТМОВ РЕШЕНИЯ"
clc;
clear all;
close all;
disp('-----------------------------------------------------');
disp('1.НЕСОВМЕСТНАЯ СИСТЕМА УРАВНЕНИИ: ');
disp('-----------------------------------------------------');
disp('СИСТЕМА ЛИНЕИНЫХ УРАВНЕНИИ ВИДА: x(1)+x(2)=1;x(1)+X(2)=3;');
disp('-----------------------------------------------------');
%1. Задание матрицы коэффициентов СЛАУ:
A=[1,1;1,1];
%1. Задание вектора свободных членов СЛАУ:
b=[1;3];
disp('РЕШЕНИЕ :');
AT=[1,1,1;1,1,3];
rang_A=rank(A);
rang_A
rang_AT=rank(AT);
rang_AT
disp('1.Получение решения невозможно, т.к. ранг матрицы коэффициентов (rang_A)');
disp('не равен рангу расширенной матрицы (rang_AT)');
x_res=linsolve(A,b);
x_res
disp('-----------------------------------------------------');
disp('3.НЕОПРЕДЕЛЕННАЯ СИСТЕМА УРАВНЕНИИ: ');
disp('-----------------------------------------------------');
disp('СИСТЕМА ЛИНЕИНЫХ УРАВНЕНИИ ВИДА: 3x(1)-x(2)=1;6x(1)-2X(2)=2;');
disp('-----------------------------------------------------');
```

Рис. 3.27 (начало)

Код программы решения несовместной, неопределенной и плохо обусловленной системы двух линейных уравнений (СЛАУ)

```
%1. Задание матрицы коэффициентов СЛАУ:
A=[3,-1;6,-2];
%1. Задание вектора свободных членов СЛАУ:
b=[1;2];
disp('РЕШЕНИЕ :');
DET_A=det(A);
DET_A
disp('3.Получение решения невозможно, т.к. определитель матрицы коэффициентов (det_A)равен нулю');
x_res=linsolve(A,b);
x_res
disp('3.ПЛОХО ОБУСЛОВЛЕННАЯ СИСТЕМА УРАВНЕНИИ: ');
disp('-----------------------------------------------------');
disp('СИСТЕМА ЛИНЕИНЫХ УРАВНЕНИИ ВИДА: 5x(1)+7x(2)=12;7x(1)+10X(2)=17;');
disp('-----------------------------------------------------');
%1. Задание абсолютных погрешностей свободных членов СЛАУ:
%deltab=[0.075 -0.095];
deltab=[0 0];
%1. Задание матрицы коэффициентов СЛАУ:
A=[5,7;7,10];
%1. Задание вектора свободных членов СЛАУ:
b=[12+deltab(1);17+deltab(2)];
disp('ВЕКТОР АБСОЛЮТНЫХ ПОГРЕШНОСТЕи СВОБОДНЫХ ЧЛЕНОВ СЛАУ "deltab":');
deltab
disp('3.Из-за незначительных погрешностей "deltab" могут получаться разные решения');
disp('-----------------------------------------------------');
%3. Графическое представление уравнений в плоскости искомых переменных
i=0;
for x1=0:0.5:2
   i=i+1;
   x11(i)=x1;
x21(i)=(12-5*x1)/7;
x22(i)=(17-7*x1)/10;
end
plot(x11,x21,x11,x22,'-.');
title ('Зав-сть x(2)=f(x(1)) для сис-мы 2-х ур.:5x(1)+7x(2)=12x и 7x(1)+10x(2)=17[2]');
xlabel('x(1)');ylabel(' x(2) — "-" ,x(2)[2] — "-.-"');
text(1,1,'O-РЕШЕНИЕ');text(0.1,1.7,'ПЛОХО ОБУСЛОВЛЕННАЯ СИСТЕМА ЛИНЕИНЫХ УРАВНЕНИИ');
text(0.2,1.2,'II-УРАВНЕНИЕ');text(0.6,1.4,'I-УРАВНЕНИЕ');
grid on;
disp('0.ОПРЕДЕЛЕНИЕ ЧИСЛА ОБУСЛОВЛЕННОСТИ СЛАУ "COND_A": ');
COND_A=cond(A);
COND_A
disp('1.ОПРЕДЕЛЕНИЕ РЕШЕНИЯ ПО ПРАВИЛУ КРАМЕРА С ПРИМЕНЕНИЕМ ФУНКЦИИ "det": ');
A1b=[4,1;-1,-1];A2b=[2,4;1,-1];
x_res(1)=det(A1b)/det(A);
x_res(2)=det(A2b)/det(A);
%x_res=x_res';
x_res
disp('3.ОПРЕДЕЛЕНИЕ РЕШЕНИЯ МЕТОДОМ ОБРАТНОИ МАТРИЦЫ С ПРИМЕНЕНИЕМ ФУНКЦИИ "inv": ');
x_res=inv(A)*b;
x_res
disp('3.ОПРЕДЕЛЕНИЕ РЕШЕНИЯ С ПРИМЕНЕНИЕМ РЕШАТЕЛЯ "linsolve": ');
x_res=linsolve(A,b);
x_res

end
```

Рис. 3.27 (окончание)

Код программы решения несовместной, неопределенной и плохо обусловленной системы двух линейных уравнений (СЛАУ)

Результаты решений, полученных по программе, приведенной на рисунке 3.27, представлены на рисунке 3.28.

```
-------------------------------------------------------
1. НЕСОВМЕСТНАЯ СИСТЕМА УРАВНЕНИИ:
-------------------------------------------------------
СИСТЕМА ЛИНЕИНЫХ УРАВНЕНИИ ВИДА: x(1)+x(2)=1;x(1)+X(2)=3;
-------------------------------------------------------
РЕШЕНИЕ :

rang_A = 1

rang_AT = 2

1.Получение решения невозможно, т.к. ранг матрицы коэффициентов (rang_A)
не равен рангу расширенной матрицы (rang_AT)
Warning: Matrix is singular to
working precision.
> In System_Eq_2_11 at 23

x_res =

  -Inf
   Inf

-------------------------------------------------------
3. НЕОПРЕДЕЛЕННАЯ СИСТЕМА УРАВНЕНИИ:
-------------------------------------------------------
СИСТЕМА ЛИНЕЙНЫХ УРАВНЕНИЙ ВИДА: 3x(1)-x(2)=1;6x(1)-2X(2)=2;
-------------------------------------------------------
РЕШЕНИЕ :

DET_A = 0
3. Получение решения невозможно, т.к. определитель матрицы коэффициентов (det_A)равен нулю
Warning: Matrix is singular to
working precision.
> In System_Eq_2_11 at 38

x_res =

   NaN
   NaN

3. ПЛОХО ОБУСЛОВЛЕННАЯ СИСТЕМА УРАВНЕНИИ:
-------------------------------------------------------
СЛАУ ВИДА: 5x(1)+7x(2)=12;7x(1)+10X(2)=17;
-------------------------------------------------------
ВЕКТОР АБСОЛЮТНЫХ ПОГРЕШНОСТЕЙ СВОБОДНЫХ ЧЛЕНОВ СЛАУ "deltab":

deltab =

    0    0

3.Из-за незначительных погрешностей "deltab" могут получаться разные решения
-------------------------------------------------------
0.ОПРЕДЕЛЕНИЕ ЧИСЛА ОБУСЛОВЛЕННОСТИ СЛАУ "COND_A":

COND_A = 223.9955

1. ОПРЕДЕЛЕНИЕ РЕШЕНИЯ ПО ПРАВИЛУ КРАМЕРА С ПРИМЕНЕНИЕМ ФУНКЦИИ "det":

x_res =

   -3.0000
   -6.0000

3.ОПРЕДЕЛЕНИЕ РЕШЕНИЯ МЕТОДОМ ОБРАТНОЙ МАТРИЦЫ С ПРИМЕНЕНИЕМ ФУНКЦИИ "inv":

x_res =

    1.0000
```

Рис. 3.28 (начало)

Результаты решения несовместной, неопределенной и плохо обусловленной системы двух линейных уравнений по программе, представленной на рисунке 3.27

```
   1.0000

3. ОПРЕДЕЛЕНИЕ РЕШЕНИЯ С ПРИМЕНЕНИЕМ РЕШАТЕЛЯ "linsolve":

x_res =

   1.0000
   1.0000

>>
```

Рис. 3.28 (окончание)

Результаты решения несовместной, неопределенной и плохо обусловленной системы двух линейных уравнений по программе, представленной на рисунке 3.27

Для первой несовместной системы (рис. 3.27):

$$x_1 + x_2 = 1,$$

$$x_1 + x_2 = 3,$$

в соответствии с методом Крамера, для получения решения необходимо определитель числителя разделить на ноль, в результате получается бесконечность (Inf) для каждого решения, а в случае второй неопределенной системы:

$$3x_1 - x_2 = 1,$$

$$6x_1 - 2\,x_2 = 2$$

значения решений не определены (NaN), так как необходимо выполнять деление нулевых определителей друг на друга.

В случае несовместных систем ранги матриц определяются с использованием функции MATLAB “rank”, определитель матрицы коэффициентов в случае неопределенных матриц — с использованием функции “det”.

Для плохо обусловленной системы:

$$\begin{aligned} 5x_1 + 7x_2 &= 12, \\ 7x_1 + 10x_2 &= 17 \end{aligned} \qquad (3.38)$$

результаты решения, представленные на рисунке 3.28, число обусловленности равно cond_A = 223.9955, и результаты, полученные по методу Крамера (–3, –6), существенно отличаются от результатов, полученных двумя другими способами (1, 1), а при графической интерпретации (рис. 3.29) прямые, соответствующие уравнениям, практически совпадают и точка их пересечения (решение) на графике не может быть точно идентифицирована.

Если в программе (рис. 3.27) на элементы вектора свободных членов $\bar{b}$ нанести возмущения в виде вектора deltab = [0,075 – 0,095], то результат решения этой системы (рис. 3.30) при использовании метода Крамера (–3, –6) не изменится, но значительно меняется при решении вторым и третьим способами (3.415, 0).

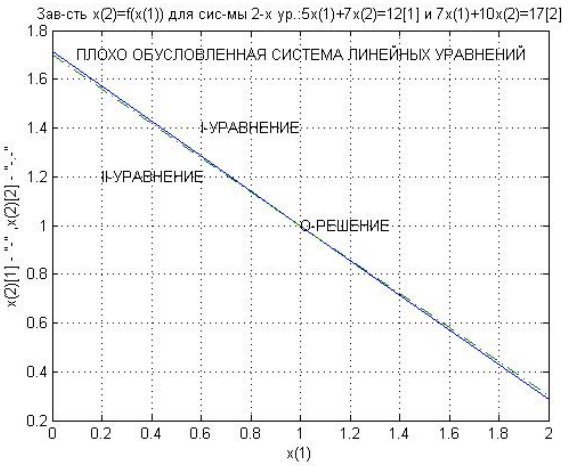


Рис. 3.29

Графическая интерпретация плохо обусловленной системы двух линейных уравнений

```
-------------------------------------------------
1.НЕСОВМЕСТНАЯ СИСТЕМА УРАВНЕНИИ:
-------------------------------------------------
СИСТЕМА ЛИНЕИНЫХ УРАВНЕНИИ ВИДА: x(1)+x(2)=1;x(1)+X(2)=3;
-------------------------------------------------
РЕШЕНИЕ :

rang_A = 1

rang_AT = 2

1.Получение решения невозможно, т.к. ранг матрицы коэффициентов (rang_A)
не равен рангу расширенной матрицы (rang_AT)
Warning: Matrix is singular to working precision.
> In System_Eq_1_2 (line 23)

x_res =

  -Inf
   Inf

-------------------------------------------------
3.НЕОПРЕДЕЛЕННАЯ СИСТЕМА УРАВНЕНИИ:
-------------------------------------------------
СИСТЕМА ЛИНЕЙНЫХ УРАВНЕНИЙ ВИДА: 3x(1)-x(2)=1;6x(1)-2X(2)=2;
-------------------------------------------------
РЕШЕНИЕ :

DET_A = 0

3.Получение решения невозможно, т.к. определитель матрицы коэффициентов (det_A)равен нулю
Warning: Matrix is singular to working precision.
> In System_Eq_1_2 (line 38)

x_res =

   NaN
```

Рис. 3.30 (начало)

Результаты решения плохо обусловленной системы двух линейных уравнений (СЛАУ) при возмущении ее свободных членов (deltab = [0.075, –0.095]) по программе, представленной на рисунке 3.27

```
   NaN

3.ПЛОХО ОБУСЛОВЛЕННАЯ СИСТЕМА УРАВНЕНИИ:
---------------------------------------------------
СИСТЕМА ЛИНЕИНЫХ УРАВНЕНИИ ВИДА: 5x(1)+7x(2)=12;7x(1)+10X(2)=15;
---------------------------------------------------
ВЕКТОР АБСОЛЮТНЫХ ПОГРЕШНОСТЕЙ СВОБОДНЫХ ЧЛЕНОВ СЛАУ "deltab":

deltab = 0.0750   -0.0950

3.Из-за незначительных погрешностей "deltab" могут получаться разные решения
---------------------------------------------------
0.ОПРЕДЕЛЕНИЕ ЧИСЛА ОБУСЛОВЛЕННОСТИ СЛАУ "COND_A":

COND_A = 223.9955

1.ОПРЕДЕЛЕНИЕ РЕШЕНИЯ ПО ПРАВИЛУ КРАМЕРА С ПРИМЕНЕНИЕМ ФУНКЦИИ "det":

x_res =

   -3.0000
   -6.0000

3.ОПРЕДЕЛЕНИЕ РЕШЕНИЯ МЕТОДОМ ОБРАТНОИ МАТРИЦЫ С ПРИМЕНЕНИЕМ ФУНКЦИИ "inv":

x_res =

    3.4150
    0.0000

3.ОПРЕДЕЛЕНИЕ РЕШЕНИЯ С ПРИМЕНЕНИЕМ РЕШАТЕЛЯ "linsolve":

x_res =

    3.4150
   -0.0000
```

Рис. 3.30 (окончание)

Результаты решения плохо обусловленной системы двух линейных уравнений (СЛАУ) при возмущении ее свободных членов (deltab = [0.075, –0.095]) по программе, представленной на рисунке 3.27

При решении СЛАУ произвольной размерности n очень важно перед применением решателей MATLAB правильно сформировать матрицу коэффициентов $\overline{\overline{A}}$ (3.32*а*) и вектор свободных членов $\overline{b}$ (3.32*б*) — см. коды программы на рисунках 3.24 и 3.27. Для определения возможных проблем, связанных с несовместностью, неопределенностью и плохой обусловленностью систем уравнений, которые могут возникать при решении термодинамических и физико-химических задач, перед решением СЛАУ рекомендуется использовать функции “rank”, “det” и “cond”. В литературе предлагаются различные методы преодоления этих проблем, в том числе и связанные с некоторыми изменениями формулировки решаемой задачи.

Следует отметить, что алгоритмы решения систем линейных уравнений (СЛАУ) широко используются при реализации итерационных методов решения систем нелинейных и дифференциальных уравнений, а также оптимизационных задач линейного и нелинейного программирования.

3.3.2. Применение решателя “fsolve” для решения системы нелинейных уравнений

Одним из известных методов решения систем нелинейных уравнений (СНУ) является итерационный метод Ньютона и его модификации:

$$\bar{x}^{(k+1)} = \bar{x}^{(k)} - \overline{\overline{y}}(\bar{x}^{(k)})^{-1}\,\bar{f}(\bar{x}^{(k)}), \qquad (3.39a)$$

где $\bar{x}^{(k+1)}$, $\bar{x}^{(k)}$ — вектор значений искомых переменных соответственно на $k+1$ и k-й итерациях; $\bar{f}(\bar{x}^{(k)})$ — вектор функций СНУ (3.29), рассчитанный при значениях искомых переменных $\bar{x}^{(k)}$,

$$\overline{\overline{y}}(\bar{x}^{(k)})^{-1} = \begin{bmatrix} \dfrac{\partial f_1(x^{(k)})}{\partial x_1^{(k)}} & \dfrac{\partial f_1(x^{(k)})}{\partial x_2^{(k)}} & \cdots & \dfrac{\partial f_1(x^{(k)})}{\partial x_n^{(k)}} \\ \vdots & \vdots & \cdots & \vdots \\ \dfrac{\partial f_n(x^{(k)})}{\partial x_1^{(k)}} & \dfrac{\partial f_n(x^{(k)})}{\partial x_2^{(k)}} & \cdots & \dfrac{\partial f_n(x^{(k)})}{\partial x_n^{(k)}} \end{bmatrix}^{-1}$$

— обратная матрица частных производных Якоби для функций системы (3.29), производные рассчитаны при значении искомых переменных $\bar{x}^{(k)}$ на k-й итерации.

Большое число модификаций этого метода обусловлено проблемами с вычислением элементов матрицы частных производных Якоби и корректным заданием начальных приближений итерационных расчетов по формуле (3.39a), что часто связано с невозможностью получения достоверного решения. Поэтому часто применяются и другие методы, в том числе и комбинированные методы оптимизации функций системы (3.29), позволяющие добиваться близких к нулю значений этих функций в точке решения.

Код программы решения СНУ для системы двух уравнений (3.31) с применением решателя “fsolve” приведен на рисунке 3.31.

```
function System_Eq_2_15
%РЕШЕНИЕ СИСТЕМ НЕЛИНЕИНЫХ УРАВНЕНИИ"
%СИСТЕМА УРАВНЕНИИ ВИДА: 2x(1)^2-x(1)*x(2)-5x(1)+1=0;x(1)+3lg(x(1))-X(2)^2=0;
clc;
clear all;
close all;
disp('-----------------------------------------------------');
disp('    РЕШЕНИЕ СИСТЕМЫ НЕЛИНЕИНЫХ УРАВНЕНИИ ВИДА:');
disp('    f(I)=2x(1)^2-x(1)*x(2)-5x(1)+1=0');
disp('    f(II)=x(1)+3lg(x(1))-x(2)^2=0');
disp('-----------------------------------------------------');
disp('Вид функций СНУ:');
disp('-----------------------------------------------------');
disp('Первой функции f(I):');
disp('-----------------------------------------------------');
disp('f(I)=2*x(1)^2-x(1)*x(2)-5*x(1)+1');
disp('-----------------------------------------------------');
disp('Второй функции f(II):');
disp('-----------------------------------------------------');
disp('f(II)=x(1)+3*log10(x(1))-x(2)^2');
```

Рис. 3.31 (начало)

Код программы решения системы двух нелинейных уравнений (СНУ)

```
disp('-----------------------------------------------------');
%1. Графическое представление уравнений в виде x(2)=f(x(1)) в плоскости искомых переменных
%для инициализации вычислительного процесса и выбора начальных приближений
%x0(1) и x0(2);
%-----------------------------------------------------;
i=0;
for x1=1:0.1:4
   i=i+1;
   x11(i)=x1;
x21(i)=2*x1+1/x1-5;
x22pl(i)=sqrt(x1+3*log10(x1));
x22mn(i)=-x22pl(i);
end
plot(x11,x21,'-.',x11,x22pl,x11,x22mn);
title ('Зав-сть x(2)=f(x(1)) для с-мы:2*x(1)^2-x(1)*x(2)-5*x(1)+1=0(I) и x(1)+3*log10(x(1))-x(2)^2=0');
xlabel('x(1)');ylabel('x(2)(I) — "-" ,x(2)(II) — "-.-'");
text(3.5,0.5,'I-УРАВНЕНИЕ');
text(3.5,-3.5,'II-УРАВНЕНИЕ');
text(3.5,3.5,'II-УРАВНЕНИЕ');
text(1.4,-1.3,'O-РЕШЕНИЕ (1)');
text(3.5,3.3,'O-РЕШЕНИЕ (2)');
grid on;
%-----------------------------------------------------
%3. Оформление функций системы уравнений для нахождения решения с
%применением решателя "fsolve";
%-----------------------------------------------------;
   function f=y(x)
      f(1)=2*x(1)^2-x(1)*x(2)-5*x(1)+1;
      f(2)=x(1)+3*log10(x(1))-x(2)^2;
   end
%-----------------------------------------------------;
%3. Решение системы уравнений с применением решателя "fsolve";
%-----------------------------------------------------;
disp('-----------------------------------------------------');
disp('     РЕАЛИЗАЦИЯ ВЫЧИСЛИТЕЛЬНОГО ПРОЦЕССА И ЕГО РЕЗУЛЬТАТЫ:');
%3.1. Задание начальных приближений решения (x0);
disp('-----------------------------------------------------');
disp(' Начальные приближения для итерационных вычислений:');
disp('-----------------------------------------------------');
x0=[1,-1]
disp('-----------------------------------------------------')
disp('Изменения параметров вычислительного процесса при последовательных приближениях:');
%3.3. Задание параметров вычислительного процесса с применением функции "optimset";
option=optimset('Display','iter','TolX',1e-9,'TolFun',1e-10,'MaxIter',100,'MaxFunEvals',250);

%3.3. Выполнение вычислений с применением решателя "fsolve":

disp('-----------------------------------------------------');
[x,f,pr]=fsolve(@y,x0,option);
disp('-----------------------------------------------------');
disp('РЕШЕНИЕ:');
disp('-----------------------------------------------------')
x
disp('-----------------------------------------------------')
disp('ЗНАЧЕНИЯ функций f(I) и f(II) при найденных решениях:');
disp('-----------------------------------------------------')
f
disp('-----------------------------------------------------')
disp('Признак успешности вычислительного процесса — 1 — ДА, другое число — сомнительно:');
disp('-----------------------------------------------------')
pr
end
```

Рис. 3.31 (окончание)

Код программы решения системы двух нелинейных уравнений (СНУ)

Для решения системы нелинейных уравнений (СНУ) применяется решатель “fsolve”, для чего необходимо:

- представить все функции СНУ (3.29) в неявном виде, как, например, для системы двух уравнений (3.31), и в виде отдельной функции программы — либо внутренней (рис. 3.31), как в данном случае, либо внешней функции (рис. 3.32):

```
function f = y(x)
f(1) = 2 · x(1)^2 − x(1) · x(2) − 5 · x(1) + 1
f(2) = x(1) + 3 · log10(x(1)) − x(2)^2
end
```

(3.39б)

Рис. 3.32

Внешняя функция при решении системы двух нелинейных уравнений (3.31)

В (3.39б) y — название функции; $x = [x(1), x(2)]$ — массив искомых неизвестных, значения которых должны быть определены в результате итерационных вычислений.

- задать начальные приближения итерационного процесса в виде (рис. 3.32):

$$x0 = [1, -1], \tag{3.40}$$

где количество чисел в массиве соответствует размерности СНУ, т. е. определяет количество уравнений в системе и число искомых корней системы уравнений (3.29);

- запустить алгоритм решения СНУ с применением решателя “fsolve” необходимо следующим образом (рис. 3.31):

$$[x, f.\, pr] = fsolve(@y, x0, option), \tag{3.41}$$

где справа в круглых скобках находятся параметры, инициализирующие вычислительный процесс: y (можно ‘y’) — обращение к функции (3.31), решаемой СНУ; $x0$ — массив начальных приближений решения; option (может не задаваться) — определяется функцией MATLAB “optimset”, которая выдает информацию об итерационном процессе и позволяет задать его параметры в виде (рис. 3.31):

$$option = optimset('Display', 'iter', 'TolX', 1e-9, 'TolFun', 1e-10, 'MaxIter', 100, 'MaxFunEvals', 250), \tag{3.42}$$

при этом ‘Display’, ‘iter’ — параметры в кавычках, выводят в Командное окно информацию о параметрах итерационного процесса; ‘TolX’, 1e-9 — точность итерационных вычислений по аргументу x — если не задано, используется точность по умолчанию; ‘TolFun’, 1e-10 — точность итерационных вычислений по функции f — если не задано, используется точность по умолчанию; ‘MaxIter’, 100 — максимальное число итераций — при отсутствии задания неограниченно; ‘MaxFunEvals’, 250 — максимальное число вычислений функций системы (3.39) — при отсутствии задания неограниченно.

Слева от решателя в квадратных скобках задаются три параметра: x — в виде массива искомых решений; f — в виде массива функций системы (3.29) при найденных решениях; pr — признак корректности вычислительного процесса (при неравенстве pr единице решение не гарантировано).

Следует отметить, что результат решения при применении решателя “fsolve” может сильно зависеть от начальных приближений $x0$ (3.40), особенно для сложных нелинейных функций (3.39). Это связано с тем, что СНУ может иметь не единственное решение, а алгоритм “fsolve” определяет одно решение, зависящее от задаваемого начального приближения.

Поэтому рекомендуется провести анализ нелинейных функций СНУ перед тем, как решать задачу. Это удается сделать не всегда, в особенности для систем большой размерности, и тогда предполагается задавать начальные приближения исходя из физического смысла решаемой задачи.

В случае рассматриваемой СНУ (3.31) этот анализ выполнен, для чего в программе (рис. 3.31) нелинейные функции представлены в виде зависимостей $x(2) = f(x(1))$ для первого и второго уравнений системы (3.31), графическое изображение которых имеет вид, представленный на рисунке 3.33:

$$\begin{aligned} &\text{Уравнение I: } x = 2x_1 - 5 + \frac{1}{x_1} \\ &\text{Уравнение II: } x_2 = \sqrt{x_1 + 3\lg x_1} \\ &\text{Уравнение II: } x_2 = -\sqrt{x_1 + 3\lg x_1}. \end{aligned} \qquad (3.43)$$

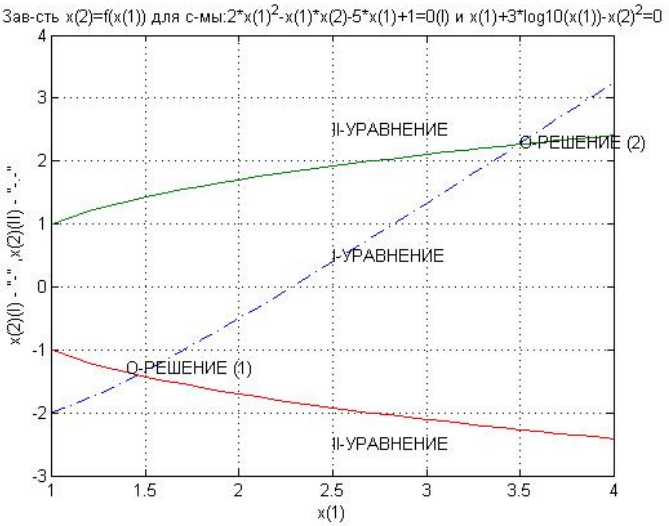


Рис. 3.33

Графическая интерпретация системы двух нелинейных уравнений с двумя решениями

Из графического отображения этих функций (рис. 3.33) следует, что СЛАУ (3.31) имеет два решения, соответствующие точкам пересечения кривой, соответствующей уравнению I, с кривыми, соответствующими уравнению II

(3.43). Поэтому при задании двух различных начальных приближений для решения СНУ (3.31) получаются два различных решения:

- для приближения $x0 = [1, -1]$ получается решение $x = [1.4589, -1.3968]$, и число итераций равно 7 (рис. 3.34);
- для приближения $x0 = [4, 4]$ получается решение $x = [3.4874, 2.2616]$, и число итераций равно 6 (рис. 3.35).

```
---------------------------------------------------
   РЕШЕНИЕ СИСТЕМЫ НЕЛИНЕИНЫХ УРАВНЕНИИ ВИДА:
   f(I)=2x(1)^2-x(1)*x(2)-5x(1)+1=0
   f(II)=x(1)+3lg(x(1))-x(2)^2=0
---------------------------------------------------
Вид функций СНУ:
---------------------------------------------------
Первой функции f(I):
---------------------------------------------------
f(I)=2*x(1)^2-x(1)*x(2)-5*x(1)+1
---------------------------------------------------
Второй функции f(II):
---------------------------------------------------
f(II)=x(1)+3*log10(x(1))-x(2)^2
---------------------------------------------------
---------------------------------------------------
   РЕАЛИЗАЦИЯ ВЫЧИСЛИТЕЛЬНОГО ПРОЦЕССА И ЕГО РЕЗУЛЬТАТЫ:
---------------------------------------------------
 Начальные приближения для итерационных вычислений:
---------------------------------------------------

x0 =  1    -1

---------------------------------------------------
Изменения параметров вычислительного процесса при последовательных приближениях:
---------------------------------------------------

                            Norm of     First-order   Trust-region
 Iteration  Func-count      f(x)        step          optimality    radius
    0          3           1                         1             1
    1          4           1           1             1             1
    2          7           0.737446    0.25          1.31          0.25
    3         10           0.387254    0.625         1.6           0.625
    4         13           0.00283585  0.187096      0.138         0.625
    5         16           3.3589e-007 0.0189624     0.00136       0.625
    6         19           1.21692e-015 0.000171625  1.06e-007     0.625
    7         22           3.02146e-030 1.18225e-008 4.08e-015     0.625

Equation solved.

fsolve completed because the vector of function values is near zero
as measured by the selected value of the function tolerance, and
the problem appears regular as measured by the gradient.

<stopping criteria details>

---------------------------------------------------
РЕШЕНИЕ:
---------------------------------------------------

x =  1.4589   -1.3968
```

Рис. 3.34 (начало)

Результат решения системы двух нелинейных уравнений с начальным приближением [1, –1] по программе, представленной на рисунке 3.31

```
-------------------------------------------------
ЗНАЧЕНИЯ функций f(I) и f(II) при найденных решениях:
-------------------------------------------------

f = 1.0e-014 *  -0.0888   -0.1110

-------------------------------------------------
Признак успешности вычислительного процесса - 1 - ДА, другое число - сомнительно:
-------------------------------------------------

pr = 1

>>
```

Рис. 3.34 (окончание)

Результат решения системы двух нелинейных уравнений с начальным приближением [1, –1] по программе, представленной на рисунке 3.31

```
-------------------------------------------------
   РЕШЕНИЕ СИСТЕМЫ НЕЛИНЕИНЫХ УРАВНЕНИИ ВИДА:
   f(I)=2x(1)^2-x(1)*x(2)-5x(1)+1=0
   f(II)=x(1)+3lg(x(1))-x(2)^2=0
-------------------------------------------------
Вид функций СНУ:
-------------------------------------------------
Первой функции f(I):
-------------------------------------------------
f(I)=2*x(1)^2-x(1)*x(2)-5*x(1)+1
-------------------------------------------------
Второй функции f(II):
-------------------------------------------------
f(II)=x(1)+3*log10(x(1))-x(2)^2
-------------------------------------------------
-------------------------------------------------
   РЕАЛИЗАЦИЯ ВЫЧИСЛИТЕЛЬНОГО ПРОЦЕССА И ЕГО РЕЗУЛЬТАТЫ:
-------------------------------------------------
 Начальные приближения для итерационных вычислений:
-------------------------------------------------

x0 =   4     4

-------------------------------------------------
Изменения параметров вычислительного процесса при последовательных приближениях:
-------------------------------------------------

                                Norm of      First-order   Trust-region
 Iteration  Func-count     f(x)          step          optimality   radius
     0          3         113.914                     93.6            1
     1          6         19.3903           1         24.6            1
     2          9        0.375054    0.918221         3.48          3.5
     3         12     0.000526921    0.178045       0.0825          3.5
     4         15    1.44014e-009   0.00727597     0.000127          3.5
     5         18    1.11752e-020   1.21751e-005   3.23e-010          3.5
     6         21    1.34106e-029   3.42246e-011    3.5e-014          3.5

Equation solved.

fsolve completed because the vector of function values is near zero
as measured by the selected value of the function tolerance, and
the problem appears regular as measured by the gradient.

<stopping criteria details>
```

Рис. 3.35 (начало)

Результат решения системы двух нелинейных уравнений с начальным приближением [4, 4] и по программе, представленной на рисунке 3.31

```
----------------------------------------------------
РЕШЕНИЕ:
----------------------------------------------------

x =  3.4874    3.2616

----------------------------------------------------
ЗНАЧЕНИЯ функций f(I) и f(II) при найденных решениях:
----------------------------------------------------

f =  1.0e-014 *

   -0.3553   -0.0888

----------------------------------------------------
Признак успешности вычислительного процесса - 1 - ДА, другое число - сомнительно:
----------------------------------------------------

pr = 1

>>
```

Рис. 3.35 (окончание)

Результат решения системы двух нелинейных уравнений с начальным приближением [4, 4] и по программе, представленной на рисунке 3.31

Полученные решения находятся в полном соответствии с графической интерпретацией (рис. 3.33) рассматриваемой задачи.

Системы нелинейных уравнений (СНУ) широко используются при математическом описании большого числа разнородных стационарных физико-химических и химико-технологических процессов с сосредоточенными параметрами (химических и фазовых равновесий, процессов разделения, реакторных процессов и т. п.). При моделировании химико-технологических процессов с рециклическими (обратными) материальными и тепловыми потоками размерности СНУ могут достигать очень больших величин, вследствие чего часто приходится снижать размерности решаемых задач декомпозиционными методами. Большинство существующих алгоритмов — соответственно решателей не удовлетворяет современным требованиям. Поэтому их приходится дорабатывать, и варьировать большое число их установочных параметров. Необходимо прикладывать серьезные усилия к разработке процедур инициализации и задания начальных приближений с учетом физического смысла решаемых задач.

Для преодоления перечисленных затруднений от прямых методов решения СНУ иногда переходят к решению оптимизационных (экстремальных) задач, когда минимизируют сумму квадратов функций всех уравнений системы (3.29). Широкое разнообразие и многообразие оптимизационных алгоритмов — от детерминированных, стохастических до интеллектуальных, эвристических и популяционных с обилием вариантов ограничений различного рода позволяет надеяться, что существующие проблемы при решении систем нелинейных уравнений в определенной степени могут быть преодолены.

3.3.3. Моделирование стационарного режима процесса химического превращения с линейной кинетической зависимостью скоростей стадий от концентраций веществ в изотермическом проточном реакторе с мешалкой путем решения системы линейных алгебраических уравнений (СЛАУ) при изменяющемся времени пребывания реакционного потока в реакторе

Система линейных алгебраических уравнений (СЛАУ) получается при математическом описании стационарного процесса в непрерывном изотермическом реакторе идеального перемешивания, когда кинетические зависимости скоростей отдельных стадий реакции линейно зависят от концентраций и стехиометрические коэффициенты участвующих в реакции веществ равны единице. В этом случае прямая задача моделирования решается при фиксированной температуре и формулируется следующим образом: при известном расходе поступающего в реактор потока и концентрациях в нем реагентов, известных значениях всех кинетических констант, соответствующих некоторой фиксированной температуре и известной стехиометрии всех стадий реакции, а также заданном значении времени пребывания в реакторе определить состав выходного потока из реактора.

Дано:

$v^{(0)}$ — расход входного потока;

$x^{(0)}$ — концентрация реагентов во входном потоке;

$\bar{k}$ — кинетические константы всех реакций;

τ — время пребывания реакционного потока в проточном реакторе как отношение реакционного объема (V) к расходу потока (v) в реакторе, $\tau = V/v$.

Определить:

$\bar{x}$ — концентрации всех компонентов реакции в выходном потоке реактора;

v — расход выходного потока.

Для простейшей последовательной реакции $A \xrightarrow{k_1} P \xrightarrow{k_2} S$ с известным мольным расходом входного потока ($v^{(0)} = n^{(0)}$), с известными мольными долями компонентов во входном потоке ($x_A^{(0)}$, $x_P^{(0)}$, $x_S^{(0)}$), с объемом реакционной смеси в молях ($V = N$) и заданными значениями кинетических констант k_1 и k_2 уравнения математического описания процесса химического превращения в реакторе записываются в виде уравнений покомпонентных балансов:

$$\begin{aligned} n^{(0)}x_A^{(0)} - nx_A + N(-k_1x_A) &= 0 \\ n^{(0)}x_P^{(0)} - nx_P + N(k_1x_A - k_2x_P) &= 0. \\ n^{(0)}x_S^{(0)} - nx_S + N(k_2x_P) &= 0 \end{aligned} \qquad (3.44)$$

С учетом того, что мольные потоки компонентов определяются по формуле $n_i = nx_i$ ($i = A, P, S$), мольные доли компонентов в реакционной смеси могут быть определены как $x_i = n_i / n$, и система (3.44) будет иметь вид:

$$
\begin{aligned}
n_A^{(0)} - n_A + N[-k_1(n_A / n)] &= 0 \\
n_P^{(0)} - n_P + N[k_1(n_A / n) - k_2(n_P / n)] &= 0, \\
n_S^{(0)} - n_S + N[k_2(n_P / n)] &= 0
\end{aligned}
\tag{3.45}
$$

и должна быть дополнена балансовым уравнением для выходного потока ($v = n$):

$$
n = n_A + n_P + n_S. \tag{3.46}
$$

Так как в условии задачи задается время пребывания в реакторе — $\tau = N/n$, то с учетом этого, а также уравнений (3.45) и (3.46) математическое описание изотермического реакционного процесса в реакторе имеет вид:

$$
\begin{aligned}
n_A^{(0)} - n_A - k_1 n_A \tau &= 0 \\
n_P^{(0)} - n_P + k_1 n_A \tau - k_2 n_P \tau &= 0 \\
n_S^{(0)} - n_S - k_2 n_P \tau &= 0 \\
-n_A - n_P - n_S + n &= 0
\end{aligned}
\tag{3.47}
$$

Исходные данные для решения поставленной задачи представлены в файле DATA.m (рис. 3.36). При этом задача решается для различных времен пребывания (τ) реакционной смеси в реакторе в интервале [1÷10] ч.

```
function DATA
%-------------------------------------------------------------------------------------------
%Программный код файла DATA.m — задание исходной информации для расчетов;
%-------------------------------------------------------------------------------------------
%Программа включает следующие
файлы:GLAV_model_linreactor_stat.m+DATA.m+model_stat_linreactor.m+model_stat_linreactor_3+REPORT.m;
%-------------------------------------------------------------------------------------------
%Программа расчета состава выходного потока в
%изотермическом реакторе идеального смешения со стехиометрической схемой
%реакции A - P -S
%-------------------------------------------------------------------------------------------
global PR n0 na0 np0 ns0 xa0 xp0 xs0 k1 k2 tau_a tau_b priz_print;
%1. Мольный расход потока сырья на входе в реактор n0 (мол.см./час)
n0=14.4;
%3. Мольные доли компонентов A,P,S в потоке сырья реактора
xa0=0.8; xp0=0.1;xs0=0.1;
%Мольные потоки компонентов i=A,P,S (мол."i"/час) во входном потоке реактора
na0=n0*xa0; np0=n0*xp0; ns0=n0*xs0;
%Константы скоростей реакций (час^(-1))
k1=0.35;k2=0.13;
%3. Левая граница интервала изменения времени пребывания в реакторе (час)
tau_a=1;
%4. Правая граница интервала изменения времени пребывания в реакторе (час)
tau_b=10;
%5. Признак решаемой СЛАУ:
%Если решается система 3-х линейных уравнений PR=1;
%Если решается система 4-х линейных уравнений PR=2;
PR=2;
%-----------------------------------------------------
%6. Форма вывода отчета о работе программы
%-----------------------------------------------------
%6.1. Если priz_print=0 (задается в файле DATA.m) — отчет выводится в командное окно MATLAB;
%6.3. Если priz_print=1 (задается в файле DATA.m) — отчет выводится в текстовый файл,
%   который размещается в той же папке MATLAB, в которой находятся все m-файлы программы;
priz_print=1;
end
```

Рис. 3.36

Файл исходных данных моделирования стационарного режима изотермического реактора идеального перемешивания с реакцией $A \rightarrow P \rightarrow S$ при заданных значениях времен пребывания

В случае равенства признака расчета 1 (PR = 1) (рис. 3.36) решается задача с применением функции model_stat_linreactor_3.m (рис. 3.36) — следующая система трех линейных уравнений при условии, что для рассматриваемой реакции $n^{(0)} = n$:

$$\begin{aligned}(1+k_1\tau)x_A + 0x_P + 0x_S &= x_A^{(0)}\\ (-k_1\tau)x_A + (1+k_2\tau)x_P + 0x_S &= x_P^{(0)},\\ 0x_A + (k_2\tau)x_P + 1x_S &= x_S^{(0)}\end{aligned} \tag{3.48}$$

или в общепринятом матричном виде:

$$\begin{bmatrix}(1+k_1)\tau & 0 & 0\\ -k_1\tau & (1+k_2\tau) & 0\\ 0 & -k_2\tau & 1\end{bmatrix}\cdot\begin{bmatrix}x_A\\ x_P\\ x_S\end{bmatrix} = \begin{bmatrix}x_A^{(0)}\\ x_P^{(0)}\\ x_S^{(0)}\end{bmatrix}. \tag{3.49}$$

Для решения этой СЛАУ с применением алгоритма model_stat_linreactor_3.m (рис. 3.37) используется метод обратной матрицы:

$$\bar{x} = \overline{\overline{A}}^{-1}\cdot\bar{b}. \tag{3.50}$$

```
function x= model_stat_linreactor_3(tau)
%-------------------------- ---------------------------------------------------------------------
%Программный код файла model_stat_linreactor(tau).m — расчет выходных концентраций
%продуктов из реактора путем решения системы линейных уравнений
%-------------------------------------------------------------------------------------------------
%Программа включает следующие файлы:
GLAV_model_linreactor_stat.m+DATA.m+model_stat_linreactor.m+model_stat_linreactor_4+REPORT.m;
%-------------------------------------------------------------------------------------------------
%Программа расчета состава выходного потока в
%изотермическом реакторе идеального смешения со стехиометрической схемой
%реакции A - P -S
%-------------------------------------------------------------------------------------------------
global xa0 xp0 xs0 k1 k2;
a(1,1)=1+k1*tau;a(1,2)=0;a(1,3)=0;b(1,1)=xa0;
a(2,1)=-k1*tau;a(2,2)=(1+k2*tau);a(2,3)=0; b(2,1)=xp0;
a(3,1)=0;a(3,2)=-k2*tau;a(3,3)=1; b(3,1)=xs0;
%Определение выходных параметров модели
x=inv(a)*b;
end
```

Рис. 3.37

Файл, в котором формируются матрица коэффициентов $\overline{\overline{A}}$ и вектор свободных членов $\bar{b}$ СЛАУ, при решении задачи моделирования в реакторе с применением мольных долей участвующих в реакции веществ (PR = 1)

В случае равенства признака расчета двум (PR = 2) (рис. 3.36) задача решается с применением функции model_stat_linreactor_4.m (рис. 3.38) — системы четырех линейных уравнений вида (3.51):

$$
\begin{aligned}
&(1+k_1\tau)n_A = n_A^{(0)}\\
&(-k_1\tau)n_A + (1+k_2\tau)n_P = n_P^{(0)}\\
&(-k_2\tau)n_P + n_S = n_S^{(0)}\\
&-n_A - n_P - n_S + n = 0
\end{aligned}, \qquad (3.51)
$$

или в матричном виде:

$$
\begin{bmatrix}
(1+k_1\tau) & 0 & 0 & 0\\
-k_1\tau & (1+k_2\tau) & 0 & 0\\
0 & -k_2\tau & 1 & 0\\
-1 & -1 & -1 & 1
\end{bmatrix}
\cdot
\begin{bmatrix} n_A\\ n_P\\ n_S\\ n \end{bmatrix}
=
\begin{bmatrix} n_A^{(0)}\\ n_P^{(0)}\\ n_S^{(0)}\\ 0 \end{bmatrix}. \qquad (3.52)
$$

Система линейных уравнений также решается методом обратной матрицы (3.50), но не относительно мольных долей в продуктовом потоке $\bar{x}$, как при значении признака PR = 1, а относительно мольных расходов компонентов (n_A, n_P, n_S) и суммарного мольного потока (n) в продуктовом потоке. При этом мольный объем реакционной смеси (N) легко определяется по формуле

$$N = n\tau. \qquad (3.53)$$

```
function x=model_stat_linreactor_4(tau)
%---------------------- ----------------------------------------------------------------------
%Программный код файла model_stat_linreactor_4 .m — расчет выходных концентраций
%продуктов из реактора путем решения системы линейных уравнений
%----------------------------------------------------------------------------------------------
%Программа включает следующие
файлы:GLAV_model_linreactor_stat.m+DATA.m+model_stat_linreactor.m+model_stat_linreactor_4+REPORT.m;
%----------------------------------------------------------------------------------------------
%Программа расчета состава выходного потока в
%изотермическом реакторе идеального смешения со стехиометрической схемой
%реакции A - P -S
%----------------------------------------------------------------------------------------------
global na0 np0 ns0 k1 k2;
a(1,1)=1+k1*tau;a(1,2)=0;a(1,3)=0;a(1,4)=0;b(1,1)=na0;
a(2,1)=-k1*tau;a(2,2)=(1+k2*tau);a(2,3)=0;a(2,4)=0;b(2,1)=np0;
a(3,1)=0;a(3,2)=-k2*tau;a(3,3)=1;a(3,4)=0; b(3,1)=ns0;
a(4,1)=-1;a(4,2)=-1;a(4,3)=-1;a(4,4)=1;b(4,1)=0;
%Определение выходных параметров модели
x=inv(a)*b;

end
```

Рис. 3.38

Файл, в котором формируются матрица коэффициентов $\bar{\bar{A}}$ и вектор свободных членов $\bar{b}$ СЛАУ, при решении задачи моделирования в реакторе с применением мольных расходов компонентов (PR = 2)

Следует отметить, что рассмотренный метод расчета (PR = 2) предпочтительнее первого (PR = 1) (рис. 3.37), так как позволяет определять расход продуктового потока n и не ограничен условием $n^{(0)} = n$.

Основная управляющая программа GLAV_model_linreactor_stat (рис. 3.39) исходя из значения признака (PR) 1 или 2 позволяет выбрать тот или

иной метод решения задачи. Здесь же (рис. 3.39) происходит считывание данных из файла DATA.m и пересчет концентраций в потоке в мольные потоки, и наоборот.

```
%-------------------------------------------------------------------------------------------------------
%Программный код файла GLAV_model_linreactor_stat.m — главная управляющая программа
%-------------------------------------------------------------------------------------------------------
%Программа включает следующие файлы:
GLAV_model_linreactor_stat.m+DATA.m+model_stat_linreactor.m+model_stat_linreactor_4+REPORT.m;
%-------------------------------------------------------------------------------------------------------
%Программа расчета состава выходного потока в
%изотермическом реакторе идеального смешения со стехиометрической схемой
%реакции A - P -S
%-------------------------------------------------------------------------------------------------------
clc;
clear all;
close all;
global PR t n0 n na np ns xa xp xs N tau_a tau_b;
DATA;
i=0;
for tau=tau_a:0.1:tau_b
    i=i+1;
    if PR==1
    x= model_stat_linreactor_3(tau);
    end
    if PR==2
    x= model_stat_linreactor_4(tau);
    end
t(i)=tau;
%-----------------------------------------------------
%Для модели model_stat_linreactor(tau)
%-----------------------------------------------------
if PR==1
xa(i)=x(1);
xp(i)=x(2);
xs(i)=x(3);
na(i)=n0*xa(i);np(i)=n0*xp(i);ns(i)=n0*xs(i);
n(i)=na(i)+np(i)+ns(i);
N(i)=tau*n(i);
end
%-----------------------------------------------------
%Для модели model_stat_linreactor_4(tau)
%-----------------------------------------------------
if PR==2
na(i)=x(1);
np(i)=x(2);
ns(i)=x(3);
n(i)=x(4);
N(i)=tau*n(i);
xa(i)=x(1)/x(4);
xp(i)=x(2)/x(4);
xs(i)=x(3)/x(4);
end
end
REPORT;
```

Рис. 3.39

Код основной управляющей программы, позволяющей решить систему уравнений математического описания процесса в изотермическом реакторе идеального перемешивания с применением мольных долей реагирующих веществ (PR = 1) либо мольных потоков реагирующих веществ (PR = 2)

Как и следовало ожидать, результаты расчета для различных времен пребывания в диапазоне [1÷10] ч по двум вариантам вычислений (PR = 1 и PR = 2) совпадают и представлены в Приложении (табл. П.2).

Графическая иллюстрация результатов для различных времен пребывания смеси в реакторе (τ) приведена на рисунке 3.40 — изменение мольных потоков компонентов (n_A, n_P, n_S) и суммарного мольного потока смеси (n) в выходном потоке; рисунок 3.41 — изменение мольного объема (N) реакционной смеси, определяемое как $N = n\tau$; рисунок 3.42 — изменение мольных долей компонентов в продуктовом потоке (x_A, x_P, x_S).

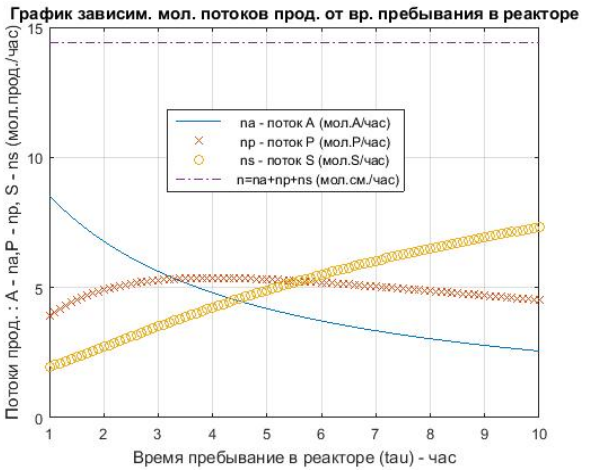


Рис. 3.40

Изменение мольных потоков веществ в зависимости от времени пребывания реакционной смеси в реакторе

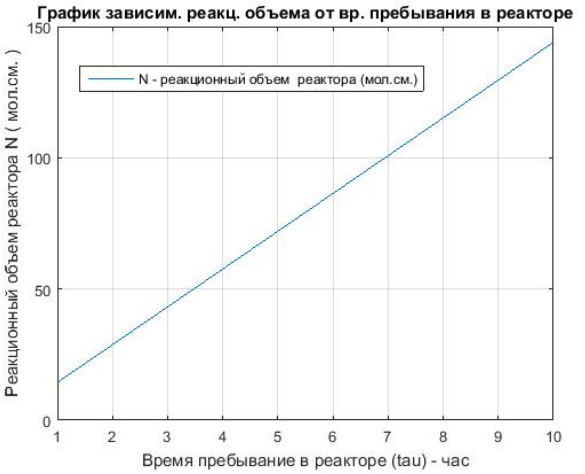


Рис. 3.41

Изменение реакционного объема с увеличением времени пребывания реакционной смеси в реакторе

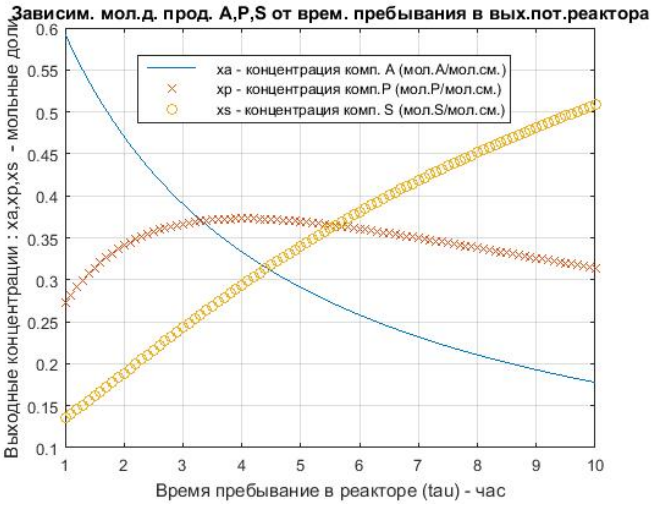


Рис. 3.42

Изменение состава продуктовых потоков в зависимости от времени пребывания реакционной смеси в реакторе

3.3.4. Моделирование стационарного режима процесса химического превращения с линейной кинетической зависимостью скоростей стадий реакции от концентрации веществ в изотермическом проточном реакторе путем решения системы нелинейных уравнений при изменяющемся реакционном объеме

В случае известного реакционного объема реактора N, даже с линейным кинетическим механизмом протекания реакции $A \xrightarrow{k_1} P \xrightarrow{k_2} S$, система уравнений математического описания процесса (3.44) не является линейной и представляет собой систему нелинейных уравнений (СНУ) следующего вида:

$$\begin{aligned} f_1 &= \frac{n_A^{(0)} - n_A}{N} - k_1\left(\frac{n_A}{n}\right) = 0 \\ f_2 &= \frac{n_P^{(0)} - n_P}{N} + k_1\left(\frac{n_A}{n}\right) - k_2\left(\frac{n_P}{n}\right) = 0 \\ f_3 &= \frac{n_S^{(0)} - n_S}{N} + k_2\left(\frac{n_P}{n}\right) = 0 \\ f_4 &= n - n_A - n_P - n_S = 0 \end{aligned} \quad . \qquad (3.54)$$

Система нелинейных уравнений (3.54) решается относительно n_A, n_P, n_S и n (в программе (рис. 3.45) это x_1, x_2, x_3, x_4) для различных реакционных объемов (N) в диапазоне 10÷100 [мол·см] (рис. 3.43 — код файла DATA.m). Для решения используется стандартный решатель MATLAB “fsolve” управляющей программы GLAV_model_linreac_N_stat.m (рис. 3.44) с функциями (3.54), при-

веденными в *m*-файле model_linreac_N_stat.m (рис. 3.45). Время пребывания в реакторе определяется по формуле:

$$\tau = N/n. \tag{3.55}$$

```
function DATA
%-------------------------------------------------------------------------------------
%Программный код файла DATA.m — задание исходной информации для расчетов;
%-------------------------------------------------------------------------------------
%Программа включает следующие
%файлы: GLAV_model_linreac_N_stat.m+DATA.m+model_linreac_N_stat.m+REPORT.m;
%-------------------------------------------------------------------------------------
%Программа расчета концентраций компонентов реакций в выходном потоке в
%изотермическом реакторе идеального смешения со стехиометрической схемой
%реакции A - P -S при заданном реакционном и параметрах входного потока.
%-------------------------------------------------------------------------------------
global n0 xa0 xp0 xs0 na0 np0 ns0 k1 k2 Nmin Nmax Ndel priz_print;
%1. Мольный расход потока сырья на входе в реактор n (мол.см./час)
n0=10;

%3. Мольные доли компонентов i= A,P,S в потоке сырья на входе в реактор xa0,xp0,xs0
(мол."i"/мол.см.)
xa0=0.7;xp0=0.15;xs0=0.15;
%Мольные потоки компонентов A,P,S (мол."i"/час) во входном потоке реактора
na0=n0*xa0; np0=n0*xp0; ns0=n0*xs0;

%3. Константы скоростей реакций (час^(-1))
k1=0.35;k2=0.13;

%4. Левая граница реакционного объема реактор (мол.см.)
Nmin=10;

%5. Правая граница реакционного объема реактор (мол.см.)
Nmax=100;

%6. Шаг увеличения реакционного объема реактор (мол.см.)
Ndel=5;
%-----------------------------------------------------
%7. Форма вывода отчета о работе программы
%-----------------------------------------------------
%7.1. Если priz_print=0 (задается в файле DATA.m) — отчет выводится в командное окно MATLAB;
%7.3. Если priz_print=1 (задается в файле DATA.m) — отчет выводится в текстовый файл,
%   который размещается в той же папке MATLAB, в которой находятся все m-файлы программы;
priz_print=1;
end
```

Рис. 3.43

Код файла исходных данных при моделировании стационарного режима изотермического реактора идеального перемешивания со стехиометрической схемой реакции $A \rightarrow P \rightarrow S$ при изменяющихся реакционных объемах

```
%-------------------------------------------------------------------------------------
%Программный код файла GLAV_model_linreac_N_stat.m — основная управляющая программа
%-------------------------------------------------------------------------------------
%Программа включает следующие
%файлы: GLAV_model_linreac_N_stat.m+DATA.m+model_linreac_N_stat.m+REPORT.m;
%-------------------------------------------------------------------------------------
%Программа расчета концентраций компонентов реакций в выходном потоке в
%изотермическом реакторе идеального смешения со стехиометрической схемой
%реакции A — P -S при заданном реакционном и параметрах входного потока.
%-------------------------------------------------------------------------------------
clc;
```

Рис. 3.44 (начало)

Основная управляющая программа моделирования стационарного режима изотермического реактора идеального перемешивания со стехиометрической схемой реакции $A \rightarrow P \rightarrow S$ при изменяющихся реакционных объемах

```
clear all;
close all;
global t xa xp xs na np ns Nmin Nmax Ndel Nreac tau n;
i=0;
DATA;
for Nreac=Nmin:Ndel:Nmax
 i=i+1;
[ni,f,pr]=fsolve('model_linreac_N_stat',[0.3,0.3,0.3,0.9]);
na(i)=ni(1);np(i)=ni(2);ns(i)=ni(3);n(i)=ni(4);
 nsum=ni(4);
 tau=Nreac/nsum;
 t(i)=tau;
 N(i)=nsum*tau;
 xa(i)=ni(1)/nsum;
 xp(i)=ni(2)/nsum;
 xs(i)=ni(3)/nsum;
end

REPORT;
```

Рис. 3.44 (окончание)

Основная управляющая программа моделирования стационарного режима изотермического реактора идеального перемешивания со стехиометрической схемой реакции $A \rightarrow P \rightarrow S$ при изменяющихся реакционных объемах

```
function f= model_linreac_N_stat(x)
%-------------------------------------------------------------------------------------------
%Программный код файла model_linrea_N_stat.m — формирование системы
%нелинейных уравнений математического описания модели стационарного процесса в реакторе
%с мешалкой в неявном виде
%-------------------------------------------------------------------------------------------
%Программа включает следующие
%файлы:GLAV_model_linreac_N_stat.m+DATA.m+model_linreac_N_stat.m+REPORT.m;
%-------------------------------------------------------------------------------------------
%Программа расчета концентраций компонентов реакций в выходном потоке в
%изотермическом реакторе идеального смешения со стехиометрической схемой
%реакции A — P -S при заданном реакционном и параметрах входного потока.
%-------------------------------------------------------------------------------------------
global na0 np0 ns0 k1 k2 Nreac
f(1)=(na0-x(1))/Nreac-k1*x(1)/x(4);
f(2)=(np0-x(2))/Nreac+k1*(x(1)/x(4))-k2*(x(2)/x(4));
f(3)=(ns0-x(3))/Nreac+k2*(x(2)/x(4));
f(4)=x(4)-x(1)-x(2)-x(3);
end
```

Рис. 3.45

Функции системы нелинейных уравнений, решаемой при моделировании стационарного режима изотермического реактора идеального перемешивания со стехиометрической схемой реакции $A \rightarrow P \rightarrow S$ при известном реакционном объеме

Программные коды файлов для решения задачи приведены на рисунках 3.43–3.45. Результаты расчетов для исходных данных, представленных в файле DATA.m (рис. 3.43), приведены в Приложении (табл. П.3). Графическая иллюстрация результатов расчета для различных реакционных объемов реактора приведена на рисунках 3.46–3.48.

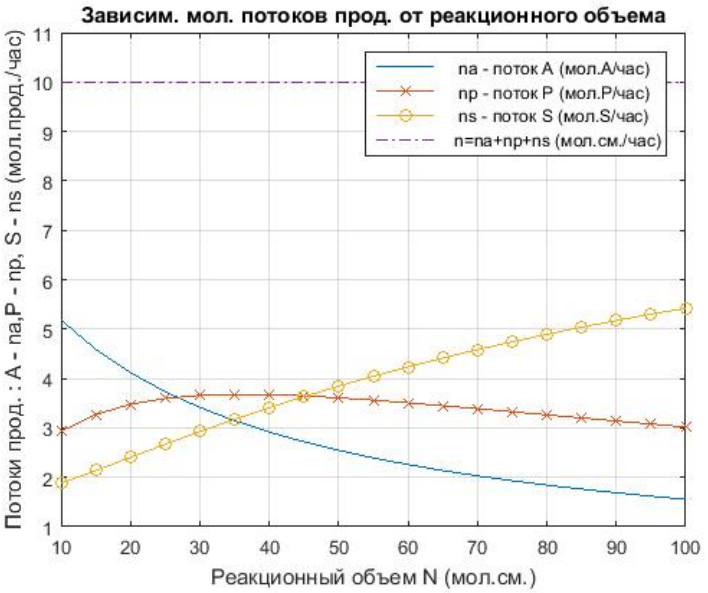


Рис. 3.46

Изменение мольных потоков и суммарного мольного потока компонентов реакции $A \rightarrow P \rightarrow S$ на выходе из изотермического реактора идеального перемешивания в зависимости от изменения реакционного объема

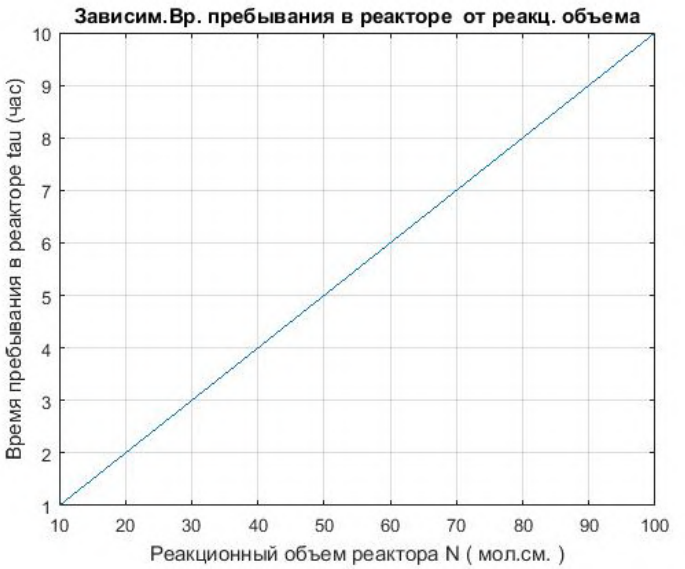


Рис. 3.47

Изменение времени пребывания реакционного потока в изотермическом реакторе идеального перемешивания с реакцией $A \rightarrow P \rightarrow S$ в зависимости от изменения реакционного объема

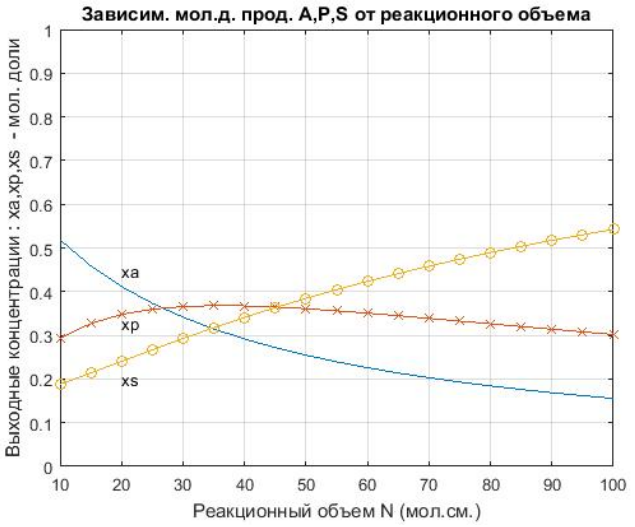


Рис. 3.48

Изменение мольных долей компонентов реакции $A \rightarrow P \rightarrow S$, протекающей в изотермическом реакторе идеального перемешивания, в зависимости от изменения реакционного объема.

3.3.5. Моделирование стационарного режима процесса химического превращения с нелинейной кинетической зависимостью скоростей стадий реакций от концентраций веществ в изотермическом проточном реакторе путем решения системы нелинейных уравнений при изменяющемся времени пребывания в реакторе

В общем случае при математическом описании стационарного процесса в непрерывном изотермическом реакторе идеального перемешивания получается система нелинейных уравнений (СНУ), так как зависимости скоростей стадий реакции от концентраций являются нелинейными, а стехиометрические коэффициенты уравнений стадий могут быть произвольными числами, в том числе и дробными.

Постановка задачи такая же, как в разделе 3.3.3, когда решается система линейных алгебраических уравнений (СЛАУ). Однако в случае системы нелинейных уравнений (СНУ) необходимо задать произвольные стехиометрические коэффициенты веществ во всех стадиях реакции, а также показатели степеней концентраций в кинетических уравнениях скоростей стадий, которые могут быть и дробными.

Исходные данные для реакции $A \xrightarrow{k_1} 2P \xrightarrow{k_2} S$ приведены в файле DATA.m (рис. 3.49) с теми же значениями констант скоростей k_1 и k_3. Однако порядки реакций отличаются: для первой реакции 0.33, а для второй — 3.

Расчет параметров продуктового потока проводится в диапазоне времен пребывания (τ) [0.5÷5.5] ч. Используются данные по мольным потокам, составы которых выражаются в мольных долях. Расход входного потока в реактор равен 10 мол. смеси/ч, а его состав в мольных долях: $x_A = 1$; $x_P = 0$; $x_S = 0$.

```
function DATA
%-------------------------------------------------------------------------------
%Программный код файла DATA.m — задание исходной информации для расчетов;
%-------------------------------------------------------------------------------
%Программа включает следующие
%файлы: GLAV_model_non_linreac_tau_stat.m+DATA.m+model_non_linreac_tau_stat.m+REPORT.m;
%-------------------------------------------------------------------------------
%Программа расчета концентраций компонентов реакций в выходном потоке
%изотермического реактора идеального смешения со стехиометрической схемой
%реакции A — 2P -S с порядком 1-ой реакции 0.33, а второй — 3.
%-------------------------------------------------------------------------------
global n0 xa0 xp0 xs0 na0 np0 ns0 k1 k2 tau_a tau_b priz_print;
%1. Мольный расход потока сырья на входе в реактор n (мол.см./час)
n0=10;

%3. Мольные доли компонентов i= A,P,S в потоке сырья на входе в реактор xa0,xp0,xs0
(мол."i"/мол.см.)
xa0=1;xp0=0;xs0=0;
%Мольные потоки компонентов A,P,S (мол."i"/час) во входном потоке реактора
na0=n0*xa0; np0=n0*xp0; ns0=n0*xs0;

%3. Константы скоростей реакций (час^(-1))
k1=0.35;k2=0.13;

%4. Левая граница интервала изменения времени пребывания в реакторе (час)
tau_a=0.05;

%5. Правая граница интервала изменения времени пребывания в реакторе (час)
tau_b=5.55;
%-------------------------------------------------------
%6. Форма вывода отчета о работе программы
%-------------------------------------------
%6.1. Если priz_print=0 (задается в файле DATA.m) — отчет выводится в командное окно MATLAB;
%6.3. Если priz_print=1 (задается в файле DATA.m) — отчет выводится в текстовый файл,
%  который размещается в той же папке MATLAB, в которой находятся все m-файлы программы;
priz_print=1;
end
```

Рис. 3.49

Код файла исходных данных для моделирования химической реакции $A \xrightarrow{k_1} 2P \xrightarrow{k_2} S$ с нелинейной кинетической зависимостью скоростей стадий реакции от концентраций веществ в изотермическом реакторе идеального перемешивания при изменении времени пребывания реакционного потока в реакторе

Система нелинейных уравнений (СНУ), описывающая процесс в реакторе, с учетом того, что $x_i = n_i / n$ ($i = A, P, S$), имеет вид:

$$\begin{aligned}
&n_A^{(0)} - n_A + N[-k_1\left(\frac{n_A}{n}\right)^{0.33}] = 0 \\
&n_P^{(0)} - n_P + N[2k_1\left(\frac{n_A}{n}\right)^{0.33} - k_2\left(\frac{n_P}{n}\right)^{2}] = 0 \\
&n_S^{(0)} - n_S + N[-k_2\left(\frac{n_P}{n}\right)^{2}] = 0 \\
&-n_A - n_P - n_S + n = 0
\end{aligned} \qquad (3.56)$$

Принимая во внимание то, что в условии задачи задано время пребывания реакционного потока в реакторе $\tau = N/n$, система уравнений может быть записана в следующем виде:

$$\begin{aligned}
&\frac{n_A^{(0)} - n_A}{n\tau} - k_1\left(\frac{n_A}{n}\right)^{0.33} = 0\\
&\frac{n_P^{(0)} - n_P}{n\tau} + 2k_1\left(\frac{n_A}{n}\right)^{0.33} - k_2\left(\frac{n_P}{n}\right)^2 = 0\\
&\frac{n_S^{(0)} - n_S}{n\tau} + k_2\left(\frac{n_P}{n}\right)^2 = 0\\
&n - n_A - n_P - n_S = 0
\end{aligned}. \qquad (3.57)$$

Для решения этой системы нелинейных уравнений используется решатель MATLAB “fsolve”, к которому осуществляется обращение из основной управляющей программы GLAV_model_non_linreac_tau_stat.m (рис. 3.50) с начальными приближениями по выходным потокам компонентов (n_i) и суммарному выходному потоку (n) для итерационных расчетов: n_A = 0.3; n_P = 0.3; n_S = 0.3; n = 0.9. Как следует из этого обращения, конкретный вид уравнений нелинейной системы представлен в программе model_non_linreac_tau_stat (рис. 3.51).

Графическая иллюстрация результатов расчётов с изменением времени пребывания смеси в реакторе (τ) приведена на рисунке 3.52 — изменение мольных потоков компонентов и суммарного потока на выходе из реактора; рисунок 3.53 — изменение реакционного объема ($N = n\tau$); рисунок 3.54 — изменение мольных долей компонентов на выходе из реактора.

```
%-------------------------------------------------------------------------------
%Программный код файла GLAV_model_non_linreac_tau_stat.m — главная управляющая программа
%-------------------------------------------------------------------------------
%Программа включает следующие
%файлы:GLAV_model_non_linreac_tau_stat.m+DATA.m+model_non_linreac_tau_stat.m+REPORT.m;
%-------------------------------------------------------------------------------
%Программа расчета концентраций компонентов реакций в выходном потоке
%изотермического реактора идеального смешения со стехиометрической схемой
%реакции A - 2P -S с порядком 1-ой реакции 0.33, а второй — 3.
%-------------------------------------------------------------------------------
clc;
clear all;
close all;
global t xa xp xs na np ns tau_a tau_b tau n N;
i=0;
DATA;
for tau=tau_a:0.5:tau_b
 i=i+1;
[ni,f,pr]=fsolve('model_non_linreac_tau_stat',[0.3,0.3,0.3,0.9]);
na(i)=ni(1);np(i)=ni(2);ns(i)=ni(3);n(i)=ni(4);
 nsum=ni(4);
 t(i)=tau;
 N(i)=nsum*tau;
 xa(i)=ni(1)/nsum;
 xp(i)=ni(2)/nsum;
 xs(i)=ni(3)/nsum;
end

REPORT;
```

Рис. 3.50

Код файла управляющей программы при моделировании химического превращения с нелинейной кинетической зависимостью скоростей стадий от концентраций веществ в реакторе идеального перемешивания при изменении времени пребывания реакционного потока в реакторе

```
function f= model_non_linrea_tau_stat(x)
%-----------------------------------------------------------------------------------------------
%Программный код файла  model_non_linreac_tau_stat.m — формирование системы
%нелинейных уравнений математического описания модели стационарного процесса в реакторе
%в неявном виде
%-----------------------------------------------------------------------------------------------
%Программа включает следующие
%файлы:GLAV_model_non_linreac_tau_stat.m+DATA.m+model_non_linreac_tau_stat.m+REPORT.m;
%-----------------------------------------------------------------------------------------------
%Программа расчета концентраций компонентов реакций в выходном потоке
%изотермического реактора идеального смешения со стехиометрической схемой
%реакции A - 2P -S с порядком 1-ой реакции 0.33, а второй — 3.
%-----------------------------------------------------------------------------------------------
global na0 np0 ns0 k1 k2 tau
f(1)=(na0-x(1))/x(4)/tau-k1*(x(1)/x(4))^0.33;
f(2)=(np0-x(2))/x(4)/tau+2*(k1*(x(1)/x(4))^0.33-k2*(x(2)/x(4))^2);
f(3)=(ns0-x(3))/x(4)/tau+k2*(x(2)/x(4))^2;
f(4)=x(4)-x(1)-x(2)-x(3);
end
```

Рис. 3.51

Код файла функций системы нелинейных уравнений, решаемой при моделировании реакции $A \xrightarrow{k_1} 2P \xrightarrow{k_2} S$ (порядок первой реакции 0,33, а второй — 2) в изотермическом реакторе идеального перемешивания

Полученные результаты расчетов для различных времен пребывания в реакторе (τ) представлены в таблице П.4 Приложения.

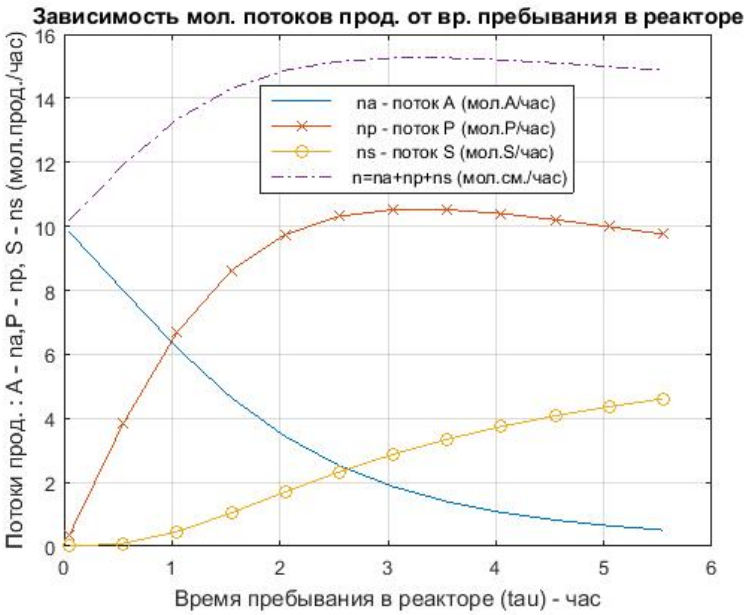


Рис. 3.52

Изменение молярных потоков компонентов веществ и суммарного молярного потока на выходе из изотермического реактора идеального перемешивания для реакции с нелинейной кинетической зависимостью скоростей стадий от концентраций веществ ($A \xrightarrow{k_1} 2P \xrightarrow{k_2} S$) в зависимости от изменения времени пребывания реакционного потока в реакторе

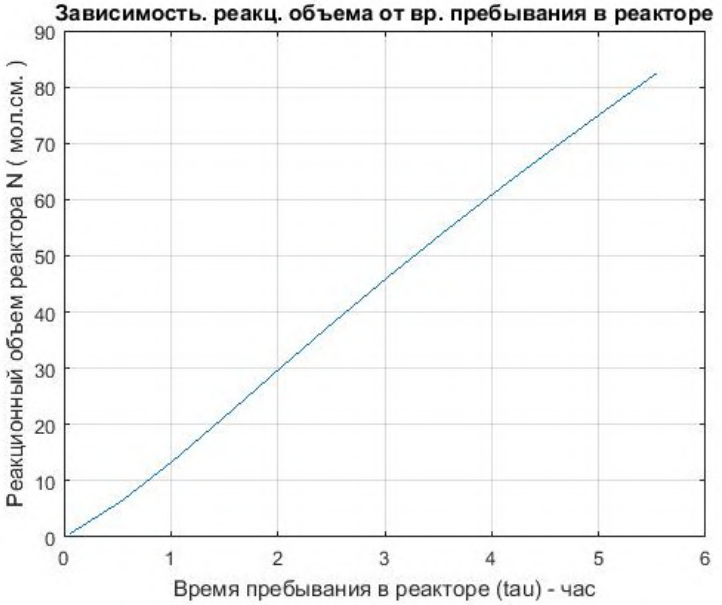


Рис. 3.53

Изменение реакционного объема изотермического реактора идеального перемешивания для реакции $A \xrightarrow{k_1} 2P \xrightarrow{k_2} S$ в зависимости от изменения времени пребывания реакционного потока в реакторе

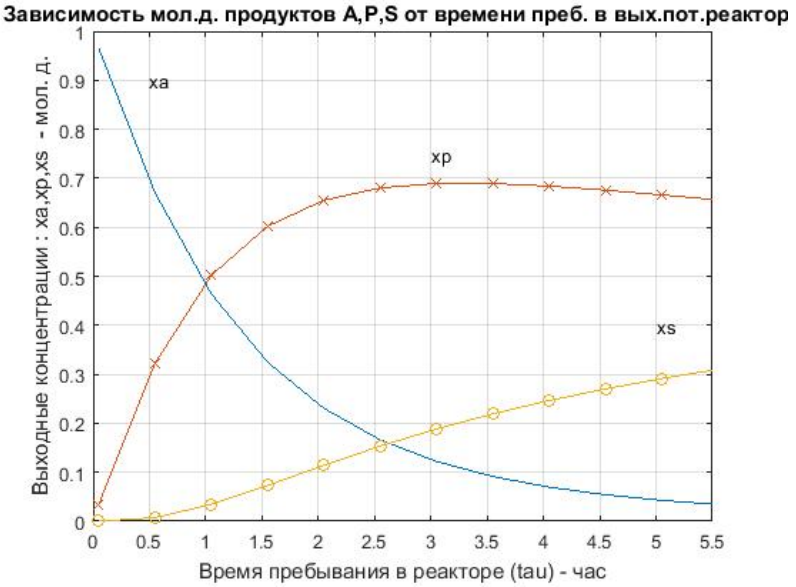


Рис. 3.54

Зависимость мольных долей компонентов в изотермическом реакторе идеального перемешивания для реакции $A \xrightarrow{k_1} 2P \xrightarrow{k_2} S$ в зависимости от изменения времени пребывания реакционного потока в реакторе

3.3.6. Моделирование стационарного режима процесса химического превращения с нелинейной кинетической зависимостью скоростей стадий реакций от концентрации веществ в изотермическом проточном реакторе путем решения системы нелинейных уравнений при изменяющемся реакционном объеме

Исходные данные для расчетов задаются в файле DATA.m (рис. 3.55), в котором приводится диапазон изменения реакционного объема — 20÷80 [мол·см], а параметры реакции $A \xrightarrow{k_1} 2P \xrightarrow{k_2} S$ такие же, как в предыдущем разделе.

Так как задается реакционный объем (N), который равен $N = n\tau$, то система нелинейных уравнений (3.57) записывается в следующем виде:

$$\begin{aligned} f_1 &= \frac{n_A^{(0)} - n_A}{N} - k_1\left(\frac{n_A}{n}\right)^{0.33} = 0 \\ f_2 &= \frac{n_P^{(0)} - n_P}{N} + k_1\left(\frac{n_A}{n}\right)^{0.33} - k_2\left(\frac{n_P}{n}\right)^{2} = 0 \\ f_3 &= \frac{n_S^{(0)} - n_S}{N} + k_2\left(\frac{n_P}{n}\right)^{2} = 0 \\ f_4 &= n - n_A - n_P - n_S = 0 \end{aligned} \quad . \qquad (3.58)$$

```
function DATA
%-------------------------------------------------------------------------------------------
%Программный код файла DATA.m — задание исходной информации для расчетов;
%-------------------------------------------------------------------------------------------
%Программа включает следующие
%файлы: GLAV_model_non_linreac_N_stat.m+DATA.m+model_non_linreac_N_stat.m+REPORT.m;
%-------------------------------------------------------------------------------------------
%Программа расчета концентраций компонентов реакций в выходном потоке
%изотермического реактора идеального смешения со стехиометрической схемой
%реакции A - 2P - S с порядком 1-ой реакции 0.33, а второй — 3.
%-------------------------------------------------------------------------------------------
global n0 xa0 xp0 xs0 na0 np0 ns0 k1 k2 Nmax Nmin Ndel priz_print;
%1. Мольный расход потока сырья на входе в реактор n (мол.см./час)
n0=10;

%3. Мольные доли компонентов i= A,P,S в потоке сырья на входе в реактор xa0,xp0,xs0 (мол."i"/мол.см.)
xa0=1;xp0=0;xs0=0;
%Мольные потоки компонентов A,P,S (мол."i"/час) во входном потоке реактора
na0=n0*xa0; np0=n0*xp0; ns0=n0*xs0;

%3. Константы скоростей реакций (час^(-1))
k1=0.35;k2=0.13;

%4. Левая граница изменения реакционного объема реактора (мол.см.)
Nmin=20;
%5. Правая граница изменения реакционного объема реактора (мол.см.)
Nmax=80;
%6. Шаг изменения реакционного объема реактора (мол.см.)
Ndel=5;
%-----------------------------------------------------------
%7. Форма вывода отчета о работе программы
%-----------------------------------------------------------
%6=7.1. Если priz_print=0 (задается в файле DATA.m) — отчет выводится в командное окно MATLAB;
%7.3. Если priz_print=1 (задается в файле DATA.m) — отчет выводится в текстовый файл,
%   который размещается в той же папке MATLAB, в которой находятся все m-файлы программы;
priz_print=1;
end
```

Рис. 3.55

Код файла исходных данных при моделировании реакции $A \xrightarrow{k_1} 2P \xrightarrow{k_2} S$ в изотермическом реакторе идеального перемешивания при различных значениях реакционного объема

Для решения этой системы уравнений с использованием решателя “fsolve” из управляющей программы GLAV_model_non_linreac_N_stat.m (рис. 3.56) выполняется обращение к функциям f_1, f_2, f_3, f_4 системы уравнений (3.58), которые представлены в функции файла model_non_linreac_N_stat.m (рис. 3.57).

```
%-------------------------------------------------------------------------------------------
%Программный код файла GLAV_model_non_linreac_N_stat.m — главная управляющая программа
%-------------------------------------------------------------------------------------------
%Программа включает следующие
%файлы: GLAV_model_non_linreac_N_stat.m+DATA.m+model_non_linreac_N_stat.m+REPORT.m;
%-------------------------------------------------------------------------------------------
%Программа расчета концентраций компонентов реакций в выходном потоке
%изотермического реактора идеального смешения со стехиометрической схемой
%реакции A — 2P -S с порядком 1-ой реакции 0.33, а второй — 3.
%-------------------------------------------------------------------------------------------
clc;
clear all;
close all;
global t xa xp xs na np ns Nmin Nmax Ndel Nreac tau n N;
i=0;
DATA;
for Nreac=Nmin:Ndel:Nmax
 i=i+1;
[ni,f,pr]=fsolve('model_non_linreac_N_stat',[0.3,0.3,0.3,0.9]);
na(i)=ni(1);np(i)=ni(2);ns(i)=ni(3);n(i)=ni(4);
 nsum=ni(4);
 tau=Nreac/nsum;
 t(i)=tau;
 N(i)=nsum*tau;
 xa(i)=ni(1)/nsum;
 xp(i)=ni(2)/nsum;
 xs(i)=ni(3)/nsum;
end

REPORT;
```

Рис. 3.56

Код файла управляющей программы при моделировании реакции $A \xrightarrow{k_1} 2P \xrightarrow{k_2} S$ в изотермическом реакторе идеального перемешивания при различных значениях реакционного объема

```
function f= model_non_linreac_N_stat(x)
%-------------------------------------------------------------------------------------------
%Программный код файла model_non_linreac_N_stat.m — формирование системы
%нелинейных уравнений математического описания модели стационарного процесса в реакторе
%в неявном виде
%-------------------------------------------------------------------------------------------
%Программа включает следующие
%файлы: GLAV_model_non_linreac_N_stat.m+DATA.m+model_non_linreac_N_stat.m+REPORT.m;
%-------------------------------------------------------------------------------------------
```

Рис. 3.57 (начало)

Код файла функций системы нелинейных уравнений, используемой при моделировании реакции $A \xrightarrow{k_1} 2P \xrightarrow{k_2} S$ в изотермическом реакторе идеального перемешивания при различных значениях реакционного объема

```
%Программа расчета концентрации компонентов реакции в выходном потоке
%изотермического реактора идеального смешения со стехиометрической схемой
%реакции A — 2P -S с порядком 1-ой реакции 0.33, а второй — 3.
%-----------------------------------------------------------------------------------
global na0 np0 ns0 k1 k2 Nreac
f(1)=(na0-x(1))/Nreac-k1*(x(1)/x(4))^0.33;
f(2)=(np0-x(2))/Nreac+2*(k1*(x(1)/x(4))^0.33-k2*(x(2)/x(4))^2);
f(3)=(ns0-x(3))/Nreac+k2*(x(2)/x(4))^2;
f(4)=x(4)-x(1)-x(2)-x(3);
end
```

Рис. 3.57 (окончание)

Код файла функций системы нелинейных уравнений, используемой при моделировании реакции $A \xrightarrow{k_1} 2P \xrightarrow{k_2} S$ в изотермическом реакторе идеального перемешивания при различных значениях реакционного объема

Результаты расчетов, зависящие от реакционного объема реактора N [мол·см], представлены в таблице П.5, а их графическая иллюстрация — на рисунках 3.58–3.60.

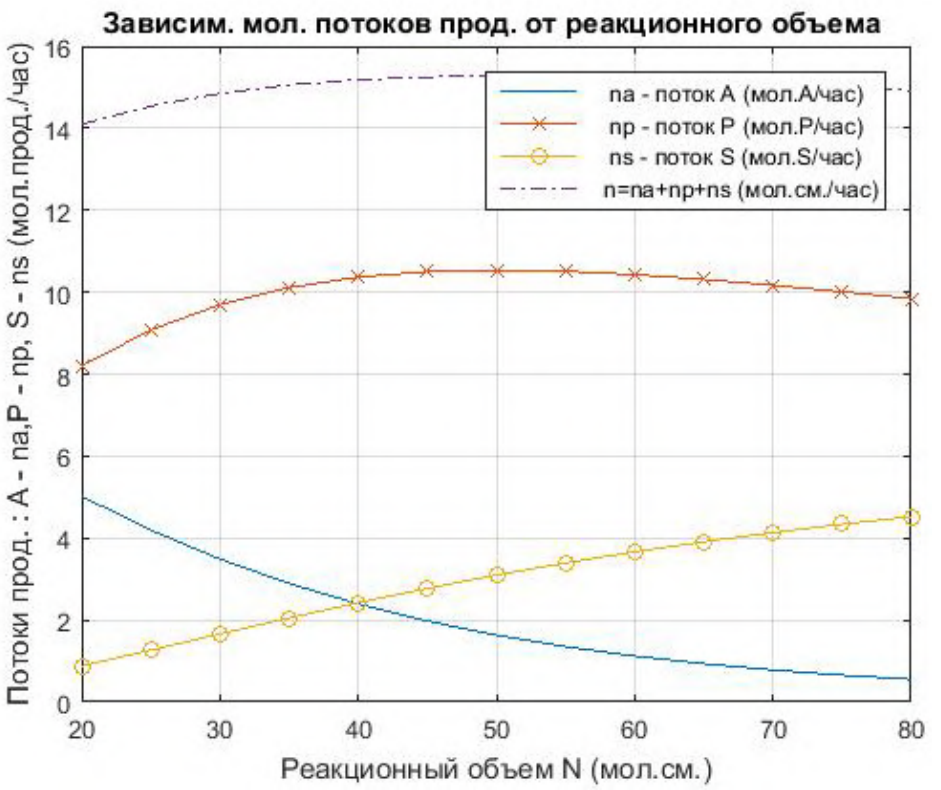


Рис. 3.58

Зависимость мольных потоков компонентов и суммарного мольного потока реакции на выходе из изотермического реактора идеального перемешивания с реакцией $A \xrightarrow{k_1} 2P \xrightarrow{k_2} S$ от изменения реакционного объема

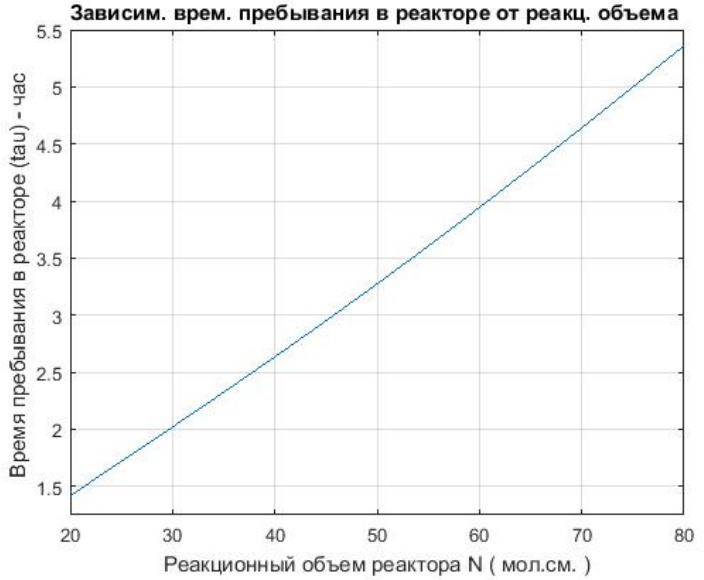


Рис. 3.59

Зависимость времени пребывания реакционного потока в изотермическом реакторе идеального перемешивания с реакцией $A \xrightarrow{k_1} 2P \xrightarrow{k_2} S$ от изменения реакционного объема

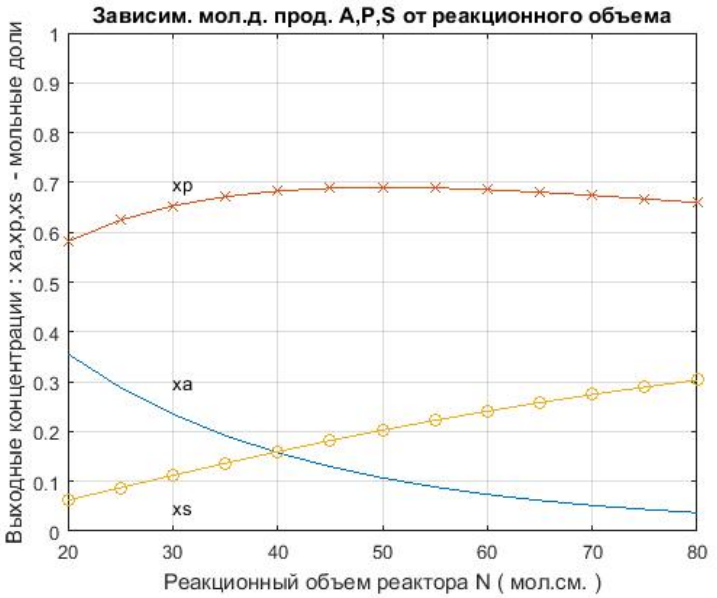


Рис. 3.60

Зависимость мольных долей компонентов в изотермическом реакторе идеального перемешивания с реакцией $A \xrightarrow{k_1} 2P \xrightarrow{k_2} S$ от изменения реакционного объема

3.4. Решение систем дифференциальных уравнений

Дифференциальные уравнения применяются для математического описания широкого класса процессов и явлений в химии и химической технологии. Прежде всего это относится к динамическим и нестационарным процессам, а также к распределенным в пространстве процессам. Эти уравнения включают в себя производные различных порядков, и для получения их решения численными методами — частного, физически обоснованного решения, — необходимо задавать дополнительные условия. Сами решения представляют собой функциональные зависимости от некоторых условно независимых переменных — времени для динамических процессов и пространственной или пространственных координат для распределенных в пространстве процессов.

Если производные, входящие в дифференциальные уравнения, зависят от одной независимой переменной, то уравнения называются обыкновенными дифференциальными уравнениями.

Если производные, входящие в дифференциальные уравнения, зависят от двух и более независимых переменных, то производные в них называются частными производными, а уравнения — дифференциальными уравнениями с частными производными.

Методы и алгоритмы решения двух типов дифференциальных уравнений отличаются. Поэтому в среде MATLAB для решения систем обыкновенных дифференциальных уравнений предлагаются всевозможные решатели, а для решения дифференциальных уравнений с частными производными реализовано специализированное приложение Partial Differential Equations Toolbox (PDE Toolbox). Как будет показано ниже, с использованием явного и неявного метода сеток для решения дифференциальных уравнений с частными производными можно избежать необходимости применения этого приложения.

3.4.1. Решение обыкновенных дифференциальных уравнений

В общем случае решатели MATLAB позволяют решать системы дифференциальных уравнений первого порядка с начальными условиями (одно дифференциальное уравнение — это частный случай системы с числом уравнений 1). Таким образом, задача решения формулируется в виде задачи Коши с начальными условиями при одном значении независимой переменной ($t = t^{(0)}$):

$$\begin{aligned}
\frac{\mathrm{d}x_1}{\mathrm{d}t} &= f_1\left(x_1, x_2, \ldots, x_n, t\right) \\
\frac{\mathrm{d}x_2}{\mathrm{d}t} &= f_2\left(x_1, x_2, \ldots, x_n, t\right) \\
&\ldots \\
\frac{\mathrm{d}x_n}{\mathrm{d}t} &= f_n\left(x_{1,1} x_2, \ldots, x_n, t\right)
\end{aligned} \tag{3.59}$$

при следующих начальных условиях:

$$x_1^{(0)} = x_1(t^{(0)}),\ x_2^{(0)} = x_2(t^{(0)}),\ \ldots,\ x_n^{(0)} = x_n(t^{(0)}). \tag{3.60}$$

Для решения с применением решателей MATLAB следует также задать интервал изменения независимой переменной с начальным значением $t^{(0)}$ и конечным значением $t^{(k)}$:

$$[t^{(0)} \div t^{(k)}]. \tag{3.61}$$

Результат решения представляет собой вектор n-функций, изменяющихся в интервале $[t^{(0)} \div t^{(k)}]$ (3.61), следующего вида:

$$\begin{matrix} x_i = x_i(t\ ) \\ i = 1, \ldots, n \end{matrix} \quad t^{(0)} \le t \le t^{(k)}, \tag{3.62}$$

при этом при любом значении t в интервале (3.61) все уравнения (3.59) при подстановке в них функций решения (3.62) должны превратиться в тождества.

В том случае, когда приходится решать системы дифференциальных уравнений второго порядка и выше, они должны быть преобразованы в системы дифференциальных уравнений первого порядка, и число уравнений в системе (3.59) увеличивается.

Можно рассмотреть такое преобразование на примере следующего дифференциального уравнения второго порядка:

$$\frac{d^2x}{dt^2} = -4\frac{dx}{dt} - 13x + \exp(\sin(t)). \tag{3.63}$$

Если принять, что $x_2 = x$:

$$x_1 = \frac{dx_2}{dt}, \tag{3.64}$$

то уравнение (3.63) может быть записано:

$$\frac{dx_1}{dt} = -4x_1 - 13x_2 + \exp(\sin(t)). \tag{3.65}$$

В результате вместо одного дифференциального уравнения второго порядка с учетом (3.64) и (3.65) получается система двух дифференциальных уравнений первого порядка:

$$\begin{matrix} \frac{dx_1}{dt} = -4x_1 - 13x_2 + \exp(\sin(t)) \\ \frac{dx_2}{dt} = x_1 \end{matrix} . \tag{3.66}$$

Искомыми решениями этой системы являются функции $x_2 = x_2(t)$ и $x_1 = x_1(t)$. Последняя функция в соответствии с (3.64) является функцией производной $dx_2/dt = dx_2/dt(t)$, а для получения частного решения задачи Коши необходимо задавать начальные условия по аналогии с (3.60).

Другая проблема решения задачи (3.59)–(3.60) связана с заданием дополнительных условий, когда они задаются при нескольких значениях независи-

мых переменных, в частном случае при двух. В этом случае для решения могут использоваться разностные схемы или, как будет показано, итерационные методы определения отсутствующих в постановке задачи дополнительных условий, в частности метод «стрельбы».

Еще одна проблема может возникнуть при реализации численных методов решения дифференциальных уравнений для так называемых жестких систем.

Система обыкновенных дифференциальных уравнений с постоянной матрицей коэффициентов $\overline{\overline{A}}$ вида:

$$\frac{dx}{dt} = \overline{\overline{A}}\bar{x}, \tag{3.67}$$

называемой жесткой, если собственные значения матрицы $\overline{\overline{A}}$ (λ_k, $k = 1, 2, \ldots, n$):

$$\overline{\overline{A}} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{22} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{bmatrix} \tag{3.68}$$

удовлетворяют следующим двум условиям:

$$\begin{gathered} Re(\lambda_k) < 0,\ k = 1,\ 2, \ldots, n, \\ S = \frac{\max\limits_{1 \le k \le n} \left| Re(\lambda_k) \right|}{\min\limits_{1 \le k \le n} \left| Re(\lambda_k) \right|} - \text{велико}, \end{gathered} \tag{3.69}$$

где *Re* — вещественная часть собственного значения; *S* — число жесткости.

На практике условия (3.69) означают, что элементы матрицы $\overline{\overline{A}}$ (3.68) на несколько порядков отличаются друг от друга и реализация явных численных методов типа метода Эйлера и метода Рунге — Кутта приводит к «раскачке» функции решения из-за ее жесткой компоненты, связанной с сильно отличающимися элементами матрицы $\overline{\overline{A}}$ (3.68), что и приводит к отсутствию сходимости. Как было показано в ранее изданном нами пособии [15], и т. д., применение неявных, как правило, итерационных методов решения не допускает «раскачки» решения и может обеспечить сходимость.

Поэтому в среде MATLAB существуют две группы методов решения обыкновенных дифференциальных уравнений:

- для нежестких систем, так называемые быстрые методы;
- для жестких систем, так называемые медленные методы.

Основная сложность при оценке жесткости систем дифференциальных уравнений состоит в том, что матрица $\overline{\overline{A}}$ редко бывает постоянной и ее элементы зависят от $x_i = x_i(t)$ и t. Теоретически можно воспользоваться формулами

(3.69) на каждом шаге решения при применении явных методов и сделать вывод о необходимости перехода к применению неявных методов. Однако такой подход нельзя считать продуктивным.

Наиболее целесообразно сначала использовать явные методы решения и при неудаче перейти к неявным методам.

Обращение к решателям MATLAB для решения системы обыкновенных дифференциальных уравнений (3.59)–(3.61) имеет вид:

$$[t, x] = \text{Название решателя ('prav', } [t^{(0)}, t^{(k)}], [x_1^{(0)}, x_2^{(0)}, \ldots, x_n^{(0)}], \qquad (3.70)$$

где название решателя, например, "ode45", соответствующего методу Рунге — Кутта 4–5-го порядка; prav — название файла, в котором формируется вектор правых частей системы дифференциальных уравнений, записанный в явном виде; $[t^{(0)}, t^{(k)}]$ — начальное и конечное значения независимой переменной при решении; $[x_1^{(0)}, x_2^{(0)}, \ldots, x_n^{(0)}]$ — начальные условия задачи, заданные при значении независимой переменной $t = t^{(0)}$ (левая граница интервала решения); t — результат решения: вектор дискретных значений независимой переменной, которым соответствуют дискретные точки функций решения; x — результат решения: матрица, каждый столбец которой соответствует дискретным точкам функций решения; число столбцов матрицы соответствует числу искомых функций решения и равно количеству заданных значений начальных условий.

Вывод результатов в соответствии с (3.70) осуществляется следующим образом:

- сначала выводятся компоненты вектора независимой переменной t, его размерность определяется используемым алгоритмом;
- затем выводится матрица x для (3.59)–(3.61), имеющая вид:

$$\begin{bmatrix} x_{11}^{(0)} & x_{12}^{(0)} & \cdots & x_{1n}^{(0)} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ x_{p1} & x_{p2} & \ldots & x_{pn} \end{bmatrix},$$

где p — размерность вектора t, определяемая выбранным решателем.

При реализации явных методов могут использоваться следующие решатели MATLAB: "ode45", "ode113", "ode23", а неявных — "ode15s", "ode23s", "ode23t" и "ode23tb".

3.4.1.1. Решение одного дифференциального уравнения явным методом с применением решателей "ode45", "ode23", "ode113"

Вид уравнения:

$$\frac{dy}{dt} = -y + \sin(yt). \qquad (3.71)$$

Начальное условие:

$$y(0) = 1.5. \qquad (3.72)$$

Диапазон изменения независимой переменной *t*:

$$t = [0 \div 35]. \tag{3.73}$$

Функция MATLAB для вычисления правой части дифференциального уравнения:

$$\begin{aligned} &\text{function}\, f = \text{prav}(t,\, y) \\ &f = -y + \sin(y \cdot t) \qquad . \\ &\text{end} \end{aligned} \tag{3.74}$$

Обращение к решателю “ode45” для решения уравнения выполняется следующим образом:

$$[t, y] = \text{ode45}(@\text{prav}, [0 - 35], 1.5). \tag{3.75}$$

Код программы решения difuravn.m приведен на рисунке 3.61, а результаты — в табличном виде на рисунке 3.62, а в графическом виде на рисунке 3.63.

```
function difuravn1
clc;
clear al;
close all;
disp('-------------------------------------------------------');
disp('    РЕШЕНИЕ ДИФФЕРЕНЦИАЛЬНОГО УРАВНЕНИЯ ВИДА (РЕШАТЕЛИ — "ode45","ode23","ode113" — явные
методы):');
disp('     dy/dt=f(t,y), где f(t,y)=-y+sin(y*t)');
disp('                    или')
disp('     dy/dt=-y+sin(y*t)');
disp('     с начальным условием y(0)=1.5 (задача Коши)');
disp('     в диапазоне изменения независимой переменной t : [0 - 35]');
disp('-------------------------------------------------------');
%-------------------------------------------------------------------
%Создание функции "prav", соответствующей правой части "f"
%дифференциального уравнения первого порядка, записанного в явном виде
%-------------------------------------------------------------------

function f=prav(t,y)
f=-y+sin(y*t);
end
%-------------------------------------------------------------------
%1. РЕШЕНИЕ ДИФФЕРЕНЦИАЛЬНОГО УРАВНЕНИЯ С ИСПОЛЬЗОВАНИЕМ РЕШАТЕЛЯ "ode45",
%реализующего метод Рунге-Кутта 4-5 порядка
%-------------------------------------------------------------------
[t,y]=ode45(@prav,[0,35],1.5);
%-------------------------------------------------------------------
%3. РЕШЕНИЕ ДИФФЕРЕНЦИАЛЬНОГО УРАВНЕНИЯ С ИСПОЛЬЗОВАНИЕМ РЕШАТЕЛЯ "ode23",
%реализующего метод Рунге-Кутта 2-3-го порядка
%-------------------------------------------------------------------
%[t,y]=ode23(@prav,[0,35],1.5);
%-------------------------------------------------------------------
%-------------------------------------------------------------------
%3. РЕШЕНИЕ ДИФФЕРЕНЦИАЛЬНОГО УРАВНЕНИЯ С ИСПОЛЬЗОВАНИЕМ РЕШАТЕЛЯ "ode113",
%реализующего метод Адамса
%-------------------------------------------------------------------
%[t,y]=ode113(@prav,[0,35],1.5);
%-------------------------------------------------------------------
disp('-------------------------------------------- ');
disp('    РЕЗУЛЬТАТ РЕШЕНИЯ — ФУНКЦИЯ y=y(t)  ');
disp('-------------------------------------------- ');
fprintf('%-5s %-10s %-10s \n','№','  t','  y(t)');
```

Рис. 3.61 (начало)

Программный код файла решения одного дифференциального уравнения явным методом

```
disp('-------------------------------------------------');
for i=1:length(t)
fprintf('%-5d %-10.5f %-10.5f \n',i,t(i),y(i));
end
disp('-------------------------------------------------');

plot(t,y,'-k');
title ('Результат решения диф.уравн.:dy/dt=-y+sin(y*t);y(0)=1.5');
xlabel ('t');
ylabel('y');
grid on;
end
```

Рис. 3.61 (окончание)

Программный код файла решения одного дифференциального уравнения явным методом

```
-------------------------------------------------
   РЕШЕНИЕ ДИФФЕРЕНЦИАЛЬНОГО УРАВНЕНИЯ ВИДА (РЕШАТЕЛИ — "ode45","ode23","ode113" — явные мето-
ды:
   dy/dt=f(t,y), где f(t,y)=-y+sin(y*t)
                 или
   dy/dt=-y+sin(y*t)
   с начальным условием y(0)=1.5 (задача Коши)
   в диапазоне изменения независимой переменной t : [0 35]
-------------------------------------------------
-------------------------------------------------
   РЕЗУЛЬТАТ РЕШЕНИЯ — ФУНКЦИЯ y=y(t)
-------------------------------------------------
№     t        y(t)
-------------------------------------------------
1    0.00000   1.50000
2    0.05024   1.42831
3    0.10048   1.36346
4    0.15071   1.30483
5    0.20095   1.25181

800  34.92112  0.08753
801  34.94422  0.08750
802  34.95816  0.08745
803  34.97211  0.08741
804  34.98605  0.08737
805  35.00000  0.08734
-------------------------------------------------
>>
```

Рис. 3.62

Результат решения одного дифференциального уравнения явным методом

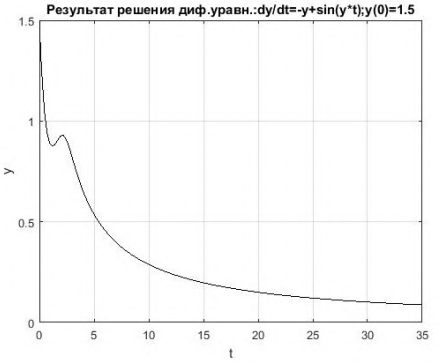


Рис. 3.63

Графическая иллюстрация результата решения одного дифференциального уравнения явным методом

3.4.1.3. Решение системы двух дифференциальных уравнений явным методом с применением решателя “ode45”

Вид системы уравнений:

$$\begin{aligned} \frac{dy_1}{dt} &= y_1 + 2y_2 - 9t \\ \frac{dy_2}{dt} &= 2y_1 + y_2 - 4\exp(t) \end{aligned} . \tag{3.76}$$

Начальные условия:

$$y_1(0) = 1; y_2(0) = 3. \tag{3.77}$$

Диапазон изменения независимой переменной:

$$t = [0 \div 2]. \tag{3.78}$$

Функция MATLAB для вычисления правых частей дифференциальных уравнений:

$$\begin{aligned} &\text{function}\, f = \text{prav}(t,\ y) \\ &f = \text{zeros}\ (2,\ 1) \\ &f(1) = y(1) + 2y(2) - 9t \\ &f(2) = 2y(1) + y(2) - 4\exp(t) \\ &\text{end} \end{aligned} . \tag{3.79}$$

Обращение к решателю “ode45” для решения системы уравнений выполняется следующим образом:

$$[t, y] = \text{ode45}(@\text{prav}, [0, 2], [1, 2]). \tag{3.80}$$

Код программы решения difursys3.m представлен на рисунке 3.64, а результаты решения — в табличном виде на рисунке 3.65, а в графическом виде на рисунке 3.66.

```
function difursys2
clc;
clear al;
close all;
disp('-----------------------------------------------------');
disp('    РЕШЕНИЕ СИСТЕМЫ ДИФ. УРАВНЕНИИ ВИДА:');
disp('    dy1/dt=f1(t,y1,y2), где f1(t,y1,y2)=y1+2y2-9t');
disp('    dy2/dt=f2(t,y1,y2), где f2(t,y1,y2)=2y1+y2-4exp(t)');
disp('-----------------------------------------------------');
disp('                  или')
disp('-----------------------------------------------------');
disp('    dy1/dt=y1+2*y2-9t');
disp('    dy2/dt=2y1+y2-4exp(t)');
disp('    с начальными условиями y1(0)=1 и y2(0)=2 (задача Коши)');
disp('    в диапазоне изменения независимой переменной t : [0 - 2]');
disp('-----------------------------------------------------');
%-------------------------------------------------------------------
%Создание функции "prav", соответствующей правым частям системы
%дифференциальных уравнений первого порядка "f1(t,y1,y2);f2(t,y1,y2);", записанного в явном
виде
%-------------------------------------------------------------------
```

Рис. 3.64 (начало)

Программный код файла решения системы двух дифференциальных уравнений явным методом

```
function f = prav(t,y)
f=zeros(2,1);
f(1)=y(1)+2*y(2)-9*t;
f(2)=2*y(1)+y(2)-4*exp(t);
end
%-------------------------------------------------------------------
%РЕШЕНИЕ СИСТЕМЫ 2-х ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ С ИСПОЛЬЗОВАНИЕМ РЕШАТЕ-
ЛЯ "ode45",
%реализующего метод Рунге-Кутта 4-5 порядка
%-------------------------------------------------------------------
[t,y]=ode45(@prav,[0 2],[1 2]);
disp('------------------------------------------------- ');
disp('    РЕЗУЛЬТАТ РЕШЕНИЯ — ФУНКЦИИ: y1=y1(t) и y2=y2(t) ');
disp('------------------------------------------------- ');
n=length(y);
   for i=1:n
   y1(i)=y(i,1);
   y2(i)=y(i,2);
   end
fprintf('%-5s %-10s %-10s %-10s \n','№','  t',' y1(t)',' y2(t)');
disp('-------------------------------------------------');
for i=1:n
fprintf('%-5d %-10.5f %-10.5f %-10.5f \n',i,t(i),y1(i),y2(i));
end
disp('-------------------------------------------------');

   plot(t,y1,'k-',t,y2,'k.');
   title('Результат решения системы 2-х диф. уравн. с нач.усл.:y1(0)=1;y2(0)=2');
   xlabel('t');ylabel('y1,y2')
   text(0.05,1.7,'y1');text(0.1,3.5,'y2');
grid;
end
```

Рис. 3.64 (окончание)

Программный код файла решения системы двух дифференциальных уравнений явным методом

```
-------------------------------------------------
   РЕШЕНИЕ СИСТЕМЫ ДИФ. УРАВНЕНИЙ ВИДА:
   dy1/dt=f1(t,y1,y2), где f1(t,y1,y2)=y1+2y2-9t
   dy2/dt=f2(t,y1,y2), где f2(t,y1,y2)=2y1+y2-4exp(t)
-------------------------------------------------
                  или
-------------------------------------------------
   dy1/dt=y1+2*y2-9t
   dy2/dt=2y1+y2-4exp(t)
   с начальными условиями y1(0)=1 и y2(0)=2 (задача Коши)
   в диапазоне изменения независимой переменной t : [0 - 2]
-------------------------------------------------
-------------------------------------------------
   РЕЗУЛЬТАТ РЕШЕНИЯ — ФУНКЦИИ: y1=y1(t) и y2=y2(t)
-------------------------------------------------
№     t          y1(t)       y2(t)
-------------------------------------------------
1    0.00000   1.00000   3.00000
2    0.01005   1.05004   3.00030
3    0.02010   1.09968   3.00120
4    0.03014   1.14893   3.00270
5    0.04019   1.19781   3.00478
6    0.09019   1.43565   3.02369
7    0.14019   1.66526   3.05630
8    0.19019   1.88758   3.10195
9    0.24019   3.10354   3.16001
10   0.29019   3.31407   3.22987
11   0.34019   3.52005   3.31095
```

Рис. 3.65 (начало)

Результат решения системы двух дифференциальных уравнений явным методом

```
12   0.39019   3.72239   3.40271
13   0.44019   3.92195   3.50463
14   0.49019   3.11963   3.61621
15   0.54019   3.31628   3.73697
16   0.59019   3.51277   3.86648
17   0.64019   3.70997   3.00430
18   0.69019   3.90875   3.15004
19   0.74019   4.10997   3.30329
20   0.79019   4.31452   3.46371
21   0.84019   4.52329   3.63093
22   0.89019   4.73717   3.80463
23   0.94019   4.95707   3.98449
24   0.99019   5.18391   4.17020
25   1.04019   5.41864   4.36149
26   1.09019   5.66223   4.55809
27   1.14019   5.91565   4.75973
28   1.19019   6.17991   4.96617
29   1.24019   6.45605   5.17717
30   1.29019   6.74514   5.39252
31   1.34019   7.04827   5.61200
32   1.39019   7.36658   5.83540
33   1.44019   7.70123   6.06254
34   1.49019   8.05345   6.29324
35   1.54019   8.42449   6.52733
36   1.59019   8.81563   6.76462
37   1.64019   9.22825   7.00498
38   1.69019   9.66377   7.24827
39   1.74019   10.12362   7.49432
40   1.79019   10.60934   7.74301
41   1.84019   11.12252   7.99420
42   1.88014   11.55343   8.19665
43   1.92010   13.00381   8.40057
44   1.96005   13.47457   8.60589
45   3.00000   13.96665   8.81256
-------------------------------------------------
>>
```

Рис. 3.65 (окончание)

Результат решения системы двух дифференциальных уравнений явным методом

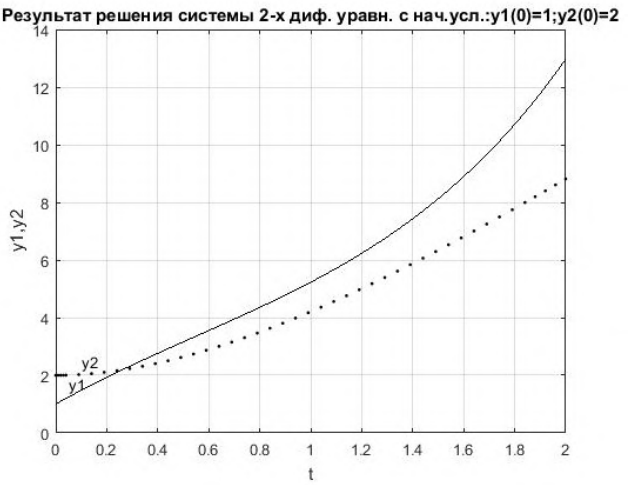


Рис. 3.66

Графическая иллюстрация результата решения системы двух дифференциальных уравнений явным методом

3.4.1.3. Решение дифференциального уравнения второго порядка явным методом с применением решателя “ode45”

Решается дифференциальное уравнение второго порядка (3.63), с начальными условиями $x(0.25) = -1$, $dx/dt\ (0.25) = 1$ в диапазоне изменения независимой переменной $t = (0.25 \div 2)$. Как было показано выше (3.64)–(3.66), уравнение может быть преобразовано в систему двух дифференциальных уравнений.

Преобразованная система (3.66) имеет вид:

$$\begin{aligned} \frac{dx_1}{dt} &= -4x_1 - 13x_2 + \exp\left(\sin\left(t\right)\right) \\ \frac{dx_2}{dt} &= x_1 \end{aligned} . \tag{3.81}$$

Начальные условия:

$$x_1(0.25) = 1;\ x_2(0.25) = -1. \tag{3.82}$$

Диапазон изменения независимой переменной:

$$t = [0.25 \div 2]. \tag{3.83}$$

Функция MATLAB для вычисления правых частей дифференциальных уравнений:

$$\begin{aligned} &\text{function}\, f = \text{prav}\left(t, x\right) \\ &f = \left[-4x\left(1\right) - 13x\left(2\right) + \exp\left(\sin\left(t\right)\right);\ x\left(1\right)\right]. \\ &\text{end} \end{aligned} \tag{3.84}$$

Фрагмент программы (рис. 3.67) обращения к решателю “ode45” для решения дифференциального уравнения второго порядка имеет вид:

$$\begin{aligned} &x0 = \left[1,\ -1\right]; \\ &\text{interval} = \left[0.25,\ 10\right]; \\ &\left[t, x\right] = \text{ode45}\left(@\,\text{prav},\ \text{interval},\ x0\right). \end{aligned} \tag{3.85}$$

Код программы решения difur3.m приведен не рисунке 3.67, а результаты решения — в табличном виде на рисунке 3.68, а в графическом виде на рисунке 3.69.

```
function difur3
clc;
clear al;
close all;
disp('-------------------------------------------------------');
disp('    РЕШЕНИЕ ДИФ. УРАВНЕНИЯ 2-го ПОРЯДКА ВИДА:');
disp('-------------------------------------------------------')
disp('    d^2x/dt^2=f(t,x,dx/dt), где f(t,x,dx/dt)=-4dx/dt-13x+exp(sin(t))');
disp('-------------------------------------------------------')
disp('    с начальными условиями x(0.25)=-1 и dx/dt(0.25)=1 (задача Коши)');
disp('    в диапазоне изменения независимой переменной t : [0.25 - 2]');
```

Рис. 3.67 (начало)

Программный код файла решения дифференциального уравнения второго порядка явным методом

```
disp('-----------------------------------------------------');
disp('                    или')
disp('-----------------------------------------------------');
disp('    после преобразования (x2=x и x1=dx/dt) к системе 2-х дифф. уравнений 1-го порядка       ');
disp('-----------------------------------------------------')
disp('    dx1/dt=-4x1-13x2+exp(sin(t))');
disp('    dx2/dt=x1');
disp('-----------------------------------------------------')
disp('    с начальными условиями x1(0.25)=1 и x2(0.25)=-1 (задача Коши)');
disp('    в диапазоне изменения независимой переменной t : [0.25 - 2]');
disp('-----------------------------------------------------');
%-------------------------------------------------------------------------
%Создание функции "prav", соответствующей правым частям системы
%дифференциальных уравнений первого порядка "f1(t,x1,x2);f2(t,x1,x2);", записанного в явном виде
%-------------------------------------------------------------------------
function f = prav(t,x)
f=[-4*x(1)-13*x(2)+exp(sin(t));x(1)];
end
%-------------------------------------------------------------------------
%РЕШЕНИЕ СИСТЕМЫ 2-х ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИИ С ИСПОЛЬЗОВАНИЕМ РЕШАТЕЛЯ
"ode45",
%реализующего метод Рунге-Кутта 4-5 порядка
%-------------------------------------------------------------------------
x0=[1,-1];
interval=[0.25 10];
[t,x]=ode45(@prav,interval,x0);
disp('----------------------------------------------------- ');
disp('    РЕЗУЛЬТАТ РЕШЕНИЯ — ФУНКЦИИ: x1=dx/dt(t) и x2=x(t) ');
disp('----------------------------------------------------- ');
n=length(x);
   for i=1:n
   x1(i)=x(i,1);
   x2(i)=x(i,2);
   end
fprintf('%-5s %-10s %-10s %-10s \n','№','  t','dx/dt(t)','  x(t)');
disp('-----------------------------------------------------');
for i=1:n
fprintf('%-5d %-10.5f %-10.5f %-10.5f \n',i,t(i),x1(i),x2(i));
end
disp('-----------------------------------------------------');

plot(t,x(:,1),'o',t,x(:,2),'-');
title('РЕШЕНИЕ ДИФ.УРАВН.2-ГО ПОРЯДКА С Н.У. x(0.25)=-1;dx/dt(0.25)=1');
xlabel('t');ylabel(' dx/dt;   x ');
text(0.3,3.4,'dx/dt(t)');text(0.4,-0.5,'x(t)');
legend('    dx/dt — производная решения','   x — функция решения');
end
```

Рис. 3.67 (окончание)

Программный код файла решения дифференциального уравнения второго порядка явным методом

```
-----------------------------------------------------
    РЕШЕНИЕ ДИФ. УРАВНЕНИЯ 2-го ПОРЯДКА ВИДА:
-----------------------------------------------------
    d^2x/dt^2=f(t,x,dx/dt), где f(t,x,dx/dt)=-4dx/dt-13x+exp(sin(t))
-----------------------------------------------------
    с начальными условиями x(0.25)=-1 и dx/dt(0.25)=1 (задача Коши)
    в диапазоне изменения независимой переменной t : [0.25 - 2]
-----------------------------------------------------
                    или
-----------------------------------------------------
    после преобразования (x2=x и x1=dx/dt) к системе 2-х дифф. уравнений 1-го порядка
-----------------------------------------------------
    dx1/dt=-4x1-13x2+exp(sin(t))
    dx2/dt=x1
```

Рис. 3.68 (начало)

Результат решения дифференциального уравнения второго порядка явным методом

```
-------------------------------------------------
   с начальными условиями x1(0.25)=1 и x2(0.25)=-1 (задача Коши)
   в диапазоне изменения независимой переменной t : [0.25 - 2]
-------------------------------------------------
-------------------------------------------------
   РЕЗУЛЬТАТ РЕШЕНИЯ — ФУНКЦИИ: x1=dx/dt(t) и x2=x(t)
-------------------------------------------------
№     t       dx/dt(t)    x(t)
-------------------------------------------------
1    0.25000   1.00000   -1.00000
2    0.25489   1.04961   -0.99499
3    0.25977   1.09796   -0.98974
4    0.26466   1.14507   -0.98426
5    0.26955   1.19095   -0.97855
------
140  9.64520   -0.09196  0.08310
141  9.71980   -0.08508  0.07650
142  9.78985   -0.07850  0.07077
143  9.85990   -0.07197  0.06550
144  9.92995   -0.06559  0.06069
145  10.00000  -0.05943  0.05631
-------------------------------------------------
>>
```

Рис. 3.68 (окончание)

Результат решения дифференциального уравнения второго порядка явным методом

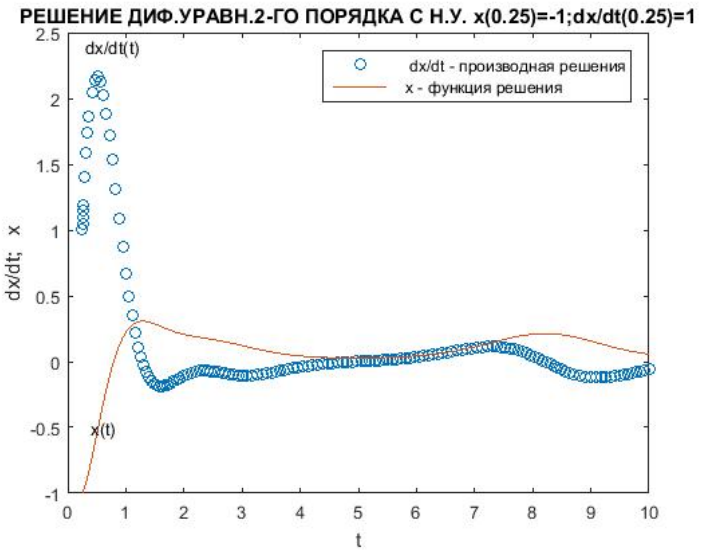


Рис. 3.69

Графическая иллюстрация результата решения дифференциального уравнения второго порядка явным методом

3.4.1.4. Решение жестких систем дифференциальных уравнений с применением решателей, названия которых оканчиваются буквами “s” и “t” — “ode15s”, “ode23s”, “ode23t”, “ode23tb”

В алгоритмах решения жестких (stiff) систем дифференциальных уравнений реализованы неявные методы, которые наряду с обычными численными методами интегрирования используют и итерационные методы, вследствие че-

го при большей достоверности они медленнее алгоритмов решения нежестких систем. Как указывалось в начале параграфа, предварительное определение числа жесткости (3.69) для систем дифференциальных уравнений довольно сложно, так как матрицы ее коэффициентов $\overline{\overline{A}}$ (3.68) являются зависимыми переменными и могут изменяться на каждом шаге решения.

Тем не менее для однородной нелинейной системы дифференциальных уравнений (3.67):

$$\frac{dx}{dt} = \overline{f}\left(t, \overline{x}\right) \tag{3.86}$$

переменные и зависимые от $\overline{x}$ и t коэффициенты матрицы $\overline{\overline{A}}$ (3.68) могут быть оценены при определенных значениях $\overline{x}$ и t, например при начальных условиях:

$$\frac{dx}{dt} = \overline{\overline{A}}\left(t^{(0)}, \overline{x}^{(0)}\right) \cdot \overline{x} \tag{3.87}$$

$$\overline{\overline{A}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}.$$

В результате может быть определено число жесткости s, которое в общем случае принимает разное значение на каждом шаге решения дифференциального уравнения. Основная проблема при этом связана с представлением элементов матрицы частных производных $\overline{\overline{A}}$ (3.87).

Пример 1. Решение жесткой системы трех однородных дифференциальных уравнений с переменными коэффициентами вида:

$$\begin{aligned} &\frac{dy_1}{dt} = -7y_1 + 7y_2, \\ &\frac{dy_2}{dt} = 157y_1 + y_2 - 1.15y_1y_3, \\ &\frac{dy_3}{dt} = 0.96y_1y_2 - 8.36y_3, \\ &\begin{bmatrix} \frac{dy_1}{dt} \\ \frac{dy_2}{dt} \\ \frac{dy_3}{dt} \end{bmatrix} = \begin{bmatrix} -7 & 7 & 0 \\ (157 - 1.15y_3) & 1 & -1.15y_1 \\ 0.96y_2 & 0.96y_1 & -8.36 \end{bmatrix} \cdot \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} \end{aligned} \tag{3.88}$$

с начальными условиями при $t = 0$: $\bar{y}^{(0)} = [-1,0,1]$, в диапазоне изменения независимой переменной $t = [0 \div 5]$ представлено на рисунке 3.70 в табличном и на рисунке 3.71 в графическом виде.

```
-------------------------------------------------
   РЕШЕНИЕ НЕЛИНЕЙНОЙ ЖЕСТКОЙ СИСТЕМЫ 3-Х ОДНОРОДНЫХ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕ-
НИИ С ПЕРЕМЕННЫМИ КОЭФФИЦИЕНТАМИ
          (РЕШАТЕЛИ - "ode23s"(метод Розенброка),"ode15s"(метод Гира),
           "ode23t"(малое число жесткости "s"),"ode23tb"(грубая оценка решения) — неявные методы):
   dy/dt=f(t,y),  где
   f(1)=-7y1+7y2
   f(2)=157y1+y2-1.15y1*y3
   f(3)=0.96y1*y2-8.36*y3
   с начальными условиями: y1(0)=-1; y2(0)=0;y3(0)=1(задача Коши)
   в диапазоне изменения независимой переменной t : [0 - 5]
-------------------------------------------------
Определение элементов матрицы правых частей диф. уравн. для вычисления их собственных чисел
a11=-7 a12=7 a13=0
a21=157-1.15y(3) a22=1 a23=-1.15y(1)
a31=0.96y(2) a32=0.96y(1) a33=-8.36
Значения элементов матрицы A при начальных условиях задачи

A =

  -7.0000   7.0000        0
 155.8500   1.0000   1.1500
      0   -0.9600   -8.3600

-------------------------------------------------
Собственные числа матрицы коэффициентов правых частей дифференциальных уравнений:
  30.2548
 -36.2535
  -8.3614

-------------------------------------------------
Число жесткости системы дифференциальных уравнений:

s =

   4.3358

-------------------------------------------------
-------------------------------------------------
Решение системы дифференциальных уравнений
-------------------------------------------------
   РЕЗУЛЬТАТ РЕШЕНИЯ — ФУНКЦИИ: y1=y1(t), y2=y2(t), y3=y3(t),
-------------------------------------------------
№     t        y1(t)     y2(t)     y3(t)
-------------------------------------------------
1    0.00000   -1.00000  0.00000   1.00000
2    0.00004   -0.99972  -0.00630  0.99966
3    0.00008   -0.99944  -0.01259  0.99933
4    0.00012   -0.99916  -0.01888  0.99899
5    0.00053   -0.99648  -0.08174  0.99564
.......
320  1.95450   0.65486   21.72452  30.23581
321  1.96220   1.79055   23.05654  28.54843
322  1.96990   3.96627   25.52644  27.18488
323  1.97760   4.24202   29.23162  26.19890
324  1.98529   5.68101   34.31782  25.69549
325  1.99299   7.35195   40.97567  25.85010
326  3.00000   9.13958   48.59544  26.79245
-------------------------------------------------
-------------------------------------------------------------------------------
Расчет по программе завершен, результаты также представлены в виде 1-ого графика
-------------------------------------------------------------------------------
>>
```

Рис. 3.70

Результаты решения жесткой нелинейной системы трех дифференциальных уравнений (3.88)

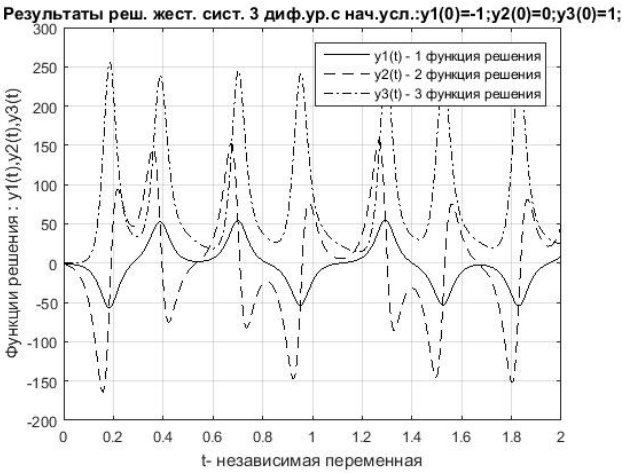


Рис. 3.71

Графическое представление результатов решения жесткой нелинейной системы трех дифференциальных уравнений (3.88)

Как видно из таблицы результатов (рис. 3.70), элементы матрицы $\overline{\overline{A}}$ (3.87), которые определены при заданных значениях начальных условий, характеризуются небольшим числом жесткости (3.69) s = 4.3358, и при этом не все вещественные части собственных значений матрицы $\overline{\overline{A}}$ (3.87) отрицательны. Однако с увеличением числа шагов решения, например уже при t = 0.7 s = 16.7769, вещественные части собственных значений матрицы $\overline{\overline{A}}$ все отрицательны:

— 13.8305;

— 0.7648 + 64.7555i;

— 0.7648 – 64.7555i.

Программный код файла решения системы (3.88) с использованием решателя “ode15s” приведен на рисунке 3.73.

```
function difursys6
clc;
clear all;
close all;
disp('-----------------------------------------------------');
disp('    РЕШЕНИЕ НЕЛИНЕЙНОЙ ЖЕСТКОЙ СИСТЕМЫ 3-Х ОДНОРОДНЫХ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ С
ПЕРЕМЕННЫМИ КОЭФФИЦИЕНТАМИ');
disp('              (РЕШАТЕЛИ — "ode23s"(метод Розенброка),"ode15s"(метод Гира),');
disp('               "ode23t"(малое число жесткости "s"),"ode23tb"(грубая оценка решения) — неявные методы):');
disp('    dy/dt=f(t,y),  где');
disp('    f(1)=-7y1+7y2 ');
disp('    f(2)=157y1+y2-1.15y1*y3');
disp('    f(3)=0.96y1*y2-8.36*y3');
disp('    с начальными условиями: y1(0)=-1; y2(0)=0;y3(0)=1(задача Коши)');
disp('    в диапазоне изменения независимой переменной t : [0 - 5]');
disp('-----------------------------------------------------');
```

Рис. 3.72 (начало)

Программный код файла решения жесткой нелинейной системы трех дифференциальных уравнений (3.88)

```
%-------------------------------------------------------------------------
%Создание функции "prav", соответствующей правой части "f"
%дифференциального уравнения первого порядка, записанного в явном виде
%-------------------------------------------------------------------------
function f= prav(t,y)
f=zeros(3,1);
f(1)=-7*y(1)+7*y(2);
f(2)=157*y(1)+y(2)-1.15*y(1)*y(3);
f(3)=0.96*y(1)*y(2)-8.36*y(3);
end
disp('Определение элементов матрицы правых частей диф. уравне. для вычисления их собственных чисел');
disp('a11=-7 a12=7 a13=0');
disp('a21=157-1.15y(3) a22=1 a23=-1.15y(1)');
disp('a31=0.96y(2) a32=0.96y(1) a33=-8.36');
disp('Значения элементов матрицы A при начальных условиях задачи');
yz=[-1 0 1];
y=yz;
A=[-7 7 0;157-1.15*y(3) 1 -1.15*y(1);0.96*y(2) 0.96*y(1) -8.36]
disp('-----------------------------------------------------');
disp('Собственные числа матрицы коэффициентов правых частей дифференциальных уравнений:');
l=eig(A);
disp(l);
disp('-----------------------------------------------------');
disp('Число жесткости системы дифференциальных уравнений:')
s=max(abs(real(l)))/min(abs(real(l)))
disp('-----------------------------------------------------');
l=eig(A);
disp('-----------------------------------------------------');
disp('Решение системы дифференциальных уравнений');
[t,y]=ode15s(@prav,[0 2],[-1 0 1]);
disp('------------------------------------------------- ');
disp('    РЕЗУЛЬТАТ РЕШЕНИЯ — ФУНКЦИИ: y1=y1(t), y2=y2(t), y3=y3(t),');
disp('------------------------------------------------- ');
fprintf('%-5s %-10s %-10s %-10s %-10s \n','№','  t','  y1(t)','  y2(t)','  y3(t)');
disp('-----------------------------------------------------');
for i=1:length(t)
   y1(i)=y(i,1);
   y2(i)=y(i,2);
   y3(i)=y(i,3);
fprintf('%-5d %-10.5f %-10.5f %-10.5f %-10.5f \n',i,t(i),y1(i),y2(i),y3(i));
end
disp('-----------------------------------------------------');
plot(t,y1,'-k',t,y2,'--k',t,y3,'-.k');
title ('Результаты реш. жест. сист. 3 диф.ур.с нач.усл.:y1(0)=-1;y2(0)=0;y3(0)=1;');
xlabel ('t- независимая переменная');
ylabel('Функции решения : y1(t),y2(t),y3(t)');
legend('y1(t) — 1 функция решения','y2(t) — 2 функция решения','y3(t) — 3 функция решения');
grid on;
 disp('-------------------------------------------------------------------------------');
disp('Расчет по программе завершен,  результаты также представлены в виде 1-ого графика');
disp('-------------------------------------------------------------------------------');
end
```

Рис. 3.72 (окончание)

Программный код файла решения жесткой нелинейной системы трех дифференциальных уравнений (3.88)

Пример 2. Решение жесткой системы четырех однородных дифференциальных уравнений с постоянными коэффициентами:

$$\frac{dy}{dt} = \overline{\overline{A}}\,\overline{y}, \tag{3.89}$$

где

$$\overline{\overline{A}} = \begin{bmatrix} 119.46 & 185.38 & 126.88 & 121.03 \\ -10.395 & -10.136 & -3.636 & 8.577 \\ -53.302 & -85.932 & -63.182 & -54.211 \\ -115.58 & -181.75 & -113.80 & -199.00 \end{bmatrix},$$

с начальными условиями при $t = 0$: $\overline{y}^{(0)} = [1,1,1,1]$, и в диапазоне изменения независимой переменной $t = [0 \div 5]$ приведено в табличном (рис. 3.73) и графическом (рис. 3.74) виде.

```
-------------------------------------------------------
   РЕШЕНИЕ ЛИНЕИНОИ ЖЕСТКОИ СИСТЕМЫ 4-Х ОДНОРОДНЫХ ДИФФЕРЕНЦИАЛЬНЫХ
УРАВНЕНИИ С ПОСТОЯННЫМИ КОЭФФИЦИЕНТАМИ ВИДА
          (РЕШАТЕЛИ — "ode23s"(метод Розенброка),"ode15s"(метод Гира),
           "ode23t"(малое число жесткости "s"),"ode23tb"(грубая оценка решения) — неявные
методы):
   dy/dt=f(t,y), где f(t,y)=Ay
              где
   матрица A(4,4)     =
   1 — строка: 119.46 185.38 126.88 121.03
   2 — строка: -10.395 -10.136 -3.636 8.577
   3 — строка: -53.302 -85.932 -63.182 -54.211
   4 — строка: -115.58 -181.75 -113.8 -199]
   с начальными условиями: y1(0)=1; y2(0)=1;y3(0)=1;y4(0)=1;(задача Коши)
   в диапазоне изменения независимой переменной t : [0 - 5]
-------------------------------------------------------
-------------------------------------------------------
Собственные числа матрицы коэффициентов правых частей дифференциальных уравне-
ний:

l =

  1.0e+02 *

 -1.5144 + 0.0000i
 -0.0020 + 0.1717i
 -0.0020 — 0.1717i
 -0.0101 + 0.0000i

-------------------------------------------------------
Число жесткости системы дифференциальных уравнений:

s =

  750.2332

-------------------------------------------------------
-------------------------------------------------------
   РЕЗУЛЬТАТ РЕШЕНИЯ — ФУНКЦИИ: y1=y1(t), y2=y2(t),y3=y3(t),y4=y4(t),
-------------------------------------------------------
№     t          y1(t)      y2(t)      y3(t)      y4(t)
-------------------------------------------------------
1    0.00000   1.00000   1.00000   1.00000   1.00000
2    0.00012   1.06573   0.99801   0.96950   0.92806
3    0.00024   1.13088   0.99589   0.93928   0.85733
4    0.00036   1.19543   0.99363   0.90935   0.78782
5    0.00124   1.65539   0.97322   0.69658   0.30848
......
299  4.87266   5.39634   -3.40798  -1.84598  -0.00791
300  4.89246   4.37413   -3.75662  -1.03548  0.45221
301  4.91226   3.86446   -3.79190  -0.11101  0.85737
302  4.93206   1.04358   -3.51233  0.82093   1.16178
303  4.95187   -0.87818  -1.95237  1.65391   1.33155
```

Рис. 3.73 (начало)

Результаты решения жесткой линейной системы четырех дифференциальных уравнений с постоянными коэффициентами (3.89)

```
304  4.97167   -3.68079  -1.17817  3.29366   1.34841
305  4.99147   -4.15961  -0.27979  3.66875   1.21172
306  5.00000   -4.65372  0.11966   3.73760   1.10929
-------------------------------------------------
-------------------------------------------------------------------------------
Расчет по программе завершен, результаты также представлены в виде 1-ого графика
-------------------------------------------------------------------------------
>>
```

Рис. 3.73 (окончание)

Результаты решения жесткой линейной системы четырех дифференциальных уравнений с постоянными коэффициентами (3.89)

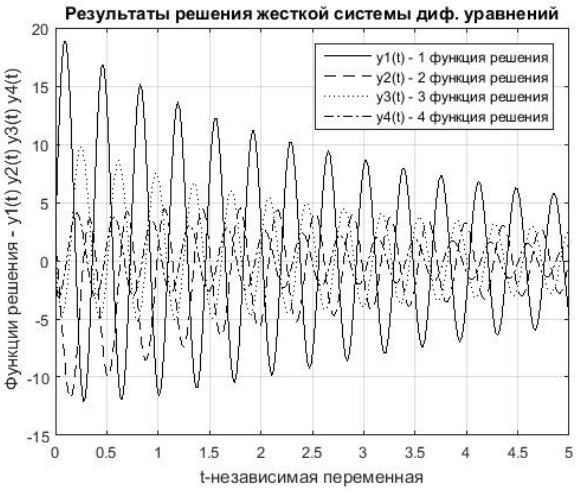


Рис. 3.74

Графическое представление результатов решения жесткой линейной системы четырех дифференциальных уравнений с постоянными коэффициентами (3.89)

Программный код файла решения с использованием решателя “ode15s” представлен на рисунке 3.75, в программе он представлен следующим образом:

$$[t, y] = \text{ode15s}\ (@\text{prav}, [0\ 5], [1\ 1\ 1\ 1]). \tag{3.90}$$

В этом случае правая часть системы дифференциальных уравнений задается в виде внутренней функции (рис. 3.75):

$$\begin{aligned} &\text{function}\, f = \text{prav}(t, y) \\ &f = A \cdot y \\ &\text{end.} \end{aligned} \tag{3.91}$$

```
function difursys5
clc;
clear all;
close all;
disp('-------------------------------------------------');
disp('    РЕШЕНИЕ ЛИНЕИНОИ ЖЕСТКОИ СИСТЕМЫ 4-Х ОДНОРОДНЫХ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИИ С
```

Рис. 3.75 (начало)

Программный код файла решения жесткой линейной системы четырех дифференциальных уравнений с постоянными коэффициентами

```
ПОСТОЯННЫМИ КОЭФФИЦИЕНТАМИ ВИДА ');
disp('              (РЕШАТЕЛИ — "ode23s"(метод Розенброка),"ode15s"(метод Гира),');
disp('               "ode23t"(малое число жесткости "s"),"ode23tb"(грубая оценка решения) — неявные методы):');
disp('    dy/dt=f(t,y), где f(t,y)=Ay');
disp('                  где')
disp('    матрица A(4,4)     = ');
disp('    1 - строка: 119.46 185.38 126.88 121.03 ');
disp('    2 - строка: -10.395 -10.136 -3.636 8.577');
disp('    3 - строка: -53.302 -85.932 -63.182 -54.211');
disp('    4 - строка: -115.58 -181.75 -113.8 -199]');
disp('    с начальными условиями: y1(0)=1; y2(0)=1;y3(0)=1;y4(0)=1;(задача Коши)');
disp('    в диапазоне изменения независимой переменной t : [0 - 5]');
disp('-----------------------------------------------------');
global A;
%-------------------------------------------------------------------
%Создание функции "prav", соответствующей правой части "f"
%дифференциального уравнения первого порядка, записанного в явном виде
%-------------------------------------------------------------------
function f=prav(t,y)
f=A*y;
end
A=[119.46 185.38 126.88 121.03;-10.395 -10.136 -3.636 8.577;-53.302 -85.932 -63.182 -54.211;-115.58 -181.75 -113.8 -199];
%-----------------------------------------------------------
%Решение системы дифференциальных уравнений
%-----------------------------------------------------------
[t,y]=ode15s(@prav,[0 5],[1 1 1 1]);
disp('-----------------------------------------------------');
disp('Собственные числа матрицы коэффициентов правых частей дифференциальных уравнений:');
l=eig(A)
disp('-----------------------------------------------------');
disp('Число жесткости системы дифференциальных уравнений:')
s=max(abs(real(l)))/min(abs(real(l)))
disp('-----------------------------------------------------');
disp('------------------------------------------------- ');
disp('    РЕЗУЛЬТАТ РЕШЕНИЯ  - ФУНКЦИИ: y1=y1(t), y2=y2(t),y3=y3(t),y4=y4(t),');
disp('------------------------------------------------- ');
n=length(y);
   for i=1:n
   y1(i)=y(i,1);
   y2(i)=y(i,2);
   y3(i)=y(i,3);
   y4(i)=y(i,4);
   end
fprintf('%-5s %-10s %-10s %-10s %-10s %-10s \n','№','  t','y1(t)','  y2(t)','y3(t)','y4(t)');
disp('-----------------------------------------------');
for i=1:n
fprintf('%-5d %-10.5f %-10.5f %-10.5f %-10.5f %-10.5f \n',i,t(i),y1(i),y2(i),y3(i),y4(i));
end
disp('-----------------------------------------------');

%-------------------------------------------------------------
%Построение графика решения
%-------------------------------------------------------------
plot(t,y(:,1),'k-',t,y(:,2),'k--',t,y(:,3),'k:',t,y(:,4),'k-.');
title('Результаты решения жесткой системы диф. уравнений');
xlabel('t-независимая переменная');ylabel(' Функции решения - y1(t) y2(t) y3(t) y4(t)');
legend('y1(t) - 1 функция решения','y2(t) - 2 функция решения','y3(t) - 3 функция решения','y4(t) - 4 функция реше-
ния');
grid;
 disp('-------------------------------------------------------------------------------');
disp('Расчет по программе завершен,  результаты также представлены в виде 1-ого графика');
disp('-------------------------------------------------------------------------------');
end
```

Рис. 3.75 (окончание)

Программный код файла решения жесткой линейной системы четырех дифференциальных уравнений с постоянными коэффициентами

Далее в программе матрица $\overline{\overline{A}}$ задается в виде массива (рис. 3.75), в котором строки матрицы отделяются точкой с запятой, а ее собственные числа ℓ определяются стандартной функцией MATLAB:

$$\ell = eig(A). \tag{3.92}$$

Число жесткости s (рис. 3.74), определяемое по формуле (3.69), характеризует жесткость рассматриваемой системы дифференциальных уравнений и равно:

$$s = 750.2332, \tag{3.93}$$

а все вещественные части собственных значений матрицы $\overline{\overline{A}}$ (3.89) отрицательны.

3.4.1.5. Решение двухточечной краевой задачи методом стрельбы

Решение систем обыкновенных дифференциальных уравнений и дифференциальных уравнений, порядок которых больше чем единица, не всегда выполняется при задании начальных условий как в случае решения задачи Коши, когда все дополнительные условия, накладываемые на искомые функции-решения, задаются при одном значении независимой переменной $t = t^{(0)}$ (3.59), (3.60). Часто для получения физически обусловленных решений приходится задавать дополнительные условия типа (3.60) при разных значениях независимых переменных, например двух ($t^{(0)}$, $t^{(k)}$) — так называемая двухточечная краевая задача, или для нескольких значений независимых переменных — так называемая многоточечная краевая задача. Даже в простейшем варианте двухточечной краевой задачи получение решения требует дополнительных усилий и усложнения вычислительного алгоритма.

Одним из распространенных методов решения двухточечных краевых задач является метод стрельбы, в соответствии с которым все отсутствующие при одном значении независимой переменной ($t^{(0)}$) дополнительные условия определяются одним из известных итерационных алгоритмов решения нелинейных уравнений (раздел 3.2) и их систем (раздел 3.3) или одномерной/многомерной оптимизацией (раздел 3.5) с критерием, обеспечивающим выполнение дополнительных условий, заданных при другом значении независимой переменной. Реализация такой процедуры расчетов сводится к многократному решению систем дифференциальных уравнений с различными генерируемыми итерационным алгоритмом дополнительными условиями до получения требуемого решения.

В качестве примера решения двухточечной краевой задачи целесообразно рассмотреть следующее дифференциальное уравнение второго порядка:

$$\frac{d^2x}{dt^2} = \left(\frac{1}{x}\right)\left(\frac{dx}{dt}\right)^2 \tag{3.94}$$

с краевыми условиями, заданными при $t^{(0)} = 0$ и $t^{(k)} = 1$:

$$x(t^{(0)}) = x(0) = 1$$
$$x(t^{(k)})) = x(1) = 3.718. \tag{3.95}$$

Преобразуя уравнение второго порядка (3.94), как это требуется для применения решателей MATLAB:

$$x_2 = x;\ x_1 = dx/dt, \tag{3.96}$$

получаем систему двух дифференциальных уравнений первого порядка вида:

$$\frac{dx_1}{dt} = \frac{x_1^2}{x_2}$$
$$\frac{dx_2}{dt} = x_1 \tag{3.97}$$

с краевыми условиями:

$$x_2(0) = 1$$
$$x_2(1) = 3.718. \tag{3.98}$$

Процедура решения системы уравнений (3.97) с краевыми условиями заключается в следующем.

1. Для получения частного решения системы в интервале [0, 1] используется заданное краевое условие при $t = t^{(0)} = 0 - x_2(0) = 1$ и задается оценка (начальное приближение) значения производной x_1 при $t = 0 - x_1(0) = x_{1p} = 0$, которое в процессе итерационных расчетов будет уточняться (корректироваться).

2. Решается система уравнений (3.97) в интервале [0, 1] при условии, что $x_2(0) = 1$ и $x_1(0) = x_{1p}$, и определяется $x_2^{расч}(1)$, значение которого зависит от x_{1p}, и эту зависимость можно представить в следующем общем виде:

$$x_2^{расч}(1) = x_2^{расч}(1)\{x_{1p}\}. \tag{3.99}$$

3. Результат расчета $x_2^{расч}(1)$ сравнивается со вторым условием (3.98) — $x_2(1) = 3.718$:

$$x_2^{расч}(1)\{x_{1p}\} = x_2(1)? \tag{3.100}$$

Если равенство (3.100) не соблюдается, то необходимо генерировать новую оценку (приближение) x_{1p}.

4. Для генерации очередных приближений x_{1p} в соответствии с (3.100) необходимо реализовать алгоритм решения нелинейного уравнения (раздел 3.2):

$$x_2^{расч}(1)\{x_{1p}\} - x_2(1) = 0. \tag{3.101}$$

5. Получение результата решения этого уравнения x_{1p}^* означает выполнение равенства (3.100) (с учетом того, что при значении решения x_{1p}^* был завершен процесс многократного решения системы дифференциальных уравнений (3.97) с различными приближениями x_{1p}, что автоматически подразумевает

корректное решение краевой задачи (3.97) и соответственно (3.94) с краевыми условиями (3.95)).

Программа решения представленной краевой задачи включает в себя два *m*-файла: функции Glav_uravn_kraev_zadach.m (рис. 3.76) и difur4.m (рис. 3.77).

```
function Glav_uravn_kraev_zadach
clc;
clear all;
close all;
global x2fix t x1 x2 n;
disp('-----------------------------------------------------');
disp('    РЕШЕНИЕ ДИФ. УРАВНЕНИЯ 2-го ПОРЯДКА ВИДА:');
disp('-----------------------------------------------------')
disp('    d^2x/dt^2=f(t,x,dx/dt), где f(t,x,dx/dt)=(1/x)(dx/dt)^2');
disp('-----------------------------------------------------')
disp('    с краевыми условиями x(0)=1  x(1)=3.718 (краевая задача)');
disp('    в диапазоне изменения независимой переменной t : [0 - 1]');
disp('-----------------------------------------------------');
disp('                  или')
disp('-----------------------------------------------------');
disp('    после преобразования (x2=x и x1=dx/dt) к системе 2-х дифф. уравнений 1-го порядка ');
disp('-----------------------------------------------------')
disp('    dx1/dt=x1^2/x2');
disp('    dx2/dt=x1');
disp('-----------------------------------------------------')
disp('    с дополнительными условиями x1(0)=x1p и x2(0)=1, (задача Коши)');
disp('    где x1p=0 — прибл. итерационно корректируемое знач. x1(0), пока в результате решения системы 2-х
диф.ур.');
disp('    не выполнится условие: x2(1)расч.=x2fix — (x2fix=3.718); - метод "стрельбы" при решении краевой зада-
чи');
disp('-----------------------------------------------------')
disp('    В этом случае решение краевой задачи сводится к решению уравнения F(x1p)=0: ');
disp('    где F(x1p)=x2(1)расч.(x1p)-x2(1)задан.=0, причем из условия задачи следует: x2(1)задан.=3.718')
disp('-----------------------------------------------------');
%Решение корректирующего уравнения методом "стрельбы"
%Требуемое условие для выполнения x2(1)=3.718;
x2fix=3.718;
%Задание нач. приближения для решения уравнения
x1p=0;
%Задание опций для отслеживания выполнения итераций
option=optimset('Display','iter');
%Решение корректирующего уравнения с использованием решателя "fzero"
[x1_proizv,Func,pr]=fzero(@difur4,x1p,option);
disp('-----------------------------------------------------');
disp('    1. РЕЗУЛЬТАТ РЕШЕНИЯ — ДОПОЛНИТЕЛЬНОЕ УСЛОВИЕ ДЛЯ ПРОИЗВОДНОИ x1=dx/dt(0)');
disp('-----------------------------------------------------');
disp(['1.1.Рассчитанное значение производной при t=0( dx/dt(0) ) = ' num2str(x1_proizv,'%10.2f') ' ']);
disp('-----------------------------------------------------');
disp(['1.3. Финальное значение функции решения уравнения ( Func ) = ' num2str(Func,'%10.4g') ' ']);
disp('-----------------------------------------------------');
disp(['1.3. Признак корректности вычислительного процесса ( pr ) = ' num2str(pr,'%10d') ' ']);
disp('-----------------------------------------------------');
disp('    3. РЕЗУЛЬТАТ РЕШЕНИЯ — ФУНКЦИИ: x1=dx/dt(t) и x2=x(t) ');
disp('-----------------------------------------------------');
fprintf('%-5s %-10s %-10s %-10s \n','№','  t','dx/dt(t)','  x(t)');
disp('-----------------------------------------------------');
for i=1:n
fprintf('%-5d %-10.5f %-10.5f %-10.5f \n',i,t(i),x1(i),x2(i));
end
disp('-----------------------------------------------------');
plot(t,x1,'o',t,x2,'-');
title('РЕШЕНИЕ ДИФ.УРАВН.2-ГО ПОРЯДКА С К.У. x(0)=1;x(1)=3.718');
```

Рис. 3.76 (начало)

Программный код основного управляющего файла решения дифференциального уравнения второго порядка (3.94) с краевыми условиями (3.95)

```
xlabel('t');ylabel(' dx/dt;   x ');
legend('    dx/dt — производная решения',' x — функция решения');

end
```

Рис. 3.76 (окончание)

Программный код основного управляющего файла решения дифференциального уравнения второго порядка (3.94) с краевыми условиями (3.95)

```
function f=difur4(x1p)
global x2fix x1 x2 n t;
%-------------------------------------------------------------------------
%Создание функции "prav", соответствующей правым частям системы
%дифференциальных уравнений первого порядка "f1(t,x1,x2);f2(t,x1,x2);", записанного в явном виде
%-------------------------------------------------------------------------
function f = prav(t,x)
f=[x(1)^2/x(2);x(1)];
end
%-------------------------------------------------------------------------
%РЕШЕНИЕ СИСТЕМЫ 2-х ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИИ С ИСПОЛЬЗОВАНИЕМ РЕШАТЕЛЯ "ode45",
%реализующего метод Рунге-Кутта 4-5 порядка
%-------------------------------------------------------------------------
x0=[x1p,1];
interval=[0 1];
[t,x]=ode45(@prav,interval,x0);
n=length(x);
 for i=1:n
   x1(i)=x(i,1);
   x2(i)=x(i,2);
 end
 f=x2(n)-x2fix;
end
```

Рис. 3.77

Программный код файла интегрирования системы двух дифференциальных уравнений при решении преобразованной краевой задачи (3.97) с краевыми условиями (3.98) с внутренней функцией правой части системы (3.97) и определением функции (3.101)

Первый *m*-файл (рис. 3.76) является основной управляющей программой, в которой решается уравнение (3.201) с использованием решателя “fzero” следующим образом:

$$\begin{aligned} & x1p = 0 \\ & \text{option} = \text{optimset}\ (\text{‘Display’},\ \text{‘iter’}) \\ & [x1\_\text{proizv},\ \text{Func},\ \text{pr}] = \text{fzero}(@\text{difur4}, x1p,\ \text{option}). \end{aligned} \qquad (3.102)$$

Сначала задается начальное приближение $x1p$, равное 0, затем устанавливается опция вывода результатов промежуточных итераций в Командное окно MATLAB, а в последнем операторе с fzero реализуется процесс многократного обращения к решению системы дифференциальных уравнений — *m*-файл difur4.m при различных приближениях $x1p$. Результат решения уравнения (3.101) в левой части последнего оператора (3.102) — $x1\_proizv$.

Программный код файла определения функции уравнения (3.101) и решения системы дифференциальных уравнений (3.97) с различными приближениями $x1p$ представлен на рисунке 3.77 и включает в себя внутреннюю функцию расчета правых частей в виде:

$$\text{function } f = \text{prav}(t, x)$$

$$f = [x(1)^2 / x(2); x(1)]. \qquad (3.103)$$

$$\text{end}$$

Решение системы дифференциальных уравнений (3.97) осуществляется с использованием решателя “ode45”:

$$[t, x] = \text{ode45 } (@\text{prav, interval, } xp), \qquad (3.104)$$

а функция уравнения (3.101) вычисляется с помощью оператора

$$f = x2(n) - x2\text{fix}, \qquad (3.105)$$

где $x2(n)$ — рассчитанное значение краевого условия при $t = t^{(k)}$; $x2\text{fix}$ — заданное значение краевого условия при $t = t^{(k)}$.

Результаты решения краевой задачи для дифференциального уравнения второго порядка приведены в табличном (рис. 3.78) и графическом (рис. 3.79) виде.

```
-------------------------------------------------
   РЕШЕНИЕ ДИФ. УРАВНЕНИЯ 2-го ПОРЯДКА ВИДА:
-------------------------------------------------
   d^2x/dt^2=f(t,x,dx/dt), где f(t,x,dx/dt)=(1/x)(dx/dt)^2
-------------------------------------------------
   с краевыми условиями x(0)=1  x(1)=3.718 (краевая задача)
   в диапазоне изменения независимой переменной t : [0 - 1]
-------------------------------------------------
                  или
-------------------------------------------------
   после преобразования (x2=x и x1=dx/dt) к системе 2-х дифф. уравнений 1-го порядка
-------------------------------------------------
   dx1/dt=x1^2/x2
   dx2/dt=x1
-------------------------------------------------
   с дополнительными условиями x1(0)=x1p и x2(0)=1, (задача Коши)
   где x1p=0 — прибл. итерационно корректируемое знач. x1(0), пока в результате решения системы 2-х диф.ур.
   не выполнится условие: x2(1)расч.=x2fix — (x2fix=3.718); — метод "стрельбы" при решении краевой задачи
-------------------------------------------------
   В этом случае решение краевой задачи сводится к решению уравнения F(x1p)=0:
   где F(x1p)=x2(1)расч.{x1p)-x2(1)задан.=0, причем из условия задачи следует: x2(1)задан.=3.718
-------------------------------------------------

Search for an interval around 0 containing a sign change:
 Func-count   a        f(a)         b        f(b)       Procedure
      1       0      -1.718         0      -1.718   initial interval
      3   -0.0282843  -1.74589   0.0282843  -1.68931   search
      5     -0.04     -1.75721     0.04     -1.67719   search
      7   -0.0565685    -1.773   0.0565685    -1.6598   search
      9     -0.08     -1.79488     0.08     -1.63471   search
     11   -0.113137    -1.82497   0.113137    -1.59821   search
     13     -0.16     -1.86586     0.16     -1.54449   search
     15   -0.226274    -1.9205   0.226274    -1.46408   search
     17     -0.32     -1.99185     0.32     -1.34087   search
     19   -0.452548    -3.08199   0.452548    -1.14569   search
     21     -0.64     -3.19071     0.64     -0.821519   search
     23   -0.905097    -3.3135   0.905097    -0.245829   search
     25     -1.28     -3.43996     1.28      0.87864   search

Search for a zero in the interval [-1.28, 1.28]:
 Func-count   x        f(x)          Procedure
     25       1.28     0.87864      initial
     26       1.28     0.87864      interpolation
```

Рис. 3.78 (начало)

Результаты решения краевой задачи для дифференциального уравнения второго порядка (3.94) с краевыми условиями (3.95)

```
  27      0.943633      -0.1487       interpolation
  28       0.99232    -0.0205143       interpolation
  29      0.999934   0.000102302       interpolation
  30      0.999896    -3.8801e-07       interpolation
  31      0.999896   -7.30127e-12       interpolation
  32      0.999896   -4.44089e-16       interpolation
  33      0.999896    4.44089e-16       interpolation

Zero found in the interval [-1.28, 1.28]
--------------------------------------------------
    1. РЕЗУЛЬТАТ РЕШЕНИЯ — ДОПОЛНИТЕЛЬНОЕ УСЛОВИЕ ДЛЯ ПРОИЗВОДНОИ x1=dx/dt(0)
--------------------------------------------------
1.1. Рассчитанное значение производной при t=0( dx/dt(0) ) = 1.00
--------------------------------------------------
1.3. Финальное значение функции решения уравнения ( Func ) = 4.441e-16
--------------------------------------------------
1.3. Признак корректности вычислительного процесса ( pr ) = 1
--------------------------------------------------
    3. РЕЗУЛЬТАТ РЕШЕНИЯ — ФУНКЦИИ: x1=dx/dt(t) и x2=x(t)
--------------------------------------------------
№      t         dx/dt(t)      x(t)
--------------------------------------------------
1    0.00000   0.99990   1.00000
2    0.02500   1.02521   1.02531
3    0.05000   1.05116   1.05127
4    0.07500   1.07776   1.07788
5    0.10000   1.10504   1.10516
.....................................
36   0.87500   3.39841   3.39866
37   0.90000   3.45912   3.45937
38   0.92500   3.52136   3.52163
39   0.95000   3.58519   3.58545
40   0.97500   3.65062   3.65090
41   1.00000   3.71772   3.71800
--------------------------------------------------
>>
```

Рис. 3.78 (окончание)

Результаты решения краевой задачи для дифференциального уравнения второго порядка (3.94) с краевыми условиями (3.95)

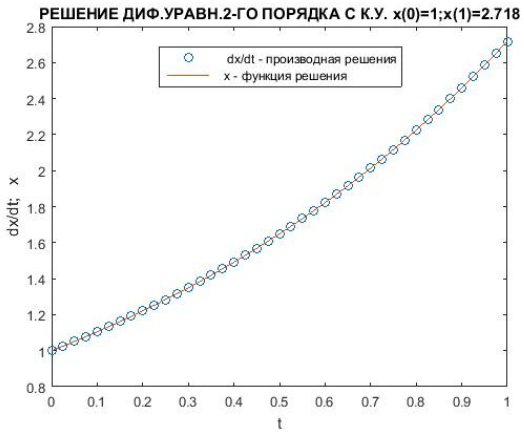


Рис. 3.79

Графическое представление результатов решения краевой задачи для дифференциального уравнения второго порядка (3.94) с краевыми условиями (3.95)

3.4.1.6. Моделирование процесса химического превращения в изотермическом периодическом реакторе идеального перемешивания со стехиометрической схемой реакции $A \xrightarrow{k_1} 2P \xrightarrow{k_2} S$ с применением решателя “ode45”

В соответствии с законами химической кинетики изменение концентраций компонентов реакционной смеси в периодическом процессе может быть описано следующим образом:

$$\begin{aligned} \frac{d(Nx_A)}{dt} &= -Nk_1x_A \\ \frac{d(Nx_P)}{dt} &= 2N\left(k_1x_A - k_2x_P^2\right), \\ \frac{d(Nx_S)}{dt} &= Nk_2x_P^2 \end{aligned} \tag{3.106}$$

где x_A, x_P, x_S — изменяющиеся во времени мольные доли компонентов реакционной смеси; k_1, k_2 — константы скоростей первой и второй реакций, неизменные в изотермическом процессе; N — изменяющийся во времени реакционный объем смеси в реакторе в молях смеси (мол. см.).

С учетом принятой системы единиц измерения моли компонентов реакции A, P, S могут быть представлены как:

$$N_A = Nx_A;\ N_P = Nx_P;\ N_S = Nx_S,$$

откуда следует, что мольные доли определяются как:

$$x_A = \frac{N_A}{N};\ x_P = \frac{N_P}{N};\ x_S = \frac{N_S}{N}. \tag{3.107}$$

В результате для реакционного объема в реакторе будет справедливо

$$N = N_A + N_P + N_S, \tag{3.108}$$

и с учетом временной зависимости этих величин будет справедливо

$$\frac{dN}{dt} = \frac{dN_A}{dt} + \frac{dN_P}{dt} + \frac{dN_S}{dt}. \tag{3.109}$$

Таким образом, принимая во внимание соотношения (3.106), (3.107) и (3.109), полная система математического описания периодического процесса в изотермическом реакторе записывается в виде системы четырех дифференциальных уравнений:

$$\begin{aligned} \frac{dN_A}{dt} &= -Nk_1\left(\frac{N_A}{N}\right) \\ \frac{dN_P}{dt} &= 2N\left[k_1\left(\frac{N_A}{N}\right) - k_2\left(\frac{N_P}{N}\right)^2\right] \end{aligned} \tag{3.110}$$

$$\frac{dN_S}{dt} = Nk_2\left(\frac{N_P}{N}\right)^2$$

$$\frac{dN}{dt} = \frac{dN_A}{dt} + \frac{dN_P}{dt} + \frac{dN_S}{dt}$$

с начальными условиями — задача Коши:

$$N(t=0) = N^{(0)};\ N_A(t=0) = N_A^{(0)};\ N_P(t=0) = N_P^{(0)};\ N_S(t=0) = N_S^{(0)}. \quad (3.111)$$

Исходя из (3.110) и (3.111), начальная загрузка периодического реактора задается в соответствии с начальными условиями (3.111) — $N^{(0)}$ и с учетом (3.107):

$$N_A^{(0)} = N^{(0)}x_A^{(0)}; N_P^{(0)} = N^{(0)}x_P^{(0)}; N_S^{(0)} = N^{(0)}x_S^{(0)}, \quad (3.112)$$

если известны мольные доли компонентов в реакционной смеси — $x_A^{(0)}$, $x_P^{(0)}$, $x_S^{(0)}$.

Первый операционный этап периодического процесса в рассматриваемой задаче задается в интервале времени [0÷10] ч и является единственным. Может быть и второй этап, например если установить другую температуру, в этом случае должны измениться константы скоростей реакций k_1 и k_2 и, соответственно, технологические параметры процесса.

Программа расчета процесса в периодическом реакторе со стехиометрической схемой реакции $A \rightarrow 2P \rightarrow S$ включает в себя три *m*-файла: GLAV_model3_batch_reactor.m (рис. 3.81), DATA.m (рис. 3.80), difpravreactor.m (рис. 3.82).

```
function DATA
%-------------------------------------------------------------------------------------------
%Программный код файла DATA.m — задание исходной информации для расчетов;
%-------------------------------------------------------------------------------------------
%Программа включает следующие
%файлы: GLAV_model3_batch_reactor.m+DATA.m+difpravreactor.m+REPORT.m
%-------------------------------------------------------------------------------------------
%Программа моделирования процесса химического прекращения в
%изотермическом периодическом реакторе со стехиометрической схемой
%реакции A - 2P - S
%-------------------------------------------------------------------------------------------
global N0 Na0 Np0 Ns0 xa0 xp0 xs0 k1 k2 t_a t_b;
%1. Начальная загрузка реактора N (мол.см.):
N0=25;
%3. Начальные концентрации компонентов A,P,S (мол."i"/мол.см.):
xa0=1;xp0=0;xs0=0;
%Начальные количества компонентов A,P,S (мол."i"):
Na0=N0*xa0;Np0=N0*xp0;Ns0=N0*xs0;
%3. Константы скоростей реакций (час^(-1))
k1=0.35;k2=0.13;
%4. Левая граница изменения времени в реакторе (час)
t_a=0;

%5. Правая граница изменения времени в реакторе (час)
t_b=10;

end
```

Рис. 3.80

Программный код файла исходных данных для моделирования процесса в изотермическом периодическом реакторе идеального перемешивания (3.110) и (3.111)

```
%-------------------------------------------------------------------------------------------------
%Программный код файла GLAV_model3_batch_reactor.m — основная управляющая програм-
ма
%-------------------------------------------------------------------------------------------------
%Программа включает следующие
%файлы: GLAV_model3_batch_reactor.m+DATA.m+difpravreactor.m+REPORT.m
%-------------------------------------------------------------------------------------------------
%Программа моделирования процесса химического прекращения в
%изотермическом периодическом реакторе со стехиометрической схемой
%реакции A - 2P - S
%-------------------------------------------------------------------------------------------------
clc;
clear all;
close all;
global xa xp xs Nl Na0 Np0 Ns0 N N0 t_a t_b t n;
DATA;
[t,Nl]=ode45(@difpravreactor,[t_a t_b],[Na0 Np0 Ns0 N0]);

n=length(t);
N=Nl(:,4);
for i=1:n
   xa(i)=Nl(i,1)/N(i);
   xp(i)=Nl(i,2)/N(i);
   xs(i)=Nl(i,3)/N(i);
end
REPORT;
```

Рис. 3.81

Программный код файла основной управляющей программы моделирования процесса в изотермическом периодическом реакторе идеального перемешивания (3.110) и (3.111)

```
function dNl = difpravreactor(t,Nl)
%-------------------------------------------------------------------------------------------------
%Программный код файла difpravreactor.m — расчет правых частей системы
%дифференциальных уравнений математического описания процесса
%-------------------------------------------------------------------------------------------------
%Программа включает следующие
%файлы: GLAV_model3_batch_reactor.m+DATA.m+difpravreactor.m+REPORT.m
%-------------------------------------------------------------------------------------------------
%Программа моделирования процесса химического прекращения в
%изотермическом периодическом реакторе со стехиометрической схемой
%реакции A - 2P - S
%-------------------------------------------------------------------------------------------------
global k1 k2
dNl=zeros(4,1);
f(1)=-Nl(4)*k1*(Nl(1)/Nl(4));
f(2)=2*Nl(4)*(k1*(Nl(1)/Nl(4))-k2*(Nl(2)/Nl(4))^2);
f(3)=Nl(4)*k2*(Nl(2)/Nl(4))^2;
dNl(1)=f(1);
dNl(2)=f(2);
dNl(3)=f(3);
dNl(4)=f(1)+f(2)+f(3);
end
```

Рис. 3.82

Программный код файла правых частей четырех дифференциальных уравнений (3.110), описывающих процесс в изотермическом периодическом реакторе идеального перемешивания

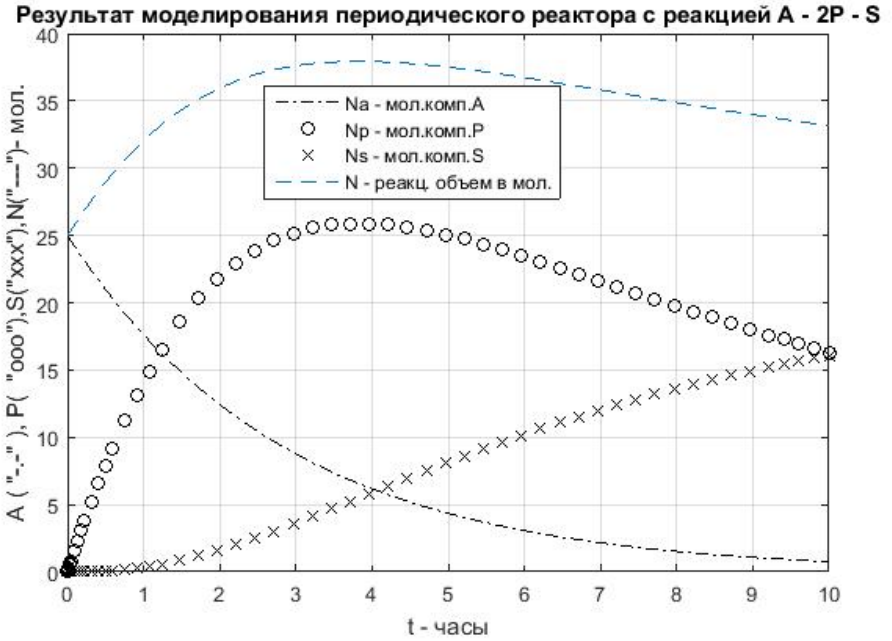


Рис. 3.83

Графическое представление результатов решения системы дифференциальных уравнений (3.110) и (3.111), описывающих процесс в изотермическом периодическом реакторе идеального перемешивания

Исходные данные для решения системы уравнений (3.110) и (3.111) с учетом (3.112) приведены в *m*-файле DATA.m (рис. 3.80).

Программный код файла GLAV_model3_batch_reactor.m (рис. 3.81) является основной управляющей программой, в которой с использованием решателя “ode45” решается система четырех дифференциальных уравнений (3.110) с начальными условиями (3.111) в интервале времени [*t_a* = 0; *t_b* = 10].

Обращение к решателю “ode45” осуществляется следующим образом (рис. 3.81):

[*t,NI*]=ode45(@difpravreactor,[*t_a, t_b*],[*Na*0, *Np*0, *N*s0, *N*0]). (3.113)

Результат изменения независимой переменной отображается в массиве *t*, а функций решений $N_A(t)$, $N_P(t)$, $N_S(t)$, $N(t)$ — в двумерном массиве *NI* (рис. 3.81).

Как следует из (3.113), правые части четырех дифференциальных уравнений (3.110) представлены в *m*-файле difpravreactor.m (рис. 3.82).

Результаты моделирования процесса в реакторе в табличном виде приведены в Приложении (табл. П.6) и в графическом виде — на рисунке 3.83. Эти же результаты представлены на рисунке 3.84, где показано изменение мольных долей компонентов во времени.

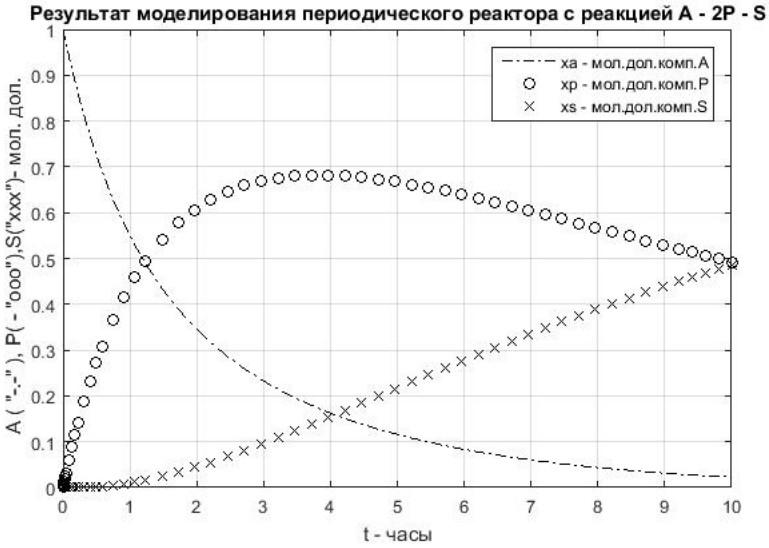


Рис. 3.84

Графическое представление изменения мольных долей компонентов реагирующей смеси в изотермическом периодическом реакторе идеального перемешивания со стехиометрической схемой реакции $A \rightarrow 2P \rightarrow S$ (3.110) и (3.111)

3.4.2. Решение дифференциальных уравнений с частными производными методом конечных разностей

Дифференциальные уравнения с частными производными включают в себя производные, которые зависят от нескольких независимых переменных.

Для математического описания химико-технологических процессов широко используются дифференциальные уравнения в частных производных второго порядка с двумя независимыми переменными ℓ и t следующего вида:

$$A(\ell,t)\frac{\partial^2 T}{\partial \ell^2} + 2B(\ell,t)\frac{\partial^2 T}{\partial \ell \partial t} + C(\ell,t)\frac{\partial^2 T}{\partial t^2} + D(\ell,t)\frac{\partial T}{\partial \ell} + \\ + E(\ell,t)\frac{\partial T}{\partial t} + F(\ell,t)T + G(\ell,t) = 0 \qquad (3.114)$$

где A, B, C, D, E, F, G — функции в общем случае независимых переменных ℓ и t.

Функция $T = T(\ell,\ t)$ является искомым решением приведенного дифференциального уравнения в частных производных.

Если $B^2 - AC < 0$, то уравнение относится к классу эллиптических, если $B^2 - AC > 0$, то к классу гиперболических, а если $B^2 - AC = 0$ — параболических. Когда $B^2 - AC$ не имеет постоянного знака, то это уравнение смешанного типа.

Для решения рассматриваемой двумерной задачи с двумя независимыми переменными можно использовать специальный пакет MATLAB PDE ToolBox, для чего достаточно набрать в Командном окне команду “pdetool”. Необходи-

мым условием применения этого пакета является знание численных методов решения дифференциальных уравнений с частными производными.

Более обоснованным с точки зрения физического смысла рассматриваемой задачи является решение собственными средствами MATLAB, в частности с применением метода конечных разностей. Несмотря на то что такой способ требует дополнительных усилий при разработке программ, он позволяет решать дифференциальные уравнения с большим чем два числом независимых переменных и более основательно вникнуть в сущность решаемой задачи.

Последний метод целесообразно проиллюстрировать на примере параболического уравнения с постоянными коэффициентами (A, B, C, D, E, F, G) следующего вида:

$$\begin{aligned} &\frac{\partial T}{\partial t} = a\frac{\partial^2 T}{\partial \ell^2} + \sin\left(\ell \cdot t\right), \\ &t^{(0)} \le t \le t^{(k)}, \end{aligned} \tag{3.115}$$

где $A = a$; $E = -1$; $G(\ell, t) = \sin(\ell \cdot t)$, а коэффициенты B, C, D, F равны нулю.

При этом задаются дополнительные условия:

— граничные условия при $\ell = 0$ и $\ell = 1$ — в данном случае константы:

$$T(0, t) = const_1 = T_{(0)};\ T(L, t) = const_2 = T_1; \tag{3.116}$$

— начальное условие при $t = t^{(0)}$ — линейная функция от независимой переменной ℓ:

$$\begin{aligned} &T\left(\ell, t^{(0)}\right) = T_{(0)} + \left(\frac{T_{(L)} - T_{(0)}}{L}\right) \cdot \ell, \\ &0 \le l \le L. \end{aligned} \tag{3.117}$$

Эти дополнительные условия называются условиями Дирихле, так как они заданы в виде непрерывных функций от независимых переменных ℓ и t. В данном случае функции граничных условий от независимой переменной t — $T_{(0)}$ и $T_{(L)}$ — считаются константами.

Следует учитывать, что могут задаваться и другие граничные и начальные условия, отличные от условий Дирихле, однако здесь они не рассматриваются.

В соответствии с методом конечных разностей частные производные в исходном уравнении заменяются их конечно-разностным представлением:

— в явной разностной схеме решения:

$$\begin{aligned} &\frac{\partial T\left(\ell, t\right)}{\partial t} = \frac{T_{i,\,j+1} - T_{i,\,j}}{\Delta t} \\ &\frac{\partial^2 T\left(\ell, t\right)}{\partial \ell^2} = \frac{T_{i,\,j+1} - 2T_{i,\,j} + T_{i-1,\,j}}{\left(\Delta \ell\right)^2}, \end{aligned} \tag{3.118}$$

где $i = 1, \ldots, N$; $j = 1, \ldots, K$;

— в неявной разностной схеме решения:

$$\frac{\partial T(\ell,t)}{\partial t} = \frac{T_{i,j} - T_{i,j-1}}{\Delta t}$$
$$\frac{\partial^2 T(\ell,t)}{\partial \ell^2} = \frac{T_{i+1,j} - 2T_{i,j} + T_{i-1,j}}{(\Delta \ell)^2}, \quad (3.119)$$

где $i = 1, \ldots, N$; $j = 1, \ldots, K$.

В результате при реализации явной разностной схемы решения (3.115) получается система конечно-разностных уравнений вида:

$$\frac{T_{i,j+1} - T_{i,j}}{\Delta t} = a\frac{T_{i,j+1} - 2T_{i,j} + T_{i-1,j}}{(\Delta \ell)^2} + \sin(\ell_i, t_j), \quad (3.120)$$

где $i = 1, \ldots, N$; $j = 1, \ldots, K$, а при реализации неявной разностной схемы:

$$\frac{T_{i,j} - T_{i,j-1}}{\Delta t} = a\frac{T_{i+1,j} - 2T_{i,j} + T_{i-1,j}}{(\Delta \ell)^2} + \sin(\ell_i, t_j), \quad (3.121)$$

где $i = 1, \ldots, N$; $j = 1, \ldots, K$.

Так как при представленной дискретизации координат по длине ℓ и по времени t индексы в уравнениях (3.120) и (3.121) изменяются от нуля, то при программной реализации изменение индексов целесообразно задавать от 1 соответственно до $(N + 1)$ и $(K + 1)$, и число участков разбиения по длине будет равно N, а по времени — K. В результате дискретные выражения для определения ℓ_i и t_j в (3.120) и (3.121) имеют вид:

$$\ell_i = 0 + \Delta\ell \cdot (i - 1),\ i = 1,\ \ldots,\ (N + 1),$$
$$t_j = t^{(0)} + \Delta t(j - 1),\ j = 1,\ \ldots,\ (K + 1). \quad (3.122)$$

Такая же индексация при программной реализации должна использоваться и при решении системы уравнений (3.120) и (3.121), причем в этом случае индекс i во избежание нулевого значения должен изменяться от 2 до $(N + 1)$.

В результате граничные условия (3.116) принимают вид:

$$T_{1,j} = T_0 = const_1$$
$$T_{N+1,j} = T_{(L)} = const_2, \quad (3.123)$$
$$j = 1,\ \ldots,\ (K + 1)$$

а линейная функция начальных условий (3.117) представляется следующим образом:

$$T_{i,j} = T_{(0)} + \frac{T_{(L)} - T_{(0)}}{N}(i - 1), \quad (3.124)$$
$$i = 1,\ \ldots, N + 1.$$

3.4.2.1. Явная разностная схема решения параболического уравнения

Если в приведенной явной разностной схеме (3.120) обозначить комплекс величин:

$$\gamma = \frac{a\Delta t}{(\Delta \ell)^2}, \tag{3.125}$$

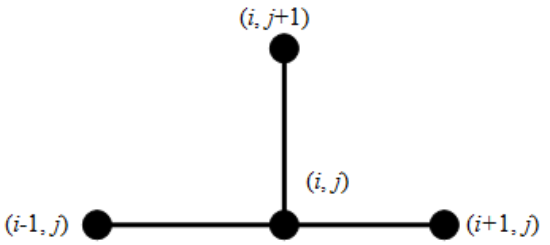
где a — коэффициент уравнения; Δt — шаг дискретизации по интервалу изменения времени $t^{(0)} \le t \le t^{(k)}$; $\Delta \ell$ — шаг дискретизации по интервалу изменения координат длины $0 \le \ell \le L$, то переменная решения $T_{i,j+1}$ может быть выражена из уравнения (3.120) следующим образом:

$$\begin{aligned} &T_{i,j+1} = gT_{i-1,j} + (1-2g)T_{i,j} + gT_{i,j+1} + \Delta t \cdot \sin(l_i, t_j), \\ &j = 1, \ \ldots, K, \\ &i = 2, \ \ldots, N, \end{aligned} \tag{3.126}$$

а дискретные значения ℓ_i, t_j определяются из соотношений (3.122).

Определение функции $T = T(\ell, t)$, или в табличном виде $T_{i,j+1}(i = 2, \ldots, N; j = 1, \ldots, K)$, основано на том, что при произвольном значении j (индекс независимой переменной t) в правой части представленного соотношения все величины известны либо из предыдущих расчетов и граничных условий, либо в начале реализации алгоритма начальных и граничных условий задачи.

Шаблон для вычислений по явной разностной схеме имеет вид:



где $i = 2, \ldots, N, j = 1, \ldots, K$.

С учетом того, что в первом слое (при $j = 1$) все значения $T_{i,j}$ известны из начальных и граничных условий (3.124), по формуле (3.126) можно рассчитать $T_{i,2}(i = 2, \ldots, N)$.

После этого из полученных значений $T_{i,j}(j = 2; i = 2, \ldots, N)$ аналогично рассчитываются все значения $T_{i,j}(j = 3; i = 2, \ldots, N)$ и так далее до $T_{i,K}(i = 2, \ldots, N)$.

В результате получается искомая функция $T = T(\ell, t)$, представленная в дискретном виде: $T_{i,j}(i = 2, \ldots, N; j = 2, \ldots, K)$.

Принято считать, что явная разностная схема сходится и устойчива, т. е. получаемое с ее помощью решение наиболее приближено к истинному, когда выполняется следующее условие для преобразованного с использованием конечно-разностного представления производных дифференциального уравнения:

$$\gamma \le \frac{1}{2}$$

или (3.127)

$$\Delta t \le \frac{(\Delta \ell)^2}{2a}.$$

Программа явного разностного метода решения параболического уравнения вида

$$\frac{\partial T}{\partial t} = a\frac{\partial^2 T}{d\ell^2} + \sin(\ell, t) \quad (3.128)$$

с граничными условиями:

$$T(0, t) = \text{const} = 1 \text{ и } T(L, t) = \text{const} = 2 \quad (3.128а)$$

и линейной функцией начального условия

$$T(\ell, 0) = 1 + \frac{2-1}{N}\ell \quad (3.128б)$$

представлена в виде двух *m*-файлов — Glav_uravn_parabol_yavn.m (рис. 3.85) и parabol_yavn.m (рис. 3.86).

В уравнении (3.128) a — постоянный коэффициент (a = 0.16); $T = T(\ell, t)$ — искомая функция решения; ℓ — независимая переменная по пространственной координате, которая изменяется в диапазоне $0 \le \ell \le 5$; t — независимая переменная по временной координате, которая изменяется в диапазоне $0 \le t \le 3$; N — количество участков дискретного разбиения диапазона изменения пространственной координаты (N = 50); K — количество участков дискретного разбиения диапазона изменения временной координаты (K = 200).

```
clc;
clear all;
close all;
fprintf('%s \n','---------------------------------------------------- ');
fprintf('%s \n',' Glav_uravn_parabol_yavn.m — основная управляющая программа ');
fprintf('%s \n','---------------------------------------------------- ');
fprintf('%s \n',' Программа включает следующие файлы: Glav_uravn_parabol_yavn.m+parabol_yavn.m ');
fprintf('%s \n','---------------------------------------------------- ');
fprintf('%s \n',' Программа решения параболического уравнения с частными производными с помо-
щью явной разностной схемы ');
fprintf('%s \n','---------------------------------------------------- ');

global T T1 T_Niplus1;
%       1. ФОРМУЛИРОВКА НАЧАЛЬНО-ГРАНИЧНОИ ЗАДАЧИ:
%-------------------------------------------------------
%       Решить параболическое уравнение с частными производными вида:
%       DT/Dt=a*D^2T/DI^2+sin(I*t), где D — обозначение частной
%       производной;I,t — независимые переменные; T(I,t) — искомое
%       решение как зависимая переменная от I и t; a — константа и параметр
%       уравнения и определить функцию T(I,t)в интервале независимых
%       переменных 0<=I<=L и 0<=t<=tk
%-------------------------------------------------------------------
```

Рис. 3.85 (начало)

Программный код файла основной управляющей программы решения параболического уравнения явным разностным методом

```
%         3. ЗАДАНИЕ ИСХОДНЫХ ДАННЫХ ДЛЯ РАСЧЕТОВ:
%-------------------------------------------------------
%1. [0,L] — интервал по независимой переменной l, на котором определяется решение:
L=5;
%3. Nl — количество участков, на которые разбивается интервал по независимой переменной l
%в соответствии с разностной схемой решения:
Nl=50;
%3. [0,tk] — - интервал по независимой переменной t, на котором определяется решение:
tk=3;
%4. Kt — количество участков, на которые разбивается интервал по независимой переменной t
%в соответствии с разностной схемой решения:
Kt=200;
%5. a — константа и параметр уравнения с частными производными:
a=0.16;
%-------------------------------------------------------
%6. Граничные условия задачи ( константы ):
%T(0,t)=const=T1;T(L,t)=const=T_Nplus1;
T1=1;
T_Nlplus1=2;
%-------------------------------------------------------
%_____________________________________________________________________________
%Начальные условия задачи при t=0(линейная функция от T(l) в виде функции:
% T(l,t=0)=T1+((T_Nplus1-T1)/Nl)*i
% где i — номер шага по независимой переменной l в интервале [0,Nl]
%задаются в специально созданной функции "parabol_yavn" для решения параболического
%уравнения методом конечных разностей с использованием явной разностной схемы
%_____________________________________________________________________________
%-------------------------------------------------------
%                3. РЕШЕНИЕ ЗАДАЧИ
%-------------------------------------------------------
%_____________________________________________________________________________
%Решение поставленной задачи и получение искомой функции T=T(l,t) в дискретном виде
%выполняется с применением специально созданной функции "parabol_yavn"
%методом конечных разностей с использованием явной разностной схемы
%_____________________________________________________________________________
%-------------------------------------------------------
[T,l,t]=parabol_yavn(Nl,Kt,L,tk,a);
%-------------------------------------------------------
%             4. ФОРМИРОВАНИЕ ОТЧЕТА О РЕЗУЛЬТАТАХ РЕШЕНИЯ ЗАДАЧИ
%-------------------------------------------------------
fprintf('%s \n','_____________________________________________________________________________');
fprintf('%s \n','Решение параболического уравнения с частными производными вида:');
fprintf('%s \n','DT/Dt=a*D^2T/Dl^2+sin(l*t), где D — обозначение частной');
fprintf('%s \n','производной;l,t — независимые переменные; T(l,t) — искомое');
fprintf('%s \n','решение как зависимая переменная от l и t; a — константа и параметр уравнения');
fprintf('%s \n','Определить функцию T(l,t)в интервале независимых переменных 0<=l<=L и 0<=t<=tk');
fprintf('%s \n','_____________________________________________________________________________');
fprintf('%s \n','------------------------------------------------- ');
fprintf('%s \n','ИСХОДНЫЕ ДАННЫЕ:');
fprintf('%s \n','------------------------------------------------- ');
fprintf('%s \n','ПАРАБОЛИЧЕСКОЕ УРАВНЕНИЕ ВИДА:');
fprintf('%s \n','------------------------------------------------- ');
fprintf('%s \n',' DT/Dt=a*D^2T/Dl^2+sin(l*t) ');
fprintf('%s \n',' где:a=0.16; 0<=l<=Nl; 0<=t<=tk и Nl=5;tk=3');
fprintf('%s \n','------------------------------------------------- ');
fprintf('%s \n',' Граничные условия: ');
fprintf('%s \n',' T(0,t)=const=T1=1;T(L,t)=const=T_Nplus1=2; ');
fprintf('%s \n','------------------------------------------------- ');
fprintf('%s \n',' Начальные условия: ');
fprintf('%s \n',' T(l,t=0)=T1+((T_Nplus1-T1)/Nl)*i ');
fprintf('%s \n','  где i — номер постоянного шага по независимой переменной l, ');
fprintf('%s \n','  которая изменяется в интервале [0,Nl] ');
fprintf('%s \n','------------------------------------------------- ');
fprintf('%s \n',' Параметры разностной схемы решения параболического уравнения: ');
fprintf('%s \n','  Nl=50 — количество одинаковых участков, на которые разбивается');
fprintf('%s \n','  интервал по независимой переменной l : 0<=l<=Nl');
fprintf('%s \n','  Kt=200 — количество одинаковых участков, на которые разбивается интервал по tk');
fprintf('%s \n','  интервал по независимой переменной t : 0<=t<=tk');
```

Рис. 3.85 (продолжение)

Программный код файла основной управляющей программы решения параболического уравнения явным разностным методом

```
fprintf('%s \n','---------------------------------------------------- ');
fprintf('%s \n','РЕЗУЛЬТАТЫ РАСЧЕТОВ:');
fprintf('%s \n','---------------------------------------------------- ');
fprintf('%s \n',' ПРЕДСТАВЛЕНИЕ ИСКОМОИ ФУНКЦИИ T(l,t) В ТАБЛИЧНОМ ВИДЕ: ');
fprintf('%s \n','---------------------------------------------------- ');
fprintf('%-5s %-10s %-10s %-10s %-10s %-10s %-10s \n','№','l','T(l,0)','T(l,0.75)','T(l,1.5)','T(l,3.25)','T(l,3)');
fprintf('%s \n','----------------------------------------------------');
for i=1:Nl+1
fprintf('%-5d %-10.5f %-10.5f %-10.5f %-10.5f %-10.5f %-10.5f %-10.5f \n',i,l(i),T(i,1),T(i,51),T(i,101),T(i,151),T(i,201));
end
fprintf('%s \n','----------------------------------------------------');
fprintf('%s \n',' ПРЕДСТАВЛЕНИЕ ИСКОМОИ ФУНКЦИИ T(l,t) В ГРАФИЧЕСКОМ ВИДЕ: ');
fprintf('%s \n','----------------------------------------------------');
%Построение графика решения
surf(l,t,T');
title('Результат решения уравнения DT/Dt=a*D^2T/Dl^2+sin(l*t)');
xlabel('l');
ylabel('t');
zlabel('T(l,t)');
fprintf('%s \n',' **************ВЫПОЛНЕНИЕ ПРОГРАММЫ УСПЕШНО ЗАВЕРШЕНО************** ');
fprintf('%s \n','----------------------------------------------------');
```

Рис. 3.85 (окончание)

Программный код файла основной управляющей программы решения параболического уравнения явным разностным методом

```
function [ l ,l,t] =parabol_yavn(Nl,Kt,L,tk,a)
%-------------------------------------------------------------------
%parabol_yavn.m — код программы решения параболического уравнения с частными
%производными методом конечных разностей с помощью явной разностной схемы
%-------------------------------------------------------------------
% Программа включает следующие файлы: Glav_uravn_parabol_neyavn.m+parabol_neyavn.m
%-------------------------------------------------------------------
% Программа решения параболического уравнения с частными производными
%с помощью явной разностной схемы
%-------------------------------------------------------------------
global T1 T_Nlplus1
%-------------------------------------------------------------------
%T(l,t)- матрица решений как функция от двух переменных длины l и времени t
%T1 — постоянное граничное условие при l=0
%T_Nlplus1 — постоянное граничное условие при l=L
%[0,L] — интервал по независимой переменной l, на котором определяется решение
%[0,tk] — интервал по независимой переменной t, на котором определяется решение
%Nl — количество участков, на которые разбивается интервал по длине L
%Kt — количество участков, на которые разбивается интервал по времени tk
%a — параметр дифференциального уравнения с частными производными
%-------------------------------------------------------------------
%вычисление шага по L
hl=L/Nl;
%вычисление шага по tk
deltat=tk/Kt;
%Из начального условия формируется массив l и первый столбец матрицы
%решений T(l,t)
for i=1:Nl+1
l(i)=(i-1)*hl;
T(i,1)=T1+(T_Nlplus1-T1)/Nl*(i-1);
%T(i,1)=funcL(x(i));
end
%Из граничных условий формируется массив t и первая и последняя строка матрицы
%решений T(l,t)
for j=1:Kt+1
t(j)=(j-1)*deltat;
T(1,j)=T1;
%T(1,j)=funcT0(t(j));
T(Nl+1,j)=T_Nlplus1;
%T(Nl+1,j)=funcTN(t(j));
end
```

Рис. 3.86 (начало)

Программный код файла реализации явного разностного метода параболического уравнения

```
gam=a*deltat/hl^2;
%Формируется матрица решений T(l,t) с использованием явной разностной схемы
for j=1:Kt
   for i=2:Nl
   T(i,j+1)=gam*T(i-1,j)+(1-2*gam)*T(i,j)+gam*T(i+1,j)+deltat*sin(l(i)*t(j));
   end
end
end
```

Рис. 3.86 (окончание)

Программный код файла реализации явного разностного метода параболического уравнения

В основной управляющей программе (рис. 3.85):

- задаются исходные данные для расчетов;
- программа решения параболического уравнения (3.128) вызывается следующим образом:

$$[T, \ell, t] = parabol\_yavn(N\ell, Kt, L, t, a);$$

- выводится результат решения — функция $T(\ell, t)[0 \le \ell \le 5, 0 \le t \le 3]$ в табличном и графическом виде.

В *m*-файле parabol_yavn.m (рис. 3.86) реализован явный разностный метод решения параболического уравнения (3.128).

Результат решения задачи (функция $T(\ell, t)$) в табличном виде представлен на рисунке 3.87, в графическом виде — на рисунке 3.88.

```
---------------------------------------------
Glav_uravn_parabol_yavn.m — основная управляющая программа
---------------------------------------------
Программа включает следующие файлы: Glav_uravn_parabol_yavn.m+parabol_yavn.m
---------------------------------------------
Программа решения параболического уравнения с частными производными с помощью явной разностной схе-
мы
---------------------------------------------
_____________________________________________________________________________
Решение параболического уравнения с частными производными вида:
DT/Dt=a*D^2T/Dl^2+sin(l*t), где D — обозначение частной
производной;l,t — независимые переменные; T(l,t) — искомое
решение как зависимая переменная от l и t; a — константа и параметр уравнения
Определить функцию T(l,t)в интервале независимых переменных 0<=l<=L и 0<=t<=tk
_____________________________________________________________________________
---------------------------------------------
ИСХОДНЫЕ ДАННЫЕ:
---------------------------------------------
ПАРАБОЛИЧЕСКОЕ УРАВНЕНИЕ ВИДА:
---------------------------------------------
DT/Dt=a*D^2T/Dl^2+sin(l*t)
где:a=0.16; 0<=l<=Nl; 0<=t<=tk и Nl=5;tk=3
---------------------------------------------
Граничные условия:
T(0,t)=const=T1=1;T(L,t)=const=T_Nplus1=2;
---------------------------------------------
Начальные условия:
T(l,t=0)=T1+((T_Nplus1-T1)/Nl)*i
 где i — номер постоянного шага по независимой переменной l,
 которая изменяется в интервале [0,Nl]
---------------------------------------------
Параметры разностной схемы решения параболического уравнения:
 Nl=50 — количество одинаковых участков, на которые разбивается
 интервал по независимой переменной l : 0<=l<=Nl
```

Рис. 3.87 (начало)

Результат решения параболического уравнения с частными производными вида (3.128) явным разностным методом

```
Kt=200 — количество одинаковых участков, на которые разбивается интервал по tk
интервал по независимой переменной t : 0<=t<=tk
-------------------------------------------------
РЕЗУЛЬТАТЫ РАСЧЕТОВ:
-------------------------------------------------
ПРЕДСТАВЛЕНИЕ ИСКОМОИ ФУНКЦИИ T(l,t) В ТАБЛИЧНОМ ВИДЕ:
-------------------------------------------------
№    l          T(l,0)    T(l,0.75) T(l,1.5)   T(l,3.25)     T(l,3)
-------------------------------------------------
1    0.00000    1.00000   1.00000   1.00000    1.00000    1.00000
2    0.10000    1.02000   1.04738   1.12558    1.23031    1.31616
3    0.20000    1.04000   1.09468   1.24998    1.45530    1.61885
4    0.30000    1.06000   1.14184   1.37204    1.66983    1.89545
5    0.40000    1.08000   1.18876   1.49064    1.86914    3.13496
6    0.50000    1.10000   1.23539   1.60470    3.04895    3.32866
7    0.60000    1.12000   1.28165   1.71323    3.20566    3.47065
8    0.70000    1.14000   1.32746   1.81529    3.33644    3.55814
9    0.80000    1.16000   1.37276   1.91006    3.43932    3.59157
10   0.90000    1.18000   1.41748   1.99681    3.51325    3.57448
.....................................................................
40   3.90000    1.78000   3.27999   1.79437    3.21733    1.81334
41   4.00000    1.80000   3.28938   1.79313    3.23634    1.79278
42   4.10000    1.82000   3.29626   1.79620    3.24216    1.79144
43   4.20000    1.84000   3.29957   1.80397    3.23429    1.80935
44   4.30000    1.86000   3.29769   1.81719    3.21275    1.84468
45   4.40000    1.88000   3.28843   1.83683    3.17833    1.89360
46   4.50000    1.90000   3.26909   1.86378    3.13307    1.95009
47   4.60000    1.92000   3.23689   1.89810    3.08098    3.00579
48   4.70000    1.94000   3.18998   1.93780    3.02898    3.05017
49   4.80000    1.96000   3.12919   1.97707    1.98778    3.07116
50   4.90000    1.98000   3.06077   3.00424    1.97218    3.05704
51   5.00000    3.00000   3.00000   3.00000    3.00000    3.00000
-------------------------------------------------
ПРЕДСТАВЛЕНИЕ ИСКОМОИ ФУНКЦИИ T(l,t) В ГРАФИЧЕСКОМ ВИДЕ:
-------------------------------------------------
**********ВЫПОЛНЕНИЕ ПРОГРАММЫ УСПЕШНО ЗАВЕРШЕНО**********
-------------------------------------------------
>>
```

Рис. 3.87 (окончание)

Результат решения параболического уравнения с частными производными вида (3.128) явным разностным методом

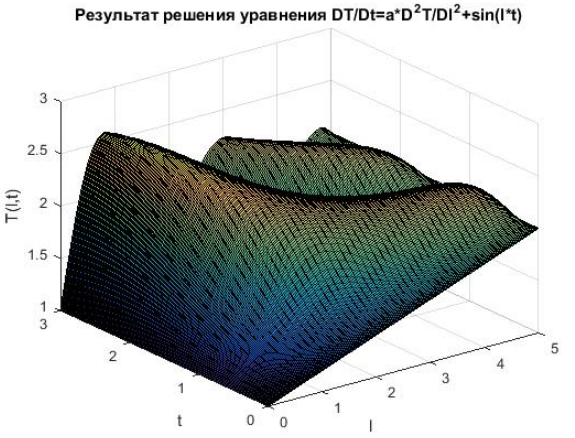


Рис. 3.88

Графическое представление результатов решения параболического уравнения с частными производными вида (3.128) явным разностным методом

3.4.2.2. Неявная разностная схема решения параболического уравнения

В этом случае дифференциальное уравнение (3.121) в конечно-разностной форме для неявной разностной схемы может быть преобразовано $\left(\gamma = \dfrac{a \cdot \Delta t}{(\Delta \ell)^2}\right)$:

$$T_{ij} - T_{i,j-1} = \gamma(T_{i+1,j} - 2T_{ij} + T_{i-1,j}) + \Delta t sin(\ell_i t_j)$$

или

$$-\gamma T_{i-1,j} + (1 + 2\gamma)T_{ij} - \gamma T_{i+1,j} = T_{i,j-1} + \Delta t sin(\ell_i t_j)$$

$$i = 2, \ldots, N; j = 2, \ldots, K + 1.$$

Для получения искомой функции $T = T(\ell, t)$ в дискретном виде последних уравнений решается система для каждого отдельного значения j (уровня) от 2 до $(K + 1)$ как линейная алгебраическая трехдиагональная система уравнений вида:

$$\begin{gathered}
-\gamma T_{1,j} + (1 + 2\gamma)T_{2,j} - \gamma T_{3,j} = T_{2,j-1} + \Delta t sin(\ell_2 t_j), \\
-\gamma T_{2,j} + (1 + 2\gamma)T_{3,j} - \gamma T_{4,j} = T_{3,j-1} + \Delta t sin(\ell_3 t_j), \\
\ldots\ldots\ldots\ldots\ldots\ldots \\
-\gamma T_{N-1,j} + (1 + 2\gamma)T_{N,j} - \gamma T_{N+1,j} = T_{N,j-1} + \Delta t sin(\ell_N t_j), \\
j = 1, \ldots, K.
\end{gathered} \tag{3.129}$$

Так как вектор правых частей системы при $j = 1 (T_{1,1}, T_{2,1}, \ldots, T_{N+1,1})$ известен из начальных условий задачи, а $T_{1,j}$ и $T_{N+1,j}$ $(j = 1, \ldots, K)$ — известные граничные условия, то искомые значения функции $T = T(\ell, t)$ определяются из преобразованного уравнения с помощью формулы

$$\begin{gathered}
T_{ij} = \frac{\gamma}{1+2\gamma}\left(T_{i-1,j} + T_{i+1,j}\right) + \frac{T_{i,j-1}}{1+2\gamma} + \frac{\Delta t}{1+2\gamma} sin\left(\ell_i t_j\right), \\
i = 2, \ldots, N; j = 2, \ldots, K+1.
\end{gathered} \tag{3.130}$$

Для решения трехдиагональной системы уравнений используются итерационные алгоритмы, в данном случае модификация метода простых итераций — метод Зейделя, в соответствии с которым на основе начальных условий в первом слое $(j = 1)$ и граничного условия $T_{1,1}$ определяются следующие искомые значения функции $T = T(\ell, t)$:

$$T_{2,1}, T_{3,1}, \ldots, T_{N,1}.$$

Далее таким же образом исходя из полученных значений функции в первом слое определяются значения функции во втором слое $(j = 2)$:

$$T_{2,2},\ T_{3,2},\ \ldots,\ T_{N,2}$$

и т. д. до последнего K-слоя ($j = K + 1$):

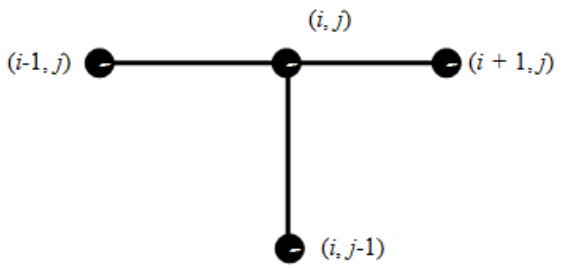
$$T_{2,K+1},\ T_{3,K+1},\ \ldots,\ T_{N,K+1}.$$

При этом для каждого слоя j ($j = 2, \ldots, K + 1$) K раз решается представленная выше трехдиагональная система уравнений (3.129) методом Зейделя при заданном начальном приближении, например $T_{2,j}$ в первом уравнении, с использованием которого из последующих уравнений до уравнения N методом исключения определяются все искомые переменные рассматриваемого слоя, в том числе и расчетное $T_{N+1,j}$ из последнего уравнения. Однако $T_{N+1,j}$ известно из заданного граничного условия. По отклонению $T^{расч}_{N+1,j}$ от $T^{граничн}_{N+1,j}$ корректируется предварительно заданное значение $T_{1,j}$ до тех пор, пока:

$$\begin{aligned} &\left|T^{расч}_{N,j} - T^{граничн}_{N,j}\right| \le \varepsilon, \\ &j = 1,\ \ldots, K+1, \end{aligned} \tag{3.131}$$

где ε — точность решения ($\varepsilon = 1 \cdot 10^{-5}$).

Шаблон решения с помощью неявной разностной схемы может быть представлен как:



где $i = 2, \ldots, N$; $j = 1, \ldots, K + 1$.

Как правило, алгоритмы решения параболических уравнений с применением неявной разностной схемы сходятся и устойчивы. Поэтому, несмотря на некоторые вычислительные сложности, связанные с необходимостью применения итерационных методов, неявные разностные схемы решения предпочтительнее явных.

Программа решения параболического уравнения неявным методом реализована также на примере уравнения (3.128) с теми же параметрами, как и при использовании явного метода. Она включает в себя два *m*-файла:

- Glav_uravn_parabol_neyavn.m (рис. 3.89) — основная управляющая программа, которая выполняет те же функции, что и управляющая программа в явном методе (3.85);
- parabol_neyavn.m (рис. 3.90), в котором реализуется итерационная процедура решения уравнения (3.128) неявным методом Зейделя и многократно решается система трехдиагональных уравнений (3.129) с использованием формулы (3.130).

```
clc;
clear all;
close all;
fprintf('%s \n','---------------------------------------------------- ');
fprintf('%s \n',' Glav_uravn_parabol_neyavn.m — основная управляющая программа ');
fprintf('%s \n','---------------------------------------------------- ');
fprintf('%s \n',' Программа включает следующие файлы: Glav_uravn_parabol_neyavn.m+parabol_neyavn.m
');
fprintf('%s \n','---------------------------------------------------- ');
fprintf('%s \n',' Программа решения параболического уравнения с частными производными с помощью
неявной разностной схемы ');
fprintf('%s \n','---------------------------------------------------- ');
global T1 T_Nlplus1;
%         1. ФОРМУЛИРОВКА НАЧАЛЬНО-ГРАНИЧНОИ ЗАДАЧИ:
%-------------------------------------------------------
%         Решить параболическое уравнение с частными производными вида:
%         DT/Dt=a*D^2T/Dl^2+sin(l*t), где D — обозначение частной
%         производной;l,t — независимые переменные; T(l,t) — искомое
%         решение как зависимая переменная от l и t; a — константа и параметр
%         уравнения и определить функцию T(l,t)в интервале независимых
%         переменных 0<=l<=L и 0<=t<=tk
%-------------------------------------------------------
%         3. ЗАДАНИЕ ИСХОДНЫХ ДАННЫХ ДЛЯ РАСЧЕТОВ:
%-------------------------------------------------------
%1. [0,L] — интервал по независимой переменной l, на котором определяется решение:
L=5;
%3. Nl — количество участков, на которые разбивается интервал по независимой переменной l
%в соответствии с разностной схемой решения:
Nl=50;
%3. [0,tk] — - интервал по независимой переменной t, на котором определяется решение:
tk=3;
%4. Kt — количество участков, на которые разбивается интервал по независимой переменной t
%в соответствии с разностной схемой решения:
Kt=200;
%5. a — константа и параметр уравнения с частными производными:
a=0.16;
%-------------------------------------------------------
%6. Граничные условия задачи (константы):
%T(0,t)=const=T1;T(L,t)=const=T_Nplus1;
T1=1;
T_Nlplus1=2;
%-------------------------------------------------------
%_______________________________________________________________________
%Начальные условия задачи при t=0(линейная функция от T(l) в виде функции:
% T(l,t=0)=T1+((T_Nplus1-T1)/Nl)*i
% где i — номер шага по независимой переменной l в интервале [0,Nl]
%задаются в специально созданной функции "parabol_neyavn" для решения параболического
%уравнения методом конечных разностей с использованием явной разностной схемы
%_______________________________________________________________________
%-------------------------------------------------------
%7. Требуемая точность решения системы разностных уравнений итерационным методом Зейделя;
eps=0.001;
%-------------------------------------------------------
%                3. РЕШЕНИЕ ЗАДАЧИ
%-------------------------------------------------------
%_______________________________________________________________________
%Решение поставленной задачи и получение искомой функции T=T(l,t) в дискретном виде
%выполняется с применением специально созданной функции "parabol_neyavn"
%методом конечных разностей с использованием неявной разностной схемы
%_______________________________________________________________________
%-------------------------------------------------------
[T,l,t,r,iter]=parabol_neyavn(Nl,Kt,L,tk,a,eps);
%-------------------------------------------------------
%              4. ФОРМИРОВАНИЕ ОТЧЕТА О РЕЗУЛЬТАТАХ РЕШЕНИЯ ЗАДАЧИ
%-------------------------------------------------------
fprintf('%s \n','_____________________________________________________________________________');
fprintf('%s \n','Решение параболического уравнения с частными производными вида:');
fprintf('%s \n','DT/Dt=a*D^2T/Dl^2+sin(l*t), где D — обозначение частной');
```

Рис. 3.89 (начало)

Программный код файла основной управляющей программы решения параболического уравнения неявным разностным методом

```
fprintf('%s \n','производной;l,t — независимые переменные; T(l,t) — искомое');
fprintf('%s \n','решение как зависимая переменная от l и t; a — константа и параметр уравнения');
fprintf('%s \n','Определить функцию T(l,t)в интервале независимых переменных 0<=l<=L и 0<=t<=tk');
fprintf('%s \n','_____');
fprintf('%s \n','---------- ');
fprintf('%s \n','ИСХОДНЫЕ ДАННЫЕ:');
fprintf('%s \n','---------- ');
fprintf('%s \n','ПАРАБОЛИЧЕСКОЕ УРАВНЕНИЕ ВИДА:');
fprintf('%s \n','---------- ');
fprintf('%s \n',' DT/Dt=a*D^2T/Dl^2+sin(l*t) ');
fprintf('%s \n',' где:a=0.16; 0<=l<=Nl; 0<=t<=tk и Nl=5;tk=3');
fprintf('%s \n','---------- ');
fprintf('%s \n',' Граничные условия: ');
fprintf('%s \n',' T(0,t)=const=T1=1;T(L,t)=const=T_Nplus1=2; ');
fprintf('%s \n','---------- ');
fprintf('%s \n',' Начальные условия: ');
fprintf('%s \n',' T(l,t=0)=T1+((T_Nplus1-T1)/Nl)*i ');
fprintf('%s \n','  где i — номер постоянного шага по независимой переменной l, ');
fprintf('%s \n','  которая изменяется в интервале [0,Nl] ');
fprintf('%s \n','---------- ');
fprintf('%s \n',' Параметры разностной схемы решения параболического уравнения: ');
fprintf('%s \n','  Nl=50 — количество одинаковых участков, на которые разбивается');
fprintf('%s \n','  интервал по независимой переменной l : 0<=l<=Nl');
fprintf('%s \n','  Kt=200 — количество одинаковых участков, на которые разбивается интервал по tk');
fprintf('%s \n','  интервал по независимой переменной t : 0<=t<=tk');
fprintf('%s \n','---------- ');
fprintf('%s \n',[' Точность итерационных вычислений методом Зейделя (eps) = ' num2str(eps,'%10.5g') ]);
fprintf('%s \n','---------- ');
fprintf('%s \n',' РЕЗУЛЬТАТЫ РАСЧЕТОВ:');
fprintf('%s \n','---------- ');
fprintf('%s \n',[' Количество итераций, при которых получен результат(iter) = ' num2str(iter,'%10d') ]);
fprintf('%s \n','---------- ');
fprintf('%s \n',' ПРЕДСТАВЛЕНИЕ ИСКОМОИ ФУНКЦИИ T(l,t) В ТАБЛИЧНОМ ВИДЕ: ');
fprintf('%s \n','---------- ');
fprintf('%-5s %-10s %-10s %-10s %-10s %-10s %-10s \n','№','l','T(l,0)','T(l,0.75)','T(l,1.5)','T(l,3.25)','T(l,3)');
fprintf('%s \n','----------');
for i=1:Nl+1
fprintf('%-5d %-10.5f %-10.5f %-10.5f %-10.5f %-10.5f %-10.5f \n',i,l(i),T(i,1),T(i,51),T(i,101),T(i,151),T(i,201));
end
fprintf('%s \n','----------');
fprintf('%s \n',' ПРЕДСТАВЛЕНИЕ ИСКОМОИ ФУНКЦИИ T(l,t) В ГРАФИЧЕСКОМ ВИДЕ: ');
fprintf('%s \n','----------');
%Построение графика решения
surf(l,t,T');
title('Результат решения уравнения DT/Dt=a*D^2T/Dl^2+sin(l*t)');
xlabel('l');
ylabel('t');
zlabel('T(l,t)');
fprintf('%s \n',' **********ВЫПОЛНЕНИЕ ПРОГРАММЫ УСПЕШНО ЗАВЕРШЕНО********** ');
fprintf('%s \n','----------');
```

Рис. 3.89 (окончание)

Программный код файла основной управляющей программы решения параболического уравнения неявным разностным методом

```
function [T,l,t,r,iter] =parabol_neyavn(Nl,Kt,L,tk,a,eps)
%-------------------------------------------------------------
%parabol_neyavn.m — код программы решения параболического уравнения с частными
%производными методом конечных разностей с помощью неявной разностной схемы
%-------------------------------------------------------
% Программа включает следующие файлы:
Glav_uravn_parabol_neyavn.m+parabol_neyavn.m
%-------------------------------------------------------
% Программа решения параболического уравнения с частными производными
%с помощью неявной разностной схемы итерационным методом Зейделя
%-------------------------------------------------------
global T1 T_Nlplus1
```

Рис. 3.90 (начало)

Программный код файла реализации неявного разностного метода при решении параболического уравнения

```
%-------------------------------------------------------------
%T(l,t)- матрица решений как функция от двух переменных длины l и времени t
%T1 — постоянное граничное условие при l=0
%T_Nlplus1 — постоянное граничное условие при l=L
%[0,L] — интервал по независимой переменной l, на котором определяется решение
%[0,tk] — интервал по независимой переменной t, на котором определяется решение
%Nl — количество участков, на которые разбивается интервал по длине L
%Kt — количество участков, на которые разбивается интервал по времени tk
%a — параметр дифференциального уравнения с частными производными
%r — точность решения системы конечных уравнений методом Зейделя
%iter — количество итераций при решении системы конечных уравнений
%-------------------------------------------------------------
%вычисление шага по L
hl=L/Nl;
%вычисление шага по tk
deltat=tk/Kt;
%Из начального условия формируется массив l и первый столбец матрицы
%решений T(l,t)
for i=1:Nl+1
l(i)=(i-1)*hl;
T(i,1)=T1+(T_Nlplus1-T1)/Nl*(i-1);
%T(i,1)=funcL(x(i));
end
%Из граничных условий формируется массив t и первая и последняя строка матрицы
%решений T(l,t)
for j=1:Kt+1
t(j)=(j-1)*deltat;
T(1,j)=T1;
%T(1,j)=funcT0(t(j));
T(Nl+1,j)=T_Nlplus1;
%T(Nl+1,j)=funcTN(t(j));
end
%Определяется матрица ошибок R и заполняется нулями
R(Nl+1,Kt+1)=0;
%Задается начальная неприемлемая точность решения
r=1;
%Задается стартовое число итераций
iter=0;
gam=a*deltat/hl^2;
%Формируется цикл WHILE для организации решения системы конечных уравнений
%итерационным методом Зейделя с точностью eps
while r>eps
%Вычисление матрицы ошибок R и пересчет значений функции T(l,t) во
%внутренних точках при решении системы конечных уравнений
%итерационным методом Зейделя с точностью eps

for i=2:Nl
   for j=2:Kt+1
      Tit=gam/(1+2*gam)*(T(i-1,j)+T(i+1,j))+...
        T(i,j-1)/(1+2*gam)+deltat*sin(l(i)*t(j))/(1+2*gam);
      R(i,j)=abs(T(i,j)-Tit);
      T(i,j)=Tit;
   end
end
%Поиск максимума в матрице ошибок
r=R(1,1);
for i=1:Nl+1
   for j=1:Kt+1
      if R(i,j)>r
         r=R(i,j);
      end
   end
end
%Счетчик итераций
iter=iter+1;
end
end
```

Рис. 3.90 (окончание)

Программный код файла реализации неявного разностного метода при решении параболического уравнения

При этом обращение к функции решения уравнения (3.128) неявным методом из управляющей программы (3.89) выполняется следующим образом:

$$[T, \ell, t, r, iter] = parabol\_neyavn(N\ell, Kt, L, tk, a, eps),$$

где r — достигнутая точность итерационных расчетов; *iter* — число итераций, в результате которых получено решение; *eps* — задаваемая точность итерационных расчетов.

Следует отметить, что в программной реализации решения задачи неявным методом (3.90) условием завершения итерационного процесса Зейделя является не только условие (3.131), но и минимизация всех абсолютных рассогласований между дискретными значениями искомой функции $T(\ell, t)$ на предыдущей и текущей итерации. При этом из множества рассогласований

$$\max \left| T_{i,j}^{\text{текущее}} - T_{i,j}^{\text{предыдущее}} \right| \le \varepsilon \quad i = 1, \ \ldots, N+1; j = 1, \ \ldots, K+1 \tag{3.132}$$

выбирается наихудшее (наибольшее) для определения условия окончания итерационного процесса.

Результат решения параболического уравнения с частными производными (3.128) неявным разностным методом представлен в табличном виде на рисунке 3.91 и несущественно отличается от результата, полученного явным методом (рис. 3.87).

```
---------------------------------------------
Glav_uravn_parabol_neyavn.m — основная управляющая программа
---------------------------------------------
Программа включает следующие файлы:
Glav_uravn_parabol_neyavn.m+parabol_neyavn.m
---------------------------------------------
Программа решения параболического уравнения с частными производными с по-
мощью неявной разностной схемы
---------------------------------------------
_____________________________________________
Решение параболического уравнения с частными производными вида:
DT/Dt=a*D^2T/Dl^2+sin(l*t), где D — обозначение частной
производной;l,t — независимые переменные; T(l,t) — искомое
решение как зависимая переменная от l и t; a — константа и параметр уравнения
Определить функцию T(l,t)в интервале независимых переменных 0<=l<=L и 0<=t<=tk
_____________________________________________
---------------------------------------------
ИСХОДНЫЕ ДАННЫЕ:
---------------------------------------------
ПАРАБОЛИЧЕСКОЕ УРАВНЕНИЕ ВИДА:
---------------------------------------------
DT/Dt=a*D^2T/Dl^2+sin(l*t)
где:a=0.16; 0<=l<=Nl; 0<=t<=tk и Nl=5;tk=3
---------------------------------------------
Граничные условия:
T(0,t)=const=T1=1;T(L,t)=const=T_Nplus1=2;
---------------------------------------------
Начальные условия:
T(l,t=0)=T1+((T_Nplus1-T1)/Nl)*i
 где i — номер постоянного шага по независимой переменной l,
 которая изменяется в интервале [0,Nl]
---------------------------------------------
```

Рис. 3.91 (начало)
Результат решения параболического уравнения с частными производными (3.128) неявным разностным методом

Параметры разностной схемы решения параболического уравнения:
NI=50 — количество одинаковых участков, на которые разбивается интервал по независимой переменной l : 0<=l<=NI
Kt=200 — количество одинаковых участков, на которые разбивается интервал по tk интервал по независимой переменной t : 0<=t<=tk

---

Точность итерационных вычислений методом Зейделя (eps) = 0.001

РЕЗУЛЬТАТЫ РАСЧЕТОВ:

Количество итераций, при которых получен результат(iter) = 74

ПРЕДСТАВЛЕНИЕ ИСКОМОИ ФУНКЦИИ T(l,t) В ТАБЛИЧНОМ ВИДЕ:

№	l	T(l,0)	T(l,0.75)	T(l,1.5)	T(l,3.25)	T(l,3)
1	0.00000	1.00000	1.00000	1.00000	1.00000	1.00000
2	0.10000	1.02000	1.04847	1.12726	1.23129	1.31476
3	0.20000	1.04000	1.09685	1.25330	1.45718	1.61615
4	0.30000	1.06000	1.14507	1.37691	1.67245	1.89147
5	0.40000	1.08000	1.19305	1.49695	1.87225	3.12965
6	0.50000	1.10000	1.24070	1.61229	3.05226	3.32198
7	0.60000	1.12000	1.28795	1.72190	3.20883	3.46254
8	0.70000	1.14000	1.33473	1.82483	3.33913	3.54863
9	0.80000	1.16000	1.38094	1.92023	3.44118	3.58074
10	0.90000	1.18000	1.42654	3.00735	3.51396	3.56254
11	1.00000	1.20000	1.47143	3.08559	3.55740	3.50046
12	1.10000	1.22000	1.51556	3.15445	3.57237	3.40325
13	1.20000	1.24000	1.55885	3.21360	3.56062	3.28125
14	1.30000	1.26000	1.60126	3.26282	3.52473	3.14570
15	1.40000	1.28000	1.64271	3.30206	3.46795	3.00788
16	1.50000	1.30000	1.68315	3.33140	3.39408	1.87844
17	1.60000	1.32000	1.72254	3.35104	3.30732	1.76660
18	1.70000	1.34000	1.76082	3.36135	3.21213	1.67965
19	1.80000	1.36000	1.79796	3.36280	3.11301	1.62253
20	1.90000	1.38000	1.83391	3.35598	3.01434	1.59758
21	3.00000	1.40000	1.86863	3.34159	1.92026	1.60454
22	3.10000	1.42000	1.90211	3.32042	1.83449	1.64070
23	3.20000	1.44000	1.93430	3.29333	1.76020	1.70126
24	3.30000	1.46000	1.96520	3.26124	1.69992	1.77979
25	3.40000	1.48000	1.99479	3.22513	1.65549	1.86882
26	3.50000	1.50000	3.02306	3.18599	1.62798	1.96049
27	3.60000	1.52000	3.04999	3.14482	1.61770	3.04717
28	3.70000	1.54000	3.07559	3.10264	1.62421	3.12207
29	3.80000	1.56000	3.09986	3.06040	1.64639	3.17971
30	3.90000	1.58000	3.12281	3.01903	1.68251	3.21633
31	3.00000	1.60000	3.14445	1.97939	1.73031	3.23008
32	3.10000	1.62000	3.16480	1.94226	1.78714	3.22111
33	3.20000	1.64000	3.18387	1.90832	1.85008	3.19142
34	3.30000	1.66000	3.20169	1.87813	1.91608	3.14468
35	3.40000	1.68000	3.21827	1.85211	1.98209	3.08579
36	3.50000	1.70000	3.23361	1.83058	3.04520	3.02049
37	3.60000	1.72000	3.24772	1.81366	3.10273	1.95481
38	3.70000	1.74000	3.26055	1.80141	3.15235	1.89461
39	3.80000	1.76000	3.27198	1.79373	3.19209	1.84511
40	3.90000	1.78000	3.28180	1.79053	3.22043	1.81049
41	4.00000	1.80000	3.28961	1.79173	3.23626	1.79360
42	4.10000	1.82000	3.29475	1.79736	3.23893	1.79575
43	4.20000	1.84000	3.29621	1.80770	3.22821	1.81653
44	4.30000	1.86000	3.29247	1.82327	3.20443	1.85375
45	4.40000	1.88000	3.28154	1.84475	3.16866	1.90332
46	4.50000	1.90000	3.26098	1.87274	3.12317	1.95917
47	4.60000	1.92000	3.22834	1.90704	3.07203	3.01316
48	4.70000	1.94000	3.18206	1.94556	3.02197	3.05520
49	4.80000	1.96000	3.12308	1.98264	1.98325	3.07389
50	4.90000	1.98000	3.05747	3.00706	1.97011	3.05804
51	5.00000	3.00000	3.00000	3.00000	3.00000	3.00000

Рис. 3.91 (продолжение)
Результат решения параболического уравнения с частными производными (3.128) неявным разностным методом

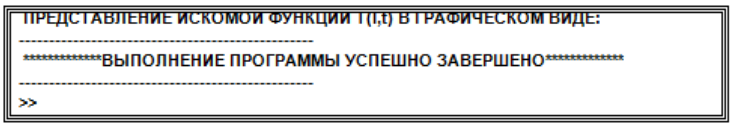


Рис. 3.91 (окончание)

Результат решения параболического уравнения с частными производными (3.128) неявным разностным методом

3.4.2.3. Моделирование нестационарного процесса теплопроводности в металлическом цилиндре при наличии внешнего источника тепла

Нестационарный процесс теплопроводности в твердом стержне, с боковой поверхности которого происходит теплообмен с окружающей средой (теплопередача по закону Ньютона), описывается дифференциальным уравнением с частными производными Фурье — Кирхгофа следующего вида:

$$\frac{\partial T}{\partial \ell} = a\frac{\partial^2 T}{\partial \ell^2} + \frac{KF}{VC\rho}\left(T_{out} - T\right), \tag{3.133}$$

где $T = T(\ell, t)$ [°C] — искомое распределение температуры в стержне длиной L (м) — $0 \le \ell \le L$, и во времени t [ч] — $0 \le t \le t^{(k)}$; T_{out} [°C] — температура окружающей среды; a [м^2/ч] — коэффициент температуропроводности; K [кДж/ч·м^2·°C] — коэффициент теплопередачи между стержнем и окружающей средой; F [м^2] — площадь поверхности теплопередачи; C [кДж/кг·К] — теплоемкость твердого тела; ρ [кг/м^3] — плотность твердого стержня.

При этом принимается, что твердый стержень — металлический, и его коэффициент теплопроводности определяется по формуле:

$$a = \frac{3600\lambda}{\rho C}, \tag{3.134}$$

где λ [кДж/с·К·м] — коэффициент теплопроводности стержня.

Для определения объема цилиндрического стержня и площади поверхности теплопередачи (F) в уравнении (3.133) используются известные соотношения:

$$\begin{aligned} F &= 2\pi RL, \\ V &= \pi R^2 L, \end{aligned} \tag{3.135}$$

где R [м] — радиус цилиндрического стержня.

Конкретные числовые значения перечисленных физико-химических параметров приведены в таблице 3.1 и в файле исходных данных DATA.m (рис. 3.92). Основная управляющая программа решения уравнения (3.133) Glav_uravn_Fourier_Kirchhoff_parabol_neyavn.m (рис. 3.93), использующая для решения неявный метод сеток, оформлена в виде *m*-файла на рисунке 3.90.

Таблица 3.1

Конструкционные и физико-химические параметры цилиндрического стержня и внешней среды

№	Название параметра	Обозначение	Величина или способ определения	Единицы измерения
1	Радиус цилиндрического стержня	R	1	м
2	Площадь цилиндрической поверхности теплопередачи	F	$2\pi RL$	$м^2$
3	Объем стержня	V	$\pi R^2 L$	$м^3$
4	Коэффициент теплопередачи	K	1440	кДж / ч·$м^2$·К
5	Коэффициент теплопроводности	λ	0,394	кДж / с·м·К
6	Теплоемкость материала стержня	C	0,322	кДж / кг·К
7	Плотность материала стержня	ρ	8940	кг / $м^3$
8	Коэффициент температуропроводности	a	$\frac{3600\lambda}{\rho C}$	$м^2$ / ч
9	Температура внешней среды	T_{out}	250	°С
10	Обобщенный коэффициент теплопередачи	$K\pi$	$\frac{KF}{VC\rho}$	°С

```
function DATA
%-------------------------------------------------------
%DATA.m — программный код файла, в котором задаются исходные данные для расчетов
%-------------------------------------------------------
 %Программа включает следующие файлы:
Glav_uravn_Fourier_Kirchhoff_parabol_neyavn.m+DATA.m+parabol_neyavn.m+REPORT.m
%-------------------------------------------------------
%Программа решения уравнения Фурье-Кирхгофа, описывающего
%нестационарный процесс теплопроводности в металлическом цилиндре
%с учетом теплопередачи между цилиндром и окружающей средой
%с помощью неявной разностной схемы итерационным методом Зейделя
%-------------------------------------------------------
%Задание исходных данных для решения параболического уравнения с частными
%производными вида, описывающего нестационарный процесс теплопроводности
%в металлическом цилиндре при наличии внешнего источника тепла с температурой Tout
% в соответствии с уравнением Фурье-Кирхгофа.
%-------------------------------------------------------
% Общий вид параболического уравнения с частными производными, описывающего
% нестационарный процесс теплопроводности в металлическом цилиндре с учетом
%теплообмена (теплопередачи) с окружающей средой:
%DT{l,t}/Dt=a*D^2T{l,t}/Dl^2+KT*FT/(V*RO*C)*(Tout-T{l,t}),
%где D — обозначение частной производной;l,t — независимые переменные;
%T(l,t) — искомая функция решения, зависимая переменная от l и t;
%a — коэффициент температуропроводности (константа — м^2/ч),
%который определяется по формуле;
%a=lambda*3600/(RO*C), где:
%lambda — коэффициент теплопроводности металла — кдж/(с*К*м)
%RO — плотность металла — кг/м^3
%C — теплоемкость металла — кдж/(кг*К)
%-------------------------------------------------------
global a KTT Tout L tk Nl Kt T1 T_Nlplus1 eps;
global lambda radius KT RO C FT V;
%-------------------------------------------------------
```

Рис. 3.92 (начало)

Программный код файла исходных данных DATA.m при моделировании нестационарного процесса теплопередачи в цилиндрическом стержне с внешним источником тепла

```
%1. Задание физико-химических данных для расчетов:
%-----------------------------------------
%lambda — коэффициент теплопроводности металла — кдж/(с*К*м)
lambda=0.394;
%RO — плотность металла — кг/м^3
RO=8940;
%C — теплоемкость металла — кдж/(кг*К)
C=0.322;
%Расчет коэффициента температуропроводности а — м^2/ч
a=lambda*3600/(RO*C);
%KT — коэффициент теплопередачи металлического цилиндра с внешней средой —
кдж/(ч*м^2*К)
KT=1440;
% radius - радиус цилиндра - м
radius=1;
%L - высота цилиндра - м
L=5;
%FT - площадь поверхности теплопередачи металлического цилиндра с внешней средой -
м^2;
FT=2*pi*radius*L;
%V — объем металлического цилиндра, в котором происходит нестационарный процесс
%теплопроводности в соответствии с законом Фурье — м^3
V=pi*radius^2*L;
%KTT — константа уравнения — ч^(-1), характеризующая процесс теплопередачи с внеш-
ней средой:
KTT=KT*FT/(V*RO*C);
%Tout — температура внешней среды — С
Tout=250;
%-----------------------------------------
%3. Задание начально-конечных данных для решения дифференциального уравнения с
частными производными.
%-----------------------------------------
%0<=l<=L — интервал (метры) по длине, на котором определяется решение:
L=5;
%0<=t<=tk — интервал по времени (часы), на котором определяется решение:
tk=3;
%Nl — количество участков, на которые разбивается интервал по длине L в
%соответствии с разностной схемой решения:
Nl=50;
%Kt — количество участков, на которые разбивается интервал по времени t в
%соответствии с разностной схемой решения:
Kt=200;
%-----------------------------------------
% Граничные условия задачи(в данном частном случае не функции от t, а константы):
%T(0,t)=const=T1;T(L,t)=const=T_Nplus1;
%T1 — начальная граничная температура — С
T1=100;
%T_Nlplus1 — конечная граничная температура — С
T_Nlplus1=200;
%-----------------------------------------
% Начальные условия при t=0(линейная функция от T(l) в виде функции от номера шага "i",
число который равно Nl )
% T(l,t=0)=T1+((T_Nplus1-T1)/Nl)*i
% где i — номер шага по длине L в интервале [0,Nl]
%-----------------------------------------
%Требуемая точность решения системы разностных уравнений итерационным методом
Зейделя;
eps=0.001;
end
```

Рис. 3.92 (окончание)

Программный код файла исходных данных DATA.m при моделировании нестационарного процесса теплопередачи в цилиндрическом стержне с внешним источником тепла

```
clc;
clear all;
close all;
%-----------------------------------------------------
%Glav_uravn_Fourier_Kirchhoff_parabol_neyavn.m — основная управляющая программа
%-----------------------------------------------------
 %Программа включает следующие файлы:
Glav_uravn_Fourier_Kirchhoff_parabol_neyavn.m+DATA.m+parabol_neyavn.m+REPORT.m
%-----------------------------------------------------
%Программа решения уравнения Фурье-Кирхгофа, описывающего
%нестационарный процесс теплопроводности в металлическом цилиндре
%с учетом теплопередачи между цилиндром и окружающей средой
%с помощью неявной разностной схемы и терационным методом Зейделя
%-----------------------------------------------------

global T a L tk Nl Kt eps iter l t;
%-----------------------------------------
%1.Формулировка физико-химической задачи.
%-----------------------------------------
%1.1.Решить параболическое уравнение Фурье-Кирхгофа с частными производными,
%описывающее нестационарный процесс теплопроводности в металлическом цилиндре при
наличии
%внешнего источника тепла с температурой Tout.
% Общий вид параболического уравнения с частными производными, описывающего
% нестационарный процесс теплопроводности в металлическом цилиндре с учетом
% теплообмена (теплопередачи) с окружающей средой:
%DT{l,t}/Dt=a*D^2T{l,t}/Dl^2+KTT*(Tout-T{l,t}),
%где D — обозначение частной производной;l,t — независимые переменные;
%T(l,t) — искомая функция решения, зависимая переменная от l и t;
%-----------------------------------------
%1.3.a — коэффициент температуропроводности (константа — м^2/ч),
%который определяется по формуле;
%a=lambda*3600/(RO*C), где:
%lambda — коэффициент теплопроводности — кдж/(с*К*м)
%RO — плотность металла — кг/м^3
%C — теплоемкость металла — кдж/(кг*К)
%-----------------------------------------
%1.3.KTT — константа уравнения — ч^(-1), характеризующая процесс теплопередачи
%между металлическим цилиндром и внешней средой:
%KTT=KT*FT/(V*RO*C);
%KT — коэффициент теплопередачи металлического цилиндра с внешней средой —
кдж/(ч*м^2*К)
% radius — радиус цилиндра — м
%radius=1;
%L — высота цилиндра — м
%FT — площадь поверхности теплопередачи металлического цилиндра с внешней средой —
м^2;
%V — объем металлического цилиндра, в котором происходит нестационарный процесс
%Tout — температура внешней среды — С
%-----------------------------------------
%  3.Формулировка начально-граничной задачи.
%-----------------------------------------
% 3.1.Определить функцию T(l,t) в интервале независимых переменных 0<=l<=L и 0<=t<=tk
%где
%0<=l<=L — интервал (метры) по длине, на котором определяется решение:
%0<=t<=tk — интервал по времени (часы), на котором определяется решение:
%Nl — количество участков, на которые разбивается интервал по длине L в
%соответствии с разностной схемой решения:
%Kt — количество участков, на которые разбивается интервал по времени t в
%соответствии с разностной схемой решения:
%-----------------------------------------
% 3.3.Граничные условия задачи (в данном частном случае не функции от времени (t), а констан-
ты):
%T(0,t)=const=T1;
%T1 —  граничная температура при l=0, не зависящая от времени (t) — С
%T(L,t)=const=T_Nplus1;
%T_Nlplus1 — граничная температура при l=L, не зависящая от времени (t)-С
%-----------------------------------------
```

Рис. 3.93 (начало)

Программный код файла Glav_uravn_Fourier_Kirchhoff_parabol_neyavn.m основной управляющей программы решения задачи моделирования нестационарного процесса теплопередачи в цилиндрическом стержне с внешним источником тепла

```
% 3.3.Начальные условия при t=0(линеиная функция от T(l) в виде функции от номера шага "i"  )
% в интервале по длине 0<=l<=L,число который равно Nl T(l,t=0)=T1+((T_Nplus1-T1)/Nl)*i
% где i — номер шага по длине L в интервале [0,Nl]
%-----------------------------------------
%3.4.eps — задаваемая точность решения системы разностных уравнений итерационным методом Зейделя;
%-----------------------------------------
DATA;
%-----------------------------------------
%Решение задачи с использованием собственной функции "parabol_neyavn", созданной для
%решения параболического уравнения итерационным методом конечных разностей
%с помощью неявной разностной схемы
%-----------------------------------------
[T,l,t,r,iter]=parabol_neyavn(Nl,Kt,L,tk,a,eps);
%-----------------------------------------
%Решение в виде функции T=T(l,t) в табличном виде
%-----------------------------------------
REPORT;
```

Рис. 3.93 (окончание)

Программный код файла Glav_uravn_Fourier_Kirchhoff_parabol_neyavn.m основной управляющей программы решения задачи моделирования нестационарного процесса теплопередачи в цилиндрическом стержне с внешним источником тепла

Для нахождения единственного решения параболического уравнения (3.133) — функции $T = T(\ell,\ t)$, задаются дополнительные условия, перечисленные в таблице 3.2.

Таблица 3.2

Уравнение Фурье — Кирхгофа и параметры алгоритма решения неявным разностным методом

1	Уравнение с частными производными параболического типа	$\frac{\partial T}{\partial \ell} = a\frac{\partial^2 T}{\partial \ell^2} + \frac{KF}{VC\rho}(T_{out} - T),$
2	Диапазоны изменения независимых переменных ℓ и t	$0 \le \ell \le 5$(м) $0 \le t \le 3$(час)
3	Параметры расчетной сетки при дискретизации (разбиении на участки) диапазона независимых переменных	По пространственной координате ℓ: $N = 50$ По временной координате t: $Kt = 200$
4	Граничные условия	$T_1 = 100$°C $T_{N+1} = 200$°C
5	Начальные условия	$T(\ell,0) = T_1 + \frac{T_{N+1} - T_1}{N}(i-1)$ $i = 1, \ldots, (N+1).$
6	Точность итерационного процесса	$\varepsilon = 0{,}001$

Постоянные граничные условия на концах стержня ($\ell = 0$ и $\ell = L$):

$$\begin{aligned} T(0,t) &= T_1 = \text{const}, \\ T(L,t) &= T_{N+1} = \text{const}, \end{aligned} \tag{3.136}$$

где N — количество участков, на которые разбивается интервал по длине L.

Линейно зависимые от пространственной координаты начальные условия ($t = 0$):

$$T(\ell,0)=T_1+\frac{T_{N+1}-T_1}{N}(i-1)$$
$$i=1,\ \ldots,\ (N+1). \tag{3.137}$$

В соответствии с условиями решения задачи (3.133) интервалы изменения независимых переменных $0 \le \ell \le L$ и $0 \le t \le t^{(k)}$ разбиваются на дискретные участки, число которых в файле DATA.m (рис. 3.92):

$$\text{для } 0 \le l \le L - N = 50,$$
$$\text{для } 0 \le t \le t^{(k)} - Kt = 200. \tag{3.138}$$

При этом задача решается неявным разностным методом, реализованным в предыдущем разделе 3.4.2.2 в виде *m*-файла parabol_neyavn.m (рис. 3.90). Для решения уравнения (3.133) вводится дополнительный постоянный коэффициент *KTT*, объединяющий все константы уравнения:

$$KTT=\frac{KF}{VC\rho}. \tag{3.139}$$

Результаты решения уравнения Фурье — Кирхгофа для цилиндрического стержня с внешним источником тепла приведены в Приложении (табл. П.7) и представлены на рисунке 3.94.

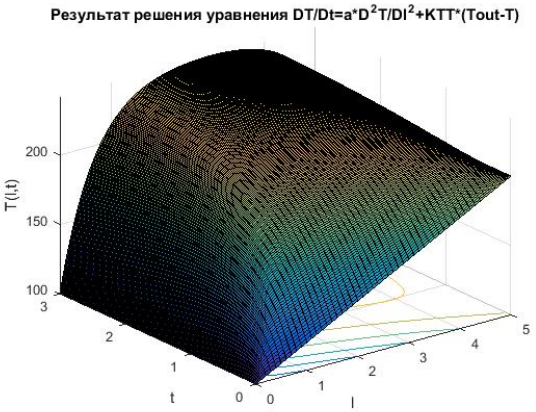


Рис. 3.94

Графическое представление результата решения уравнения Фурье — Кирхгофа для цилиндрического стержня с внешним источником тепла

3.5. Решение оптимизационных задач

В общем случае задачи оптимизации относятся к классу экстремальных задач, сущность которых сводится к определению таких параметров задачи — оптимальных параметров, которые обеспечивают экстремальное (минимальное или максимальное) значение некоторой целевой функции (функционала).

При оптимизации процессов с условно сосредоточенными или дискретно распределенными параметрами вдоль пространственных координат или во времени вычисляются оптимальные параметры, которые соответствуют некоторому экстремуму дискретной целевой функции типа критерия метода наименьших квадратов (МНК). Такие задачи называются задачами дискретной оптимизации, они и рассматриваются в настоящем разделе.

В свою очередь, при оптимизации процессов с непрерывно-распределенными параметрами вдоль пространственных координат или во времени вычисляются оптимальные функции (минимали или максимали), которым соответствует экстремальное значение некоторого интеграла, называемого функционалом. Задачи такого типа решаются методами вариационного исчисления и в настоящей книге не рассматриваются. Тем более что часто задачи оптимизации непрерывно распределенных процессов иногда могут быть сведены к задачам дискретной оптимизациии.

Задача дискретной оптимизации может решаться:

а) без ограничений на параметры процесса, в том числе и на искомые оптимальные параметры — так называемая безусловная оптимизация;

б) с ограничениями на искомые оптимальные параметры — так называемые ограничения первого рода;

в) с ограничениями на искомые оптимальные параметры и одновременно на любые конструктивные и физико-химические параметры процесса в виде систем линейных и нелинейных уравнений (так называемые ограничения второго рода).

Такие задачи с нелинейными целевыми функциями и линейными/нелинейными ограничениями принято относить к задачам нелинейного программирования.

При решении задач одномерной оптимизации, когда определяется один оптимальный параметр, обычно используют решатель MATLAB “fminbnd”, при многомерной безусловной оптимизации — “fminsearch”, при многомерной оптимизации с ограничениями первого и второго рода — “fmincon”. В некоторых случаях, как, например, при определении коэффициентов многочленов (полиномов) произвольной степени, может использоваться решатель “polyfit”, а параметров нелинейных по параметрам функций, зависящих от одной независимой переменной, — решатель “lsqcurvefit”. В первом приближении этих решателей достаточно для решения разнородных оптимизационных задач, однако следует иметь в виду, что существуют более современные и совершенные решатели MATLAB, которые подробно представлены в различных информационных и других источниках.

Перечисленные решатели используются в основном для решения трех типов задач компьютерного моделирования химико-технологических процессов.

1. Оптимизации процесса — определение оптимальных режимных и конструктивных параметров физико-химического процесса и технологии при выбранном технологическом, экономическом или технико-экономическом критерии оптимальности — целевой функции. При решении задачи предполагается наличие адекватной компьютерной модели процесса, а целевая функция (кри-

терий оптимальности) выбирается исходя из цели решаемой задачи. При поиске оптимальный параметров стремятся достигнуть максимального (в некоторых случаях наибольшего) или минимального (в некоторых случаях наименьшего) значения целевой функции — критерия оптимальности.

2. Структурной и параметрической идентификации — определение структуры и коэффициентов систем линейных, нелинейных и дифференциальных уравнений, описывающих химико-технологический процесс по экспериментальным данным. В этом случае определяются (уточняются) вид системы уравнений и коэффициенты уравнений математического описания путем нахождения минимального значения критерия рассогласования (невязки) расчетных и экспериментальных величин параметров процесса.

3. Решения систем линейных или нелинейных уравнений, описывающих технологический процесс нахождения таких корней, которые обеспечивают минимальное (наименьшее) отклонение функций всех уравнений, записанных в неявном виде, от нуля. Найденные таким образом корни превращают каждое из уравнений системы в верное числовое равенство (тождество).

3.5.1. Общая процедура решения оптимизационных задач

Для решения оптимизационных задач необходимо следующее.

1. Выбрать или сформировать целевую функцию $R = R(\bar{x})$ — критерий оптимальности, который должен быть количественным, чувствительным (по возможности заметно изменяться в зависимости от изменения ресурсов оптимизации — оптимизирующих переменных) и иметь экстремальное (минимальное или максимальное) значение в рассматриваемом интервале решения задач.

2. Выявить обоснованные ресурсы оптимизации — оптимизирующие переменные $\bar{x}$, влияющие на целевую функцию R и позволяющие обеспечить ее экстремальный характер.

3. Располагать надежным алгоритмом оптимизации, реализованным в данном случае в виде стандартного решателя MATLAB, для определения оптимального решения — в общем случае оптимального значения ресурса оптимизации — оптимизирующих переменных $\bar{x}^{opt}$ и экстремальной (минимальной или максимальной) величины целевой функции — критерия оптимальности R^{opt}.

3.5.1.1. Выбор целевых функций при решении оптимизационных задач

Выбор целевой функции для трех типов задач оптимизации отличается друг от друга.

Так, *при оптимизации химико-технологического процесса (задача первого типа)* критерий оптимальности может быть технологическим, например — выход целевого продукта — экономическим; годовые затраты на процесс — термодинамическим; эксергетический показатель и т. п. Понятно, что в одних случаях следует определять минимум целевой функции, а в других — максимум. Очевидно также, что не всегда удается ограничиться одним критерием оптимальности и приходится использовать несколько критериев, их совмещение возможно, например, с помощью аддитивного (общего) критерия оптимально-

сти с весовыми коэффициентами. В таких случаях принято говорить о решении многокритериальных задач оптимизации. При этом возникают проблемы, связанные с выбором весовых коэффициентов и согласованием разнотипных частных экстремумов (отдельных из множества) целевых функций — одни имеют минимумы, а другие максимумы. Как правило, всегда определяется минимальное значение целевой функции, а когда необходимо определять максимум, то он преобразуется путем изменения знака к виду, требующему определения минимума целевой функции.

Очень серьезной проблемой при решении оптимизационных задач наряду с многоэкстремальностью (рис. 3.95) является и сложный, в частности овражный, характер рельефа целевой функции (рис. 3.96).

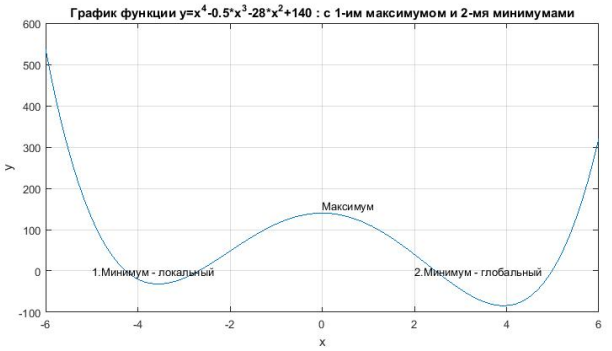


Рис. 3.95

Пример многоэкстремальной целевой функции $y = f(x)$ с глобальным минимумом (наименьшим значением)

На рисунке 3.95 приведен пример многоэкстремальной целевой функции при решении задачи одномерной оптимизации. При поиске минимума следует находить глобальный минимум (наименьшее значение), так как определение локального минимума может быть некорректным результатом, не соответствующим физическому смыслу решаемой задачи.

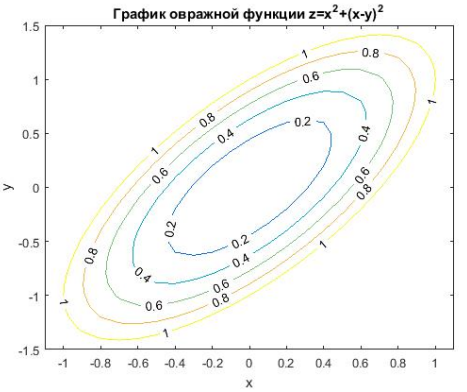


Рис. 3.96

Пример целевой функции $z = f(x, y)$ с овражным рельефом

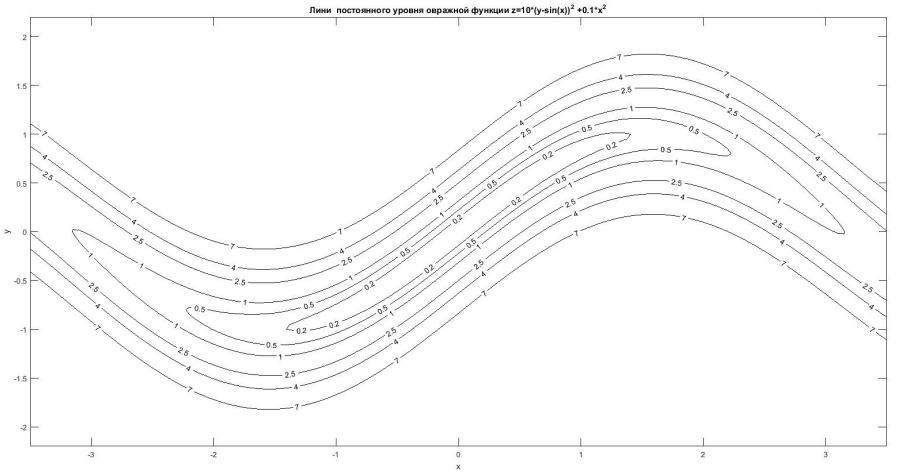


Рис. 3.97

Пример целевой функции $z = f(x, y)$ со сложным овражным рельефом

Сложнее решать задачи такого типа для случая многомерной оптимизации. На рисунках 3.96–3.99 представлены линии уровня различных типов рельефов овражных и многоэкстремальных целевых функций при решении задачи двумерной оптимизации. Решение такого типа задач требует разработки специальных методов инициализации решений оптимизационных задач и тщательного выбора специальных методов и алгоритмов, позволяющих как определить глобальный экстремум (рис. 3.98 и 3.99), так и справиться с проблемами решения при наличии оврагов (рис. 3.96 и 3.97).

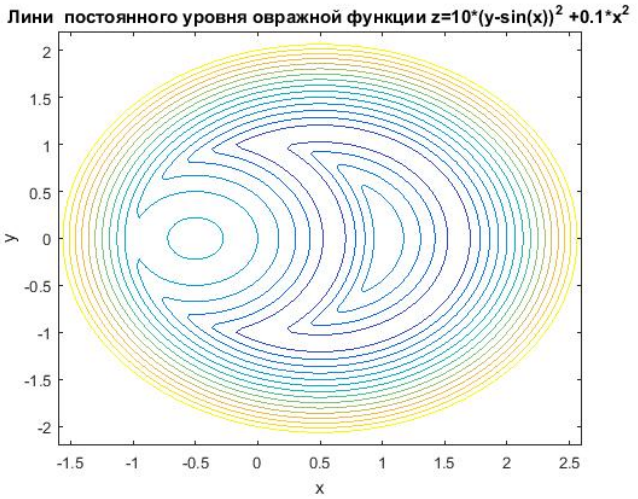


Рис. 3.98

Пример овражной многоэкстремальной целевой функции $z = f(x, y)$

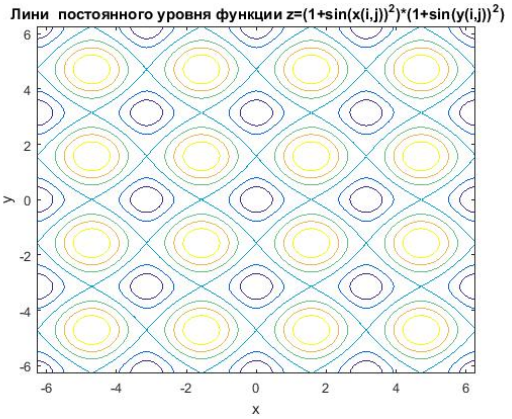


Рис. 3.99

Пример многоэкстремальной целевой функции $z = f(x, y)$

При решении второго типа оптимизационных задач химической технологии — структурной и параметрической оптимизации — целевая функция R представляет собой рассогласование (невязку) между рассчитанными по математической модели параметрами процесса и экспериментальными данными. Это рассогласование должно быть минимизировано за счет определения оптимальных коэффициентов (параметров) математической модели и ее структуры — числа и вида уравнений математического описания.

Если обозначить экспериментальные данные в N опытных точках ($y_i^{\text{эксп}}$; $i = 1,\ldots, N$) при известных значениях влияющих на них параметров процесса — независимых переменных (x_i; $i = 1, \ldots, N$), а рассчитанные по модели те же значения переменных ($y_i^{\text{расч}}$; $i = 1,\ldots, N$), зависящие от тех же независимых переменных (x_i; $i = 1,\ldots, N$) и от некоторых коэффициентов модели $\overline{a} = \left[a_1,\ldots,a_p\right]$, то для оценки рассогласования расчетных и экспериментальных значений параметров процесса ($y_i^{расч}$ и $y_i^{эксп}$) можно использовать три критерия.

1. Критерий метода наименьших квадратов (МНК):

$$Cr = \sum_{i=1}^{n} w_i \left(y_i^{\text{расч}}\left(\overline{a}, \overline{x}_i\right) - y_i^{\text{эксп}}\left(\overline{x}_i\right)\right)^2. \qquad (3.140)$$

2. Критерий метода наименьших модулей (МНМ):

$$Cr = \sum_{i=1}^{n} w_i \left| y_i^{\text{расч}}\left(\overline{a}, \overline{x}_i\right) - y_i^{\text{эксп}}\left(\overline{x}_i\right)\right|. \qquad (3.141)$$

3. Критерий Чебышева — минимаксный критерий (ММК):

$$Cr = \min\left(\max_{1 \le i \le N}\left(w_i \left| y_i^{расч}\left(\overline{a}, \overline{x}_i\right) - y_i^{эксп}\left(\overline{x}_i\right)\right|\right)\right). \qquad (3.142)$$

Весовые коэффициенты w_i, $i = 1, \ldots, N$, в этих формулах определяют важность включения рассогласования между экспериментальными $y_i^{\text{эксп}}$, $i = 1,\ldots, N$,

и рассчитанными по модели параметрами процесса ($y_i^{\text{расч}}$, i = 1, …, N) в соответствующие критерии (3.140)–(3.142). Чем больше весовой коэффициент, тем более приоритетным (важным) считается i-е рассогласование (измерение), которое должно влиять на результаты решения идентификационной задачи. Существуют различные методы оценки весовых коэффициентов, как, например, эмпирические, статистические и т. п., которые могут выбираться исходя из физического смысла решаемой задачи.

Необходимо отметить, что величина критериев в этих формулах зависит не только от коэффициентов математических моделей — $\bar{y}(\bar{x},\bar{a})$, но и от структуры, вида и числа уравнений математического описания. Поэтому с их использованием и путем их минимизации решаются задачи как параметрической, так и структурной идентификации математических моделей. По приоритету наиболее употребляемым является критерий МНК, за ним следует минимаксный метод Чебышева ММК, и в последнюю очередь критерий МНМ.

Теоретические аспекты решаемой задачи идентификации тесно связаны с математической статистикой и регрессионным анализом, что позволяет более обоснованно определять весовые коэффициенты в соотношениях (3.140)–(3.142), в частности путем использования обратной величины дисперсии экспериментальных измерений.

Завершая обсуждение критериев (3.140)–(3.142), важно подчеркнуть, что при их использовании ресурсами оптимизации являются различные виды уравнений математического описания — задача структурной идентификации, и коэффициенты уравнений математического описания моделей — задача параметрической идентификации.

К третьему типу оптимизационных задач можно отнести решение систем линейных, а чаще нелинейных уравнений (одно нелинейное уравнение рассматривается как частный случай системы нелинейных уравнений) вида:

$$\begin{aligned} f_1(x_1, x_2, \ \ldots, x_n) &= 0, \\ f_2(x_1, x_2, \ \ldots, x_n) &= 0, \\ &\vdots \\ f_n(x_1, x_2, \ \ldots, x_n) &= 0, \end{aligned} \tag{3.143}$$

где $f_i(x_1, x_2, \ \ldots, x_{\text{n}})$ — в общем случае нелинейные функции системы нелинейных уравнений, зависящие от определяемых переменных, которые необходимо вычислить путем итерационных расчетов.

Для решения задачи используются рассмотренные выше критерии (3.140)–(3.142) с той лишь разницей, что вместо $\bar{y}^{\text{расч}}$ в них включаются функции $f_i(x_1, x_2, \ \ldots, x_{\text{n}})$, а вместо $\bar{y}^{\text{эксп}}$ записывается ноль (0), так как в результате решения необходимо обеспечить нулевые (с определенной точностью) значения всех функций системы (3.143).

Отсюда следует, что одним из результатов решения должно быть не просто минимальное (возможно наименьшее) — глобальный минимум при нали-

чии нескольких минимумов значения одного из критериев (3.140)–(3.142), но и этот критерий должен быть близок к нулю с определенной точностью. Это и есть в данном случае результат решения оптимизационной задачи, т. е. целевая функция близка к нулю.

Найденные одновременно оптимальные значения определяемых переменных x_1^{opt}, x_2^{opt}, …, x_n^{opt} при этом являются решениями или корнями системы уравнений (3.143).

3.5.1.2. Выбор ресурсов оптимизации — оптимизирующих переменных

Выбор ресурсов оптимизации — оптимизирующих переменных — зависит от типа решаемой оптимизационной задачи: оптимизации процесса, структурной и параметрической идентификации, решения систем линейных и нелинейных уравнений, и основное требование к ним состоит в том, чтобы их изменение оказывало заметное влияние на перечисленные целевые функции, как, например, (3.140)–(3.142), в особенности в окрестностях их возможных экстремальных значений. В противном случае найденные оптимальные значения ресурсов оптимизации не будут однозначно характеризовать оптимальное решение. Это может привести к получению физически необоснованных и плохо определенных решений, что сведет на нет решение оптимизационной задачи.

Для оптимизационных задач первого типа целевой функцией будет технологический, экономический, эксергетический и т. п. критерий оптимальности. В этом случае, как правило, решается задача оптимизации с ограничениями первого и второго рода и определяется максимум или минимум целевой функции в зависимости от физического смысла критерия оптимальности.

В задачах второго типа с критериями вида (3.140)–(3.142) оптимизирующие параметры — это коэффициенты системы уравнений математического описания процесса, и решается задача параметрической идентификации. В случае более сложной задачи структурной идентификации количественное представление оптимизирующих переменных в виде чисел невозможно, так как необходимо определить оптимальный вид уравнений, которые адекватно описывают процесс. В этом случае решается задача минимизации критериев вида (3.140)–(3.142), и могут присутствовать ограничения первого и второго рода, вытекающие из физического смысла решаемой задачи.

В третьем типе оптимизационных задач ресурсами оптимизации — оптимизирующими переменными — являются определяемые переменные системы уравнений (3.143) и необходимо найти минимальное значение критериев вида (3.140)–(3.142) с перечисленными выше модификациями (раздел 3.5.1.1). При этом также могут присутствовать ограничения первого и второго рода.

3.5.1.3. Стандартные решатели MATLAB для решения оптимизационных задач “fminbnd”, “fminsearch”, “fmincon”, “polyfit”, “lsqcurvefit”

Выбор стандартных решателей оптимизационных задач зависит от типа решаемой задачи, ее физического смысла, рельефа целевой функции (критерия оптимальности), свойств ресурсов оптимизации (оптимизирующих переменных) и наличия ограничений первого и второго рода.

Процедура выбора решателей будет продемонстрирована на примере стандартных решателей — “fminbnd”, “polyfit”, “lsqcurvefit”, “fminsearch”, “fmincon”, первый из которых может применяться исключительно для решения задач одномерной оптимизации, второй и третий — для решения задач многомерной параметрической идентификации для процессов, зависящих от одной влияющей на их состояние переменной, а два последних — для решения задач как одномерной, так и многомерной оптимизации.

Каждый из перечисленных решателей может включать в себя один или несколько алгоритмов оптимизации, которые при корректном задании начального приближения для расчетов ресурсов оптимизации (предварительной оценки оптимальных значений оптимизирующих переменных) позволяют методом последовательных приближений (итераций) найти искомые оптимальные величины оптимизирующих переменных.

Поэтому важнейшим предварительным этапом решения оптимизационных задач является исследование поведения целевой функции и задание физически обоснованного начального приближения для итерационных расчетов в виде либо некоторого стартового значения искомой переменной или переменных, либо интервала поиска искомых переменных (ограничения первого рода).

В общем случае такая задача носит название инициализации при решении оптимизационной задачи и требует тщательного анализа характера целевой функции, физического смысла задачи и возможного поведения оптимизирующих переменных.

На следующем этапе осуществляется выбор стандартного решателя MATLAB, который может быть реализован на компьютере путем задания следующих основных параметров:

- название решателя (в нашем случае одного из пяти — “fminbnd”, “fminsearch”, “fmincon”, “polyfit”, “lsqcurvefit”);
- такого вида целевой функции, чтобы возможно было рассчитать ее значение при различных значениях оптимизирующих переменных, генерируемых решателем MATLAB в процессе итерационных расчетов;
- начального приближения оптимизирующих переменных для вычислений и/или интервала итерационного поиска оптимизирующих переменных.

В некоторых случаях, в частности при решении задач нелинейного программирования, необходимо дополнительно задать:

- систему линейных и нелинейных ограничений, накладываемых на параметры решаемой задачи;
- ограничения в виде системы нелинейных неравенств и равенств;
- ограничения в виде системы линейных неравенств и равенств.

Для контроля за ходом вычислительного процесса и управления им могут задаваться параметры стандартной функции “optimset” или “optimoptions” с указанием вывода в Командное окно MATLAB последовательных итераций с значениями оптимизирующих переменных и соответствующих им значений целевых функций, а также возможно задание точности вычислений, отличающееся от задаваемого по умолчанию.

В то время как решатели “fminbnd”, “fminsearch” и “fmincon” в общем случае могут применяться для решения всех трех типов оптимизационных задач, решатели “polyfit” и “lsqcurvefit” используются исключительно для решения задач второго типа — параметрической идентификации, аппроксимирующей функции с одной независимой переменной x вида $y = f(x)$. При этом в случае решателя “polyfit” аппроксимирующая функция представляет собой многочлен (полином) произвольной степени, которую надо задать для решения, а алгоритм определения коэффициентов многочлена сводится к известной процедуре решения систем линейных уравнений (раздел 3.3.1). С использованием решателя “lsqcurvefit” можно определить коэффициенты произвольной аппроксимирующей функции $f(x)$, что важно, нелинейной. В этом случае реализуется итерационный алгоритм поиска минимума целевой функции (3.140) с весовыми коэффициентами (w_i, $i = 1, \ldots, N$), равными единице.

3.5.2. Решение задач одномерной оптимизации

Основным решателем, используемым для решения задач одномерной оптимизации, является “fminbnd”, но могут применяться и решатели “fminsearch” и “fmincon”, предназначенные для решения задач многомерной оптимизации.

Для иллюстрации работы перечисленных решателей используется функция (рис. 3.95):

$$y = x^4 - 0.5x^3 - 23x^2 + 140 \qquad (3.144)$$

с двумя минимумами и одним максимумом внутри интервала изменения независимой переменной x (–6, 6) — незамкнутого интервала. В случае рассмотрения замкнутого интервала по x — [–6, 6], наибольшее значение функции (глобальный максимум) будет соответствовать значению x = –6 (рис. 3.95). Код программы построения графика этой функции приведен на рисунке 3.100.

```
function Grafik_ODNOM
%Построить график функции y=x^4-0.5*x^3-28*x^2+140 в интервале [-6;+6];
clc;
clear al;
close all;
a=-6;
b=6;
h=(b-a)/100;
x=a:h:b;
y=x.^4-0.5*x.^3-28*x.^2+140;
plot(x,y);
title('График функции y=x^4-0.5*x^3-28*x^2+140 : с 1-им максимумом и 2-мя ми-
нимумами');
xlabel('x');ylabel('y');
text(2,0,'3.Минимум — глобальный');
text(-5,0,'1.Минимум — локальный');
text(0,160,'Максимум');
grid on;
end
```

Рис. 3.100

Программный код файла построения графика функции $y = x^4 - 0.5x^3 - 23x^2 + 140$

3.5.2.1. Применение решателя “fminbnd”

Решатель “fminbnd” применяется исключительно для решения задач одномерной оптимизации. Целевая функция (3.144) в простейшем случае не обязательно должна быть оформлена в виде отдельного *m*-файла f.m (рис. 3.101), тогда решатель “fminbnd” для определения глобального минимума $y_{\min}$ в незамкнутом интервале (–6, 6) записывается следующим образом:

$$[x\text{opt}, y\text{min}] = \text{fminbnd}(@f', -6,6), \tag{3.145}$$

а выражение для функции *f* (3.144) можно также непосредственно записать в конструкцию решателя (рис. 3.102):

$$[x\text{opt}, y\text{min}] = \text{fminbnd}(‘x\text{^}4 - 0.5x\text{^}3 - 28x\text{^}2 + 140’, 0, 6). \tag{3.146}$$

```
function y=f(x)
y=x^4-0.5*x^3-28*x^2+140;
end
```

Рис. 3.101

Программный код *m*-файла целевой функции при решении задачи одномерной оптимизации

```
function min_odnom_local_2=global_bnd
%Вид функции представлен в решателе fminbnd
clc;
clear al;
option=optimset('Display','iter','TolX',1.e-4);
[xopt,ymin]=fminbnd('x^4-0.5*x^3-28*x^2+140',-6,6,option)
end
```

Рис. 3.102

Программный код файла решения задачи оптимизации с решателем “fminbnd” и целевой функцией, записанной непосредственно в конструкцию решателя

В выражение для решателя “fminbnd”, как следует из программного кода (рис. 3.102), может быть включена стандартная функция optimset для отслеживания хода выполнения вычислительного процесса при решении оптимизационной задачи и для задания точности вычислений по аргументу *x*:

$$\begin{aligned} &\text{option} = \text{optimset}(‘\text{Display}’,\ ‘\text{iter}’,\ ‘\text{TolX}’,\ 1\cdot e^{-4};\\ &[x\text{opt}, y\text{min}] = \text{fminbnd}(x^4 - 0.5\cdot x^3 - 28\cdot x^2 + 140,\ 0,\ 6,\ \text{option}), \end{aligned} \tag{3.147}$$

где ‘Display’, ‘iter’ — означает вывод в Командное окно MATLAB изменения значений аргументов и целевых функций на каждой итерации; ‘TolX’, $1\cdot e^{-4}$ — задание точности окончания итерационного процесса по аргументу *x*, в данном случае она соответствует точности вычислений по умолчанию.

Результаты решения оптимизационной задачи по программе (рис. 3.102) с информацией о последовательных итерациях вычислительного процесса представлены на рисунке 3.103. Программные коды *m*-файлов для определения локальных минимумов 1 и 2 (рис. 3.95) представлены соответственно на рисунках 3.104 и 3.105 с ограниченными незамкнутыми интервалами по аргументу *x* — (–6, 0) и (0, 6). Результат расчета минимума 1:

$$xopt = -3.5589,$$
$$ymin = -31.6817, \tag{3.148}$$

а минимума 2, как и следовало ожидать, совпадает с данными глобального минимума (рис. 3.103).

```
Func-count    x          f(x)         Procedure
     1      -1.41641     89.2718      initial
     2       1.41641     86.4302      golden
     3       3.16718    -56.1326      golden
     4       4.24922    -77.9114      golden
     5       4.17327    -80.6709      parabolic
     6       3.88754    -84.1374      parabolic
     7       3.61239    -78.6659      golden
     8       3.92414    -84.2568      parabolic
     9       3.93223    -84.2622      parabolic
    10       3.93392    -84.2624      parabolic
    11       3.93385    -84.2624      parabolic
    12       3.93382    -84.2624      parabolic
    13       3.93389    -84.2624      parabolic

Optimization terminated:
 the current x satisfies the termination criteria using OPTIONS.TolX of 1.000000e-04

xopt =  3.9339

ymin = -84.2624
```

Рис. 3.103

Результаты определения глобального минимума функции (3.144) по программе (рис. 3.102)

```
function min_odnom_local_1_bnd
%Вид функции представлен в решателе fminbnd
option=optimset('Display','iter','TolX',1.e-4);
[xopt,ymin]=fminbnd('x^4-0.5*x^3-28*x^2+140',-6,0,option)
end
```

Рис. 3.104

Программный код файла для определения локального минимума функции (3.144) (рис. 3.95) внутри интервала [–6, 0]

```
function min_odnom_local_2_bnd
%Вид функции представлен в решателе fminbnd
clc;
clear al;
option=optimset('Display','iter','TolX',1.e-4);
[xopt,ymin]=fminbnd('x^4-0.5*x^3-28*x^2+140',0,6,option)
end
```

Рис. 3.105

Программный код файла для определения локального минимума функции (3.144) (рис. 3.95) внутри интервала [0, 6]

При определении минимума целевой функции (рис. 3.95) ее знак в программном коде соответствующего файла (рис. 3.106) должен быть изменен на противоположный, и, соответственно, знак получаемой оптимальной функции будет противоположным знаку максимального значения целевой функции. В результате знак максимума целевой функции будет противоположным знаку оптимальной функции, полученной решателем “fminbnd”.

```
function max_odnom_local_bnd
%Вид функции представлен в решателе fminbnd
clc;
clear al;
option=optimset('Display','iter','TolX',1e-4);
[xopt,y]=fminbnd('-(x^4-0.5*x^3-28*x^2+140)',-6,6,option)
ymax=-y
end
```

Рис. 3.106

Программный код файла для определения локального максимума функции (3.144) (рис. 3.95) внутри интервала [–6, 6]

Необходимость изменения знака целевой функции в программном коде MATLAB при поиске минимумов связана с тем обстоятельством, что алгоритм поиска экстремума “fminbnd”, как в равной степени и все другие решатели поиска экстремумов MATLAB, обеспечивает нахождение минимума целевой функции.

Результат определения максимума целевой функции (3.144) по программе (рис. 3.106) представлен на рисунке 3.107.

```
 Func-count     x          f(x)         Procedure
     1      -1.41641     -89.2718       initial
     2       1.41641     -86.4302       golden
     3      -3.16718      24.3624       golden
     4     -0.0348806     -139.966        parabolic
     5      -0.019309     -139.99        parabolic
     6      0.00301089      -140        parabolic
     7      5.06432e-06     -140        parabolic
     8      -3.8269e-05     -140       parabolic
     9      3.83977e-05     -140        parabolic

Optimization terminated:
 the current x satisfies the termination criteria using OPTIONS.TolX of 1.000000e-04

xopt =  5.0643e-06

y = -140.0000

ymax =  140.0000
```

Рис. 3.107

Результаты определения максимума целевой функции (3.144) (рис. 3.95) по программе (рис. 3.106)

3.5.2.2. Применение решателя “fminsearch”

Решатель “fminsearch” может применяться для решения задач как одномерной, так и многомерной оптимизации. Его отличительная особенность состоит в том, что он требует задания только вида целевой функции и начального приближения итерационных расчетов, т. е. для получения достоверного решения необходима корректная оценка искомого решения — правильное задание начального приближения в итерационном процессе. По существу, речь идет об инициализации итерационных расчетов оптимизационной задачи, что возможно как на основании знания физического смысла предполагаемого решения, так и специальных процедур инициализации — в простейшем случае анализа ха-

рактера изменения целевой функции в зависимости от оптимизирующей переменной (аргумента).

Очевидно, что этот решатель мало пригоден для определения глобального экстремума, и в зависимости от задания тех или иных начальных приближений (оценок) параметров решения могут получаться различные локальные экстремумы.

Для одномерной функции (3.144) на рисунках 3.108 и 3.109 приведены коды программ с решателем “fminsearch” с различными начальными приближениями: –6 и +6 соответственно. Как и следовало ожидать, при запуске программы для приближения –6 получается результат, соответствующий локальному минимуму (3.148), а при начальном приближении 6 — глобальному минимуму (рис. 3.103).

```
function min_odnom_local_1
clc;
clear al;
%3.1. Вычисление первого локального минимума 1 в интервале [-6,+6] с решателем fminsearch;
%Вид функции y=x^4-0.5*x^3-28*x^2+140 задается в решателе fminsearch
option=optimset('Display','iter','TolX',1.e-4);
[xopt,ymin]=fminsearch('x^4-0.5*x^3-28*x^2+140',-6,option)
end
```

Рис. 3.108

Программный код файла для определения минимума функции (3.144) (рис. 3.95) с решателем “fminsearch” и с начальным приближением –6

```
function min_odnom_local_2
clc;
clear al;
%Вычисление второго локального минимума 2 (глобального минимума)
%в интервале [-6,+6] с решателем fminsearch;
%Вид функции y=x^4-0.5*x^3-28*x^2+140 задается в решателе fminsearch
option=optimset('Display','iter','TolX',1.e-4);
[xopt,ymin]=fminsearch('x^4-0.5*x^3-28*x^2+140',6,option)
end
```

Рис. 3.109

Программный код файла для определения минимума функции (3.144) (рис. 3.95) с решателем “fminsearch” и с начальным приближением 6

В коде программы (рис. 3.110) для определения максимума целевой функции (3.144) — рисунок 3.95, как было показано выше, необходимо изменить ее знак и обращение к решателю “fminsearch”:

$$[x\text{opt}, y] = \text{fminsearch}('-(x^4 - 0.5 \cdot x^3 - 28 \cdot x^2 + 140)',\ 3.83)$$
$$y\text{max} = -y. \tag{3.149}$$

```
function max_odnom_local_search
%Вид функции задается в решателе fminsearch
[xopt,y]=fminsearch('-(x^4-0.5*x^3-28*x^2+140)',3.83)
ymax=-y
end
```

Рис. 3.110

Программный код файла для определения максимума функции (3.144) (рис. 3.95) с решателем “fminsearch”

В результате получаются оптимальные параметры целевой функции, аналогичные результатам, приведенным на рисунке 3.107.

Важнейшим условием применения этого решателя является обоснованное задание начального приближения итерационных расчетов оптимизационной задачи.

3.5.2.3. Применение решателя “fmincon”

Решатель “fmincon”, как и решатель “fminsearch”, используется для решения одномерных и многомерных задач оптимизации.

Для решения одномерных задач оптимизации он может быть записан в следующем укороченном виде:

$$[x\text{opt}, y, f] = fmincon(‘func’, x0, xa, xb), \qquad (3.150)$$

где func — целевая функция, записанная в явном виде (как и для решателей “fminbnd” и “fminsearch”), или ее название, когда она задается отдельной функцтей (рис. 3.101); $x0$ — начальное приближение итерационных расчетов в замкнутом интервале $[xa, xb]$; xa — левая граница интервала поиска оптимального решения; xb — правая граница интервала поиска оптимального решения.

В квадратных скобках слева от решателя “fmincon” записывается результат решения оптимизационной задачи:

$x$opt — оптимальное значение аргумента x;

y — оптимальное значение целевой функции;

f — числовая характеристика вычислительного процесса:

f
- > 0 — решение найдено с требуемой точностью;
- $= 0$ — достигнуто максимальное число итераций;
- < 0 — решение неудовлетворительное.

Такие же параметры могут задаваться и для ранее рассмотренных решателей “fminbnd” и “fminsearch”. Особенность решателя “fmincon” состоит в том, что можно одновременно задавать как начальное приближение итерационных расчетов, так и границы интервала поиска $[xa, xb]$. Следует отметить, что можно обойтись и без задания границ поиска экстремума — xa, xb.

Для определения первого локального минимума функции (3.144) в интервале [–6, 6] задается начальное приближение (–1), а второго — начальное приближение (+1). Коды программ приведены соответственно на рисунках 3.111 и 3.112.

```
function min_odnom_global_con1
clc;%Вид функции задается в решателе
fmincon
clear al;
[xopt,y,f]=fmincon('(x^4-0.5*x^3-
28*x^2+140)',-1,-6,6)
end
```

Рис. 3.111

Программный код файла для определения минимума функции (3.144) (рис. 3.95) с решателем “fmincon” и с начальным приближением –1

```
function min_odnom_global_con
clc;%Вид функции представлен в решателе
fmincon
clear al;
[xopt,y,f]=fmincon('(x^4-0.5*x^3-
28*x^2+140)',1,-6,6)
end
```

Рис. 3.112

Программный код файла для определения минимума функции (3.144) (рис. 3.95) с решателем “fmincon” и с начальным приближением 1

Код программы для определения максимума функции (3.144) с начальным приближением (1) приведен на рисунке 3.113. Следует отметить, что при задании начального приближения для расчетов (–5) получается неудовлетворительный результат.

```
function max_odnom_local_con0
clc;%Вид функции представлен в решателе
fmincon`
clear al;
[xopt,y,f]=fmincon('-(x^4-0.5*x^3-
28*x^2+140)',1,-6,6)
ymax=-y
end
```

Рис. 3.113

Программный код файла для определения максимума функции (3.144) (рис. 3.95) с решателем “fmincon”

3.5.3. Решение задач многомерной оптимизации

Разработано множество решателей задач многомерной оптимизации, которые используются в среде MATLAB. Часть из них поставляется со стандартным оптимизационным пакетом, а остальные (и их большинство) приходится дополнительно приобретать у фирм-дистрибьютеров или официально скачивать из Интернета. Они отличаются как используемыми в решателе алгоритмами оптимизации, возможностями варьирования различных алгоритмов и их параметрами, так и применимостью к решению разнотипных задач, как, например, задач оптимизации процессов химической технологии, идентификации параметров математических моделей, задач решения систем уравнений математического описания химико-технологических процессов.

В настоящем разделе будут представлены два решателя из стандартного оптимизационного пакета MATLAB — “fminsearch” и “fmincon”, а также два решателя — “polyfit” и “lsqcurvefit” — для решения задач параметрической идентификации функций с одной независимой переменной. Для первых двух решателей в этом случае определяется не одно, а несколько значений переменных процесса, а для двух последних решателей — несколько значений коэффициентов уравнений с одной независимой переменной.

При применении первого решателя — “fminsearch” — для определения оптимальных значений оптимизирующих переменных (ресурсов оптимизации) достаточно задать их начальное значение в виде вектора $\bar{x} = [x_1, x_2, \ldots, x_n]$, компоненты которого будут изменяться в соответствии с алгоритмом оптимизации, используемом в решателе. Применение этого решателя позволяет не ограничивать область изменения оптимизирующих переменных $\bar{x}$.

В этом случае оптимизационная задача формулируется в следующем виде:

$$\min_{\bar{x}} R(\bar{x}), \tag{3.151}$$

т. е. необходимо найти оптимальные значения оптимизирующих переменных $\bar{x}$ — элементы вектора $\bar{x}^{onm} = \left[x_1^{onm}, x_2^{onm}, \ldots, x_n^{onm} \right]$, которые при подстановке в целевую функцию $R(x)$ обеспечивают ее минимальное значение $R^{min}\left(\bar{x}^{onm}\right)$.

Второй решатель — “fmincon” — допускает ограничение нижнего и верхнего предела изменения оптимизирующих переменных. Одновременно он также требует задания начального приближения итерационных расчетов $\bar{x}$ внутри задаваемого замкнутого интервала изменения оптимизационных переменных. В этом случае при задании нижних ($\bar{x}^{min}$) и верхних ($\bar{x}^{max}$) границ изменения оптимизирующих переменных ($\bar{x}$) принято считать, что задача оптимизации сформулирована с ограничениями первого рода:

$$\min_{\bar{x}^{\min} \le \bar{x} \le \bar{x}^{\max}} R(\bar{x}). \tag{3.152}$$

Задача оптимизации в этом случае формулируется так же, как и для (3.151), с той лишь разницей, что на оптимизирующие переменные $\bar{x}$ наложены ограничения первого рода (3.152).

Следует отметить, что выбор того или иного решателя зависит от типа решаемой задачи, в том числе и от того, стоит ли ограничивать область поиска оптимального решения $\bar{x}^{onm}$ замкнутым интервалом $[\bar{x}^{min}, \bar{x}^{max}]$, т. е. ограничениями первого рода.

Применение решателя “polyfit” ограничено исключительно определением коэффициентов многочленов (полиномов) произвольной степени по известному массиву данных, чаще всего экспериментальных — задача параметрической идентификации. Этот решатель может использоваться также для определения параметров функций с одной независимой переменной, которые путем определенных алгебраических преобразований могут быть представлены в виде многочлена (полинома) произвольной степени. Следует отметить, что алгоритм решателя “polyfit” реализует безытерационный процесс решения системы линейных алгебраических уравнений.

Решатель “lsqcurvefit” позволяет определять параметры произвольной функции с одной независимой переменной по массиву известных данных с алгоритмом поиска минимума, используемого, например, в решателе “fminsearch”. При этом используется целевая функция (3.140) с весовыми коэффициентами (w_i, $i = 1, \ldots, N$), равными единице.

3.5.3.1. Решение многомерных оптимизационных задач без ограничений на оптимизирующие переменные с применением решателей “fminsearch” и “fmincon”

В соответствии с формулировкой задачи (3.151) при записи решателей “fminsearch” и “fmincon” не надо задавать отрезки для оптимизирующих переменных с нижними ($\overline{x}^{min}$) и верхними ($\overline{x}^{max}$) границами, в которых осуществляется поиск их оптимальных значений. В этом случае следует задавать только начальные стартовые значения (приближения) оптимизирующих переменных. Определение этих приближений принято называть инициализацией процедуры решения, и чем оно удачнее, тем больше шанс получить корректное решение $\overline{x}^{onm}$.

Существуют различные методы инициализации задачи, т. е. определение оценок (приближений) искомых оптимальных значений $\overline{x}^{onm}$. Чаще всего это приближенное знание параметров решения исходя из физического смысла задачи и возможного ее упрощенного решения, а также на основе исследования характера изменения целевой функции. В любом случае процесс инициализации решения следует считать важным этапом решения оптимизационной задачи, предваряющим итерационный поиск оптимального решения.

3.5.3.1.1. Применение решателя “fminsearch”

В этом решателе реализован модифицированный алгоритм Нелдера — Мида для поиска минимума целевой функции, и в коде программы *m*-языка MATLAB он в краткой форме записывается в следующем виде:

$$[xopt, ymin, pr, out] = fminsearch('function', x0), \qquad (3.153)$$

где справа от знака равенства в скобках указаны задаваемые значения: $x0$ — начальное приближение вектора оптимизирующих переменных $\overline{x}$ (размерность этого вектора определяет в решателе число поисковых переменных; ‘*function*’ — название *m*-файла function.m целевой функции или целевая функция в явном виде (3.102), минимум которой ищется.

Слева от знака равенства в квадратных скобках указаны результаты расчетов: *xopt* — вектор рассчитанных оптимальных значений оптимизирующих переменных $\overline{x}$; *ymin* — минимальное значение целевой функции (3.151), вид которой задается в *m*-файле с именем function.m; *pr* — признак корректности полученного результата; *out* — некоторые параметры вычислительного процесса, как, например, название алгоритма оптимизации, число выполненных в процессе расчетов итераций при поиске решения и т. п.

При необходимости более тщательно контролировать вычислительный процесс, реализуемый решателем “fminsearch”, а также регулировать параметры вычислений, например точность, решатель записывается в полной форме (3.155) с функцией optimset — option = optimset (3.154):

$$\begin{aligned} &\text{option} = \text{optimset}(\text{‘Display’}, \text{‘iter’}, \text{‘TolX’},\\ &1e-4, \text{‘TolFun’}, 1e-4, \text{‘MaxIter’}, 2500, \qquad (3.154)\\ &\qquad\qquad \text{‘MaxFunEvals’}, 1500); \end{aligned}$$

$$[\bar{x}opt, ymin, pr, out] = fminsearch('function', \bar{x}0, option), \qquad (3.155)$$

где функции в круглых скобках: ‘Display’, ‘iter’ — вывод в Командное окно MATLAB параметров вычислительного процесса на каждой итерации; ‘TolX’ — задаваемая числом без кавычек сразу после этой символьной переменной точность вычислений по оптимизирующей переменной $\bar{x}$ (аргументу); ‘TolFun’ — задаваемая числом без кавычек сразу после этой переменной точность вычислений по целевой функции; ‘MaxIter’ — задаваемое числом без кавычек сразу после этой переменной максимально допустимое число итераций; ‘MaxFunEvals’ — задаваемое числом без кавычек сразу после этой переменной максимально допустимое число вычислений целевой функции.

Применение решателя “fminsearch” рассматривается на примере простейшей целевой функции с двумя аргументами:

$$z = x^2 + (x - y)^3. \qquad (3.156)$$

Программный код для объемного графического представления приведен на рисунке 3.114, а результат его реализации — на рисунке 3.115.

```
function Grafik_MNOGO_6
clc;
clear all;
close all;
[x,y]=meshgrid(-1.1:0.1:1.1,-1.5:0.1:1.5);
z=x.^2+(x-y).^2;
meshc(x,y,z);
title('График функции z=x^2+(x-y)^2');
xlabel('x');
ylabel('y') ;
zlabel('z');
end
```

Рис. 3.114

Программный код файла объемного графического представления функции (3.156)

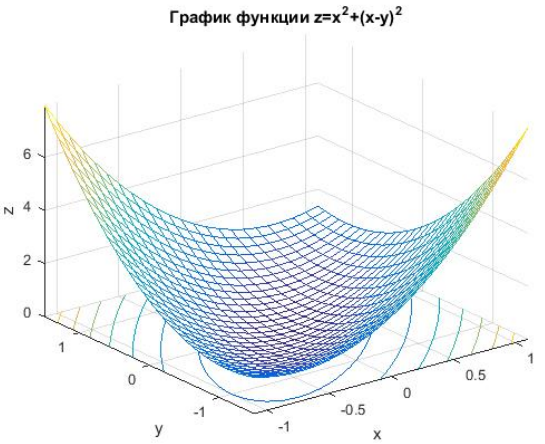


Рис. 3.115

Объемное графическое представление функции (3.156), полученное по программе, приведенной на рисунке 3.114

Программный код для плоского графического представления функции (3.156) с линиями постоянного уровня приведен на рисунке 3.116, результат его реализации — на рисунке 3.117.

```
%Построение овражного типа рельефа
clc;
clear al;
close all;
[x,y]=meshgrid(-1.1:0.1:1.1,-1.5:0.1:1.5);
z=x.^2+(x-y).^2;
v=0:0.2:1;
contour(x,y,z,v);
title(' График овражной функции z=x^2+(x-y)^2 ');
xlabel('x');
ylabel('y');
```

Рис. 3.116

Программный код файла для плоского графического представления функции (3.156) с линиями постоянного уровня (*v*)

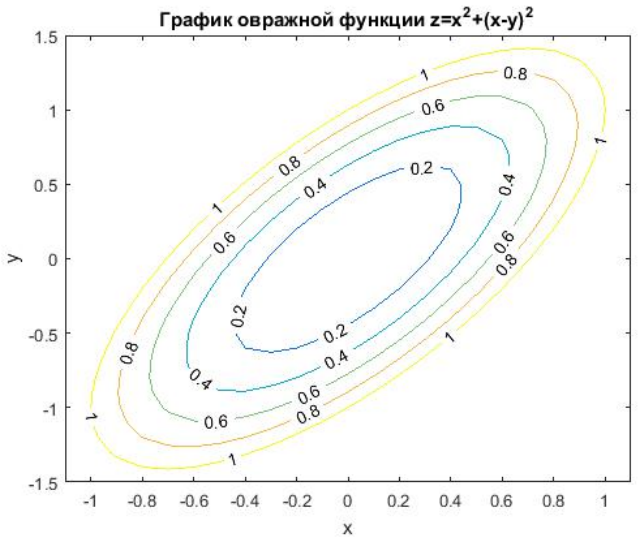


Рис. 3.117

Плоское графическое представление функции (3.156) с линиями постоянного уровня, полученное по программе, приведенной на рисунке 3.116

Результаты графического представления целевой функции (3.156), которая является овражной, могут быть использованы для задания начального приближения итерационных вычислений, т. е. решения задачи инициализации для поиска решения оптимизационной задачи.

Программный код процедуры определения оптимальных значений x и y, обеспечивающих оптимальное значение целевой функции z, приведен на рисунке 3.118. Сама целевая функция (3.156) представлена отдельным файлом extr26.m на рисунке 3.119. Оптимизирующие переменные при этом рассматриваются как элементы вектора $x = [x(1), x(2)]$, где $x(1) = x$ и $x(2) = y$ (3.156), а $z6 = z$ (рис. 3.118 и 3.119).

```
%Требует функцию extr26
clc;
clear al;
%---------------------------------------
%disp('1.ПРИМЕНЕНИЕ РЕШАТЕЛЯ fminsearch');
%---------------------------------------
%disp('1.1. Краткая форма');
%x=fminsearch(@extr26,[0.5,0.5])
%[x,y,pr,out]=fminsearch(@extr26,[0.5,0.5])
%---------------------------------------
%disp('1.3. Полная форма');
%option=optimset('Display','iter','TolX',1.0e-4,'TolFun',1.0e-4,'MaxIter',250,'MaxFunEvals',1500);
%[x,y,pr,out]=fminsearch(@extr26,[3,-3],option)
%---------------------------------------
disp('3.ПРИМЕНЕНИЕ РЕШАТЕЛЯ fmincon');
%---------------------------------------
%disp('3.1. Краткая форма');
%[x,y,pr,out]=fmincon(@extr26,[3,-3])
%---------------------------------------
disp('3.3. Полная форма');
options = optimoptions('fmincon','Algorithm','active-set','Display','iter',...
   'TolX',1.0e-8,'MaxIter',1000,'MaxFunEvals',2500)
[x,y,pr,out]=fmincon(@extr26,[0.5,0.5],[],[],[],[],[-1,-2],[5,2],@nonlincon,options)
%---------------------------------------
```

Рис. 3.118

Программный код файла определения минимума функции (3.156) с применением решателей “fmincon” и “fminsearch”

```
function z6=extr26(x)
z6=x(1)^2+(x(1)-x(2))^2;
end
```

Рис. 3.119

Программный код целевой функции (3.156), используемой для решения задачи оптимизации программой, приведенной на рисунке 3.118

Как следует из программного кода (рис. 3.118 — краткая форма закомментирована), допускается вывод в качестве результатов только оптимальных значений оптимизирующих переменных:

$$x = fminsearch(\text{extr26}, [0.5, 0.5]). \qquad (3.157)$$

Вывод всех результатов для краткой формы записи “fminsearch” (3.153) для целевой функции (3.156) в соответствии с программными кодами (рис. 3.118 и 3.119) приведен на рисунке 3.120, а вывод результатов по полной форме записи “fminsearch” — на рисунке 3.121. Различие числа выполненных итераций (рис. 3.120 и 3.121) связано с тем, что задавались различные начальные приближения (рис. 3.118) для выполнения итерационных расчетов.

```
1.ПРИМЕНЕНИЕ РЕШАТЕЛЯ fminsearch
1.1. Краткая форма

x = 1.0e-04 *

   -0.0606   -0.3927

y = 1.1394e-09

pr = 1
```

Рис. 3.120 (начало)

Результаты определения минимума функции (3.156) с применением краткой формы решателя “fminsearch” с начальными приближениями $x(1) = 0.5$ и $x(2) = 0.5$ по программе, приведенной на рисунке 3.119

```
out =

   iterations: 38
    funcCount: 75
    algorithm: 'Nelder-Mead simplex direct search'
      message: 'Optimization terminated:
 the current x satisfies the termination cri...'
>>
```

Рис. 3.120 (окончание)

Результаты определения минимума функции (3.156) с применением краткой формы решателя “fminsearch” с начальными приближениями $x(1) = 0.5$ и $x(2) = 0.5$ по программе, приведенной на рисунке 3.119

```
1.3. Полная форма

 Iteration  Func-count     min f(x)         Procedure
     0          1            45
     1          3            45          initial simplex
     2          5          43.3956         expand
     3          7          37.7227         expand
     4          9          31.2754         expand
     5         11          21.1951         expand
     6         13          11.6083         expand
     7         15          7.62084         reflect
     8         16          7.62084         reflect
     9         18          7.62084         contract inside
    10         20          7.62084         contract inside
    11         22          6.93164         expand
    12         24          4.82801         expand
    13         25          4.82801         reflect
    14         27          1.81408         expand
    15         29          1.81408         contract inside
    16         31          0.232506        expand
    17         32          0.232506        reflect
    18         34          0.232506        contract inside
    19         36          0.104463        contract inside
    20         38          0.0349432       contract inside
    21         40          0.0349432       contract inside
    22         42          0.0250967       reflect
    23         44          0.00435554      contract inside
    24         46          0.00435554      contract inside
    25         48          0.00435554      contract inside
    26         50          0.000846813     contract inside
    27         52          0.000341676     contract inside
    28         54          0.000341676     contract inside
    29         56          0.000177603     contract inside
    30         58          8.09107e-05     contract inside
    31         60          5.21006e-05     contract inside
    32         62          1.43902e-05     contract outside
    33         64          9.49323e-06     contract inside
    34         66          5.62909e-06     contract inside
    35         68          3.01259e-06     contract inside
    36         70          1.28599e-06     contract inside
    37         72          4.87712e-07     contract inside
    38         74          4.32305e-07     contract outside
    39         76          1.61648e-07     contract inside
    40         78          8.12062e-08     contract inside
    41         80          3.58611e-08     contract outside
    42         82          3.75603e-08     contract inside
    43         84          4.96227e-09     contract inside
    44         86          4.96227e-09     contract inside
    45         88          4.96227e-09     contract outside
    46         90          7.89746e-10     contract inside
    47         91          7.89746e-10     reflect
```

Рис. 3.121 (начало)

Результаты определения минимума функции (3.156) с применением полной формы решателя “fminsearch” с начальными приближениями $x(1) = 3$ и $x(2) = –3$ по программе, приведенной на рисунках 3.118 и 3.119

```
    48        93     7.89746e-10       contract inside

Optimization terminated:
 the current x satisfies the termination criteria using OPTIONS.TolX of 1.000000e-04
 and F(X) satisfies the convergence criteria using OPTIONS.TolFun of 1.000000e-04


x =
   1.0e-04 *

   -0.0989   -0.3620

y = 7.8975e-10

pr = 1

out =
    iterations: 48
     funcCount: 93
     algorithm: 'Nelder-Mead simplex direct search'
       message: 'Optimization terminated:
 the current x satisfies the termination ...'
>>
```

Рис. 3.121 (окончание)

Результаты определения минимума функции (3.156) с применением полной формы решателя “fminsearch” с начальными приближениями $x(1) = 3$ и $x(2) = –3$ по программе, приведенной на рисунках 3.118 и 3.119

3.5.3.1.2. Применение решателя “fmincon”

Краткая форма записи решателя “fmincon” с применением программы, приведенной на рисунках 3.118 и 3.119, позволяет найти решение оптимизационной задачи исходя только из начальных приближений итерационного процесса (3.153):

$$[xopt, ymin, pr, out] = fmincon('function', x0). \qquad (3.158)$$

Этот решатель использует для оптимизации алгоритм **interior-point** (по умолчанию), который отличен от алгоритма, реализованного в решателе “fminsearch”. Поэтому результаты расчетов, представляемые в квадратных скобках (3.158), могут отличаться.

Так, например, для рассматриваемого примера (3.156) получены отличные от предыдущих расчетов результаты параметров вычислительного процесса (рис. 3.122). С рассмотренным решателем получено решение за 6 итераций и при 23 расчетах значений целевой функции. С использованием решателя “fminsearch” — за 38 итераций и при 75 расчетах целевой функции (рис. 3.120).

```
3.ПРИМЕНЕНИЕ РЕШАТЕЛЯ fmincon
3.1. Краткая форма

Local minimum found that satisfies the constraints.

Optimization completed because the objective function is non-decreasing in
feasible directions, to within the default value of the function tolerance,
and constraints are satisfied to within the default value of the constraint tolerance.

<stopping criteria details>
```

Рис. 3.122 (начало)

Результаты определения минимума функции (3.156) с применением краткой формы записи решателя “fmincon” с начальными приближениями $x(1) = 3$ и $x(2) = –3$ по программе, приведенной на рисунке 3.118

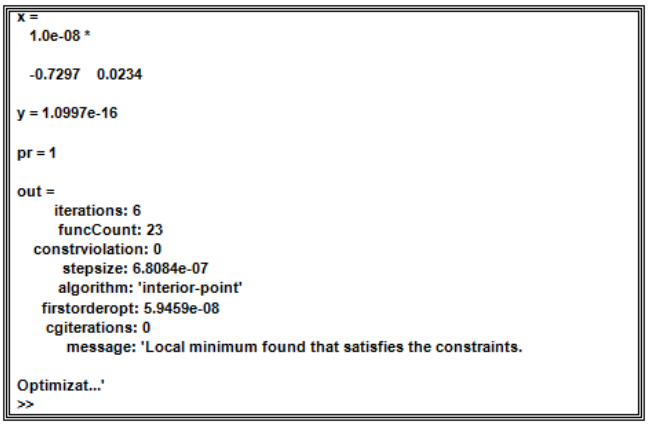


Рис. 3.122

Результаты определения минимума функции (3.156) с применением краткой формы записи решателя “fmincon” с начальными приближениями $x(1) = 3$ и $x(2) = –3$ по программе, приведенной на рисунке 3.118

Следует отметить, что при использовании решателя “fmincon” в краткой форме записи (3.158) для решения оптимизационных задач без ограничений на оптимизирующие переменные нельзя воспользоваться целым рядом его достоинств, относящихся к регулированию параметров вычислительного процесса.

3.5.3.2. Решение многомерной оптимизационной задачи с ограничениями первого рода на оптимизирующие переменные с применением решателя “fmincon”

Решение многомерной оптимизационной задачи с ограничениями первого рода возможно только с решателем “fmincon”. При этом запись решателя в программном коде *m*-языка MATLAB должна иметь вид расширенной формы (3.159) — см. полную форму записи решателя “fmincon” в программе на рисунке 3.118:

$$[xopt, ymin, pr, out] = fmincon(‘function’, x0, [], [], [], [], xmin, xmax), \quad (3.159)$$

где $x0$ — вектор начальных приближений для итерационных расчетов; *xmin* — вектор левых границ интервала поиска решения $x$opt; *xmax* — вектор правых границ интервала поиска решения $x$opt.

Наличие четырех пустых квадратных скобок в конструкции решателя “fmincon” связано с тем, что в общем случае этот решатель предназначен для решения задач нелинейного программирования (НЛП), при формулировке которых могут задаваться ограничения в виде **линейных неравенств** (первые две квадратные скобки — для записи в них идентификатора матрицы коэффициентов и идентификатора вектора свободных членов) и в виде **линейных равенств** (последние две квадратные скобки — для записи в них идентификатора матрицы коэффициентов и идентификатора вектора свободных членов соответственно). Так как в рассматриваемом случае этих ограничений нет, то в записи решателя стоят пустые квадратные скобки.

Для записи полной формы решателя “fmincon” с возможностью поитерационного контроля процесса сходимости решения и выбора алгоритма из четырех возможных — **interior-point** (по умолчанию), **active-set**, **sqp**, **trust-region-reflective**[3], используется функция “optimoptions”, и тогда полная форма конструкции этого решателя имеет, например, следующий вид (см. (3.154) и (3.155) — программный код файла на рис. 3.118):

$$options = optimoptions$$
$$(\text{‘}fmincon\text{’}, \text{‘Algorithm’}, \text{‘active-set’}, \text{‘Display’}, \tag{3.160}$$
$$\text{‘iter’}, \text{‘TolX’}, 1e-8, \text{‘MaxIter’}, 1000, \text{‘MaxFunEvals’}, 2500);$$

$$[xopt, ymin, pr, out] = fmincon(\text{‘}function\text{’}, x0,$$
$$[], [], [], [], xmin, xmax, \tag{3.161}$$
$$\text{‘}nonlincon\text{’}, options).$$

Как следует из (3.160), в данном случае для “fmincon” выбирается алгоритм active-set и, как обычно, задаются параметры вычислительного процесса.

Наличие в записи решателя функции nonlincon (3.101) также связано с использованием “fmincon” для решения задач нелинейного программирования (НЛП).

При этом в отдельной функции nonlincon (x) (рис. 3.123) необходимо записать ограничения второго рода в виде неравенств (g1) и равенств (g2) — один из атрибутов нелинейного программирования (НЛП). Так как здесь нет этих ограничений, то файл nonlincon.m пустой (рис. 3.123).

Результат решения оптимизационной задачи (3.161) с использованием полной формы “fmincon” (рис. 3.118) приведен на рисунке 3.124. Как видно из этого, результат решения получен с использованием алгоритма active-set за две итерации.

```
function [g1,g2] = nonlincon(x)
g1=[];
g2=[];
end
```

Рис. 3.123

Программный код ограничений второго рода при их отсутствии, когда для определения целевой функции используется полная форма решателя “fmincon”

```
3.ПРИМЕНЕНИЕ РЕШАТЕЛЯ fmincon
3.3. Полная форма

options =

 fmincon options:
```

Рис. 3.124 (начало)

Результаты определения минимума функции (3.156) с применением полной формы записи решателя “fmincon” с начальными приближениями $x_1 = 0.5$ и $x_2 = 0.5$, границами поиска $-1 \le x \le 5$, $-2 \le x \le 2$ по программе, приведенной на рисунке 3.118

[3] Подробнее об этих алгоритмах см. в описании и в справке (Help) пакета MATLAB.

```
Options used by current Algorithm ('active-set'):
(Other available algorithms: 'interior-point', 'sqp', 'trust-region-reflective')

Set by user:
          Algorithm: 'active-set'
            Display: 'iter'
        MaxFunEvals: 2500
            MaxIter: 1000
              TolX: 1.0000e-08

Default:
      DerivativeCheck: 'off'
        Diagnostics: 'off'
       DiffMaxChange: Inf
       DiffMinChange: 0
      FinDiffRelStep: 'sqrt(eps)'
        FinDiffType: 'forward'
       FunValCheck: 'off'
        GradConstr: 'off'
          GradObj: 'off'
        MaxSQPIter: '10*max(numberOfVariables,numberOfInequalities+numberOfB...'
         OutputFcn: []
          PlotFcns: []
      RelLineSrchBnd: []
 RelLineSrchBndDuration: 1
            TolCon: 1.0000e-06
         TolConSQP: 1.0000e-06
            TolFun: 1.0000e-06
         TypicalX: 'ones(numberOfVariables,1)'
       UseParallel: 0

Show options not used by current Algorithm ('active-set')


                         Max     Line search  Directional  First-order
Iter F-count       f(x)  constraint  steplength  derivative  optimality Procedure
   0     3       0.25        -1.5
   1     8      0.125       -1.25        0.25          -1         0.5
   2    11  7.57207e-16          -1           1      -0.447    5.77e-08

Local minimum found that satisfies the constraints.

Optimization completed because the objective function is non-decreasing in
feasible directions, to within the default value of the function tolerance,
and constraints are satisfied to within the default value of the constraint tolerance.

<stopping criteria details>

No active inequalities.

x =

  1.0e-07 *

  -0.2515   -0.3632

y =

  7.5721e-16


pr =

     1


out =
```

Рис. 3.124 (продолжение)

Результаты определения минимума функции (3.156) с применением полной формы записи решателя “fmincon” с начальными приближениями $x_1 = 0.5$ и $x_2 = 0.5$, границами поиска $-1 \le x \le 5$, $-2 \le x \le 2$ по программе, приведенной на рисунке 3.118

```
      iterations: 2
       funcCount: 11
   lssteplength: 1
        stepsize: 0.5590
       algorithm: 'active-set'
  firstorderopt: 5.7742e-08
 constrviolation: -1.0000
         message: 'Local minimum found that satisfies the constraints.

Optimiza...'

>>
```

Рис. 3.124 (окончание)

Результаты определения минимума функции (3.156) с применением полной формы записи решателя “fmincon” с начальными приближениями $x_1 = 0.5$ и $x_2 = 0.5$, границами поиска $-1 \le x \le 5$, $-2 \le x \le 2$ по программе, приведенной на рисунке 3.118

3.5.3.3. Решение задач нелинейного программирования (НЛП) с применением решателя “fmincon”

Задачи НЛП относятся к классу оптимизационных задач, при решении которых необходимо найти экстремальное (минимальное или максимальное) значение в общем случае нелинейной целевой функции с ограничениями не только *первого рода* на оптимизирующие переменные, но и с линейными и/или нелинейными ограничениями *второго рода,* накладываемыми на любые переменные математического описания процессов или явлений.

Многие задачи в химии и химической технологии формулируются как типичные задачи нелинейного программирования. Это прежде всего относится к задачам определения оптимальных условий (параметров) химико-технологических процессов и к задачам параметрической идентификации уравнений математического описания. Основные проблемы в этом случае сводятся к возможной многоэкстремальности целевых функций (критериев оптимальности) и, соответственно, к необходимости реализации на компьютерах эффективных алгоритмов оптимизации, позволяющих надежно определять глобальный экстремум. Поэтому решатели, предлагаемые для решения задач НЛП, включают в себя несколько алгоритмов оптимизации для поиска глобального экстремума, которые можно варьировать в зависимости от конкретной задачи.

К такому типу решателей относится и решатель MATLAB “fmincon”, в котором реализованы четыре различных алгоритма оптимизации[4]:

- interior-point (по умолчанию);
- active-set;
- sqp;
- trust-region-reflective.

Решатель “fmincon” в программном коде на *m*-языке программирования записывается в следующем виде:

$$\begin{aligned}&[x\text{opt},\ y\text{min},\ \text{pr},\ \text{out}] = \text{fmincon('function'},\ x0,\ \text{A},\ \text{b},\\ &\text{Aeq},\ \text{beq},\ x\text{min},\ x\text{max,'}\textit{nonlincon}\text{'},\ \textit{options}),\end{aligned} \tag{3.162}$$

[4] Подробнее об алгоритмах оптимизации см. в справке (Help) пакета MATLAB.

где:

а) информация о результатах расчетов:

xopt — вектор получаемого решения: оптимальных значений оптимизирующих переменных;

ymin — рассчитанное минимальное значение целевой функции;

pr — признак реализованного вычислительного процесса: при положительном целом числе результат удовлетворительный;

out — информация о реализованном вычислительном процессе: число выполненных итераций, число вычислений значений целевой функции в процессе расчетов и т. п.;

б) информация о задаваемых значениях параметров для вычислений:

'function' — название *m*-файла function.m целевой функции;

*x*0 — вектор начальных приближений оптимизирующих переменных;

A — матрица коэффициентов системы линейных неравенств вида:

$$Ax \le b; \tag{3.163}$$

b — вектор правых частей системы линейных неравенств;

Aeq — матрица коэффициентов системы линейных равенств вида:

$$Aeq \cdot x = beq; \tag{3.164}$$

beq — вектор правых частей системы линейных равенств;

xmin — вектор левых границ интервала изменения оптимизирующих переменных при запуске решателя;

xmax — вектор правых границ интервала изменения оптимизирующих переменных при запуске решателя;

'nonlincon' — название *m*-файла nonlincon.m линейных и/или нелинейных ограничений в виде вектора неравенств *g*1 и/или вектора равенств *g*2 (рис. 3.123);

options — параметры этой переменной (3.160) задаются функцией optimoptions, которая позволяет варьировать оптимизационные алгоритмы решателя "fmincon", организовать поитерационный вывод параметров вычислительного процесса в Командное окно MATLAB, регулировать погрешности вычислительного процесса при решении оптимизационной задачи.

В случае отсутствия ограничений в виде линейных неравенств (3.163) и линейных равенств (3.164) вместо указанных матриц и векторов в записи решателя должны стоять четыре пустые квадратные скобки.

При отсутствии нелинейных ограничений в виде нелинейных неравенств *g*1 и равенств *g*2 *m*-файл функции *nonlincon*(*x*) имеет вид (рис. 3.123):

$$\begin{aligned} &function[g1, g2] = nonlincon(x) \\ &g1 = []; \\ &g2 = []; \\ &\text{end.} \end{aligned} \tag{3.165}$$

Задание параметров массива *options* в (3.162) осуществляется функцией “*optimoptions*” (3.160) и предваряет запись решателя “fmincon” (3.162).

Конкретные способы задания параметров решателя “fmincon” (3.162) для систем линейных неравенств (3.163) и (3.164), а также нелинейной функции ограничений в виде *nonlincon*(x) (3.165) иллюстрируется приводимыми ниже примерами.

3.5.3.3.1. Решение задачи НЛП при наличии одного линейного ограничения в виде неравенства

Формулировка задачи НЛП.

Найти минимум функции с двумя оптимизирующими переменными x_1 и x_2:

$$y = 100\left(x_2 - x_1^2\right) + \left(1 - x_1\right) \tag{3.166}$$

при условии, что

$$x_1 + 2x_2 \le 1, \tag{3.167}$$

и с начальными приближениями для расчетов:

$$x_1^0 = -1 \text{ и } x_2^0 = 3. \tag{3.168}$$

Программный код *m*-файла решения, в котором для отображения получаемых результатов создается отдельный текстовый файл — Результаты НЛП_1.txt, представлен на рисунке 3.125. Вид целевой функции записан в отдельном файле FL.m (рис. 3.126), а линейные ограничения в виде неравенства (3.167) оформлены в коде программы (рис. 3.125) в следующем виде (3.163):

$$\text{A} = [1{,}2;\ 0{,}0];\ \text{b} = [1;\ 0]. \tag{3.169}$$

```
function GLAV_nlpmin_1
clc;
clear al;
%вектор начальных приближений
x0=[-1,2];
%дополнительное линейное ограничение в виде неравенства:
%        x(1)+2*x(2)<=1
A=[1,2;0,0];b=[1;0];
%поиск минимума функции FL при ограничениях A*x<=b
[XOPT,YMIN,pr]=fmincon(@FL,x0,A,b);
%открыть файл для записи
ff=fopen('результаты НЛП_1.txt','wt');
fprintf(ff,'\n ---------------------------------\n');
fprintf(ff,'РЕЗУЛЬТАТЫ РЕШЕНИЯ ЗАДАЧИ НЛП № 1');
fprintf(ff,'\n ---------------------------------');
fprintf(ff,'\n 1.Оптимальные значения оптимизирующих переменных (XOPT) :\n');
for i=1:2
   fprintf(ff,'%13g',XOPT(i));
end;
fprintf(ff,'\n ---------------------------------');
fprintf(ff,'\n 3.Минимальное значение целевой функции (YMIN): \n');
```

Рис. 3.125 (начало)

Программный код файла решения задачи НЛП для функции (3.166) с одним линейным неравенством (3.167) и с начальными приближениями $x_1^{(0)} = -1$ и $x_2^{(0)} = 3$

```
fprintf(ff,'%13g',YMIN);
fprintf(ff,'\n ----------------------------------');
fprintf(ff,'\n 3.Признак качества вычислительного процесса (pr):\n');
fprintf(ff,'%7g',pr);
fprintf(ff,'\n ----------------------------------');
fprintf(ff,'\n Процесс решения задачи завершен');
fprintf(ff,'\n ----------------------------------');
%закрыть файл текстовый файл
fclose(ff);
end
```

Рис. 3.125 (окончание)

Программный код файла решения задачи НЛП для функции (3.166) с одним линейным неравенством (3.167) и с начальными приближениями $x_1^{(0)} = -1$ и $x_2^{(0)} = 3$

```
function y=FL(x)
%целевая функция
y=100*(x(2)-x(1)^2)^2+(1-x(1))^2;
end
```

Рис. 3.126

Программный код целевой функции для решения задачи НЛП с функцией (3.166) и ограничением (3.167) по программе, приведенной на рисунке 3.125

Здесь также задаются начальные приближения для итерационных расчетов (рис. 3.125):

$$x0 = [-1, 2]. \tag{3.170}$$

Результаты вычислений приведены в текстовом файле (рис. 3.127).

```
-----------------------------------
РЕЗУЛЬТАТЫ РЕШЕНИЯ ЗАДАЧИ НЛП № 1
-----------------------------------
1.Оптимальные значения оптимизирующих переменных (XOPT) :
   0.502203    0.248899
-----------------------------------
3.Минимальное значение целевой функции (YMIN):
   0.248897
-----------------------------------
3.Признак качества вычислительного процесса (pr):
    1
-----------------------------------
Процесс решения задачи завершен
-----------------------------------
```

Рис. 3.127

Результаты решения задачи НЛП с целевой функцией (3.166) по программе, приведенной на рисунке 3.125

Следует отметить, в программном коде (рис. 3.125) обращение к решателю “fmincon” (3.162) может быть записано и в краткой форме:

$$xopt = fmincon(@FL, x0, A, b), \tag{3.171}$$

а матрица и вектор (3.160) для задания линейного неравенства (3.167) — в сокращенном виде (рис. 3.125):

$$\mathrm{A} = [1, 2]; \mathrm{b} = 1. \tag{3.172}$$

3.5.3.3.2. Решение задачи НЛП при наличии одного нелинейного ограничения в виде неравенства и с ограничениями первого рода на оптимизирующие переменные

Задача решается на примере поиска минимума целевой функции (3.166) в области, ограниченной окружностью, координаты центра которой $x_1^c = \frac{1}{3}$; $x_2^c = \frac{1}{3}$ и радиус $r = \frac{1}{3}$.

Нелинейное ограничение в этом случае имеет вид

$$\left(x_1 - \frac{1}{3}\right)^2 + \left(x_2 - \frac{1}{3}\right) \le \left(\frac{1}{3}\right)^2. \qquad (3.173)$$

Оно оформлено в файле GL.m (рис. 3.128) в неявном виде и приравнено к g_1:

$$g_1 = \left(x_1 - \frac{1}{3}\right)^2 + \left(x_2 - \frac{1}{3}\right) - \left(\frac{1}{3}\right)^2, \qquad (3.174)$$

а g_2 = [], так как нелинейных ограничений в виде равенств нет.

```
function [g1,g2]=GL(x)
g1=(x(1)-1/3)^2+(x(2)-1/3)^2-(1/3)^2;
g2=[];
end
```

Рис. 3.128

Программный код файла нелинейного ограничения к программе решения задачи НЛП с целевой функцией (3.166) по программе, приведенной на рисунке 3.129

Программный код файла решения задачи с целевой функцией FL.M (рис. 3.126), ограничениями первого рода на оптимизирующие переменные $0 \le x_1 \le 0.5$ и $0.2 \le x_2 \le 0.8$ и начальными приближениями для вычислений $x_1^0 = 0.25$ и $x_2^0 = 0.25$ представлен на рисунке 3.129, а результаты расчетов — на рисунке 3.130.

```
function GLAV_nlpmin_2
clc;
clear al;
%вектор начальных приближений
x0=[0.25,0.25];
%дополнительные ограничения
XMIN=[0,0.2];
XMAX=[0.5,0.8];
A=[];b=[];Aeq=[];beq=[];
%поиск минимума функции FL при ограничениях A*x<=b
nonlincon=@GL;
[XOPT,YMIN,pr]=fmincon('FL',x0,A,b,Aeq,beq,XMIN,XMAX,nonlincon);
%открыть файл для записи
ff=fopen('результаты НЛП_3.txt','wt');
fprintf(ff,'\n ------------------------------------\n');
fprintf(ff,'РЕЗУЛЬТАТЫ РЕШЕНИЯ ЗАДАЧИ НЛП № 2');
```

Рис. 3.129 (начало)

Программный код файла решения задачи НЛП с ограничениями первого рода и нелинейным ограничением второго рода (рис. 3.128) с целевой функцией (3.166)

```
fprintf(ff,'\n ----------------------------------------');
fprintf(ff,'\n 1.Оптимальные значения оптимизирующих переменных (XOPT) :\n');
for i=1:2
   fprintf(ff,'%13g',XOPT(i));
end;
fprintf(ff,'\n ----------------------------------------');
fprintf(ff,'\n 3.Минимальное значение целевой функции (YMIN): \n');
fprintf(ff,'%13g',YMIN);
fprintf(ff,'\n ----------------------------------------');
fprintf(ff,'\n 3.Признак качества вычислительного процесса (pr):\n');
fprintf(ff,'%7g',pr);
fprintf(ff,'\n ----------------------------------------');
fprintf(ff,'\n Процесс решения задачи завершен');
fprintf(ff,'\n ----------------------------------------');
%закрыть файл текстовый файл
fclose(ff);
end
```

Рис. 3.129 (окончание)

Программный код файла решения задачи НЛП с ограничениями первого рода и нелинейным ограничением второго рода (рис. 3.128) с целевой функцией (3.166)

```
----------------------------------------
РЕЗУЛЬТАТЫ РЕШЕНИЯ ЗАДАЧИ НЛП № 2
----------------------------------------
1.Оптимальные значения оптимизирующих переменных (XOPT) :
      0.5        0.25
----------------------------------------
3.Минимальное значение целевой функции (YMIN):
      0.25
----------------------------------------
3.Признак качества вычислительного процесса (pr):
    1
----------------------------------------
Процесс решения задачи завершен
----------------------------------------
```

Рис. 3.130

Результаты решения задачи НЛП с целевой функцией (3.166) по программе, приведенной на рисунке 3.129.

3.5.3.3.3. Решение задачи НЛП при наличии одного нелинейного ограничения второго рода в виде равенства и с частичными ограничениями первого рода на оптимизирующие переменные

Формулировка задачи НЛП.

Найти минимум функции

$$y = x_1^2 + x_2^2 - 1 \tag{3.175}$$

при условии, что

$$x_1 \cdot x_2 = 4 \tag{3.176}$$

и

$$x_1 \geq 0;\ x_2 \geq 0. \tag{3.177}$$

Программный код файла решения приведен на рисунке 3.131, целевой функции — на рисунке 3.132, а нелинейного ограничения в виде равенства (3.176) в неявном виде — на рисунке 3.133.

```
function GLAV_nlpmin_3
clc;
clear al;
%вектор начальных приближений
x0=[0.1,0.1];
%дополнительные ограничения
lx=[0,0];
%поиск минимума функции FL при ограничениях 2-го рода GL
[XOPT,YMIN,pr]=fmincon(@FL,x0,[],[],[],[],lx,[],@GL);
%открыть файл для записи
ff=fopen('результаты НЛП_3.txt','wt');
fprintf(ff,'\n ---------------------------------\n');
fprintf(ff,'РЕЗУЛЬТАТЫ РЕШЕНИЯ ЗАДАЧИ НЛП № 3');
fprintf(ff,'\n ---------------------------------');
fprintf(ff,'\n 1.Оптимальные значения оптимизирующих переменных (XOPT) ;\n');
for i=1:2
   fprintf(ff,'%7g',XOPT(i));
end;
fprintf(ff,'\n ---------------------------------');
fprintf(ff,'\n 3.Минимальное значение целевой функции (YMIN): \n');
fprintf(ff,'%7g',YMIN);
fprintf(ff,'\n ---------------------------------');
fprintf(ff,'\n 3.Признак качества вычислительного процесса (pr):\n');
fprintf(ff,'%7g',pr);
fprintf(ff,'\n ---------------------------------');
fprintf(ff,'\n Процесс решения задачи завершен');
fprintf(ff,'\n ---------------------------------');
%закрыть файл текстовый файл
fclose(ff);
end
```

Рис. 3.131

Программный код файла решения задачи НЛП с целевой функцией (3.175)

```
function y=FL(x)
%целевая функция
y=x(1)^2+x(2)^2-1;
end
```

Рис. 3.132

Программный код целевой функции при решении задачи НЛП по программе, приведенной на рисунке 3.131

```
function [g1,g2]=GL(x)
g1=[];
g2=x(1)*x(2)-4;
end
```

Рис. 3.133

Программный код файла ограничений второго рода при решении задачи НЛП по программе, приведенной на рисунке 3.131.

Так как нелинейных ограничений в виде неравенств нет, то g_1 в функции GL(x) (рис. 3.133) приравнивается к пустым квадратным скобкам g_1 = [], по аналогии отсутствие верхних границ изменения оптимизирующих переменных

при поиске минимума (3.175) приводит к тому, что *xmax* = [] (3.161) в записи решателя (рис. 3.131).

При задании начальных приближений для решения: $x_1^0 = 0.1$ и $x_2^0 = 0.1$ (рис. 3.131) получается результат, представленный на рисунке 3.134.

```
-------------------------------------
РЕЗУЛЬТАТЫ РЕШЕНИЯ ЗАДАЧИ НЛП № 3
-------------------------------------
1.Оптимальные значения оптимизирующих переменных (XOPT) :
   2     2
-------------------------------------
3.Минимальное значение целевой функции (YMIN):
   7
-------------------------------------
3.Признак качества вычислительного процесса (pr):
   1
-------------------------------------
Процесс решения задачи завершен
-------------------------------------
```

Рис. 3.134

Результат решения задачи НЛП по программе, приведенной на рисунке 3.131

3.5.3.3.4. Решение задачи НЛП при наличии трех ограничений второго рода, двух в виде неравенств и одного в виде равенства и с частичными ограничениями первого рода на оптимизирующие переменные

Формулировка задачи НЛП.

Найти минимум функции

$$y = x_1^2 - x_2 \tag{3.178}$$

при условии, что

$$\begin{gathered} x_1 \geq 1, \\ x_1^2 - x_2 \leq 26, \\ x_1 + x_2 = 6 \end{gathered} \tag{3.179}$$

и

$$x_1 \geq 0;\ x_2 \geq 0. \tag{3.180}$$

Для решения условия (3.179) должны быть переписаны в неявном виде:

$$-x_1 + 1 \leq 0, \tag{3.181a}$$

$$x_1^2 + x_2^2 - 26 \leq 0, \tag{3.181b}$$

$$x_1 + x_2 - 6 = 0 \tag{3.181c}$$

и оформлены в виде *m*-файла GL.m (рис. 3.135) с функциями g_{11} и g_{12} от x_1 и x_2:

$$g_{11} = -x_1 + 1,$$
$$g_{12} = x_1^2 + x_2^2 - 26, \tag{3.182}$$

характеризующими компоненты вектора $[g_{11}, g_{12}]$ с ограничениями в виде неравенств (3.181b), и переменную, характеризующую ограничение в виде равенства (3.181c):

$$g_2 = x_1 + x_2 - 6. \tag{3.183}$$

```
function [g1,g2]=GL(x)
g11=-x(1)+1;
g12=x(1)^2+x(2)^2-26;
g1=[g11,g12];
g2=x(1)+x(2)-6;
end
```

Рис. 3.135

Программный код файла ограничений второго рода при решении задачи НЛП с целевой функцией (3.178) по программе, приведенной на рисунке 3.136

Результат решения задачи НЛП с начальными приближениями оптимизирующих переменных $x_1^0 = 8$ и $x_2^0 = 6$ приведен на рисунке 3.138.

```
clc;
clear all;
% Вектор начальных приближений;
x0=[8 6];
% Дополнительные ограничения;
lx=[0 0];
[XOPT,YMIN,pr]=fmincon('FL',x0,[],[],[],[],lx,[],'GL');
%открыть файл для записи
ff=fopen('результаты НЛП_4.txt','wt');
fprintf(ff,'\n ---------------------------------------\n');
fprintf(ff,'РЕЗУЛЬТАТЫ РЕШЕНИЯ ЗАДАЧИ НЛП № 4');
fprintf(ff,'\n ---------------------------------------');
fprintf(ff,'\n 1.Оптимальные значения оптимизирующих переменных (XOPT) :\n');
for i=1:2
   fprintf(ff,'%7g',XOPT(i));
end;
fprintf(ff,'\n ---------------------------------------');
fprintf(ff,'\n 3.Минимальное значение целевой функции (YMIN): \n');
fprintf(ff,'%7g',YMIN);
fprintf(ff,'\n ---------------------------------------');
fprintf(ff,'\n 3.Признак качества вычислительного процесса (pr):\n');
fprintf(ff,'%7g',pr);
fprintf(ff,'\n ---------------------------------------');
fprintf(ff,'\n Процесс решения задачи завершен');
fprintf(ff,'\n ---------------------------------------');
%закрыть файл текстовый файл
fclose(ff);
```

Рис. 3.136

Программный код файла решения задачи НЛП с целевой функцией (3.178)

Программный код файла решения задачи НЛП представлен на рисунке 3.136, а целевая функция (3.178) оформлена в виде отдельного файла FL.m (рис. 3.137).

```
function R=FL(x)
%целевая функция
R=x(1)^2-x(2);
end
```

Рис. 3.137

Программный код целевой функции при решении задачи НЛП по программе, приведенной на рисунке 3.136.

```
-------------------------------------
РЕЗУЛЬТАТЫ РЕШЕНИЯ ЗАДАЧИ НЛП № 4
-------------------------------------
1.Оптимальные значения оптимизирующих переменных (XOPT) :
    1     5
-------------------------------------
3.Минимальное значение целевой функции (YMIN):
   -4
-------------------------------------
3.Признак качества вычислительного процесса (pr):
    1
-------------------------------------
Процесс решения задачи завершен
-------------------------------------
```

Рис. 3.138

Результаты решения задачи НЛП с целевой функцией (3.178)

3.5.3.3.5. Решение задачи НЛП с тремя оптимизирующими переменными при наличии двух ограничений второго рода в виде равенств и с частичными ограничениями первого рода

Формулировка задачи НЛП.

Найти минимум функции

$$y = 1000 - x_1^2 - 2x_2^2x_3^2 - x_1x_2 \tag{3.184}$$

при условии, что

$$\begin{aligned} x_1^2 + x_2^2 + x_3^2 &= 25, \\ 8x_1 + 14x_2 + 7x_3 &= 56 \end{aligned} \tag{3.185}$$

и

$$x_1 \geq 0,\ x_2 \geq 0,\ x_3 \geq 0. \tag{3.186}$$

Ограничения в виде равенств оформлены отдельным файлом GL.m (рис. 3.139), в программном коде которого каждое равенство записано неявно, как компоненты вектора $g2 = [g21, g22]$, где

$$\begin{aligned} g21 &= x_1^2 + x_2^2 + x_3^2 - 25, \\ g22 &= 8x_1 + 14x_2 + 7x_3 - 56. \end{aligned} \tag{3.187}$$

Вектор g_1, характеризующий ограничения в виде неравенств, записывается как $g1$=[], так как их нет.

```
function [g1,g2]=GL(x)
g21=x(1)^2+x(2)^2+x(3)^2-25;
g22=8*x(1)+14*x(2)+7*x(3)-56;
g2=[g21,g22];
g1=[];
end
```

Рис. 3.139

Программный код файла ограничений второго рода при решении задачи НЛП по программе, приведенной на рисунке 3.140.

Программный код файла решения задачи при задании начальных приближений на оптимизирующие переменные $x_1^0 = 35$, $x_2^0 = 0.2$, $x_3^0 = 3.5$ представлен на рисунке 3.140, а отдельно оформленная целевая функция (3.184) в виде файла FL.m — на рисунке 3.141. Результаты решения задачи НЛП приведены на рисунке 3.143.

```
clc;
clear al;
% Вектор начальных приближений;
x0=[3.5 0.2 3.5];
% Дополнительные ограничения;
lx=[0 0 0];
[XOPT,YMIN,pr]=fmincon('FL',x0,[],[],[],[],lx,[],'GL');
%открыть файл для записи
ff=fopen('результаты НЛП_5.txt','wt');
fprintf(ff,'\n ---------------------------------------\n');
fprintf(ff,'РЕЗУЛЬТАТЫ РЕШЕНИЯ ЗАДАЧИ НЛП № 5');
fprintf(ff,'\n ---------------------------------------');
fprintf(ff,'\n 1.Оптимальные значения оптимизирующих переменных (XOPT) ;\n');
for i=1:3
   fprintf(ff,'%13g',XOPT(i));
end;
fprintf(ff,'\n ---------------------------------------');
fprintf(ff,'\n 3.Минимальное значение целевой функции (YMIN): \n');
fprintf(ff,'%13g',YMIN);
fprintf(ff,'\n ---------------------------------------');
fprintf(ff,'\n 3.Признак качества вычислительного процесса (pr):\n');
fprintf(ff,'%13g',pr);
fprintf(ff,'\n ---------------------------------------');
fprintf(ff,'\n Процесс решения задачи завершен');
fprintf(ff,'\n ---------------------------------------');
%закрыть файл текстовый файл
fclose(ff);
```

Рис. 3.140

Программный код файла решения задачи НЛП с целевой функцией (3.184)

```
function R=FL(x)
%целевая функция
R=1000-x(1)^2-2*x(2)^2-x(3)^2-x(1)*x(2);
end
```

Рис. 3.141

Программный код целевой функции при решении задачи НЛП с целевой функцией (3.184) по программе, приведенной на рисунке 3.140

```
---------------------------------------
РЕЗУЛЬТАТЫ РЕШЕНИЯ ЗАДАЧИ НЛП № 5
---------------------------------------
1.Оптимальные значения оптимизирующих переменных (XOPT) :
 3.16111e-06     1.63795      4.7241
---------------------------------------
3.Минимальное значение целевой функции (YMIN):
    973.317
---------------------------------------
3.Признак качества вычислительного процесса (pr):
          1
---------------------------------------
Процесс решения задачи завершен
---------------------------------------
```

Рис. 3.142

Результаты решения задачи НЛП с целевой функцией (3.181) по программе, приведенной на рисунке 3.140.

3.5.3.4. Параметрическая идентификация линейных и нелинейных уравнений с одной или несколькими независимыми переменными с применением решателей “polyfit”, “lsqcurvefit”, а также “fminsearch” и “fmincon”.

В этом случае речь идет об оптимизационной задаче, в результате решения которой определяются коэффициенты (параметры) уравнения исходя из массива данных, в частности экспериментальных. При этом оптимальными считаются параметры (коэффициенты) уравнения, вычисленные путем минимизации одного из критериев ((3.140), (3.141) или (3.142)), характеризующих рассогласование между известными значениями зависимых переменных $y^{\text{эксп}}$ и расчетными значениями $y^{\text{расч.}}$ при одних и тех же значениях независимых переменных $\overline{x}$.

Следует различать:

а) уравнения многочленов (полиномов) с одной независимой переменной x, которые являются линейными относительно коэффициентов:

$$y^{расч} = \sum_{j=1}^{m+1} a_j x^{j-1} = a_1 + a_2 x + a_3 x^2 + \ldots + a_m x^{m-1} + a_{m+1} x^m, \qquad (3.188)$$

где m — степень многочлена; a_1, a_2, …, a_{m+1} — определяемые коэффициенты (параметры) уравнения;

б) нелинейные уравнения с одной независимой переменной и тремя коэффициентами a_1, a_2, a_3, например:

$$y^{\text{расч}} = a_1 e^{-a_2 x} + a_3 \sin(x); \qquad (3.189)$$

в) уравнения с несколькими независимыми переменными, линейные относительно коэффициентов с четырьмя коэффициентами, например:

$$y^{\text{расч}} = a_1 + a_2 x_1 x_2 + a_3 x_1^2 + a_4 x_2^2; \qquad (3.190)$$

г) уравнения с несколькими независимыми переменными и пятью коэффициентами, нелинейные относительно коэффициентов:

$$y^{\text{расч}} = a_1 + a_2 x_1^{a_3} x_2^{a_4} + a_5 ln x_3. \qquad (3.191)$$

Для определения коэффициентов многочленов (полиномов) (3.188) в среде MATLAB предлагается использовать решатель “polyfit”. Определение коэффициентов нелинейных уравнений с одной независимой переменной (3.189) может быть выполнено с использованием решателя “lsqcurvefit”. Для определения линейных относительно параметров (коэффициентов) уравнений с несколькими независимыми переменными (3.190) необходимо реализовать процедуру решения системы линейных алгебраических уравнений (СЛАУ). Такая же процедура может быть применена для определения коэффициентов многочленов произвольной степени (3.188), так как уравнение многочлена (полинома) линейно относительно коэффициентов.

Коэффициенты уравнений с несколькими независимыми переменными, нелинейными относительно коэффициентов (3.191), могут быть определены с использованием решателей MATLAB “fminsearch” или “fmincon”. Эти решатели могут быть применены также для определения коэффициентов всех рассмотренных четырех типов уравнений ((3.188)–(3.191)).

Для решения задачи параметрической идентификации уравнения необходимы массивы данных, которые могут быть представлены в следующем виде:

— для уравнения с одной независимой переменной x (N — число данных):

№	x	$y^{\text{эксп}}$
1	x_1	$y_1^{\text{эксп}}$
2	x_2	$y_2^{\text{эксп}}$
⋮	⋮	⋮
N	x_N	$y_N^{\text{эксп}}$

— для уравнения с несколькими независимыми переменными $\bar{x} = [x_1, x_2, \ldots, x_n]$ (n — число независимых переменных; N — число данных):

№	$\bar{x}$	$y^{\text{эксп}}$
1	$\overline{x_1}$	$y_1^{\text{эксп}}$
2	$\overline{x_2}$	$y_2^{\text{эксп}}$
⋮	⋮	⋮
N	$\overline{x_N}$	$y_N^{\text{эксп}}$

Задача идентификации коэффициентов линейного или нелинейного уравнения осуществляется с использованием таких данных путем нахождения наименьшего значения одного из критериев рассогласования между расчетны-

ми $y^{\text{расч}}$ и заданными (экспериментальными) $y^{\text{эксп}}$ значениями зависимой переменной y ((3.140)–(3.142)).

3.5.3.4.1. Определение коэффициентов линейной зависимости поверхностного натяжения изопропилбензола от температуры

Решение задачи параметрической идентификации линейных и нелинейных уравнений различными способами целесообразно проиллюстрировать на примере определения двух коэффициентов уравнения $y = a_1 + a_2 x$ зависимости поверхностного натяжения изопропилбензола от температуры.

Массив данных (таблица экспериментов с размерностями физических величин) в этом случае имеет вид:

№	*T*	*SIGMA*
	°C	эрг/см^2
1	20	28.70
2	25	27.68
3	30	27.17
4	40	26.09
5	50	25.08
6	60	24.07
7	70	23.01
8	80	23.20
9	90	21.20

В качестве критерия рассогласования, минимум которого необходимо определить путем решения оптимизационной задачи, выбирается критерий метода наименьших квадратов (МНК) (3.140) с весовыми коэффициентами, равными единице:

$$SIGMA^{\text{расч}} = a_1 + a_2 T. \tag{3.192}$$

Для линейных по коэффициентам уравнений или уравнений, которые путем алгебраических преобразований могут быть приведены к линейному по коэффициентам виду, для определения минимума критерия (3.140) принято пользоваться необходимыми условиями существования экстремума функции Cr ($\bar{a}$) многих переменных — в данном случае переменными являются искомые коэффициенты.

В конкретном случае рассматриваемой задачи критерий (3.140) имеет вид

$$Cr = \sum_{i=1}^{N} \left(a_1 + a_2 T_i - SIGMA_i^{\text{эксп}} \right)^2, \tag{3.193}$$

а необходимые условия экстремума:

$$\frac{\partial Cr}{\partial a_1} = 0; \ \frac{\partial Cr}{\partial a_2} = 0. \tag{3.194}$$

В результате получается система линейных алгебраических уравнений следующего вида:

$$\begin{cases} (N)a_1 + \left(\sum_{i=1}^{N} T_i\right) a_2 = \sum_{i=1}^{N} SIGMA_i^{\text{эксп}} \\ \left(\sum_{i=1}^{N} T_i\right) a_1 + \left(\sum_{i=1}^{N} T_i^2\right) a_2 = \sum_{i=1}^{N} SIGMA_i^{\text{эксп}} \cdot T_i \end{cases}. \tag{3.195}$$

Искомые коэффициенты (параметры) a_1 и a_2 являются решениями системы уравнений (3.195).

Для программной реализации этого метода определения коэффициентов линейных уравнений, в частности многочлена (полинома) степени m, предлагается сформировать матрицу $\overline{\overline{\Phi}}$ и вектор $\overline{b}$:

$$\overline{\overline{\Phi}} = \begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^m \\ 1 & x_2 & x_2^2 & \cdots & x_2^m \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_N & x_N^2 & \cdots & x_N^m \end{bmatrix}; \quad \overline{y}^{\,эксп} = \begin{bmatrix} y_1^{эксп} \\ y_2^{эксп} \\ \vdots \\ y_N^{эксп} \end{bmatrix}, \tag{3.196}$$

и формула для определения коэффициентов многочлена степени m имеет вид

$$\overline{a} = \left(\overline{\overline{\Phi}}^T \overline{\overline{\Phi}}\right)^{-1} \overline{\overline{\Phi}}^T \overline{y}^{\text{эксп}}. \tag{3.197}$$

Для рассматриваемой конкретной задачи (3.192) матрица $\overline{\overline{\Phi}}$ и вектор $\overline{b}$ имеют вид:

$$\overline{\overline{\Phi}} = \begin{bmatrix} 1 & T_1 \\ 1 & T_2 \\ \vdots & \vdots \\ 1 & T_N \end{bmatrix}; \quad \overline{b} = \begin{bmatrix} SIGMA_1^{\text{эксп}} \\ SIGMA_2^{\text{эксп}} \\ \vdots \\ SIGMA_N^{\text{эксп}} \end{bmatrix}. \tag{3.198}$$

Рассмотренный метод определения коэффициентов уравнения многочлена первой степени (m = 1) реализован в программе (рис. 3.143 — priznak = 1), включающей в себя три m-файла: рисунки 3.143–3.145.

```
%-------------------------------------------------------------
%Программный код файла GLAV_THERMO_5.m — основная управляющая программа
%-------------------------------------------------------------
%Программа включает следующие файлы:
GLAV_THERMO_5.m+DATA.m+coef_calcul.m+extr.m+mnk.m+REPORT.m
%-------------------------------------------------------------
%Программа расчета коэффициентов аппроксимационной зависимости
%поверхностного натяжения изопропилбензола от температуры (T):
%1. Если priznak=1 (задается в файле DATA.m) — методом линейной регрессии
%с получением уравнения вида: SIGMA=a(1)+a(2)*T ;
%2. Если priznak=2 (задается в файле DATA.m) — с применением решателя polyfit
%с получением уравнения вида: SIGMA=a(1)*T +a(2));
```

Рис. 3.143 (начало)

Программный код файла основной управляющей программы определения коэффициентов линейного уравнения $SIGMA = a_1 + a_2 T$

```
%3. Если priznak=3 (задается в файле DATA.m) — с применением решателя fminsearch
%с поиском минимального рассогласования между расчетными и
%экспериментальными значениями SIGMA и с получением уравнения вида:
SIGMA=a(1)+a(2)*T;
%4. Если priznak=4 (задается в файле DATA.m) — с применением решателя fmincon
%с поиском минимального рассогласования между расчетными и
%экспериментальными значениями SIGMA и с получением уравнения вида:
SIGMA=a(1)+a(2)*T;
%5. Если priznak=5 (задается в файле DATA.m) — с применением решателя
%lsqcurvefit с поиском минимального квадратичного рассогласования между расчет-
ными и
%экспериментальными значениями SIGMA (критерий метода наименьших квадратов-
МНК)
%и с получением уравнения вида: SIGMA=a(1)+a(2)*T;
%6. Если priznak=6 (задается в файле DATA.m) — решение задачи с включением
%стандартного инструмента MATLAB для построения графика и его редактирования,
%а также автоматическим выбором многочлена 1-ой степени для аппроксимации
%данных с уравнения вида: SIGMA=a(1)+a(2)*T;
%-----------------------------------------------------------------------
clc;
clear all;
close all;
global  priznak
DATA;
coef_calcul(priznak);
if priznak~=6
REPORT;
end
```

Рис. 3.143 (окончание)

Программный код файла основной управляющей программы определения коэффициентов линейного уравнения $SIGMA = a_1 + a_2 T$

```
function DATA
%-----------------------------------------------------------------
%Программный код файла DATA.m — задание исходных данных для расчетов
%-----------------------------------------------------------------
%Программа включает следующие файлы:
GLAV_THERMO_5.m+DATA.m+coef_calcul.m+extr.m+mnk.m+REPORT.m
%-----------------------------------------------------------------
%Программа расчета коэффициентов аппроксимационной зависимости
%поверхностного натяжения изопропилбензола от температуры (T):
%1. Если priznak=1 (задается в файле DATA.m) — методом линейной регрессии
%с получением уравнения вида: SIGMA=a(1)+a(2)*T ;
%2. Если priznak=2 (задается в файле DATA.m) — с применением решателя polyfit
%с получением уравнения вида: SIGMA=a(1)*T +a(2));
%3. Если priznak=3 (задается в файле DATA.m) — с применением решателя fminsearch
%с поиском минимального рассогласования между расчетными и
%экспериментальными значениями SIGMA и с получением уравнения вида: SIGMA=a(1)+a(2)*T;
%4. Если priznak=4 (задается в файле DATA.m) — с применением решателя fmincon
%с поиском минимального рассогласования между расчетными и
%экспериментальными значениями SIGMA и с получением уравнения вида: SIGMA=a(1)+a(2)*T;
%5. Если priznak=5 (задается в файле DATA.m) — с применением решателя
%lsqcurvefit с поиском минимального квадратичного рассогласования между расчетными и
%экспериментальными значениями SIGMA (критерий метода наименьших квадратов-МНК)
%и с получением уравнения вида: SIGMA=a(1)+a(2)*T;
%6. Если priznak=6 (задается в файле DATA.m) — решение задачи с включением
%стандартного инструмента MATLAB для построения графика и его редактирования,
%а также автоматическим выбором многочлена 1-ой степени для аппроксимации
%данных с уравнения вида: SIGMA=a(1)+a(2)*T;
%-----------------------------------------------------------------------
global T SIGMA comp PARAM priznak priz_print aregr;
format short;
%1. Название индивидуального вещества ;
comp='ИЗОПРОПИЛБЕНЗОЛ';
%-----------------------------
```

Рис. 3.144 (начало)

Программный код файла исходных данных DATA.m для определения коэффициентов линейного уравнения $SIGMA = a_1 + a_2 T$

```
%2. Используемые данные по зависимости поверхностного натяжения
  %индивидуального вещества — SIGMA('эрг.см^2 )  от температуры T ( C ) :
T=[20 25 30 40 50 60 70 80 90];
SIGMA=[28.7 27.68 27.17 26.09 25.08 24.07 23.01 22.2 21.2];
%-----------------------------------
%3. Число параметров уравнения аппроксимации SIGMA[эрг/см^2]=a(1)+a(2)*T[C}:
PARAM=2;
%4. Начальные приближения итерационных расчетов при
%минимизации критерия рассогласования между расчетными и
%экспериментальными значениями SIGMA:
aregr=[5,-5];
%5. Метод расчета коэффициентов методом линеаризованной регрессии путем задания
% значения параметра:
priznak=5;
%-----------------------------------------------------------------------
% 4. Форма вывода отчета о работе программы:
%1. Если priz_print=0 (задается в файле DATA.m) — отчет выводится в командное окно MATLAB;
%2. Если priz_print=1 (задается в файле DATA.m) — отчет выводится в текстовый файл, который
размещается в той же...
%папке MATLAB, в которой находятся все m-файлы программы;
priz_print=1;
end
```

Рис. 3.144 (окончание)

Программный код файла исходных данных DATA.m для определения коэффициентов линейного уравнения $SIGMA = a_1 + a_2 T$

```
function coef_calcul(priznak)
%-----------------------------------------------------------------
%Программный код файла coef_calcul.m  — определение коэффициентов
%аппроксимирующих уравнений
%-----------------------------------------------------------
%Программа включает следующие файлы: GLAV_THERMO_5.m+DATA.m+coef_calcul.m+extr.m+mnk.m+REPORT.m
%-----------------------------------------------------------
%Программа расчета коэффициентов аппроксимационной зависимости
%поверхностного натяжения изопропилбензола от температуры (T):
%1. Если priznak=1 (задается в файле DATA.m) — методом линейной регрессии
%с получением уравнения вида: SIGMA=a(1)+a(2)*T ;
%2. Если priznak=2 (задается в файле DATA.m) — с применением решателя polyfit
%с получением уравнения вида: SIGMA=a(1)*T +a(2));
%3. Если priznak=3 (задается в файле DATA.m) — с применением решателя fminsearch
%с поиском минимального рассогласования между расчетными и
%экспериментальными значениями SIGMA и с получением уравнения вида: SIGMA=a(1)+a(2)*T;
%4. Если priznak=4 (задается в файле DATA.m) — с применением решателя fmincon
%с поиском минимального рассогласования между расчетными и
%экспериментальными значениями SIGMA и с получением уравнения вида: SIGMA=a(1)+a(2)*T;
%5. Если priznak=5 (задается в файле DATA.m) — с применением решателя
%lsqcurvefit с поиском минимального квадратичного рассогласования между расчетными и
%экспериментальными значениями SIGMA (критерий метода наименьших квадратов-МНК)
%и с получением уравнения вида: SIGMA=a(1)+a(2)*T;
%6. Если priznak=6 (задается в файле DATA.m) — решение задачи с включением
%стандартного инструмента MATLAB для построения графика и его редактирования,
%а также автоматическим выбором многочлена 1-ой степени для аппроксимации
%данных с уравнения вида: SIGMA=a(1)+a(2)*T;
%-----------------------------------------------------------------
global T SIGMA  a  N obusl TT PP PPL  SIGMA_calc aregr crit comp
obusl=0;
N=length(T);
TT=T';
PP=SIGMA';
PPL=PP;
switch priznak
   case 1
%1. Решение задачи численным методом линейной регрессии с использованием
%стандартных функций MATLAB
```

Рис. 3.145 (начало)

Программный код файла для определения коэффициентов линейного уравнения $SIGMA = a_1 + a_2 T$ пятью разными способами

```
%Формирование матрицы коэффициентов линеиной системы уравнении
for i=1:N
   for j=1:2
      FF(i,j)=TT(i)^(j-1);
   end
end

FFT=FF';
II=FFT*FF;
BB=FFT*PPL;
IIINV=inv(II);
a=IIINV*BB;
obusl=norm(II)*norm(IIINV);
for i=1:N
SIGMA_calc(i)=a(1)+a(2)*TT(i);
end
%-------------------------------


   case 2

%2. Решение задачи с использованием стандартных решателей MATLAB
%Определение коэффициентов многочлена 1-ой степени Pcalc2m=a(1)*T^1+a(2)
%решателем MATLAB — polyfit
a=polyfit(T,SIGMA,1);
%Определение расчетных значений поверхностного натяжения с применением
%найденных значений коэффициентов функцией MATLAB — polyval
SIGMA_calc=polyval(a,T);
%-------------------------------
case 3
%Вычисление вектора коэффициентов aregr методом нелинейной регрессии
%с функцией рассогласования экспериментальных расчетных значений extr.m;
%Определение оптимальных параметров aregr=[A,B и C], которые обеспечивают минимум критерия рассогла-
сования');
[aregr,crit,pr,out]=fminsearch('extr',aregr)
a=aregr;
case 4
%Вычисление вектора коэффициентов aregr методом нелинейной регрессии
%с функцией рассогласования экспериментальных расчетных значений extr.m;
%Определение оптимальных параметров aregr=[A,B и C], которые обеспечивают минимум критерия рассогла-
сования');
[aregr,crit,pr,out]=fmincon('extr',aregr)
a=aregr;
case 5
%Вычисление вектора коэффициентов AA со стандартной функцией MATLAB"lsqcurvefit"
%с вектором приближений для расчетов aregr для функции, заданной в файле mnk.m;
[AA,crit]=lsqcurvefit('mnk',aregr,T,SIGMA)
a=AA;
case 6
%Решение задачи с включением стандартного инструмента MATLAB для построения
% графика и его редактирования, а также автоматическим выбором многочлена 1-ой
% степени для аппроксимации данных
plot(T,SIGMA,'*');
title([comp ':Расчет поверхностного натяжения по форм.: SIGMA=a(1)+a(2)*T']);
xlabel('T-[C]');
ylabel('SIGMA[эрг.см^2]) — ("***" — эксперимент; "-" — аппрок.ф-ла');
end

end
```

Рис. 3.145 (окончание)

Программный код файла для определения коэффициентов линейного уравнения $SIGMA = a_1 + a_2 T$ пятью разными способами

В основной управляющей программе (рис. 3.143) считываются данные из файла DATA.m (рис. 3.144), вызывается файл вычислений coef_calcul и определяются коэффициенты с учетом (3.195), (3.196), (3.198) и по матричной формуле (3.197) — при условии, что priznak = 1 и $m = 1$.

Результаты расчетов коэффициентов уравнения (3.192) представлены в табличном (рис. 3.146) и графическом (рис. 3.147) виде.

Программа расчета коэффициентов аппроксимационных зависимостей поверхностного натяжения изопропилбензола(Р) от температуры (Т) :

---

1. Если priznak=1 (задается в файле DATA.m) — методом линеаризованой регрессии с получением уравнения вида: SIGMA=a(1)+a(2)*T

---

2. Если priznak=2 (задается в файле DATA.m) — с применением стандартных функций и решателей MATLAB с получением уравнения вида: SIGMA=a(1)*T+a(2)

---

3. Если priznak=3 (задается в файле DATA.m) — с применением решателя fminsearch с поиском минимального рассогласования между расчетными и экспериментальными значениями SIGMA и с получением уравнения вида: SIGMA=a(1)+a(2)*T;

---

4. Если priznak=4 (задается в файле DATA.m) — с применением решателя fmincon с поиском минимального квадратичного рассогласования (метод наименьших квадратов — МНК) между расчетными и экспериментальными значениями SIGMA и с получением уравнения вида: SIGMA=a(1)+a(2)*T;

---

5. Если priznak=5 (задается в файле DATA.m) — с применением решателя lsqcurvefit с поиском минимального квадратичного рассогласования (метод наименьших квадратов — МНК) между расчетными и экспериментальными значениями SIGMA и с получением уравнения вида: SIGMA=a(1)+a(2)*T;

---

6. Если priznak=6 (задается в файле DATA.m) — решение задачи с включением стандартного инструмента MATLAB для построения графика и его редактирования, а также автоматическим выбором многочлена 1-ой степени для аппроксимации данных с уравнения вида: SIGMA=a(1)+a(2)*T

Программа включает следующие файлы: GLAV_THERMO_5.m+DATA.m+coef_calcul.m+extr.m+mnk.m+REPORT.m

---

ИСХОДНЫЕ ДАННЫЕ ДЛЯ РАСЧЕТОВ

---

1. Признак расчетов (priznak) = 1
2.Индивидуальное вещество: ИЗОПРОПИЛБЕНЗОЛ
3.Определение коэффициентов аппроксимационной зависимости поверхностного натяжения индивидуального вещества от температуры для уравнения вида: SIGMA[эрг/см^2]=a(1)+a(2)*T[C]

РЕЗУЛЬТАТЫ РАСЧЕТОВ

Параметры уравнения давления пара: SIGMA[эрг/см^2]=a(1)+a(2)*T[C]

1. Коэффициенты уравнения
Первый коэффициент (a(1)) = 30.3721
Второй коэффициент (a(2)) = -0.1035
2. Результаты сравнения опытных (SIGMA_exp) и расчетных (SIGMA_calc) данных о поверхностном натяжении вещества

№	T(C)	SIGMA_exp(эрг/см^2)	SIGMA_calc(эрг/см^2)	Ошибка(эрг/см^2)
1	20.0	28.7000	28.3012	3.9880e-01
2	25.0	27.6800	27.7835	-1.0347e-01
3	30.0	27.1700	27.2657	-9.5733e-02
4	40.0	26.0900	26.2303	-1.4027e-01
5	50.0	25.0800	25.1948	-1.1480e-01
6	60.0	24.0700	24.1593	-8.9333e-02
7	70.0	23.0100	23.1239	-1.1387e-01

Рис. 3.146 (начало)

Результаты определения коэффициентов линейного уравнения $SIGMA = a_1 + a_2 T$ методом линейной регрессии

```
8    80.0          22.2000          22.0884          1.1160e-01
9    90.0          21.2000          21.0529          1.4707e-01
_____

3. Средняя погрешность описания в точке ( er ) = 0.1461 эрг/см^2
4. Минимальное значение суммарного квадратичного критерия рассогласования ( summ ) = 0.2668
5. Минимальное значение суммарного критерия рассогласования ( norm_summ ) = 0.5165
6. Остаточная дисперсия ( SR) = 0.0381
7. Суммарное квадратичное отклонение от среднего значения опытных данных (SES) = 53.8764
8. Дисперсия среднего значения опытных данных ( SE) = 6.7345
9. Коэффициент детерминации ( KDET1) = 0.9950
10. Индекс регрессии ( RKDET2) = 0.9972
-------------------------------------------------------------------
Коэффициенты уравнения определены путем решения линейной системы уравнений
-------------------------------------------------------------------
Относительный коэффициент обусловленности линейной системы уравнений (obusl) = 1.873e+04
-------------------------------------------------------------------
-------------------------------------------------------------------
-------------------------------------------------------------------
```

Рис. 3.146 (окончание)

Результаты определения коэффициентов линейного уравнения $SIGMA = a_1 + a_2 T$ методом линейной регрессии

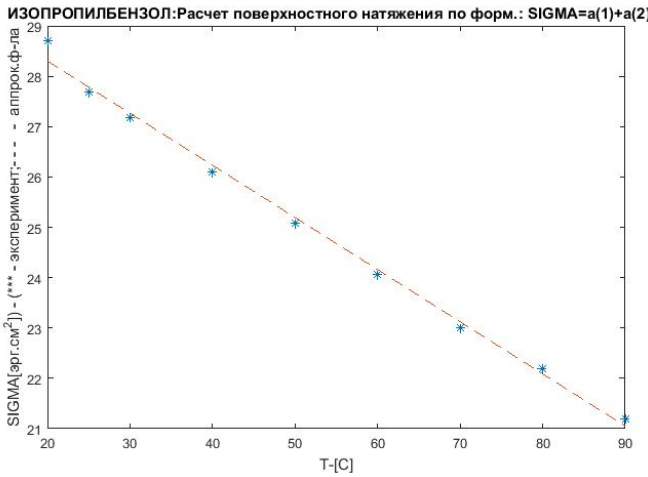


Рис. 3.147

Графическое представление описания данных зависимости поверхностного натяжения изопропанола от температуры с использованием линейного уравнения $SIGMA = a_1 + a_2 T$

Следует отметить, что матричная формула (3.197) применима для параметрической идентификации любых линейных и линеаризованных уравнений и для ее корректного использования необходимо правильно сформировать матрицу $\overline{\overline{\Phi}}$ и вектор $\overline{y}^{\text{эксп}}$ (3.196).

В среде MATLAB реализован решатель “polyfit” (priznak = 2 в программе на рис. 3.143–3.145), позволяющий определять коэффициенты многочлена произвольной степени по формулам (3.196) и (3.197) по массиву данных:

$$x = [x_1, x_2, \ldots, x_n]; y = [y_1, y_2, \ldots, y_n], \qquad (3.199)$$

который записывается в виде

$$a = polyfit(x, y, m), \tag{3.200}$$

где a — вектор-строка искомых коэффициентов; x — вектор-строка данных о независимых переменных (3.198); m — степень многочлена.

В отличие от многочлена (3.188), для этого решателя многочлен степени m записывается в виде

$$y^{\text{расч}} = a_{m+1}x^m + a_m x^{m-1} + \ldots + a_2 x + a_1. \tag{3.201}$$

Поэтому для решаемой задачи определения коэффициентов многочлена первой степени, описывающего зависимость поверхностного натяжения изопропанола от температуры, формула (3.192) имеет вид

$$SIGMA = a_1 T + a_2, \tag{3.202}$$

и, соответственно, обращения к выполнению действий, предусмотренных в решателе “polyfit”, имеют вид (рис. 3.145 — priznak = 2):

$$a = polyfit(T, SIGMA, 1). \tag{3.203}$$

Для определения расчетных значений зависимой переменной используется функция polyval в виде (рис. 3.145 — priznak = 2):

$$SIGMA\_cal = polyval(a, T). \tag{3.204}$$

В этом случае результаты решения такие же, как при определении коэффициентов, когда priznak = 1. Однако порядок коэффициентов в найденном векторе $\bar{a} = [a_1, a_2]$ будет не такой, как в предыдущем случае, т. е. будет соответствовать (3.202), а не (3.192) (рис. 3.146).

На рисунке 3.146 представлен анализ результатов параметрической идентификации уравнения $SIGMA = a_1 + a_2 T$ (priznak = 1). Он включает в себя следующие параметры.

1. Значения коэффициентов:

$$\begin{aligned} a_1^{calc} &= 30.3721, \\ a_2^{calc} &= -0.1035. \end{aligned} \tag{3.205}$$

2. Распределение ошибок расчетов в каждой экспериментальной точке:

$$\begin{gathered} der_i = SIGMA\_exp_i - SIGMA\_calc_i, \\ SIGMA\_calc_i = a_1 + a_2 T, \\ i = 1, \ldots, N. \end{gathered} \tag{3.206}$$

3. Среднюю погрешность расчетов в отдельной экспериментальной точке:

$$er = \frac{\sum_{i=1}^{N} \left| SIGMA\_exp_i - SIGMA\_calc_i \right|}{N} = 0.146 \text{ эрг/см}^2. \tag{3.207}$$

4. Минимальное значение суммарного по всем экспериментальным точкам квадратичного критерия рассогласования экспериментальных и расчетных значений *SIGMA* (*summ*):

$$summ = \sum_{i=1}^{N} \left(der_i \right)^2 = 0.2668. \tag{3.208}$$

5. Минимальное значение суммарного по всем экспериментальным точкам рассогласования экспериментальных и расчетных значений *SIGMA* (*summ*):

$$sum = \sqrt{summ}. \tag{3.209}$$

6. Остаточную дисперсию (SR):

$$SR = \frac{\sum_{i=1}^{N} \left(der_i \right)^2}{N - p} = 0.0381, \tag{3.210}$$

где p — число коэффициентов ($p = 2$).

7. Суммарное квадратичное отклонение от среднего значения опытных данных зависимой переменной:

$$SES = \sum_{i=1}^{N} \left(SIGMA\_exp_i - SIGMA\_exp^{cp} \right)^2 = 53.8764, \tag{3.211}$$

где $SIGMA\_exp^{cp} = \frac{\sum_{i=1}^{N} SIGMA\_exp_i}{N}$.

8. Дисперсию среднего значения (SE):

$$SE = \frac{SES}{N - 1} = 6.7345. \tag{3.212}$$

9. Коэффициент детерминации (KDET1):

$$KDET1 = 1 - \frac{summ}{SES} = 0.995. \tag{3.213}$$

10. Индекс регрессии (RKDET2):

$$RKDET2 = \sqrt{1 - \frac{summ}{SES}} = 0.9973. \tag{3.214}$$

Так как для определения коэффициентов уравнения (3.192) решается система линейных алгебраических уравнений (СЛАУ) (3.195), то определяется также коэффициент относительной обусловленности СЛАУ (*obusl*) для случая, когда priznak = 1 (рис. 3.146):

$$obusl = norm\left(\overline{\overline{\Phi}}^T \overline{\overline{\Phi}} \right) \cdot norm\left(\overline{\overline{\Phi}}^T \overline{\overline{\Phi}} \right)^{-1} = 1.873e + 04,$$

где norm — нормы прямой и обратной матриц $\overline{\overline{\Phi}}^T \overline{\overline{\Phi}}$.

Коэффициенты уравнения (3.192) могут быть определены не только путем решения системы линейных уравнений, но и прямым методом минимизации критерия (3.193) (priznak = 3, 4, 5):

$$min\sum_{i=1}^{N}\left(a_1 + a_2 T_1 - SIGMA\_\exp_i\right)^2. \qquad (3.215)$$

В результате оптимизационная задача (3.215) формулируется следующим образом: необходимо определить такие значения коэффициентов a_1 и a_2 из области их допустимых значений a_1^{don} и a_2^{don}, которые обеспечат минимальное значение критерия (3.193). В общем случае таким образом формулируется задача параметрической идентификации нелинейного уравнения. Если рассматривать линейное уравнение как частный случай нелинейного уравнения, то процедура (3.215) подходит и для линейных уравнений*.

Определение коэффициентов нелинейных уравнений в частном случае линейных уравнений может быть выполнено также с использованием следующих решателей MATLAB:

— “fminsearch” (priznak = 3 — рис. 3.144): минимизируемый критерий рассогласования экспериментальных и расчетных значений зависимой переменной (3.193) задается в отдельном *m*-файле extr.m (3.148).

```
function zcrit = extr(a)
%Программный код файла extr.m — расчет критерия рассогласования расчетных и
%экспериментальных значений методом наименьших квадратов
%-----------------------------------------------------------
%Программа включает следующие файлы:
GLAV_THERMO_5.m+DATA.m+coef_calcul.m+extr.m+mnk.m+REPORT.m
%-----------------------------------------------------------
%Программа расчета коэффициентов аппроксимационной зависимости
%поверхностного натяжения изопропилбензола от температуры (T) различными методами
%-----------------------------------------------------------------------
global SIGMA T;
zc=0;
for i=1:length(T)
   zc=zc + (SIGMA(i)-(a(1)+a(2)*T(i)))^2;
end
zcrit=zc;
end
```

Рис. 3.148

Программный код файла формирования критерия минимизации при параметрической идентификации уравнения $SIGMA = a_1 + a_2 T$ с применением решателей “fminsearch” и “fmincon”

Обращение к выполнению алгоритма решения оптимизационной задачи имеет вид (рис. 3.145):

$$[aregr, crit, pr, out] = fminsearch(‘extr’, aregr), \qquad (3.216)$$

* При определении коэффициентов линейных уравнений по возможности используют методы линейной регрессии (рис. 3.144, 3.145).

где aregr справа — начальное приближение коэффициентов, слева — результат определения коэффициентов;

— “fmincon” (priznak = 4 — рис. 3.146): минимизируемый критерий задается тем же файлом extr.m (рис. 3.148). Обращение к выполнению алгоритма решения оптимизационной задачи имеет такой же вид, как в случае priznak = 3, только вместо решателя “fminsearch” записывается “fmincon” (рис. 3.145):

$$[aregr, crit, pr, out] = fmincon(‘extr’, aregr); \qquad (3.217)$$

— “lsqcurvefit” (priznak = 5 — рис. 3.145): минимизируемый критерий сформирован в самом решателе в виде критерия квадратичного рассогласования экспериментальных и расчетных значений зависимой переменной (3.193), а в отдельном файле mnk.m (рис. 3.149) формируется вектор-функция с конкретным видом уравнений для расчета значений зависимых переменных в каждой экспериментальной точке.

```
function SIGMA_C=mnk(ao,T)
%Программный код файла mnk.m — задание вида аппроксимируемой функции
%уравнения зависимости поверхностного натяжения вещества от температуры SIGMA_C=ao(1)+ao(2)*T
%-----------------------------------------------------------
%Программа включает следующие файлы: GLAV_THERMO_5.m+DATA.m+coef_calcul.m+extr.m+mnk.m+REPORT.m
%-----------------------------------------------------------
%Программа расчета коэффициентов аппроксимационной зависимости
%поверхностного натяжения изопропилбензола от температуры (T) различными методами
%-----------------------------------------------------------
for i=1:length(T)
SIGMA_C(i)=ao(1)+ao(2)*T(i);
end
end
```

Рис. 3.149

Программный код файла формирования вектор-функции $SIGMA_i = aa_1 + aa_2 T_i$ ($i = 1, \ldots, N$) для всех опытных точек при параметрической идентификации уравнения $SIGMA = a_1 + a_2 T$ с применением решателя “lsqcurvefit”

Обращение к выполнению алгоритма решения оптимизационной задачи имеет вид (рис. 3.145):

$$[AA, crit] = lsqcurvefit(‘mnk’, aregr, x, y), \qquad (3.218)$$

где *aregr* — начальные приближения для определения искомых коэффициентов *AA*; *x* — массив независимых переменных в разных экспериментальных точках; *y* — массив зависимой переменной в разных экспериментальных точках; *AA* = = [*a*(1), *a*(2)] — определяемые оптимальные значения коэффициентов.

Для задачи определения коэффициентов a_1 и a_2 зависимости $SIGMA = a_1 + + a_2 T - x = T^{эксп}$ и $y = SIGMA^{эксп}$ (рис. 3.144 и 3.145).

Следует отметить, что при использовании решателей “fminsearch” (3.216) и “fmincon” (3.217) четвертый параметр в квадратных скобках решателей — в данном случае *out*, позволяет получить информацию о параметрах вычислительного процесса. Так, для определения параметров уравнения (3.192) с применением решателя “fminsearch” с алгоритмом Нелдера — Мида потребовалось 73 итерации, 137 расчетов целевой функции (рис. 3.150), а с применением ре-

шателя “fmincon” с алгоритмом interior-point — 6 итераций и 26 расчетов целевой функции.

```
aregr = 30.3721   -0.1035

crit = 0.2668

pr = 1

out =
    iterations: 73
     funcCount: 137
     algorithm: 'Nelder-Mead simplex direct search'
       message: 'Optimization terminated:
 the current x satisfies...'
```

Рис. 3.150

Данные о параметрах вычислительного процесса при определении коэффициентов уравнения $SIGMA = aregr_1 + aregr_2 T$ по массиву экспериментальных данных с применением решателя “fminsearch”

```
crit = 0.2668

pr = 1

out =
        iterations: 6
         funcCount: 26
    constrviolation: 0
          stepsize: 3.1626e-05
         algorithm: 'interior-point'
     firstorderopt: 1.3784e-07
      cgiterations: 1
           message: 'Local minimum found that satisfies the·const...'
```

Рис. 3.151

Данные о параметрах вычислительного процесса при определении коэффициентов уравнения $SIGMA = aregr_1 + aregr_2 T$ по массиву экспериментальных данных с применением решателя “fmincon”

Результаты параметрической идентификации уравнения зависимости поверхностного натяжения изопропанола от температуры $SIGMA = a_1 + a_2 T$ пятью разными способами (priznak = 1, 2, 3, 4, 5 — рис. 3.144), как и следовало ожидать, дают одинаковые результаты (рис. 3.146).

Необходимо иметь в виду, что решатели “polyfit” и “lsqcurvefit” (рис. 3.144, 3.145) могут использоваться только для уравнений с одной независимой переменной: “polyfit” — для уравнений многочленов любой степени, “lsqcurvefit” — для любых линейных и нелинейных по коэффициентам уравнений. При этом для обоих решателей для решения оптимизационной задачи поиска минимума целевой функции используется один и тот же критерий метода наименьших квадратов (3.193): в случае “polyfit” коэффициенты определяются исходя из необходимого условия существования экстремума функции многих переменных путем решения системы линейных алгебраических уравнений

(СЛАУ), а в случае “lsqcurvefit” — путем непосредственного нахождения минимума целевой функции (3.193).

При параметрической идентификации линейных по коэффициентам уравнений с несколькими независимыми переменными рекомендуется использовать известные методы решения систем линейных алгебраических уравнений (СЛАУ) (priznak = 1 — рис. 3.145), когда система линейных уравнений хорошо обусловлена (3.214). В этом случае при определении параметров уравнений может использоваться критерий МНК с весовыми коэффициентами (3.140), что позволяет учитывать важность и вклад отдельных опытных точек, в частности особых точек, в критерий — целевую функцию, подлежащую минимизации.

Решатели “fminsearch” и “fmincon” (priznak = 3, 4 — рис. 3.145) при решении задач параметрической идентификации являются универсальными в том смысле, что они позволяют решать задачи определения коэффициентов:

- линейных и нелинейных уравнений с одной и несколькими независимыми переменными;
- самостоятельно создавать целевые функции ((3.140)–(3.142)) (рис. 3.148), подлежащие минимизации;
- выбирать оптимальные алгоритмы поиска минимума, как, например, для решателя “fmincon” (priznak = 4 — рис. 3.145);
- регулировать параметры алгоритмов оптимизации, например, точность вычислений, максимальное число итераций и т. п.

3.5.3.5. Применение решателей “fminbnd” и “fminsearch” для оптимизации процессов в химических реакторах (первый тип оптимизационных задач)

Для оптимизации процессов химических превращений в реакторах необходимо:

- располагать адекватной моделью протекающего в реакторе химико-технологического процесса, которая обычно представляет собой реализованный на компьютере алгоритм решения уравнений математического описания процессов в реакторе;
- сформировать целевую функцию (технологический, экономический, термодинамический и т. п. критерии оптимальности), зависящую от технологических, физико-химических и режимных параметров процесса;
- выбрать соответствующий решатель, в данном случае “fminbnd” или “fminsearch”, который совместно с реализованным алгоритмом компьютерной модели позволяет решить рассматриваемую оптимизационную задачу путем определения минимума или максимума (в зависимости от критерия оптимальности) целевой функции.

При этом задача оптимизации процесса в реакторе превращается в стандартную экстремальную задачу вычислительной математики — поиск максимума или минимума целевой функции.

3.5.3.5.1. Определение оптимального времени пребывания реакционного потока в проточном изотермическом реакторе с мешалкой

Математическая модель

Рассматривается стационарный режим процесса в проточном реакторе с мешалкой при постоянной температуре, в котором протекает простейшая последовательная реакция:

$$A \xrightarrow{k_1} P \xrightarrow{k_2} S.$$

При математическом описании процесса в реакторе структура движущегося потока описывается моделью идеального перемешивания, и с учетом химической реакции получается математическая модель в виде системы линейных алгебраических уравнений (СЛАУ) вида:

$$\begin{aligned} &n^{(0)}x_A^{(0)} - nx_A + N\left(-k_1 x_A\right) = 0, \\ &n^{(0)}x_P^{(0)} - nx_P + N\left(k_1 x_A - k_2 x_P\right) = 0, \\ &n^{(0)}x_S^{(0)} - nx_S + N\left(k_2 x_P\right) = 0, \\ &x_A + x_P + x_S = 1, \end{aligned} \tag{3.219}$$

где $n^{(0)}$, $x_A^{(0)}$, $x_P^{(0)}$, $x_S^{(0)}$ — мольный расход и соответственно мольные доли компонентов в поступающем в реактор реакционном потоке; n, x_A, x_P, x_S — мольный расход и соответственно мольные доли компонентов в продуктовом потоке; N — мольный объем реакционной смеси; k_1, k_2 — кинетические константы первой и второй реакций.

С учетом последнего стехиометрического соотношения системы уравнений (3.219) и уравнений, описывающих скорости химических реакций: $n^{(0)} = n$, время пребывания реакционной смеси в реакторе определяется по формуле

$$\tau = \frac{N}{n}. \tag{3.220}$$

В результате с учетом того, что $x_A^{(0)} = 1$, $x_P^{(0)} = 0$, $x_S^{(0)} = 0$, система линейных уравнений (3.219) для определения состава продуктового потока при заданном времени пребывания (τ) и известных кинетических константах (k_1, k_2) принимает вид:

$$\begin{aligned} &\frac{x_A^{(0)} - x_A}{\tau} - k_1 x_A = 0 \\ &-\frac{x_P}{\tau} + k_1 x_A - k_2 x_P = 0 \end{aligned} \tag{3.221}$$

или в матричном виде для систем линейных алгебраических уравнений (СЛАУ) ($\overline{\overline{A}}\,\overline{x} = \overline{b}$):

$$\begin{bmatrix} (1+k_1\tau) & 0 \\ k_1\tau & (-1-k_2\tau) \end{bmatrix} \begin{bmatrix} x_A \\ x_P \end{bmatrix} = \begin{bmatrix} x_A^{(0)} \\ 0 \end{bmatrix}, \tag{3.222}$$

где

$$\overline{\overline{A}} = \begin{bmatrix} (1+k_1\tau) & 0 \\ k_1\tau & (-1-k_2\tau) \end{bmatrix};\ \overline{b} = \begin{bmatrix} x_A^{(0)} \\ 0 \end{bmatrix};\ \overline{x} = \begin{bmatrix} x_A \\ x_P \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}.$$

Система (3.221) может быть решена аналитически или в общем случае (3.222) методом обратной матрицы, что и реализовано в расчетном модуле программы (рис. 3.154).

Программные коды файлов решения рассматриваемой оптимизационной задачи приведены на рисунках 3.152–3.154. Исходные данные для решения системы уравнений (3.222) при различных значениях времени пребывания (τ) представлены в программном коде файла (рис. 3.153). Скрипт главной управляющей программы, в котором задается изменение времени пребывания (τ) в диапазоне [tau_a ÷ tau_b] с шагом 0.5 и происходит обращение к решению задачи одномерной оптимизации — определению оптимального времени пребывания потока в реакторе — приведен на рисунке 3.153.

```
%-------------------------------------------------------------------------------------
%Программный код файла GLAV_maximum_phi_p.m — главная управляющая программа
%-------------------------------------------------------------------------------------
%Программа включает следующие
%файлы: GLAV_maximum_phi_p.m+DATA.m+model1_stat_tau.m+REPORT.m;
%-------------------------------------------------------------------------------------
%Программа расчета оптимального времени пребывания (tau — час) в
%изотермическом реакторе идеального смешения со стехиометрической схемой
%реакции A — P -S
%-------------------------------------------------------------------------------------
clc;
clear all;
close all;
global tau_opt phi_p_max tau_a tau_b xa xp  xaa xpp t;
DATA;
i=0;
for tau=tau_a:0.5:tau_b
i=i+1;
x= model1_stat_tau(tau);
t(i)=tau;
xaa(i)=xa;
xpp(i)=xp;
end
[tau_opt,phi_p]=fminbnd('model1_stat_tau',tau_a,tau_b);
phi_p_max=-phi_p;
REPORT;
```

Рис. 3.152

Программный код файла главной управляющей программы определения выхода целевого продукта *P* в изотермическом проточном реакторе с мешалкой при различном времени пребывания в нем реакционной смеси и нахождения оптимального времени пребывания реакционной смеси в реакторе

```
function DATA
%-------------------------------------------------------------------------------------------
%Программный код файла DATA.m — задание исходной информации для расчетов;
%-------------------------------------------------------------------------------------------
%Программа включает следующие
%файлы: GLAV_maximum_phi_p.m+DATA.m+model1_stat_tau+REPORT.m;
%-------------------------------------------------------------------------------------------
%Программа расчета оптимального времени пребывания (tau — час) в
%изотермическом реакторе идеального смешения со стехиометрической схемой
%реакции A — P -S
%-------------------------------------------------------------------------------------------
global xa0 k1 k2 tau_a tau_b priz_print;
%1. Концентрация реагента A в долях в реакции A-P-S (мольные доли)
xa0=1;
%3. Константы скоростей реакций (час^(-1))
k1=0.35;k2=0.13;
%3. Левая граница поиска оптимального времени пребывания в реакторе (час)
tau_a=1;
%4. Правая граница поиска оптимального времени пребывания в реакторе (час)
tau_b=10;
%Форма вывода отчета о работе программы:
%1. Если priz_print=0 (задается в файле DATA.m) — отчет выводится в командное окно MATLAB;
%3. Если priz_print=1 (задается в файле DATA.m) — отчет выводится в текстовый файл,
%который размещается в той же папке MATLAB, в которой находятся все m-файлы программы;
priz_print=1;
end
```

Рис. 3.153

Программный код файла исходных данных для определения оптимального времени пребывания реакционной смеси в проточном реакторе с мешалкой

```
function phi_p= model1_stat_tau(tau)
%-------------------------------------------------------------------------------------------
%Программный код файла model1_stat_tau — расчет выхода продукта P
%-------------------------------------------------------------------------------------------
%Программа включает следующие
%файлы: GLAV_maximum_phi_p.m+DATA.m+model1_stat_tau.m+REPORT.m;
%-------------------------------------------------------------------------------------------
%Программа расчета оптимального времени пребывания (tau — час) в
%изотермическом реакторе идеального смешения со стехиометрической схемой
%реакции A — P — S
%-------------------------------------------------------------------------------------------
global xa0 k1 k2 xa xp;
a(1,1)=1+k1*tau;a(1,2)=0;b(1)=xa0;
a(2,1)=tau*k1;a(2,2)=(-1-tau*k2); b(2)=0;
%Определение выходных параметров модели
x=inv(a)*b';xa=x(1);xp=x(2);
%Вычисление критерия оптимальности (целевой функции);
phi_p=-x(2)/xa0;
end
```

Рис. 3.154

Программный код файла решения системы уравнений математического описания процесса при различном значении времени пребывания потока в изотермическом проточном реакторе с мешалкой и определения целевой функции (phi_p) при оптимизации реактора

Целевая функция (критерий оптимальности)

Используется технологический критерий оптимальности — выход целевого продукта P, определяемый по формуле, с учетом того, что $n = n^{(0)}$:

$$\Psi_P = \frac{nx_P}{n^{(0)}x_A^{(0)}} = \frac{x_P}{x_A^{(0)}}. \qquad (3.223)$$

В соответствии с оптимизационной задачей следует найти максимум критерия (3.223) и соответствующее ему оптимальное значение времени пребывания в диапазоне [tau_a ÷ tau_b]. Так как решатели MATLAB всегда рассчитывают минимум целевой функции, то для решателя “fminbnd” (рис. 3.152) задается Ψ_P (3.223) с противоположным знаком (рис. 3.154):

$$\Psi_P = -\frac{x_P}{x_A^{(0)}}, \qquad (3.224)$$

минимум которой и определяется, что соответствует максимуму целевой функции (3.223).

Выбор решателя

Так как решается задача одномерной оптимизации с определением оптимального времени пребывания потока в реакторе (tau_opt) в определенном диапазоне [tau_a ÷ tau_b], то выбирается решатель “fminbnd” (рис. 3.152).

Результаты расчетов в табличном виде приведены в Приложении (табл. П.8). Графическая зависимость мольной доли целевого продукта *P* от времени пребывания (tau) потока в реакторе представлена на рисунке 3.155.

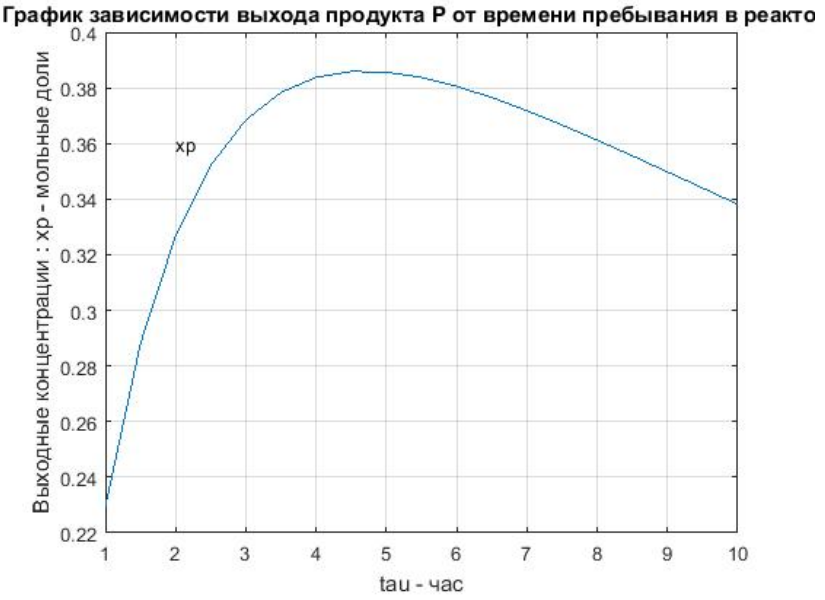


Рис. 3.155

Зависимость мольной доли целевого продукта *P* от времени пребывания потока реакционной смеси в изотермическом проточном реакторе с мешалкой, в котором протекает реакция $A \rightarrow P \rightarrow S$

3.5.3.5.2. Определение оптимальной температуры в проточном изотермическом реакторе с мешалкой

Математическая модель

В этом случае структура движущегося потока представляется моделью идеального перемешивания и система уравнений математического описания процесса в реакторе, в котором протекает обратимая реакция $A \underset{k_2}{\overset{k_1}{\rightleftarrows}} P$, может быть записана в виде:

$$
\begin{gathered}
n^{(0)}x_A^{(0)} - nx_A + N\left(-k_1x_A + k_2x_P\right) = 0, \\
n^{(0)}x_P^{(0)} - nx_P + N\left(k_1x_A - k_2x_P\right) = 0, \\
k_1 = A_1 e^{-\frac{E_1}{RT}}, \\
k_2 = A_2 e^{-\frac{E_2}{RT}}, \\
x_A + x_P = 1,
\end{gathered}
\tag{3.225}
$$

где A_1, E_1 и A_2, E_2 — предэкспоненциальные множители и энергии активации первой и второй реакций, остальные обозначения такие же, как для системы (3.219).

С учетом последнего стехиометрического соотношения для концентраций компонентов в (3.225) и выражений для скоростей реакций $n^{(0)} = n$, введя в рассмотрение время пребывания (τ) реакционного потока в реакторе (3.220), систему (3.225) при $x_A^{(0)} = 1$ можно записать:

$$
\begin{gathered}
\left(1 + k_1\tau\right)x_A + \left(-k_2\tau\right)x_P = x_A^{(0)}, \\
(k_1t)x_A + (-1 - k_2t)x_P = 0,
\end{gathered}
\tag{3.226}
$$

где $k_i = A_i e^{-E_i/RT}$, $i = 1, 2$;

Решение системы двух линейных уравнений при разных температурах и известном τ, а также E_1, A_2, E_2 относительно концентраций A и P может быть выполнено аналитически или при записи уравнения (3.226) в матричном виде (СЛАУ) ($\overline{\overline{A}}\bar{x} = \bar{b}$):

$$
\overline{\overline{A}} = \begin{bmatrix} \left(1 + k_1\tau\right) & -k_2\tau \\ k_1\tau & \left(-1 - k_2\tau\right) \end{bmatrix}; \; \bar{b} = \begin{bmatrix} x_A^{(0)} \\ 0 \end{bmatrix},
\tag{3.227}
$$

методом обратной матрицы, что и реализовано с использованием программы (рис. 3.158).

Программные коды файлов решения рассматриваемой оптимизационной задачи приведены на рисунках 3.156–3.158. Исходные данные для решения системы уравнений (3.227) при различных значениях температуры представлены

в программном коде файла (рис. 3.157). Скрипт главной управляющей программы, в которой задается изменение температуры (T) в диапазоне [T_a ÷ T_b] с шагом 0.5 и происходит обращение к решению задачи одномерной оптимизации и определения оптимальной температуры, приведен на рисунке 3.156.

```
%-------------------------------------------------------------------
%Программный код файла GLAV_maximum2_phi_p.m — главная управляющая программа
%-------------------------------------------------------------------
%Программа включает следующие
%файлы: GLAV_maximum2_phi_p+DATA.m+model2_stat_T+REPORT.m
%-------------------------------------------------------------------
%Программа расчета оптимальной температуры (T — K) в
%изотермическом реакторе идеального смешения со стехиометрической схемой
%реакции A = P где A-P(k1=A(1)exp(-E(1)/R/T)); (P-A k2=A(2)exp(-E(2)/R/T))' );
%-------------------------------------------------------------------
clc;
clear all;
close all;
global T_opt phi_p_max T_a T_b TT xa xp xam xpm;
DATA;
i=0;
for T=T_a:0.5:T_b
    i=i+1;
  x= model2_stat_T(T);
  TT(i)=T;
xam(i)=xa;
xpm(i)=xp;
end
%Реализация алгоритма поиска максимума функции одной переменной
[T_opt,phi_p]=fminbnd('model2_stat_T',T_a,T_b);
phi_p_max=-phi_p;
REPORT;
```

Рис. 3.156

Программный код файла основной управляющей программы определения зависимости концентрации целевого продукта от температуры в изотермическом проточном реакторе с мешалкой и оптимального значения температуры

```
function DATA
%-------------------------------------------------------------------
%Программный код файла DATA.m — задание исходной информации для расчетов;
%-------------------------------------------------------------------
%Программа включает следующие
%файлы: GLAV_maximum2_phi_p+DATA.m+model2_stat_T+REPORT.m
%-------------------------------------------------------------------
%Программа расчета оптимальной температуры (T - K) в
%изотермическом реакторе идеального смешения со стехиометрической схемой
%реакции A = P где A-P(k1=A(1)exp(-E(1)/R/T)); (P-A k2=A(2)exp(-E(2)/R/T))' );
%-------------------------------------------------------------------
global xa0 A E R tau T_a T_b priz_print;
A(2,1)=zeros;E(2,1)=zeros;
%Концентрация реагента A в долях в реакции A-P-S (мольные доли)
xa0=1;
%Время пребывания в реакторе (мин)
tau=10;
%Параметры уравнения Аррениуса для первой реакции
% а) Предэкспоненциальный множитель (мин^(-1)):
A(1,1)=70;
%б) Энергия активации (кал/моль):
E(1,1)=2500;
%Параметры уравнения Аррениуса для второй реакции
% а) Предэкспоненциальный множитель (мин^(-1)):
A(2,1)=100;
%б) Энергия активации (кал/моль):
```

Рис. 3.157 (начало)

Программный код файла исходных данных для определения оптимальной температуры в изотермическом проточном реакторе с мешалкой

```
E(2,1)=5000;
%Левая граница температурного интервала исследования (К);
T_a=350;
%Правая граница температурного интервала исследования (К);
T_b=370;
%Универсальная газовая постоянная (кал/моль/К):
R=1.9872;
%Форма вывода отчета о работе программы:
%1. Если priz_print=0 (задается в файле DATA.m) — отчет выводится в командное
окно MATLAB;
%3. Если priz_print=1 (задается в файле DATA.m) — отчет выводится в текстовый
файл,
%который размещается в той же папке MATLAB, в которой находятся все m-файлы
программы;
priz_print=1;
end
```

Рис. 3.157 (окончание)

Программный код файла исходных данных для определения оптимальной температуры в изотермическом проточном реакторе с мешалкой

```
function phi_p= model2_stat_T(T)
%-------------------------------------------------------------------------------------
%Программный код файла model2_stat_T .m — расчет выходных концентраций
%продуктов из реактора и выхода целевого продукта P
%-------------------------------------------------------------------------------------
%Программа включает следующие
%файлы: GLAV_maximum2_phi_p+DATA.m+model2_stat_T+REPORT.m
%-------------------------------------------------------------------------------------
%Программа расчета оптимальной температуры (T — K) в
%изотермическом реакторе идеального смешения со стехиометрической схемой
%реакции A = P где A-P(k1=A(1)exp(-E(1)/R/T)); (P-A k2=A(2)exp(-E(2)/R/T))' );
%-------------------------------------------------------------------------------------
global xa0 A E R tau xa xp;
for i=1:2
   k(i)=A(i,1)*exp(-E(i,1)/R/T);
end
a=zeros(2,2);b=zeros(2,1);
a(1,1)=1+k(1)*tau;a(1,2)=-k(2)*tau;b(1)=xa0;
a(2,1)=tau*k(1);a(2,2)=(-1-tau*k(2)); b(2)=0;
%Определение выходных параметров модели
x=inv(a)*b;xa=x(1);xp=x(2);
phi_p=-x(2)/xa0;
end
```

Рис. 3.158

Программный код файла решения системы уравнений математического описания процесса в изотермическом проточном реакторе с мешалкой

Целевая функция (критерий оптимальности)

Используется технологический критерий (3.223).

Выбор решателя

Используется решатель для решения задач одномерной оптимизации “fminbnd” в диапазоне температур [T_a ÷ T_b] (рис. 3.156).

Результаты расчетов в табличном виде приведены в Приложении (табл. П.9). Графическая зависимость мольной доли целевого продукта P от температуры T в реакторе представлена на рисунке 3.159.

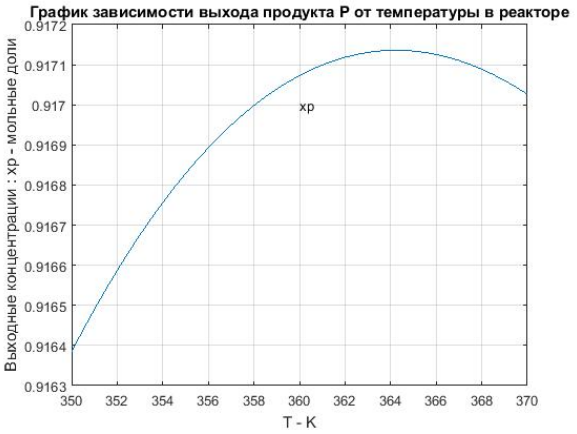


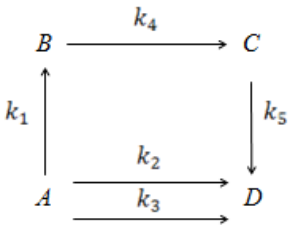
Рис. 3.159

Графическое представление зависимости мольной доли целевого продукта *P* от температуры в изотермическом проточном реакторе с мешалкой

3.5.3.5.3. Определение оптимальной температуры и оптимального времени пребывания реакционной смеси в изотермическом проточном реакторе

Математическая модель

Рассматривается стационарный режим процесса в проточном трубчатом реакторе при постоянной температуре, в котором протекает реакция:



При математическом описании процесса в реакторе структура движущегося потока описывается моделью идеального вытеснения, и с учетом химической реакции получается математическая модель в виде системы обыкновенных дифференциальных уравнений:

$$\begin{aligned} n\frac{dx_A}{d\ell} &= \frac{N}{L}\left(-k_1x_A - k_2x_A - k_3x_A\right) \\ n\frac{dx_B}{d\ell} &= \frac{N}{L}\left(k_1x_A - k_4x_B\right) \\ n\frac{dx_C}{d\ell} &= \frac{N}{L}\left(k_4x_B - k_5x_C\right) \\ n\frac{dx_D}{d\ell} &= \frac{N}{L}\left(k_2x_A + k_3x_A + k_5x_C\right) \end{aligned}, \qquad (3.228)$$

где L — условная длина реакционной зоны проточного реактора, которая изменяется в диапазоне $0 \le \ell \le L$; остальные обозначения те же, что и в (3.219).

Целесообразно в уравнениях системы (3.228) перейти от координаты ℓ $(0 \le \ell \le L)$ к координате времени пребывания потока в трубчатом реакторе, которая изменяется в диапазоне $t\_a \le \tau \le t\_b$, с использованием следующего соотношения:

$$d\tau = \frac{N}{n}\left(\frac{d\ell}{L}\right). \tag{3.229}$$

В результате система уравнений математического описания процесса (3.228) принимает вид:

$$\begin{aligned}
\frac{dx_A}{d\tau} &= -\left(k_1 + k_2 + k_3\right)x_A \\
\frac{dx_B}{d\tau} &= k_1 x_A - k_4 x_B \\
\frac{dx_C}{d\tau} &= k_4 x_B - k_5 x_C \\
\frac{dx_D}{d\tau} &= (k_1 + k_3)x_A - k_5 x_C
\end{aligned} \tag{3.230}$$

с начальными условиями, задаваемыми с учетом стехиометрических соотношений для мольных долей концентраций ($x_A + x_B + x_C + x_D = 1$):

$$x_A^{(0)} = 1;\ x_B^{(0)} = 0;\ x_C^{(0)} = 0;\ x_D^{(0)} = 0. \tag{3.230а}$$

При этом используются следующие температурные зависимости для кинетических параметров пяти химических реакций:

$$\begin{aligned}
k_1 &= 1.02\exp\left(-808.08\left(\frac{1}{T} - \frac{1}{658}\right)\right); \\
k_2 &= 0.93\exp\left(-707.07\left(\frac{1}{T} - \frac{1}{658}\right)\right); \\
k_3 &= 0.386\exp\left(-757.57\left(\frac{1}{T} - \frac{1}{658}\right)\right); \\
k_4 &= 3.28\exp\left(-505.05\left(\frac{1}{T} - \frac{1}{658}\right)\right); \\
k_5 &= 1.084\exp\left(-757.57\left(\frac{1}{T} - \frac{1}{658}\right)\right).
\end{aligned} \tag{3.231}$$

Решение системы дифференциальных уравнений (3.230) с начальными условиями (3.230а) и с учетом (3.231) при различных температурах (T) и разном времени пребывания реакционного потока в реакторе (τ) может быть выполне-

но с использованием решателя MATLAB “ode45” (рис. 3.160). При этом задание верхней границы интегрирования *t_b* системы (3.230) при постоянном значении расхода поступающего потока в реактор (*n*) соответствует заданию геометрического объема реакционной смеси (*V*), который при плотности выходного потока (*ρ*) может быть определен по формуле

$$V = \frac{t\_b \cdot n}{\rho}. \qquad (3.232)$$

Знание этой величины дает возможность определять другие геометрические параметры трубчатого реактора, в частности длину, площадь поперечного сечения и т. п. Поскольку результаты (3.228) получены для гидродинамической модели потока, соответствующей идеальному вытеснению, при выборе геометрии аппарата следует обеспечить условия движения потока, близкие к идеальному вытеснению.

Программные коды файлов решения рассматриваемой оптимизационной задачи приведены на рисунках 3.160–3.163. Исходные данные для решения системы уравнений (3.230) с (3.230а) и (3.231) приведены на рисунке 3.161. Скрипт основной управляющей программы, в котором происходит обращение к решению двумерной задачи оптимизации — одновременного определения оптимальной температуры и оптимального времени пребывания реакционного потока в реакторе, а также задается изменение параметров выходного потока из реактора в зависимости от времени пребывания (при оптимальной температуре) и от температуры (при оптимальном времени пребывания) представлен на рисунке 3.160.

```
%-------------------------------------------------------------------------------------------
%Программный код файла GLAV_optim3_PFR_reactor_T_t.m — основная управляющая программа
%-------------------------------------------------------------------------------------------
%Программа включает следующие
%файлы: GLAV_optim5_batch_reactor_T_t.m+DATA.m+model_pfr+difpravreactor.m+REPORT.m
%-------------------------------------------------------------------------------------------
%Программа расчета максимального выхода продукта C в
%изотермическом реакторе идеального вытеснения со стехиометрической схемой
%реакций A — B (k1); A - D (k2); A - D (k3); B - C (k4); C - D (k5); где ki=Ai*exp(-Ei*(1/T-1/TR),(i=1,...5);
%------------------ ----------------------------------------------------------------------
clc;
clear all;
close all;
global t_a t_b tt y n nt T0 T_opt phi_p_max tbb yr1 yr2 yr3 yr4 yr1T yr2T yr3T yr4T nT TT;
DATA;
regr=[T0,t0];
%Реализация алгоритма поиска максимума выхода продукта C как функции двух переменных: T -
%температура и t — время пребывания в реакторе идеального вытеснения
%методом многомерной оптимизации
[regrcalc,phi_p]=fminsearch('model_pfr',regr);
 phi_p_max=-phi_p;
 T_opt=regrcalc(1);
   t_b=regrcalc(2);
%1. Моделирование реактора в зависимости от времени пребывания при оптим. температуре

i=0;
t_a=0;
```

Рис. 3.160 (начало)
Программный код файла основной управляющей программы определения оптимальной температуры и оптимального времени пребывания реакционной смеси в изотермическом трубчатом реакторе

```
for tbb=t_b-0.8:0.05:t_b+0.3
   i=i+1;
      regr(1)=T_opt;
regr(2)=tbb;
phipT=model_pfr(regr);
yr1(i)=y(n,1);
yr2(i)=y(n,2);
yr3(i)=y(n,3);
yr4(i)=y(n,4);
%[tT,y]=ode45(@difpravreactor,[t_a t_b],[xa0 0 0 0]);
tt(i)=tbb;
end
nt=length(tt);
%3. Моделирование реактора в зависимости от температуры при оптим. времени пребывания
iT=0;
for TTC=T_opt-30:1:T_opt+30
   iT=iT+1;
      regr(1)=TTC;
regr(2)=regrcalc(2);
phipT=model_pfr(regr);
yr1T(iT)=y(n,1);
yr2T(iT)=y(n,2);
yr3T(iT)=y(n,3);
yr4T(iT)=y(n,4);
%[tT,y]=ode45(@difpravreactor,[t_a t_b],[xa0 0 0 0]);
TT(iT)=TTC;
end
nT=length(TT);
REPORT;
```

Рис. 3.160 (окончание)

Программный код файла основной управляющей программы определения оптимальной температуры и оптимального времени пребывания реакционной смеси в изотермическом трубчатом реакторе

```
function DATA
%-------------------------------------------------------------------------------------------
%Программный код файла DATA.m — задание исходной информации для расчетов;
%-------------------------------------------------------------------------------------------
%Программа включает следующие
%файлы: GLAV_optim3_PFR_reactor_T_t.m+DATA.m+model_pfr+difpravreactor.m+REPORT.m
%-------------------------------------------------------------------------------------------
%Программа расчета максимального выхода продукта C в
%изотермическом реакторе идеального вытеснения со стехиометрической схемой
%реакций A - B (k1); A - D (k2); A - D (k3); B - C (k4); C - D (k5); где ki=Ai*exp(-Ei*(1/T-1/TR),(i=1,...5);
%-------------------------------------------------------------------------------------------
global xa0 TR t_a t0 A1 E1  A2 E2 A3 E3 A4 E4 A5 E5 priz_print;

%Концентрация реагента A  (мольные доли)
xa0=1;
%Константы скоростей реакций: ki=Ai*exp(-Ei*(1/T-1/TR), где i=1,...5;
%a) Первая реакция —  k1:
%a.1) предэкспоненциальный множитель (c^(-1))
A1=1.02;
%a.2) экспоненциальный множитель (C)
E1=808.08;
%b) Вторая реакция — k2:
%b.1) предэкспоненциальный множитель (c^(-1))
A2=0.93;
%b.2) экспоненциальный множитель (C)
E2=707.07;
%c) Третья реакция —  k3:
%c.1) предэкспоненциальный множитель (c^(-1))
A3=0.386;
```

Рис. 3.161 (начало)

Программный код файла исходных данных программы определения оптимальной температуры и оптимального времени пребывания реакционной смеси в изотермическом трубчатом реакторе

```
%c.2) экспоненциальный множитель (C)
E3=757.57;
%d) Четвертая реакция — k4:
%d.1) предэкспоненциальный множитель (c^(-1))
A4=3.28;
%d.2) экспоненциальный множитель (C)
E4=505.05;
%e) Пятая реакция — k5:
%e.1) предэкспоненциальный множитель (c^(-1))
A5=1.084;
%e.2) экспоненциальный множитель (C)
E5=757.57;
%f. Базовая температура (C)
TR=658;
%3. Левая граница изменения времени пребывания в реакторе (c)
t_a=0;

%4. Начальное приближение при расчетах для правой границы изменения времени пребывания
в реакторе (c)
t0=0.5;
%5. Начальное приближение при расчетах для температуры в   реакторе (C)
T0=500;
%Форма вывода отчета о работе программы:
%1. Если priz_print=0 (задается в файле DATA.m) — отчет выводится в командное окно MATLAB;
%3. Если priz_print=1 (задается в файле DATA.m) — отчет выводится в текстовый файл,
%который размещается в той же папке MATLAB, в которой находятся все m-файлы программы;
priz_print=1;

end
```

Рис. 3.161 (окончание)

Программный код файла исходных данных программы определения оптимальной температуры и оптимального времени пребывания реакционной смеси в изотермическом трубчатом реакторе

```
function dy = difpravreactor(t,y)
%-------------------------------------------------------------------------------------------
%Программный код файла difpravreactor.m — расчет правых частей системы
%дифференциальных уравнений математического описания процесса
%-------------------------------------------------------------------------------------------
%Программа включает следующие
%файлы: GLAV_optim3_PFR_reactor_T_t.m+DATA.m+model_pfr+difpravreactor.m+REPORT.m
%----------------- ---------------------------------------------------------------------------
%Программа расчета максимального выхода продукта C в
%изотермическом реакторе идеального вытеснения со стехиометрической схемой
%реакций A - B (k1); A - D (k2); A - D (k3); B - C (k4); C - D (k5); где ki=Ai*exp(-Ei*(1/T-1/TR),(i=1,...5);
%----------------- ---------------------------------------------------------------------------
global k1 k2 k3 k4 k5;
dy=zeros(4,1);
dy(1)=-(k1+k2+k3)*y(1);
dy(2)=k1*y(1)-k4*y(2);
dy(3)=k4*y(2)-k5*y(3);
dy(4)=(k2+k3)*y(1)+k5*y(3);
end
```

Рис. 3.162

Программный код файла правых частей системы дифференциальных уравнений (3.230), описывающих процесс в изотермическом трубчатом реакторе

```
function phi_p=model_pfr(regr)
%-----------------------------------------------------------------------------------------------
%Программный код файла model+pfr.m — расчет выхода целевого продукта C при
%известной температуре T и времени пребывания t_b в реакторе идеального
%вытеснения
%-----------------------------------------------------------------------------------------------
%Программа включает следующие
%файлы: GLAV_GLAV_optim3_PFR_reactor_T_t.m+DATA.m+model_pfr+difpravreactor.m+REPORT.m
%-----------------------------------------------------------------------------------------------
%Программа расчета максимального выхода продукта C в
%изотермическом реакторе идеального вытеснения со стехиометрической схемой
%реакций A - B (k1); A - D (k2); A - D (k3); B - C (k4); C - D (k5); где ki=Ai*exp(-Ei*(1/T-1/TR),(i=1,...5);
%-----------------------------------------------------------------------------------------------
global xa0 t_a t_b t tbb y n T k1 k2 k3 k4 k5 TR A1 E1  A2 E2 A3 E3 A4 E4 A5 E5 priz;
T=regr(1);
t_b=regr(2);
if priz==1
   t_b=tbb;
end
k1=A1*exp(-E1*(1/T-1/TR));
k2=A2*exp(-E2*(1/T-1/TR));
k3=A3*exp(-E3*(1/T-1/TR));
k4=A4*exp(-E4*(1/T-1/TR));
k5=A5*exp(-E5*(1/T-1/TR));
[t,y]=ode45(@difpravreactor,[t_a  t_b],[xa0 0 0 0]);
n=length(t);
tk=t(n);yr(1)=y(n,1);yr(2)=y(n,2);yr(3)=y(n,3); yr(4)=y(n,4);
phi_p=-yr(3)/xa0;
end
```

Рис. 3.163

Программный код файла решения системы дифференциальных уравнений, описывающих процесс в изотермическом трубчатом реакторе

Программа решения системы дифференциальных уравнений (3.230) с правой частью (рис. 3.162) приведена на рисунке 3.163.

Целевая функция (критерий оптимальности)

Используется технологический критерий оптимальности — выход целевого продукта *C*. Так как в соответствии с кинетическими зависимостями в (3.228) и стехиометрическим соотношением о равенстве суммы мольных долей компонентов реакции единице расход реакционного потока в реакторе не изменяется, то выход целевого продукта *C* определяется по формуле ($x_A^{(0)} = 1$):

$$\Psi_C = \frac{x_C}{x_A^{(0)}} = x_C, \tag{3.233}$$

и эта величина зависит от температуры (T) и времени пребывания (τ) реакционного потока в реакторе (рис. 3.163).

Выбор решателя

В качестве решателя MATLAB для определения максимума Ψ_C (3.233) или минимума:

$$\Psi_C^* = -x_C, \tag{3.234}$$

что то же самое, что максимума (3.230), выбирается решатель “fminsearch” для определения минимума функции (3.231) многомерной оптимизационной задачи.

Результаты расчетов в табличном виде приведены в Приложении (табл. П.10). Графические зависимости мольной доли компонентов реакционной смеси в продуктовом потоке в зависимости от температуры и времени пребывания в реакторе приведены соответственно на рисунках 3.164 и 3.165. Как видно из этих графиков, температурная зависимость существенно меньше, чем зависимость от времени пребывания реакционного потока в реакторе.

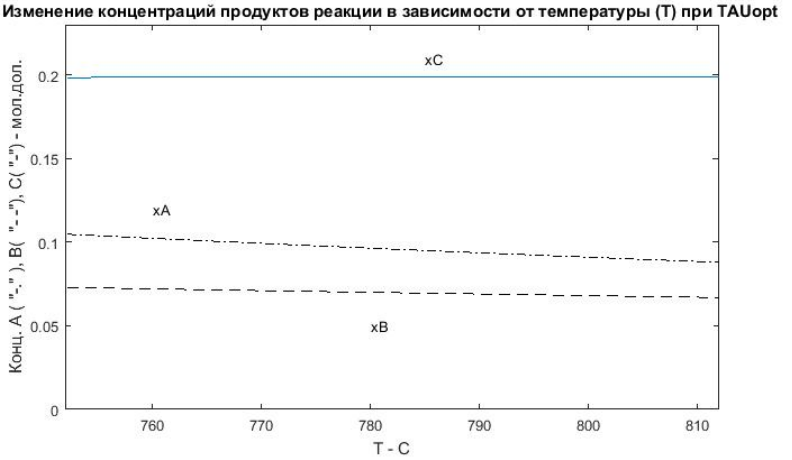


Рис. 3.164

График зависимости состава продуктового потока от температуры в изотермическом трубчатом реакторе при оптимальном времени пребывания потока

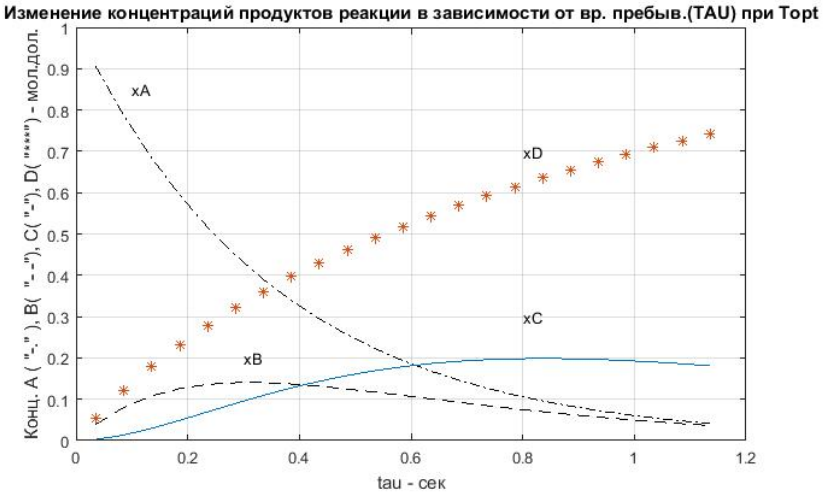


Рис. 3.165

График зависимости состава продуктового потока от времени пребывания в изотермическом трубчатом реакторе при оптимальной температуре потока

3.5.3.6. Применение алгоритмов решения систем линейных алгебраических уравнений (СЛАУ) и решателей “polyfit”, “lsqcurvefit”, “fminsearch”, “fmincon” для решения задач параметрической идентификации физико-химических и эмпирических математических моделей процессов (второй тип оптимизационных задач)

При решении задачи параметрической идентификации математической модели химико-технологического процесса определяются коэффициенты уравнений математического описания процесса исходя из массива экспериментальных данных о моделируемом объекте. Необходимо отметить, что выборка (статистика) опытных данных может состоять из различных и по объему, и по типу используемых физически измеренных величин, что, безусловно, влияет на рассчитанные значения параметров (коэффициентов) уравнений математического описания процесса. Поэтому обратную задачу параметрической идентификации принято относить к классу слабоформализованных задач как не имеющую единственного решения.

Другая проблема связана с выбором подлежащей оптимизации целевой функции оптимизационной задачи, которая должна характеризовать рассогласование между рассчитанными по модели и измеренными опытным путем параметрами процесса, так как не все требуемые физико-химические параметры процесса поддаются экспериментальным измерениям. Одновременно нужно выбирать как минимум один из приведенных ранее критериев (3.140)–(3.142) или их комбинации (часто с коррекциями) с корректными ограничениями первого и второго рода, что усугубляет неформализуемость задачи идентификации.

Применение статистических методов обработки результатов экспериментов методами корреляционного, регрессионного и дисперсионного анализа позволяет в какой-то степени, но далеко не в полной, компенсировать указанный недостаток.

Процедура расчета физической величины, входящей в критерий рассогласования (целевую функцию) рассчитанных по модели и экспериментальных величин, отличается для процессов, описываемых физико-химическими и эмпирическими моделями. В случае физико-химических моделей для определения расчетной физической величины, входящей в минимизируемую целевую функцию, порой приходится решать сложную систему нелинейных, дифференциальных или интегродифференциальных уравнений, в то время как для эмпирических моделей эта величина обычно рассчитывается по линейным или нелинейным относительно коэффициентов уравнениям, которые представляют собой аналитические формулы, записанные в явном виде.

Таким образом, для решения задачи параметрической идентификации математической модели необходимо:

- наличие алгоритма решения системы уравнений математического описания процесса — для физико-химических моделей, либо аналитических выражений, описывающих взаимосвязь физических параметров процесса друг от друга, — для эмпирических моделей;

- сформировать подлежащую минимизации целевую функцию (критерий рассогласования расчетных и экспериментальных физических величин) типа (3.140), (3.141), (3.142);
- выбрать решатель MATLAB (в данном случае из рассмотренного ограниченного набора решателей) с соответствующим алгоритмом минимизации целевой функции, в частности:

а) для линейных по коэффициентам эмпирических моделей произвольного вида — алгоритм решения системы линейных алгебраических уравнений (СЛАУ);

б) для линейных по коэффициентам эмпирических моделей полиномиального вида — решатель “polyfit”;

в) для нелинейных по коэффициентам эмпирических моделей произвольного вида, зависящих от одной независимой переменной, — решатель “lsqcurvefit”;

г) для физико-химических моделей произвольного вида — решатель “fminsearch” или “fmincon”.

3.5.3.6.1. Параметрическая идентификация кинетических констант химической реакции при постоянной температуре с физико-химической моделью процесса

Рассматривается химическая реакция $A \xrightarrow{k_1} P \xrightarrow{k_2} S$ при постоянной температуре и известном массиве экспериментальных данных об изменении мольных долей компонентов A (x_A) и P (x_P) во времени (t — ч):

№	t	x_A	x_P
1	0	1	0
2	0.104	0.969	0.037
3	0.523	0.855	0.141
4	1.023	0.736	0.251
5	3.773	0.322	0.545
6	4.023	0.299	0.554
7	5.523	0.191	0.577
8	6.273	0.153	0.573
9	9.023	0.064	0.508
10	10.00	0.049	0.477

Необходимо определить кинетические константы k_1 и k_2 при постоянной температуре реакции.

Физико-химическая математическая модель процесса

Математическая модель процесса в соответствии с законом действующих масс и с учетом стехиометрического соотношения $x_A + x_P + x_S = 1$ представляет собой систему двух обыкновенных дифференциальных уравнений вида:

$$\frac{dx_A}{dt} = -k_1 x_A$$
$$\frac{dx_P}{dt} = k_1 x_A - k_2 x_P \qquad (3.235)$$

с начальными условиями:

$$x_A^{(0)} = 1;\ \ x_P^{(0)} = 0.$$

Система (3.235) имеет аналитическое решение, что неприемлемо в общем случае для реакций со сложной стехиометрией. Поэтому она решается с использованием решателя MATLAB “ode45” (рис. 3.168) при различных кинетических коэффициентах k_1 и k_2.

Следует отметить, что для данных кинетических уравнений реакций и с учетом стехиометрического соотношения для концентраций компонентов ($x_A + x_P + x_S = 1$) система уравнений математического описания (3.235) справедлива и для периодического реактора с любой начальной суммарной загрузкой N, так как она не изменяется при протекании реакции и определяется по формуле

$$N = \sum_{i=A,P,S} N_i, \qquad (3.236)$$

где $N_i = Nx_i$, $i = A, P, S$ — число молей отдельных компонентов реакции.

Целевая функция (критерий рассогласования расчетных и опытных значений концентраций во всех экспериментальных точках)

Для определения коэффициентов k_1 и k_2 необходимо найти минимальное значение целевой функции вида (3.168):

$$Cr = \sum_{i=1}^{10} \left(x_{Ai}^{\text{расч}} - x_{Ai}^{\text{эксп}} \right)^2 + \left(x_{Pi}^{\text{расч}} - x_{Pi}^{\text{эксп}} \right)^2, \qquad (3.237)$$

так как массив экспериментальных данных состоит из десяти опытных точек.

Выбор решателя

Так как решается задача многомерной (двумерной — определяются k_1 и k_2) оптимизации, для поиска минимума используется решатель “fminsearch” (также можно было бы воспользоваться и решателем “fmincon”).

Программные коды файлов для решения рассматриваемой задачи параметрической идентификации, в том числе и для периодического изотермического реактора идеального перемешивания, приведены на рисунках 3.166–3.169. Исходные данные для расчетов представлены в файле на рисунке 3.167, а в главной управляющей программе (рис. 3.166) выполняется обращение к решателю “fminsearch” для определения минимума критерия (3.237), который формируется в файле на рисунке 3.168. В этом файле решается система дифференциальных уравнений (3.235), правая часть которой формируется в файле на рисунке 3.169.

```
%-------------------------------------------------------------------------------------------
%Программный код файла GLAV_ident4_batch_reactor.m — главная управляющая программа
%-------------------------------------------------------------------------------------------
%Программа включает следующие
%файлы: GLAV_ident4_batch_reactor.m+DATA.m+difpravreactor.m+Criterium.m+REPORT.m
%-------------------------------------------------------------------------------------------
%Программа определения кинетических констант по экспериментальным данным в
%изотермическом периодическом реакторе со стехиометрической схемой
%реакции A - P - S где A - P (k1); P - S (k2)
%-------------------------------------------------------------------------------------------
clc;
clear all;
close all;
global  n aaexp tt xae xpe k10 k20 crit;
DATA;
%pause
n=length(aaexp);
for i=1:n
tt(i)=aaexp(i,1);xae(i)=aaexp(i,2);xpe(i)=aaexp(i,3);
end
aregr=[k10 k20];
[aregr,crit,pr,out]=fminsearch('Criterium',aregr)
k1=aregr(1);k2=aregr(2);
REPORT;
```

Рис. 3.166

Программный код файла основной управляющей программы для определения кинетических коэффициентов химической реакции при постоянной температуре по экспериментальным данным

```
function DATA
%-------------------------------------------------------------------------------------------
%Программный код файла DATA.m — задание исходной информации для расчетов;
%-------------------------------------------------------------------------------------------
%Программа включает следующие
%файлы: GLAV_ident4_batch_reactor.m+DATA.m+difpravreactor.m+Criterium.m+REPORT.m
%-------------------------------------------------------------------------------------------
%Программа определения кинетических констант по экспериментальным данным в
%изотермическом периодическом реакторе со стехиометрической схемой
%реакции A - P - S, где A - P (k1); P - S (k2)
%-------------------------------------------------------------------------------------------
global xa0 k10 k20 t_a t_b aaexp priz_print;

%Концентрация реагента A в долях в реакции A-P-S (мольные доли)
xa0=1;
%Экспериментальные данные: для каждой опытной точки построчно — время (t-
%час), концентрация исходного компонента (A — мольн. доли), концентрация
%целевого компонента (P — мольн. доли);
aaexp=[0.000,1.000, 0.000; 0.104,0.969,0.037;0.523,0.855,0.141;1.023,0.736,0.251;
3.773,0.322,0.545;...
  4.023,0.299,0.554;5.523,0.191,0.577; 6.273,0.153,0.573;
9.023,0.067,0.50833;10.000,0.0498,0.477];

%Начальные приближения констант скоростей реакций (час^(-1))

k10=0.01;k20=1;
%3. Левая граница изменения времени в реакторе при определении констант скоростей
реакций(час)
t_a=0;

%4. Правая граница изменения времени в реакторе при определении констант скоростей
реакций(час)
t_b=10;

%5. Форма вывода отчета о работе программы:
```

Рис. 3.167 (начало)

Программный код файла исходных данных для определения кинетических коэффициентов химической реакции по опытным данным

```
%1. Если priz_print=0 (задается в файле DATA.m) — отчет выводится в командное окно
MATLAB;
%3. Если priz_print=1 (задается в файле DATA.m) — отчет выводится в текстовый файл,
%который размещается в той же папке MATLAB, в которой находятся все m-файлы про-
граммы;
priz_print=1;
end
```

Рис. 3.167 (окончание)

Программный код файла исходных данных для определения кинетических коэффициентов химической реакции по опытным данным

```
function Cr = Criterium(aregr)
%-------------------------------------------------------------------------------
%Программный код файла Criterium.m — расчет квадратичного критерия рассогласования между
%опытными и расчетными концентрациями компонентов реакции;
%-------------------------------------------------------------------------------
%Программа включает следующие
%файлы: GLAV_ident4_batch_reactor.m+DATA.m+difpravreactor.m+Criterium.m+REPORT.m
%-------------------------------------------------------------------------------
%Программа определения кинетических констант по экспериментальным данным в
%изотермическом периодическом реакторе со стехиометрической схемой
%реакции A - P - S где A - P (k1); P - S (k2)
%-------------------------------------------------------------------------------
global tt t_a t_b xa0 k1 k2 yar ypr xae  n xpe;
k1=aregr(1);
k2=aregr(2);
yar(1)=xa0;ypr(1)=0;
for i=2:n
 t_b=tt(i);

[t,y]=ode45(@difpravreactor,[t_a t_b],[xa0 0]);

nn=length(t);
yar(i)=y(nn,1);
ypr(i)=y(nn,2);
end

Cr=0;
for i=1:10
   Cr=Cr+(yar(i)-xae(i))^2+(ypr(i)-xpe(i))^2;
end

end
```

Рис. 3.168

Программный код файла для расчета целевой функции и решения системы дифференциальных уравнений при определении кинетических коэффициентов химической реакции при постоянной температуре

```
function dy = difpravreactor(t,y)
%-------------------------------------------------------------------------------
%Программный код файла difpravreactor.m — расчет правых частей системы
%дифференциальных уравнений математического описания процесса
%-------------------------------------------------------------------------------
%Программа включает следующие
%файлы: GLAV_ident4_batch_reactor.m+DATA.m+difpravreactor.m+Criterium.m+REPORT.m
%-------------------------------------------------------------------------------
%Программа определения кинетических констант по экспериментальным данным в
%изотермическом периодическом реакторе со стехиометрической схемой
%реакции A - P - S где A - P (k1); P - S (k2)
%-------------------------------------------------------------------------------
global k1 k2
```

Рис. 3.169 (начало)

Программный код файла для расчета правых частей системы дифференциальных уравнений при определении кинетических коэффициентов химической реакции при постоянной температуре

```
dy=zeros(2,1);
dy(1)=-k1*y(1);
dy(2)=k1*y(1)-k2*y(2);
end
```

Рис. 3.169 (окончание)

Программный код файла для расчета правых частей системы дифференциальных уравнений при определении кинетических коэффициентов химической реакции при постоянной температуре

Результаты определения кинетических коэффициентов химической реакции приведены в Приложении (табл. П.11), а графическое представление соответствия найденных по модели с помощью кинетических параметров экспериментальных значений физических величин представлено на рисунке 3.170.

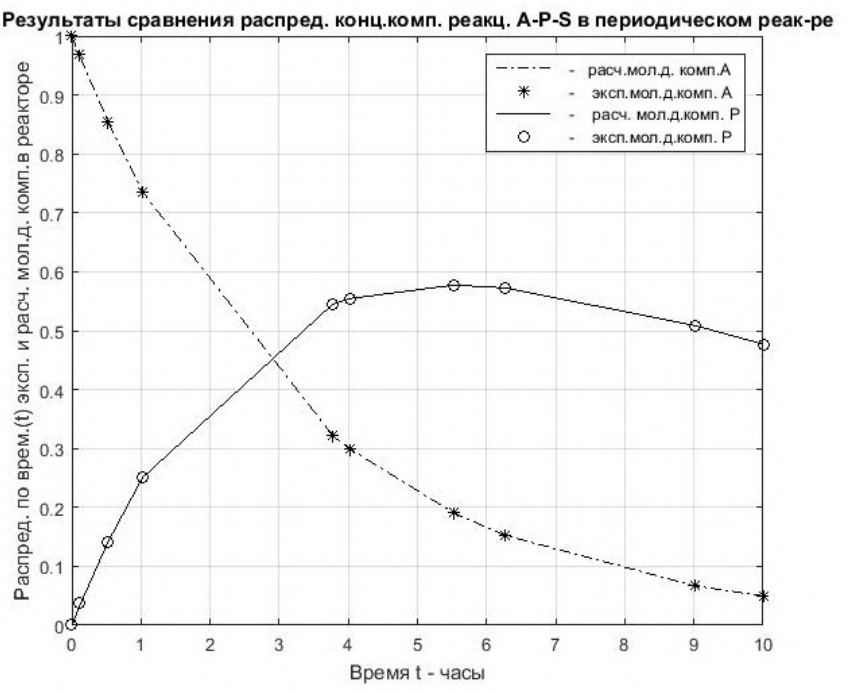


Рис. 3.170

Графическое представление результатов сравнения опытных и расчетных параметров при найденных кинетических коэффициентах химической реакции

3.5.3.6.2. Параметрическая идентификация эмпирической линеаризованной модели описания зависимости давления насыщенного пара индивидуального вещества от температуры

Эмпирическая математическая модель зависимости давления насыщенного пара ацетона от температуры представляется тремя уравнениями следующего вида:

$$P = exp\left(a_1 + a_2 T + a_3 T^2\right),$$
$$P = exp\left(a_1 + a_2 T + a_3 T^2 + a_4 T^3\right), \qquad (3.238)$$
$$P = \exp\left(a_1 + \frac{a_2}{T} + a_3 lnT + a_4 T^2\right),$$

где a_1, a_2, a_3, a_4 — искомые параметры (коэффициенты) уравнений зависимости давления (P) насыщенного пара ацетона от температуры (T).

Таблица экспериментальных данных для определения коэффициентов уравнений (3.238) имеет вид:

№	T	$P^{эксп}$
n/n	°C	мм рт. ст.
1	–2	60
2	7.7	100
3	23.7	200
4	39.5	400
5	56.5	760
6	78.6	1520
7	113	3800
8	144.5	7600

Выбор вида эмпирических уравнений принято относить к задачам структурной идентификации, решение которых требует дополнительных усилий и здесь не рассматриваются.

Все представленные в (3.238) уравнения могут быть легко линеаризованы относительно коэффициентов логарифмированием:

$$lnP = a_1 + a_2 T + a_3 T^2,$$
$$lnP = a_1 + a_2 T + a_3 T^2 + a_4 T^3, \qquad (3.239)$$
$$lnP = a_1 + a_2 \frac{1}{T} + a_3 lnT + a_4 T^2.$$

Это дает возможность при использовании критерия наименьших квадратов (МНК) на основе использования необходимых условий экстремума функции многих переменных (переменные в данном случае — это искомые коэффициенты) применять не поисковые итерационные методы их определения, а решать систему линейных алгебраических уравнений (СЛАУ) без итераций, например методом обратной матрицы.

Первые два уравнения (3.239) представляют собой многочлен соответственно второй и третьей степени, и для определения их коэффициентов можно использовать решатель MATLAB “polyfit” (рис. 3.173 — priznak = 2 и 4). При этом эти уравнения для использования этого решателя необходимо записать в следующем виде:

$$lnP = \tilde{a}_1 T^2 + \tilde{a}_2 T + \tilde{a}_3,$$
$$lnP = \tilde{a}_1 T^3 + \tilde{a}_2 T^2 + \tilde{a}_3 T + \tilde{a}_4. \qquad (3.240)$$

Для последнего уравнения (3.239) четыре коэффициента определяются решением системы линейных уравнений (рис. 3.173 — priznak = 5), полученной на основе учета необходимых условий существования экстремума функции многих переменных, в этом случае целевой функции следующего вида:

$$Cr = \sum_{i=1}^{8} \left(lnP_i^{\text{расч}} - lnP_i^{\text{эксп}} \right)^2. \tag{3.241}$$

Система линейных алгебраических уравнений (СЛАУ) для определения коэффициентов последнего уравнения (3.239):

$$lnP = a_1 + a_2 \frac{1}{T} + a_3 lnT + a_4 T^2 \tag{3.242}$$

получается, когда это выражение подставляется вместо $lnP^{\text{расч}}$ в критерий (3.241), который аналитически дифференцируется по всем четырем коэффициентам и приравнивается нулю:

$$\begin{gathered} \frac{\partial Cr}{\partial a_i} = 0, \\ i = 1,\ 2,\ 3,\ 4. \end{gathered} \tag{3.243}$$

Для решения СЛАУ (3.243) методом обратной матрицы предлагается использовать матричную формулу (рис. 3.173):

$$\bar{a} = \left(\overline{\overline{\Phi}}^T \overline{\overline{\Phi}} \right)^{-1} \overline{\overline{\Phi}}^T \bar{y}, \tag{3.244}$$

которая реализована в приведенной на рисунке 3.178, 1, 3, 5 программе.

В матричной формуле (3.244):

$$\bar{a} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} \text{ — искомые коэффициенты;}$$

$$\overline{\overline{\Phi}} = \begin{bmatrix} 1 & T_1^{-1} & lnT_1 & T_1^2 \\ 1 & T_2^{-1} & lnT_2 & T_2^2 \\ 1 & T_3^{-1} & lnT_3 & T_3^2 \\ 1 & T_4^{-1} & lnT_4 & T_4^2 \\ 1 & T_5^{-1} & lnT_5 & T_5^2 \\ 1 & T_6^{-1} & lnT_6 & T_6^2 \\ 1 & T_7^{-1} & lnT_7 & T_7^2 \\ 1 & T_8^{-1} & lnT_8 & T_8^2 \\ & & & \end{bmatrix} \text{ — матрица независимых переменных;} \tag{3.245}$$

$$y = \begin{bmatrix} lnP_1 \\ lnP_2 \\ lnP_3 \\ lnP_4 \\ lnP_5 \\ lnP_6 \\ lnP_7 \\ lnP_8 \end{bmatrix}$$ — вектор зависимых переменных.

С использованием формулы (3.244) (рис. 3.173) могут быть определены коэффициенты первых двух уравнений (3.239), которые представляют собой линеаризованные уравнения второй и третьей степени. Однако определение этих коэффициентов с использованием решателя “polyfit” (3.240) представляется более рациональным (priznak 2 и 4 на рис. 3.173).

Программные коды файлов для решения рассматриваемой задачи оптимизации приведены на рисунках 3.171–3.173. Исходные данные для расчетов представлены в файле на рисунке 3.172. В основной управляющей программе происходит обращение к программе расчета коэффициентов пятью различными способами в зависимости от параметра priznak = 1, 2, 3, 4, 5. Когда значение признака равно 1 или 3, коэффициенты уравнений многочленов второй или третьей степени определяются по формуле (3.244), когда признак равен двум или четырем, эти коэффициенты определяются с использованием решателя “polyfit”, для признака, равного пяти, определяются четыре коэффициента уравнения Риделя (3.242) по формуле (3.244) с учетом матрицы $\overline{\overline{\Phi}}$ и вектора $\overline{y}$ (3.245).

```
%-------------------------------------------------------------------------------------------
%Программный код файла GLAV_THERMO3.m  — основная управляющая программа
%-------------------------------------------------------------------------------------------
%Программа включает следующие файлы: GLAV_THERMO3.m+DATA.m+coef_calcul.m+REPORT.m
%-------------------------------------------------------------------------------------------
%Программа расчета коэффициентов аппроксимационных зависимостей
%давления насыщенного пара вещества (P) от температуры (T) методом линейной регрессии:
%1. Если priznak=1 (задается в файле DATA.m) — методом линеаризованной регрессии
%с получением уравнения вида: P=exp(a(1)+a(2)*T +a(3)*T^2);
%3. Если priznak=2 (задается в файле DATA.m) — с применением стандартных
%функций  и решателей MATLAB  с получением уравнения вида: P=exp(a(1)*T^2+a(2)*T +a(3));
%3. Если priznak=3 (задается в файле DATA.m) — методом линеаризованной регрессии
%с получением уравнения вида: P=exp(a(1)+a(2)*T +a(3)*T^2+a(4)*T^3);
%4. Если priznak=4 (задается в файле DATA.m) — с применением стандартных
%функций и решателей MATLAB  с получением уравнения вида:  P=exp(a(1)*T^3+a(2)*T^2 +a(3)*T
+a(4));
%5. Если priznak=5 (задается в файле DATA.m) — методом линеаризованной регрессии
%с получением уравнения вида: P=exp(a(1)+a(2)/TT(i)+a(3)*log(TT(i))+a(4)*TT(i)^2);;
%-------------------------------------------------------------------------------------------
%-------------------------------------------------------------------------------------------
clc;
clear all;
```

Рис. 3.171 (начало)

Программный код основной управляющей программы параметрической идентификации трех линеаризованных уравнений, описывающих температурную зависимость давления насыщенного пара ацетона

```
close all;
global  priznak
DATA;
coef_calcul(priznak);
REPORT
```

Рис. 3.171 (окончание)

Программный код основной управляющей программы параметрической идентификации трех линеаризованных уравнений, описывающих температурную зависимость давления насыщенного пара ацетона

```
function DATA
%-------------------------------------------------------------------------------------------
%Программный код файла DATA.m — задание исходных данных для расчетов
%-------------------------------------------------------------------------------------------
%Программа включает следующие файлы: GLAV_THERMO3.m+DATA.m+coef_calcul.m+REPORT.m
%-------------------------------------------------------------------------------------------
%Программа расчета коэффициентов аппроксимационных зависимостей
%давления насыщенного пара вещества (P) от температуры (T) методом линейной регрессии:
%1. Если priznak=1 (задается в файле DATA.m) — методом линеаризованной регрессии
%с получением уравнения вида: P=exp(a(1)+a(2)*T +a(3)*T^2);
%3. Если priznak=2 (задается в файле DATA.m) — с применением стандартных
%функций и решателей MATLAB с получением уравнения вида: P=exp(a(1)*T^2+a(2)*T +a(3));
%3. Если priznak=3 (задается в файле DATA.m) — методом линеаризованной регрессии
%с получением уравнения вида: P=exp(a(1)+a(2)*T +a(3)*T^2+a(4)*T^3);
%4. Если priznak=4 (задается в файле DATA.m) — с применением стандартных
%функций и решателей MATLAB с получением уравнения вида: P=exp(a(1)*T^3+a(2)*T^2 +a(3)*T +a(4));
%5. Если priznak=5 (задается в файле DATA.m) — методом линеаризованной регрессии
%с получением уравнения вида: P=exp(a(1)+a(2)/TT(i)+a(3)*log(TT(i))+a(4)*TT(i)^2);;
%-------------------------------------------------------------------------------------------
global T P comp priznak priz_print;
%1. Название индивидуального вещества:
comp='АЦЕТОН';
%-----------------------------------------------
%3. Используемые данные по зависимости поверхностного натяжения
 %индивидуального вещества — P(мм.рт.ст. )   от температуры T ( C ) :
T=[-3.0 7.7 23.7 39.5 56.5 78.6 113.0 144.5];
P=[60 100 200 400 760 1520 3800 7600];
%-----------------------------------------------
%3. Метод расчета коэффициентов методом линеаризованной регрессии путем задания
% значения параметра:
priznak=3;
%-------------------------------------------------------------------------------------------
% 4. Форма вывода отчета о работе программы:
%1. Если priz_print=0 (задается в файле DATA.m) — отчет выводится в командное окно MATLAB;
%3. Если priz_print=1 (задается в файле DATA.m) — отчет выводится в текстовый файл, который раз-
мещается в той же...
%папке MATLAB, в которой находятся все m-файлы программы;
priz_print=1;
end
```

Рис. 3.172

Программный код исходных экспериментальных данных для определения коэффициентов линеаризованных уравнений температурной зависимости давления насыщенного пара ацетона

```
function coef_calcul(priznak)
%-------------------------------------------------------------------------------------------
%Программный код файла coef_calcul.m — определение коэффициентов
%аппроксимирующих уравнений
%-----------------------------------------------------------------------------
```

Рис. 3.173 (начало)

Программный код файла для вычисления коэффициентов различных линеаризованных уравнений, описывающих температурную зависимость давления насыщенного пара ацетона

```
%Программа включает следующие файлы: GLAV_THERMO3.m+DATA.m+coef_calcul.m+REPORT.m
%-------------------------------------------------------------------------
%Программа расчета коэффициентов аппроксимационных зависимостей
%давления насыщенного пара вещества (P) от температуры (T) методом линейной регрессии:
%1. Если priznak=1 (задается в файле DATA.m) — методом линеаризованной регрессии
%с получением уравнения вида: P=exp(a(1)+a(2)*T +a(3)*T^2);
%3. Если priznak=2 (задается в файле DATA.m) — с применением стандартных
%функций и решателей MATLAB с получением уравнения вида: P=exp(a(1)*T^2+a(2)*T +a(3));
%3. Если priznak=3 (задается в файле DATA.m) — методом линеаризованной регрессии
%с получением уравнения вида: P=exp(a(1)+a(2)*T +a(3)*T^2+a(4)*T^3);
%4. Если priznak=4 (задается в файле DATA.m) — с применением стандартных
%функций и решателей MATLAB с получением уравнения вида: P=exp(a(1)*T^3+a(2)*T^2 +a(3)*T +a(4));
%5. Если priznak=5 (задается в файле DATA.m) — методом линеаризованной регрессии
%с получением уравнения вида: P=exp(a(1)+a(2)/TT(i)+a(3)*log(TT(i))+a(4)*TT(i)^2);;
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
global T P  a  N obusl TT PP PPL  Pcalc
obusl=0;
N=length(T);
TT=T';
PP=P';
PPL=log(PP);
switch priznak
   case 1
%1. Решение задачи численным методом линейной регрессии с использованием
%стандартных функций MATLAB
%TM=T';PM=P';

%Формирование матрицы коэффициентов линейной системы уравнений
for i=1:N
   for j=1:3
      FF(i,j)=TT(i)^(j-1);
   end
end

FFT=FF';
II=FFT*FF;
BB=FFT*PPL;
IIINV=inv(II);
a=IIINV*BB;
obusl=norm(II)*norm(IIINV);
for i=1:N
Pcalc(i,1)=exp(a(1)+a(2)*TT(i)+a(3)*TT(i)^2);
end


%-------------------------------------------------


   case 2

%3. Решение задачи с использованием стандартных решателей MATLAB
%Определение коэффициентов многочлена 2-ой степени Pcalc2m=a(1)*T^2+a(2)*T+a(3)
%решателем MATLAB  — polyfit
a=polyfit(TT,PPL,2);
%Определение расчетных значений поверхностного натяжения с применением
%найденных значений коэффициентов функцией MATLAB — polyval
%Pcalc=exp(polyval(a,TT));

%Определение суммарного среднеквадратичного отклонения расчетных и заданных
%значений поверхностного натяжений для всех аппроксимируемых данных
for i=1:N
Pcalc(i,1)=exp(a(1,1)*TT(i)^2+a(1,2)*TT(i)+a(1,3));
end


%-------------------------------------------------
```

Рис. 3.173 (продолжение)

Программный код файла для вычисления коэффициентов различных линеаризованных уравнений, описывающих температурную зависимость давления насыщенного пара ацетона

```
   case 3
%Формирование матрицы коэффициентов линейной системы уравнений
for i=1:N
   for j=1:4
      FF(i,j)=TT(i)^(j-1);
   end
end
FFT=FF';
II=FFT*FF;BB=FFT*PPL;
IIINV=inv(II);
obusl=norm(II)*norm(IIINV);
a=IIINV*BB;
for i=1:N
Pcalc(i,1)=exp(a(1)+a(2)*TT(i,1)+a(3)*TT(i,1)^2+a(4)*TT(i,1)^3);
end


   case 4
%Определение коэффициентов многочлена 2-ой степени Pcalc2m=a(1)*T^2+a(2)*T+a(3)
%решателем MATLAB  — polyfit
a=polyfit(TT,PPL,3);
%Определение расчетных значений поверхностного натяжения с применением
%найденных значений коэффициентов функцией MATLAB — polyval
Pcalc=polyval(a,TT);
%Определение суммарного среднеквадратичного отклонения расчетных и заданных
%значений поверхностного натяжений для всех аппроксимируемых данных
for i=1:N
%Pcalc(i,1)=exp(a(1)*TT(i)^3+a(2)*TT(i)^2+a(3)*TT(i)+a(4));
end
   case 5
      %Ридель
TT=TT+273.15;
PP=PP*101325/760;
PPL=log(PP);
for i=1:N

      FF(i,1)=1;FF(i,2)=1/TT(i);FF(i,3)=log(TT(i));FF(i,4)=TT(i)^2;

end
FFT=FF';
II=FFT*FF;BB=FFT*PPL;
IIINV=inv(II);
a=IIINV*BB;
obusl=norm(II)*norm(IIINV);
for i=1:N
%Pcalc(i,1)=exp(a(1)+a(2)/TT(i)+a(3)*log(TT(i))+a(4)*TT(i)^2);
end

end

end
```

Рис. 3.173 (окончание)

Программный код файла для вычисления коэффициентов различных линеаризованных уравнений, описывающих температурную зависимость давления насыщенного пара ацетона

Результаты определения коэффициентов уравнения многочлена третьей степени $P = exp(a_1T^3 + a_2T^2 + a_3T + a_4)$ (priznak = 4) с использованием решателя “polyfit” приведены в Приложении (табл. П.12). В этом случае используется второе линеаризованное уравнение в виде (3.240). Некоторые статистические параметры для оценки точности описания данных нелинеаризованных уравнений оцениваются с помощью формул, приведенных в разделе 3.5.3.4.1 (формулы (3.206)–(3.214)). Совпадение расчетных и экспериментальных значений давления насыщенного пара ацетона при найденных с применением решателя “polyfit” значениях четырех коэффициентов в графическом виде показано на графиках как для линеаризованного уравнения $P = exp(a_1T^3 + a_2T^2 + a_3T + a_4)$

(рис. 3.174), так и для нелинеаризованного (оригинального) уравнения (рис. 3.175).

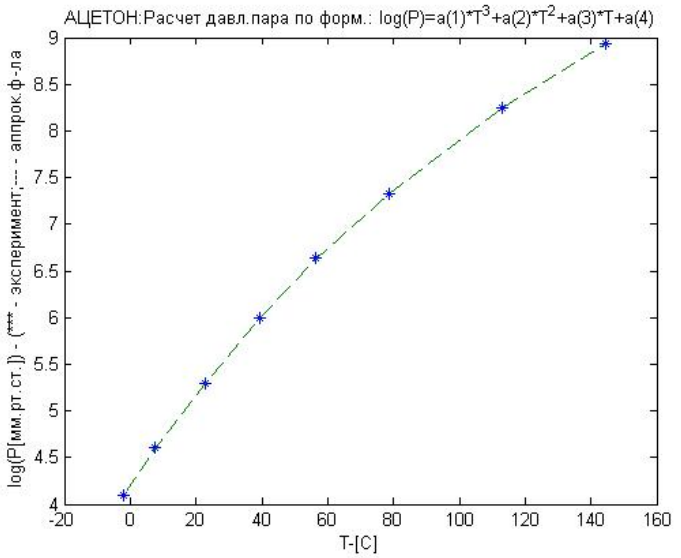


Рис. 3.174

Графическое представление совпадения расчетных и экспериментальных давлений насыщенных паров ацетона при разных температурах с использованием логарифмов последнего линеаризованного уравнения многочлена (3.237)

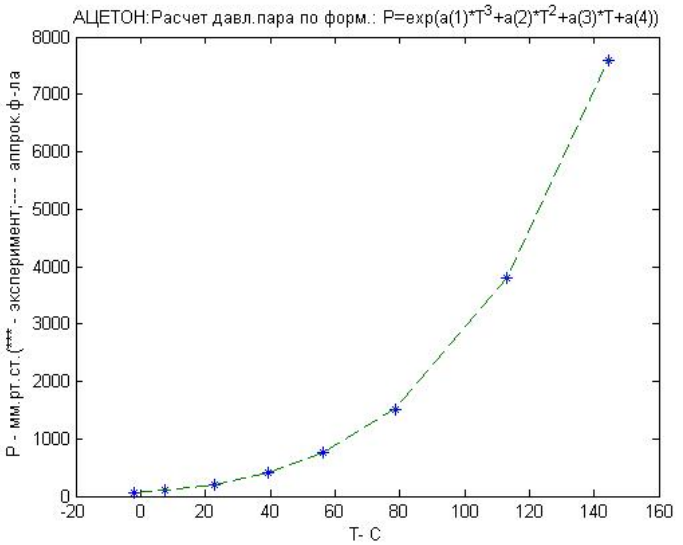


Рис. 3.175

Графическое представление совпадения расчетных и экспериментальных значений давлений насыщенных паров ацетона с использованием параметров уравнения (3.237)

Совпадение расчетных и экспериментальных значений давлений насыщенных паров ацетона при разных температурах с использованием параметров третьего уравнения (3.238) приведено на рисунке 3.176.

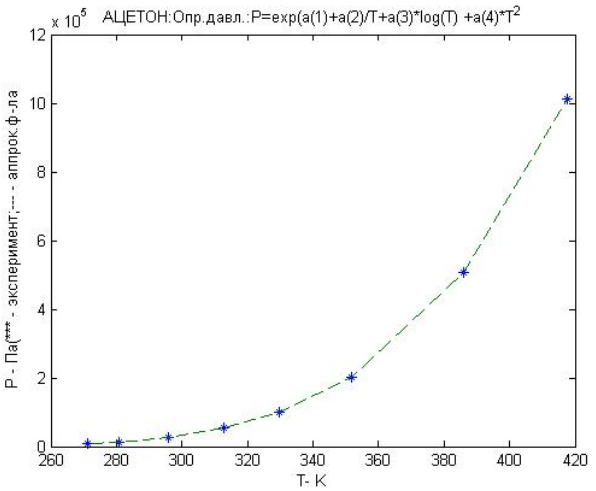


Рис. 3.176

Графическое представление совпадения расчетных и экспериментальных значений давлений насыщенных паров ацетона при разных температурах с использованием параметров третьего уравнения (3.238)

3.5.3.6.3. Параметрическая идентификация эмпирического нелинейного уравнения Антуана для описания температурной зависимости давления насыщенного пара индивидуального вещества

Уравнение Антуана широко используется для описания температурной зависимости давления насыщенного пара индивидуального вещества и содержит три параметра (коэффициента) — A, B, C:

$$P^{\text{расч}} = exp\left(A + \frac{B}{C+T} \right). \qquad (3.246)$$

При решении задачи параметрической идентификации этого уравнения для ацетона можно воспользоваться массивом опытных данных, используемых для определения коэффициентов уравнения из предыдущего раздела (3.238).

Так как уравнение является нелинейным относительно коэффициентов A, B и C, для их определения в общем случае решается задача минимизации критерия рассогласования между расчетными и экспериментальными значениями давления насыщенного пара ацетона с использованием алгоритма многомерной оптимизации (число искомых параметров равно трем — A, B, C).

Программные коды файлов программы определения коэффициентов уравнения Антуана приведены на рисунках 3.177–3.181. Исходные данные для расчетов представлены в файле на рисунке 3.178. Файл основной управляющей

программы (рис. 3.177) позволяет реализовать три метода определения параметров:

— priznak = 1 — методом решения системы линейных алгебраических уравнений — файл lin.m (рис. 3.179);

— priznak = 2 — с использованием решателя “fminsearch”, целевая функция для которого формируется в файле extr.m на рисунке 3.180;

— priznak = 3 — с использованием решателя “lsqcurvefit”, аппроксимирующая функция в векторном виде для которого приведена в файле mnk.m на рисунке 3.181.

Целевая функция обычно представляет собой критерий метода наименьших квадратов (МНК):

$$Cr = \sum_{i=1}^{8} \left(P_i^{\text{расч}} - P_i^{\text{эксп}} \right)^2 . \qquad (3.247)$$

Такой способ определения коэффициентов A, B, C может быть реализован с использованием решателя “fminsearch” (рис. 3.177 — case 2) или “lsqcurvefit” (рис. 3.177 — case 3).

```
%Программный код файла GLAV_THERMO3.m — основная управляющая программа
%-------------------------------------------------------
%Программа GLAV_THERMO3.m+DATA.m+lin.m+extr.m+mnk.m+REPORT.m
%-------------------------------------------------------
%Программа расчета коэффициентов уравнения Антуана для аппроксимации зависимости
%давления насыщенного пара индивидуального вещества по опытным данным;);
%-------------------------------------------------------
%1. Если priznak=1 (задается в файле DATA.m) — Программа расчета параметров уравнения Антуана
P(мм.рт.ст.)=exp(A+B/(C+T(K))методом
%линейной регрессии путем определения коэффициентов линеаризованного
%уравнения T*log(P)=a(1)+a(2)*T+a(3)*log(P) по экспериментальным данным
%о зависимости давлений насыщенных паров индивидуального вещества от температуры
%-------------------------------------------------------
%3. Если priznak=2 (задается в файле DATA.m) — Программа расчета параметров уравнения Антуана
P(мм.рт.ст.)=exp(A+B/(C+T(K)) методом
%нелинейной регрессии путем определения коэффициентов по экспериментальным данным
%о зависимости давлений насыщенных паров индивидуального вещества от температуры
%-------------------------------------------------------
%3. Если priznak=3 (задается в файле DATA.m) — Программа расчета параметров уравнения Антуана
P(мм.рт.ст.)=exp(A+B/(C+T(K))методом
% методом нелинейной регрессии со стандартной функцией MATLAB "lsqcurvefit" путем определения
коэффициентов  по экспериментальным данным
%о зависимости давлений насыщенных паров индивидуального вещества от температуры
%-------------------------------------------------------
clc;
clear all;
close all;
DATA;
global a T P TT crit N priznak aregr A B C;
N=length(T);TT=T+273.15;P=P';TT=TT';
switch priznak
   case 1
%ВЫЧИСЛЕНИЕ КОЭФФИЦИЕНТОВ ЛИНЕАРИЗОВАННОГО УРАВНЕНИЯ — [a] с применением
%функции linear.m и преобразование вектора [a] в коэффициенты A,B,C для уравнения в
% экспоненциальном форме;
[a,A,B,C]=lin(TT,P,N);
   case 2
%Вычисление вектора коэффициентов aregr методом нелинейной регрессии
%с функцией рассогласования экспериментальных расчетных значений extr.m;
```

Рис. 3.177 (начало)

Программный код файла основной управляющей программы для определения коэффициентов уравнения Антуана для расчета давления насыщенного пара ацетона тремя способами

```
%Определение оптимальных параметров aregr=[A,B и C], которые обеспечивают минимум критерия рас-
согласования');
[aregr,crit]=fminsearch('extr',aregr);
A=aregr(1);B=aregr(2);C=aregr(3);
   case 3
%Вычисление вектора коэффициентов AA со стандартной функцией MATLAB "lsqcurvefit"
%с вектором приближений для расчетов aregr для функции, заданной в файле mnk.m;
AA=lsqcurvefit('mnk',aregr,TT',P');
A=AA(1);B=AA(2);C=AA(3);
end
%Формирование отчета о выполненных расчетах
REPORT;
```

Рис. 3.177 (окончание)

Программный код файла основной управляющей программы для определения коэффициентов уравнения Антуана для расчета давления насыщенного пара ацетона тремя способами

Первый способ определения коэффициентов с решателем “fminsearch” (case 2 — можно использовать и решатель “fmincon”) является более универсальным, так как можно сформировать более сложный критерий рассогласования, чем (3.247) (рис. 3.180), в частности с весовыми коэффициентами.

```
function DATA
%Программный код файла DATA.m — задание исходной информации
%-----------------------------------------------------------
%Программа GLAV_THERMO3.m+DATA.m+lin.m+extr.m+mnk.m+REPORT.m
%-----------------------------------------------------------------
%1. Если priznak=1 (задается в файле DATA.m) — Программа расчета параметров уравнения Антуана
P(мм.рт.ст.)=exp(A+B/(C+T(K))методом
%линейной регрессии путем определения коэффициентов линеаризованного
%уравнения T*log(P)=a(1)+a(2)*T+a(3)*log(P) по экспериментальным данным
%о зависимости давлений насыщенных паров индивидуального вещества от температуры
%-----------------------------------------------------------------

%3. Если priznak=2 (задается в файле DATA.m) — Программа расчета параметров уравнения Антуана
P(мм.рт.ст.)=exp(A+B/(C+T(K)) методом
%нелинейной регрессии путем определения коэффициентов по экспериментальным данным
%о зависимости давлений насыщенных паров индивидуального вещества от температуры
%-----------------------------------------------------------------
%3. Если priznak=3 (задается в файле DATA.m) — Программа расчета параметров уравнения Антуана
P(мм.рт.ст.)=exp(A+B/(C+T(K))методом
% методом нелинейной регрессии со стандартной функцией MATLAB "lsqcurvefit" путем определения коэффи-
циентов  по экспериментальным данным
%о зависимости давлений насыщенных паров индивидуального вещества от температуры
%-----------------------------------------------------------------
global T P comp priznak aregr priz_print;
%1. Название индивидуального вещества:
comp='АЦЕТОН';
%-------------------------------
%3. Используемые данные по зависимости давления насыщенного пара
 %индивидуального вещества — P( мм.рт.ст. )  от температуры T ( C ) :
T=[-3.0 7.7 23.7 39.5 56.5 78.6 113.0 144.5];
P=[60 100 200 400 760 2*760 5*760 10*760];
%-------------------------------
%3. Метод расчета коэффициентов уравнения Антуана:
% 1. Линейной регрессией (Priznak=1);
% 3. Нелинейной регрессией (Priznak=2);
% 3. Стандартной функцией "lsqcurvefit" (Priznak=3);
priznak=1;
%-------------------------------
if or(priznak==2,priznak==3)
   aregr=[10 -3000 -30];
```

Рис. 3.178 (начало)

Программный код файла исходных данных для определения коэффициентов уравнения Антуана тремя способами

```
end
%-----------------------------------------
% 4. Форма вывода отчета о работе программы:
%1. Если priz_print=0 (задается в файле DATA.m) — отчет выводится в командное окно MATLAB;
%3. Если priz_print=1 (задается в файле DATA.m) — отчет выводится в текстовый файл, который размещается в
той же папке MATLAB, в которой находятся все m-файлы программы;
priz_print=1;
end
```

Рис. 3.178 (окончание)

Программный код файла исходных данных для определения коэффициентов уравнения Антуана тремя способами

```
function [a,A,B,C]=lin(TT,P,N)
%Программный код файла lin.m — определение коэффициентов аппроксимируемой функции
%уравнения Антуана P(мм.рт.ст.)=exp(A+B/(C+T(K)) в линеаризованном виде
%(a(1),a(2),a(3) -Tlog(P)=a(1)+a(2)*T(i)+a(3)*log(P(i)) и преобразование их
%к виду (A,B,C),используемому в стандартном уравнении Антуана
%-----------------------------------------------------------
%Программа включает следующие файлы: GLAV_THERMO3.m+DATA.m+lin.m+extr.m+mnk.m+REPORT.m
%-----------------------------------------------------------
%Программа расчета коэффициентов уравнения Антуана для аппроксимации зависимости
%давления насыщенного пара индивидуального вещества по опытным данным:');
%-----------------------------------------------------------
% Этот файл включается в программу, когда priznak=1 (задается в файле DATA.m) —
%при расчете параметров уравнения Антуана P(мм.рт.ст.)=exp(A+B/(C+T(K)) методом
%линейной регрессии со стандартным решателем MATLAB "inv" для обращения матрицы коэффициентов
%системы линейных алгебраических уравнений
%-----------------------------------------------------------
%ВЫЧИСЛЕНИЕ КОЭФФИЦИЕНТОВ ЛИНЕАРИЗОВАННОГО УРАВНЕНИЯ
%Формирование векторов и матриц для линеаризованной регрессии с целевой
%функцией crit=SUM[T(i)*log(P(i))-a(1)+a(2)*T(i)+a(3)*log(P(i))]^2
%          где(i=1,N); a(1)=A*C+B;a(2)=A;a(3)=-C
%-----------------------------------------------------------
global crit obusl;
for i=1:N
   PP(i)=log(P(i));
   y(i,1)=TT(i)*PP(i);
   FF(i,1)=1;
   FF(i,2)=TT(i);
   FF(i,3)=PP(i);
end
%Определение коэффициента относительной обусловленности системы уравнений
FFsys=FF'*FF;
obusl=norm(FFsys)*norm(inv(FFsys));
%Определение коэффициентов по линеаризованному критерию путем решению
%системы линейных алгебраических уравнений
a=inv(FF'*FF)*FF'*y;
%Вычисление расчетных значений выходной переменной с найденными
%коэффициентами линеаризованной модели
yrlin=FF*a;
%Вычисление критерия рассогласования расчетных и экспериментальных значений
%для линеаризованной модели
crit=(yrlin-y)'*(yrlin-y);
%Пересчет коэффициентов для исходного не линеаризованного уравнения
%Антуана
A=a(2);C=-a(3);B=a(1)-A*C;

end
```

Рис. 3.179

Программный код файла для определения коэффициентов линеаризованного уравнения Антуана

```
function zcrit = extr(a)
%Программный код файла extr.m — расчет критерия рассогласования расчетных и
%экспериментальных значений методом наименьших квадратов
%-------------------------------------------------------
%Программа включает следующие файлы: GLAV_THERMO3.m+DATA.m+lin.m+extr.m+mnk.m+REPORT.m
%-------------------------------------------------------------------------------------------
%Программа расчета коэффициентов уравнения Антуана для аппроксимации зависимости
%давления насыщенного пара индивидуального вещества по опытным данным:');
%-------------------------------------------------------------------------------
% Этот файл включается в программу, когда priznak=2 (задается в файле DATA.m) —
%при расчете параметров уравнения Антуана P(мм.рт.ст.)=exp(A+B/(C+T(K)) методом
%нелинейной регрессии путем со стандартным решателем MATLAB "fminsearch" путем
%определения коэффициентов по экспериментальным данным
%о зависимости давлений насыщенных паров индивидуального вещества от температуры
%-------------------------------------------------------------------------------------------
global P TT;
zc=0;
for i=1:length(TT)
   zc=zc + (P(i)-exp(a(1)+a(2)/(a(3)+TT(i))))^2;
end
zcrit=zc;
end
```

Рис. 3.180

Программный код файла целевой функции при определении коэффициентов уравнения Антуана с использованием решателя “fminsearch”

```
function P=mnk(ao,TT)
%Программный код файла mnk.m — задание вида аппроксимируемой функции
%уравнения Антуана P(мм.рт.ст.)=exp(A+B/(C+T(K))
%-------------------------------------------------------
%Программа включает следующие файлы: GLAV_THERMO3.m+DATA.m+lin.m+extr.m+mnk.m+REPORT.m
%-------------------------------------------------------
%Программа расчета коэффициентов уравнения Антуана для аппроксимации зависимости
%давления насыщенного пара индивидуального вещества по опытным данным:');
%-------------------------------------------------------
% Этот файл включается в программу, когда priznak=3 (задается в файле DATA.m) —
%при расчете параметров уравнения Антуана P(мм.рт.ст.)=exp(A+B/(C+T(K)) методом
%нелинейной регрессии со стандартным решателем MATLAB "lsqcurvefit"
%путем определения коэффициентов по экспериментальным данным
%о зависимости давлений насыщенных паров индивидуального вещества от температуры
%-------------------------------------------------------
for i=1:length(TT)
P(i)=exp(ao(1)+ao(2)/(ao(3)+TT(i)));
end
end
```

Рис. 3.181

Программный код файла для задания вида уравнения Антуана при определении его коэффициентов с использованием решателя “lsqcurvefit”

Второй способ с решателем “lsqcurvefit” (case 3) более удобен, так как целевая функция (3.247) автоматически формируется решателем “lsqcurvefit”. В этом случае необходимо задать вид аппроксимирующей функции — аналитическое выражение уравнения Антуана (3.245).

Чтобы избежать применения поисковых итерационных методов, путем линеаризации уравнения Антуана можно реализовать метод определения коэффициентов путем решения системы линейных алгебраических уравнений по формуле (3.244) (рис. 3.177 — case 1). В этом случае после линеаризации получается линейное относительно коэффициентов уравнение вида:

$$(TlnP)^{\text{расч}} = a_1 + a_2T + a_3lnP. \tag{3.248}$$

При этом новые коэффициенты a_1, a_2 и a_3 связаны с исходными коэффициентами следующим образом:

$$a_2 = A;\ a_3 = -C;\ a_1 = B + A \cdot C. \tag{3.249}$$

Определяются коэффициенты a_1, a_2 и a_3 путем использования необходимых условий существования экстремума целевой функции вида:

$$Cr = \sum_{i=1}^{8}\left[\left(T_ilnP_i\right)^{\text{расч}} - \left(T_ilnP_i\right)^{\text{эксп}}\right]^2, \tag{3.250}$$

в результате чего получается матричная формула (3.244) для определения a_1, a_2 и a_3, которые в соответствии с (3.249) пересчитываются, и получаются исходные коэффициенты уравнения Антуана (рис. 3.179).

Совпадение результатов расчетов давлений насыщенного пара с использованием найденных коэффициентов уравнения с опытными данными представлено на графиках (рис. 3.182 и 3.183). Как и следовало ожидать, результаты, полученные при линеаризации уравнения Антуана (линейная регрессия, рис. 3.177 — case 1), несколько отличаются от результатов, полученных с использованием решателя “fminsearch” (рис. 3.177 — case 2) или “lsqcurvefit” (рис. 3.177 — case 3), но на значения давлений насыщенных паров это мало влияет. Кстати, применение решателей “fminsearch” и “lsqcurvefit” (нелинейная регрессия, рис. 3.176 — case 2 и 3) приводит к абсолютно совпадающим результатам.

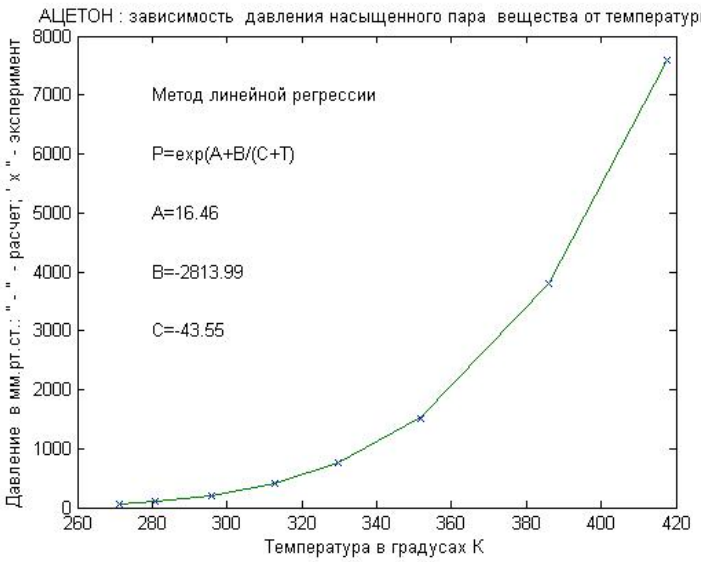


Рис. 3.182

Графическое представление результатов определения коэффициентов уравнения Антуана методом линейной регрессии

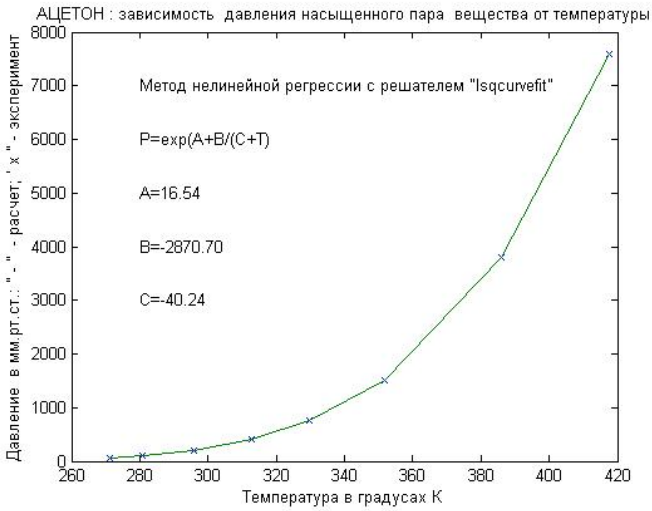


Рис. 3.183

Графическое представление результатов определения коэффициентов уравнения Антуана методом нелинейной регрессии с применением решателя “lsqcurvefit”

Результаты определения коэффициентов уравнения Антуана в табличном виде приведены в приложении (табл. П.13 — линейная регрессия, табл. П.14 — нелинейная регрессия). Формулы для оценки ошибки и погрешностей результатов, в том числе и статистические, такие же, как в разделе 3.5.3.4.1 (формулы (3.206)–(3.214)).

3.5.3.7. Применение решателей “fminbnd”, “fminsearch” и “fmincon” для решения систем уравнений математического описания химико-технологических процессов оптимизационным методом (третий тип оптимизационных задач)

Большое число химико-технологических, термодинамических и т. п. процессов описывается системами уравнений, часто трансцендентных, нелинейных и большой размерности. Традиционные алгоритмы их решения (разделы 3.2 и 3.3) часто не позволяют получить корректные результаты, что связано с большими проблемами инициализации начальных приближений итерационных расчетов, наличием сложных линейных и нелинейных ограничений на искомые и варьируемые переменные процессов.

Бурно развивающееся направление разработки сложных оптимизационных алгоритмов, применяемых в различных областях науки и техники, предоставляет широкие возможности их применения при компьютерном моделировании в химии и химической технологии. При решении систем уравнений путем решения оптимизационной задачи их следует записать в неявном виде, так чтобы правые части всех уравнений были равны нулю, и добиваться при реализации оптимизационного алгоритма таких значений искомых переменных — это и будет решением, которое обеспечит наименьшее отклонение от нуля правых частей всех уравнений системы.

В качестве минимизируемой целевой функции можно использовать аддитивный или мультипликативный критерий, включающий в себя все левые части уравнений, что очень важно — его результирующее значение должно несущественно отличаться от нуля. Такой подход позволяет максимально эффективно учитывать, например, ограничения первого и второго рода при решении оптимизационной задачи, в частности в случае решения задач нелинейного программирования.

Система нормальных уравнений математического описания химико-технологического процесса в неявном виде может быть записана:

$$\begin{aligned} &f_i(x_1, x_2, \ \ldots, \ x_s) = 0, \\ &i = 1, \ 2, \ \ldots, \ s, \end{aligned} \tag{3.251}$$

где x_1, x_2, …, x_s — определяемые переменные; $f_i(x_1, x_2, \ \ldots, \ x_s)$ — линейные и нелинейные функции перечисленных определяемых переменных; s — число уравнений системы (3.251).

Расчет значений определяемых переменных системы (3.251) может осуществляться путем нахождения минимума целевой функции, например, следующего вида:

$$Cr = \sum_{i=1}^{s} \left[f_i\left(x_1, x_2, \ldots, x_s\right) \right]^2, \tag{3.252}$$

который не должен сильно отличаться от нуля, что является одной из разновидностей критериев (3.140)–(3.142) с перечисленными ранее модификациями для оптимизационных задач третьего типа.

При решении одного нелинейного уравнения целесообразно использовать решатель “fminbnd” с целевой функцией вида:

$$Cr = [f(x)]^2. \tag{3.253}$$

Если решается система уравнений без ограничений первого и второго рода, можно воспользоваться решателем “fminsearch” или “fmincon” с целевой функцией (3.252).

При наличии ограничений первого и/или второго рода используются целевые функции (3.253) для одного уравнения и (3.252) для системы уравнений и решатели “fminbnd” (одно уравнение с ограничением первого рода) и/или “fminsearch” (одно или система уравнений), а также “fmincon” (одно или система уравнений с различными ограничениями, в том числе и второго рода).

3.5.3.7.1. Определение равновесного состава реакции синтеза аммиака при температуре T = 450°C и давлении P = 600 атм при известной зависимости константы химического равновесия K_T от температуры оптимизационным методом

Из термодинамики известно, что зависимость константы равновесия реакции синтеза аммиака от температуры задается уравнением

$$K_T = exp\left(-11.95 + \frac{5505.314}{T[C] + 273.15}\right). \tag{3.254}$$

Стехиометрическое соотношение реакции синтеза аммиака имеет вид

$$N_2 + 3H_2 \rightleftarrows 2NH_3 \tag{3.255}$$

Так как реакция протекает в газовой фазе, для константы равновесия будет справедливо:

$$K_R = \frac{p_{NH_3}}{\left(p_{N_2}\right)^{1/2} \cdot \left(p_{H_2}\right)^{3/2}}, \tag{3.256}$$

где p_{NH_3}, p_{N_2}, p_{H_2} — парциальные давления компонентов в равновесной системе.

Парциальные давления компонентов при заданном давлении в системе (P = 600 атм) определяются по формулам:

$$p_{NH_3} = P \cdot y_{NH_3};\ \ p_{N_2} = P \cdot y_{N_2};\ \ p_{H_2} = P \cdot y_{H_2}, \tag{3.257}$$

где y_{NH_3}, y_{N_2}, y_{H_2} — мольные доли компонентов в газовой фазе; P — общее давление, равное 600 атм.

Задача заключается в определении равновесных составов y_{NH_3}, y_{N_2} и y_{H_2} при условии, что K_R в (3.256) равно K_T при температуре 450°C и определено по формуле (3.254), т. е. равновесные мольные доли компонентов должны удовлетворять уравнению

$$K_T - \frac{p_{NH_3}}{\left(Pp_{N_2}\right)^{\frac{1}{2}} \cdot \left(p_{H_2}\right)^{\frac{3}{2}}} = 0 \tag{3.258}$$

Это уравнение может быть решено относительно одной неизвестной в качестве которой выбирается степень превращения (конверсия) азота, определяемая по формуле

$$\xi_{N_2} = \frac{n_{N_2}^{(0)} - n_{N_2}}{n_{N_2}^{(0)}}, \tag{3.259}$$

где $n_{N_2}^{(0)}$, n_{N_2} — начальное и текущее число молей компонента N_2.

В соответствии с (3.259) и стехиометрией реакции (3.255) текущее число молей каждого компонента реакции определяется следующим образом:

$$\begin{aligned} n_{N_2} &= n_{N_2}^{(0)}\left(1 - \xi_{N_2}\right); \\ n_{H_2} &= 3n_{N_2}^{(0)}\left(1 - \xi_{N_2}\right); \\ n_{NH_3} &= 2\xi_{N_2} \cdot n_{N_2}^{(0)}. \end{aligned} \tag{3.260}$$

В результате мольные доли компонентов реакционной смеси могут быть определены по формуле

$$y_i = \frac{n_i}{n_{N_2} + n_{H_2} + n_{NH_3}}, \quad i = N_2,\ H_2,\ NH_3, \tag{3.261}$$

которые должны быть подставлены в выражения для парциальных давлений компонентов (3.257), а последние — в уравнение (3.258).

Таким образом получается, что уравнение (3.258) содержит одну неизвестную — степень превращения азота ξ_{N_2}.

Определение этой величины при известном равновесном K_T (3.254) позволит по формулам (3.260) и (3.261) определить равновесные концентрации всех компонентов химической реакции.

Уравнение (3.258) решается методом одномерной оптимизации с минимизируемым критерием вида (3.253) (программная реализация — рис. 3.186):

$$Cr = \left[K_T - K_R\left(\xi_{N_2}\right) \right]^2. \tag{3.262}$$

При этом K_R определяется по формуле (3.256) (программная реализация — рис. 3.186) с y_{NH_3}, y_{N_2}, y_{H_2}, которые вычисляются по формулам (3.261) (программная реализация — рис. 3.187).

Поиск минимума критерия (3.262) выполняется с использованием решателя “fminbnd” (программная реализация — рис. 3.184) в интервале для определяемой переменной $0.00001 \le \xi_{N_2} \le 0.99999$ (программная реализация — рис. 3.185).

Программные коды файлов решения рассматриваемой задачи приведены на рисунках 3.184–3.187. Исходные данные для расчетов задаются в файле на рисунке 3.185. В основной управляющей программе (рис. 3.184) происходит обращение к решателю задачи одномерной оптимизации “fminbnd”, целевая функция для которой представлена в файле на рисунке 3.186. Определение состава реакционной смеси по значению степени превращения азота ξ_{N_2} осуществляется в файле, представленном на рисунке 3.187.

```
%-------------------------------------------------;
%Программный код файла GLAV_THERMO10.m — основная управляющая программа
%и определение равновесных конверсий реакции при различных давлениях и температурах;
%-------------------------------------------------;
%Программа включает следующие файлы:
%GLAV_THERMO10.m+DATA.m+Funcm.m+Sostavm.m+REPORT.m;
%-------------------------------------------------;
%Расчёт равновесного состава химической реакции синтеза аммиака N2+3H2=2NH3
%при заданном давлении и температуре
%-------------------------------------------------;
clc;
clear all;
close all;
global T0 P P0 T A1 B1 TN KT;
global ksia ksib ksires funcres ya yb yc;
global jj Pt Tt KTt KSI YAt YBt YCt;
```

Рис. 3.184 (начало)

Программный код файла основной управляющей программы для определения равновесного состава реакции синтеза аммиака

```
kequil=inline('exp(AA/T+BB)');
DATA;
jj=0;
P=P0;
T=T0;
TC=T;
    T=TC+TN;
    KT=kequil(A1,B1,T);
    [ksires,funcres,pr,out]=fminbnd('Funcm',ksia,ksib);
    [ya yb yc]=Sostavm(ksires);
    Pt=P;Tt=TC;KTt=KT;KSI=ksires;
    YAt=ya;YBt=yb;YCt=yc;
    REPORT;
```

Рис. 3.184 (окончание)

Программный код файла основной управляющей программы для определения равновесного состава реакции синтеза аммиака

```
function DATA
%-----------------------------------------------;
%Программный код файла DATA.m — задание исходных данных для расчетов;
%-----------------------------------------------;
%Программа включает следующие файлы:
%GLAV_THERMO10.m+DATA.m+Funcm.m+Sostavm.m+REPORT.m;
%-----------------------------------------------;
%Расчёт равновесного состава химической реакции синтеза аммиака N2+3H2=2NH3
%при заданном давлении и температуре
%-----------------------------------------------;
global ksia  ksib T0 P0 A1 B1 TN;
global n ksi_calc;
global ka kb kc kam  kbm kcm na0 priz_print;
%-----------------------------------------------;
%1. Уравнение химической реакции:
%1.1. N2+3H2=2NH3
%1.3.(1)A+(3)B->(2)C;
%где:A-азот,B-водород(b),C-аммиак(c);
%1.3. Определение приведенных стехиометрических коэффициентов реакции
ka=1;kb=3;kc=2;
kam=ka/kc;kbm=kb/kc;kcm=kc/kc;
%1.4. Общее число компонентов реакции:
n=3;
%1.5. Число молей компонента A в начале реакции:
na0=1;
%-----------------------------------------------;
%3. Коэффициенты уравнения температурной зависимости константы равновесия
%химической реакции вида K=exp(A1/T[K]+B1), рассчитанных без учета температурной
% энтальпии реакции V (признак KCONST=1):
A1=5505.314;B1=-11.95;
%-----------------------------------------------;
%3. Базовая температура(TN) в K:
TN=273.15;
%-----------------------------------------------;
%4. Данные для вычислений численными методами:
%-----------------------------------------------;
%4.1. Начальное приближение (интервал поиска) для итерационных расчетов
% при вычислении равновесной конверсии(ksi):
ksia=0.00001;ksib=0.99999;
%5. Давление в системе P0 в атм:
P0=600;
%6. Температура в системе T0 в C:
T0=450;
%7. Графическая интерпретация определения равновесной конверсии
%при известных P и T путем определения
%минимума, характеризующего рассогласование в квадрате известного
%и рассчитанного значения константы равновесия (ksi_calc=1 — да,ksi_calc=0 — нет);
```

Рис. 3.185 (начало)

Программный код файла исходных данных для определения равновесного состава реакции синтеза аммиака

```
ksi_calc=1;
%8. Форма вывода отчета о работе программы:
%8.1. Если priz_print=0 (задается в файле DATA.m) — отчет выводится в командное ок-
но MATLAB;
%8.3. Если priz_print=1 (задается в файле DATA.m) — отчет выводится в текстовый
файл, который размещается в той же папке MATLAB, в которой находятся все m-файлы
программы;
priz_print=1;
end
```

Рис. 3.185 (окончание)

Программный код файла исходных данных для определения равновесного состава реакции синтеза аммиака

```
function y=Funcm(ksi)
%-----------------------------------------------;
%Программный код файла Funcm.m — вычисления функции уравнения, используемого
%для определения равновесной конверсии реакции;
%-----------------------------------------------;
%Программа включает следующие файлы:
%GLAV_THERMO10.m+DATA.m+Funcm.m+Sostavm.m+REPORT.m;
%-----------------------------------------------;
%Расчёт равновесного состава химической реакции синтеза аммиака N2+3H2=2NH3
%при заданном давлении и температуре
%-----------------------------------------------;
global KT kam kbm kcm P;
global ya yb yc;
[ya yb yc]=Sostavm(ksi);
yn=[ya yb yc];
pa=(P*ya)^kam;pb=(P*yb)^kbm;pc=(P*yc)^kcm;
KR=pc/pa/pb;
y=(KT-KR)^2;
end
```

Рис. 3.186

Программный код файла минимизируемой целевой функции при определении равновесного состава реакции синтеза аммиака

```
function [ya yb yc] = Sostavm(ksi)
%-----------------------------------------------;
%Программный код файла Sostavm.m — нормировка рассчитываемых равновесных
%составов в мольных долях;
%и определение равновесных конверсий реакции при различных давлениях и температурах;
%-----------------------------------------------;
%Программа включает следующие файлы:
%GLAV_THERMO10.m+DATA.m+Funcm.m+Sostavm.m+REPORT.m;
%-----------------------------------------------;
%Расчёт равновесного состава химической реакции синтеза аммиака N2+3H2=2NH3
%при заданном давлении и температуре
%-----------------------------------------------;
global na0 ka kb kc
na=ka*na0*(1-ksi);
nb=kb*na0*(1-ksi);
nc=kc*na0*ksi;
nsum=na+nb+nc;
ya=na/nsum;yb=nb/nsum;yc=nc/nsum;
end
```

Рис. 3.187

Программный код файла для определения состава реакционной смеси при синтезе аммиака по конверсии азота

Результат определения равновесной степени превращения азота (конверсии) и соответствующие равновесные концентрации компонентов в реакционной смеси приведены в Приложении (табл. П.15). На рисунке 3.188 представлено графическое изображение изменения целевой функции *Cr* (3.262) в зависимости от конверсии азота (ξ_{N_2}).

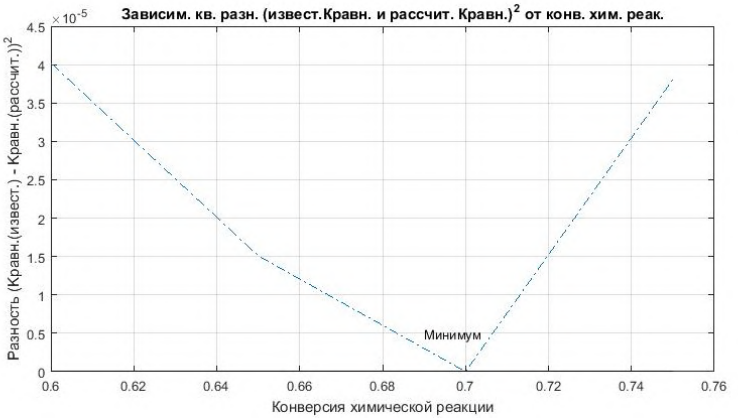


Рис. 3.188

Графическая иллюстрация характера изменения целевой функции при определении равновесного состава реакции синтеза аммиака методом одномерной оптимизации

3.5.3.7.2. Определение равновесного состава двухстадийной реакции получения водорода из метана и водяного пара (начальное мольное соотношение $\frac{n^{(0)}_{H_2O}}{n^{(0)}_{CH_4}} = 5$ при температуре T = 600°C и давлении P = 10 атм при известных значениях констант равновесия первой $K_1 = 0.5$ и второй $K_2 = 3.54$ реакций)

В этом случае две одновременно протекающие реакции записываются следующим образом:

$$\begin{aligned} &(1)\ CH_4 + H_2O \rightleftarrows CO + 3H_2, \\ &(2)\ CO + H_2O \rightleftarrows CO_2 + H_2. \end{aligned} \tag{3.263}$$

Мольный баланс компонентов с учетом мольных превращений в первой (Δn_1) и во второй (Δn_2) реакции записывается:

$$\begin{aligned} n_{CH_4} &= 1 - \Delta n_1, \\ n_{H_2O} &= 5 - \Delta n_1 - \Delta n_2, \\ n_{CO} &= \Delta n_1 - \Delta n_2, \\ n_{H_2} &= 3\Delta n_1 + \Delta n_2, \\ n_{CO_2} &= \Delta n_2. \end{aligned} \tag{3.264}$$

Суммарное число молей определяется следующим образом:

$$n_T = n_{CH_4} + n_{H_2O} + n_{CO} + n_{H_2} + n_{CO_2}, \tag{3.265}$$

и для мольных долей компонентов реакционной смеси будет справедливо:

$$\begin{aligned} &y_i = \frac{n_i}{n_T}, \\ &i = CH_4,\ H_2O,\ CO,\ H_2,\ CO_2. \end{aligned} \tag{3.266}$$

С учетом стехиометрии газофазной реакции (3.263) равновесные условия описываются двумя уравнениями:

$$\begin{aligned} K_{r1} &= \frac{p_{CO} \cdot p_{H_2}^3}{p_{CH_4} \cdot p_{H_2O}}, \\ K_{r2} &= \frac{p_{CO_2} \cdot p_{H_2}}{p_{CO} \cdot p_{H_2O}} \end{aligned} \tag{3.267}$$

с соответствующими выражениями для парциальных давлений компонентов следующего вида:

$$\begin{aligned} &p_i = P \cdot y_i, \\ &i = CH_4,\ H_2O,\ CO,\ H_2,\ CO_2, \end{aligned} \tag{3.268}$$

где p_i — парциальные давления компонентов реакции; P — общее давление в системе.

Определение равновесного состава реакционной смеси происходит с учетом того, что константы равновесия K_1 и K_2 известны и заданы в условии задачи, а K_{r1} и K_{r2}, рассчитываемые по формулам (3.267), должны быть:

$$K_{r1} = K_1 \text{ и } K_{r2} = K_2. \tag{3.269}$$

Отсюда для определения равновесного состава химической реакции в соответствии с (3.267) и (3.269) необходимо решить систему двух уравнений:

$$\begin{aligned} K_1 - \frac{p_{CO} \cdot p_{H_2}^3}{p_{CH_4} \cdot p_{H_2O}} &= 0, \\ K_2 - \frac{p_{CO_2} \cdot p_{H_2}}{p_{CO} \cdot p_{H_2O}} &= 0. \end{aligned} \tag{3.270}$$

Все парциальные давления в этих уравнениях с учетом (3.268), а также (3.264)–(3.266) зависят от двух величин — мольных превращений в первой реакции (Δn_1) и во второй реакции (Δn_2).

Для решения системы двух уравнений (3.270) относительно Δn_1 и Δn_2 методом оптимизации с учетом (3.267) формируется минимизируемая целевая функция вида (3.252) (программная реализация — рис. 3.193):

$$Cr = (K_1 - K_{r1})^2 + (K_2 - K_{r2})^2. \tag{3.271}$$

Решение двумерной оптимизационной задачи с критерием (3.271) осуществляется с использованием решателя “fminsearch” (программная реализа-

ция — рис. 3.191), целевая функция для которого представлена на рисунке 3.193. Если при найденных в результате решения системы (3.268) Δn_1 и Δn_2 значения критерия (3.268) близки к нулю, то по соотношениям (3.264)–(3.266) вычисляются равновесные составы химической реакции (программная реализация — рис. 3.192 и 3.191).

Программные коды файлов решения рассматриваемой задачи представлены на рисунках 3.189–3.193. Исходные данные задаются в файле на рисунке 3.190. В основной управляющей программе происходит обращение к программе определения равновесного состава (рис. 3.191), в частности к решению задачи двумерной оптимизации с целевой функцией (рис. 3.189) и программе непосредственного расчета равновесного состава (рис. 3.191).

Результаты расчетов равновесного состава представлены в Приложении (табл. П.16).

```
%-------------------------------------------------------------------;
%Программный код файла GLAV_THERMO11.m — основная управляющая программа;
%-------------------------------------------------------------------;
%Программа включает следующие файлы: GLAV_THERMO11.m+DATA.m+Equil.m
%+nonlin2opt.m+sostav.m+REPORT.m
%-------------------------------------------------------------------;
%Расчёт равновесного состава 2-х одновременно протекающих химических реакций
%при заданном давлении P и температуре T   A+B->C+3D (k1) и C+B->E+D (k2)
%-------------------------------------------------------------------;
clc;
clear all;
global x0
DATA;
x=Equil(x0);
REPORT;
```

Рис. 3.189

Программный код файла основной управляющей программы для определения равновесного состава в двух реакциях получения водорода из метана и водяного пара

```
function DATA
%-------------------------------------------------------;
%Программный код файла DATA.m — задание исходных данных для расчетов
%-------------------------------------------------------;
%Программа включает следующие файлы: GLAV_THERMO11.m+DATA.m+Equil.m
%+nonlin2opt.m+sostav.m+REPORT.m
%-------------------------------------------------------;
%Расчёт равновесного состава 2-х одновременно протекающих химических реакций
%при заданном давлении P и температуре T   A+B->C+3D (k1) и C+B->E+D (k2)
%-------------------------------------------------------;
global T P k1 k2 x0 n nA0 nB0 priz_print;
%-------------------------------------------------------;
%1. Уравнения одновременно протекающих газофазных реакций:
%1.1. A+B->C+3D;
%1.3. C+B->E+D;
%где: A-метан(a), B-водяной пар (b), C-угарный газ (c), D-водород(d),
%E-углекислый газ(e);
%-------------------------------------------------------;
%3. Количественные характеристики реакций:
%3.1. Общее число компонентов реакции:
n=5;
%3.3. Исходное число молей метана (A):
nA0=1;
%3.3. Исходное число молей водяного пара (B):
```

Рис. 3.190 (начало)

Программный код файла для задания исходных данных для определения равновесного состава в двух реакциях получения водорода из метана и водяного пара

```
nB0=5;
%-----------------------------------------------;
%3. Исходные данные для расчётов:
%3.1. Давление в системе (P) в атм:
P=10;
%-----------------------------------------------;
%3.3. Температура в системе (T) в C:
T=600;
%-----------------------------------------------;
%4. Константа равновесия 1-ой реакции k1:
k1=0.5;
%-----------------------------------------------;
%5. Константа равновесия 2-ой реакции k2:
k2=3.54;
%-----------------------------------------------;
%6. Физико-химические константы:
%7.1. Rg-Универсальная газовая постоянная Rg=8.314 (дж/моль*K)
%Rg=8.314;
%7.3. Базовая температура (K)
%TN=298.15;
%-----------------------------------------------;
%7. Начальные приближения для итерационных расчетов определения
%равновесной мольной доли превращенного реагента B (водяного пара) в реакции 1 x0(1) и реак-
ции 2 x0(2)
x0=[0.25,0.25];
%-----------------------------------------------;
%8. Форма вывода отчета о работе программы:
%1. Если priz_print=0 (задается в файле DATA.m) — отчет выводится в командное окно MATLAB;
%3. Если priz_print=1 (задается в файле DATA.m) — отчет выводится в текстовый файл, который
размещается в той же папке MATLAB, в которой находятся все m-файлы программы;
priz_print=1;
%-----------------------------------------------;
end
```

Рис. 3.190 (окончание)

Программный код файла для задания исходных данных для определения равновесного состава в двух реакциях получения водорода из метана и водяного пара

```
function x = Equil(x0)
%-----------------------------------------------;
%Программный код файла Equil.m — расчет равновесного состава путем определения
мольных превращений 2-х одновременно протекающих реакций: x(1) =Δn1 и x(2) =Δn2

%-----------------------------------------------;
%Программа включает следующие файлы: GLAV_THERMO11.m+DATA.m+Equil.m
%+nonlin2opt.m+sostav.m+REPORT.m
%-----------------------------------------------;
%Расчёт равновесного состава 2-х одновременно протекающих химических реакций
%при заданном давлении P и температуре T  A+B->C+3D (k1) и C+B->E+D (k2)
%-----------------------------------------------;
global P y xx ff ex;
[x,ff,ex]=fminsearch('nonlin2opt',x0);
xx=x;
py = sostav( x );
y=py/P;
end
```

Рис. 3.191

Программный код файла для непосредственного определения равновесного состава в двух одновременно протекающих реакциях получения водорода из метана и водяного пара

```
function py = sostav(x)
%-----------------------------------------------------;
%Программный код файла sostav.m - расчет парциальных давлений компонентов
%химической реакции при известной мольной доле непрореагировавших реагентов
%метана и водяного пара
%-----------------------------------------------------;
%Программа включает следующие файлы: GLAV_THERMO11.m+DATA.m+Equil.m
%+nonlin2opt.m+sostav.m+REPORT.m
%-----------------------------------------------------;
%Расчёт равновесного состава 2-х одновременно протекающих химических реакций
%при заданном давлении P и температуре T  A+B->C+3D (k1) и C+B->E+D (k2)
%-----------------------------------------------------;
global nA0 nB0 P
n(1)=nA0-x(1);n(2)=nB0-x(1)-x(2);
n(3)=x(1)-x(2);n(4)=3*x(1)+x(2);n(5)=x(2);
nsum=sum(n);
for i=1:5
py(i)=P*n(i)/nsum;
end
end
```

Рис. 3.192

Программный код файла для расчета парциальных давлений компонентов реакции получения водорода из метана и водяного пара

```
function ff = nonlin2opt(x)
%-----------------------------------------------------;
%Программный код файла nonlin2opt.m — решение системы 2-х нелинейных уравнений;
%-----------------------------------------------------;
%Программа включает следующие файлы: GLAV_THERMO11.m+DATA.m+Equil.m
%+nonlin2opt.m+sostav.m+REPORT.m
%-----------------------------------------------------;
%Расчёт равновесного состава 2-х одновременно протекающих химических реакций
%при заданном давлении P и температуре T :  A+B->C+3D (k1) и C+B->E+D (k2)
global k1 k2
py = sostav(x);
kr1=py(3)*py(4)^3/py(1)/py(2);
kr2=py(5)*py(4)/py(3)/py(2);
f(1)=k1-kr1;
f(2)=k2-kr2;
ff=f(1)^2+f(2)^2;
end
```

Рис. 3.193

Программный код файла для формирования целевой функции при определении равновесного состава реакции получения водорода из метана и водяного пара

ПРИЛОЖЕНИЕ. РЕЗУЛЬТАТЫ КОМПЬЮТЕРНОГО МОДЕЛИРОВАНИЯ ХИМИКО-ТЕХНОЛОГИЧЕСКИХ ПРОЦЕССОВ. (ПРОГРАММНЫЕ КОДЫ ФАЙЛОВ ПРИВЕДЕНЫ В ГЛАВЕ 3)

Таблица П.1. Результаты определения равновесных температур и мольных объемов жидкой и паровой фаз по уравнению состояния SRK при различных давлениях

Программные коды файлов для вычислений — рисунки 3.14–3.18.

```
-------------------------------------------------
%Программа определения температуры кипения индивидуального вещества
%при заданном давлении по 5-коэффициентному аппроксимирующему уравнению Риделя для
%зависимости давления насыщенного пара от температуры и мольных объемов жидкости и пара
%в заданном диапазоне давлений по уравнению состояния Soave-Redlich-Kwong (SRK)
-------------------------------------------------
Программа включает следующие файлы:
Glav_model_SRK_P.m+DATA.m+EqP.m+ABSRK0.m+REPORT.m
_____________________________________________________________
ИСХОДНЫЕ ДАННЫЕ
-------------------------------------------------
1.Название индивидуального вещества (comp) = Этиловый спирт
-----------------------------------------------------
3.Критическая температура (TKR) = 513.92 K
3.Критическое давление (PKR) = 60.68 атм
4.Ацентрический фактор Питцера (omega) = 0.6350
-----------------------------------------------------
5.Коэффициенты 5-коэффициентного уравнения Риделя для определения давления нас. пара вещества:
          P[Па]=exp(A+B/T[K]+C*log(T[K])+D*T[K]^E)
5.1. Первый коэффициент ( A ) = 74.4750
5.3. Второй коэффициент ( B ) = -7164.3000
5.3. Третий коэффициент ( C ) = -7.3270
5.4. Четвертый коэффициент ( D ) = 3.134e-06
5.5. Пятый коэффициент ( E ) = 3.0000
_____________________________________________________
6.1. Левая граница интервала изменения давления ( Pmin ) = 640.00 мм.рт.ст.
-----------------------------------------------------
6.3. Правая граница интервала изменения давления( Pmax ) = 4000.00 мм.рт.ст.
-----------------------------------------------------
6.3. Шаг изменения давления( Pstep ) = 60.00 мм.рт.ст.
-----------------------------------------------------
7. Оценка приближения по температуре при определении T-кипения ( T0 ) = 400.19 K
-----------------------------------------------------------------
-----------------------------------------------------------------
                РЕЗУЛЬТАТЫ РАСЧЕТОВ
-----------------------------------------------------------------
1.Результаты расчета температур кипения (Ткип) при различных давлениях
путем решения трансцендентного уравнения вида P[Па]-exp(A+B/T[K]+Cln(T[K])+DT[K]^E=0
использующим 5-коэффициентное аппроксимирующее уравнение Риделя для расчета давл. нас. пара
```

Продолжение табл. П.1

№	P(мм.рт.ст.)	Ткип(C)	VV(дм^3/мол.)	VL(дм^3/мол.)	VS(дм^3/мол.)
1	640.0	74.005	33.18008	0.07591	0.57133
2	700.0	76.233	30.48704	0.07618	0.56315
3	760.0	78.308	28.20719	0.07643	0.55568
4	820.0	80.252	26.25143	0.07667	0.54879
5	880.0	83.081	24.55468	0.07690	0.54241
52	3700.0	124.881	6.20688	0.08377	0.41806
53	3760.0	125.431	6.11016	0.08388	0.41674
54	3820.0	125.974	6.01638	0.08399	0.41544
55	3880.0	126.510	5.92540	0.08410	0.41416
56	3940.0	127.039	5.83711	0.08421	0.41290
57	4000.0	127.562	5.75138	0.08432	0.41167

3. Заданное давление, при граф. интерпр. корня уравнения, опред. Т-кипения (Pdata) = 640.00 мм.рт.ст.

Выведено 5 графиков

Расчет по программе завершен, результаты также представлены в виде 5 отдельных графиков

Таблица П.2. Результаты моделирования стационарного режима изотермического реактора идеального перемешивания со стехиометрической схемой реакции $A \rightarrow P \rightarrow S$ при изменении времени пребывания в аппарате

Программные коды файлов для вычислений — рисунки 3.36–3.39.

ПРОГРАММА МОДЕЛИРОВАНИЯ ИЗОТЕРМИЧЕСКОГО РЕАКТОРА ИДЕАЛЬНОГО ПЕРЕМЕШИВАНИЯ
РЕАКЦИЯ: A - P - S

Программа включает следующие файлы:
GLAV_model_linreactor_stat.m+DATA.m+model_stat_linreactor.m+model_stat_linreactor_4+REPORT.m

ИСХОДНЫЕ ДАННЫЕ

1.Мольный расход потока сырья на входе в реактор (n0) = 14.40 мол.см./час

1.Концентрация реагента A на входе в реактор (xa0) = 0.80 мольные доли

3.Концентрация реагента A на входе в реактор (xp0) = 0.10 мольные доли

3.Концентрация реагента A на входе в реактор (xs0) = 0.10 мольные доли

4.Константа скорости первой элементарной реакции (k1) = 0.35 час^(-1)
5.Константа скорости второй элементарной реакции (k2) = 0.13 час^(-1)

6.Левая граница интервала поиска оптимального времени пребывания в реакторе (tau_a) = 1.00 час

---

7. Правая граница интервала поиска оптимального времени пребывания в реакторе (tau_b) = 10.00 час

_____

РЕЗУЛЬТАТЫ РАСЧЕТОВ

1.Концентрации продуктов на выходе из реактора

№	tau (час)	xa (мол.д.)	xp(мол.д.)	xs (мол.д.)	na (мол.А/час)	np (мол.Р/час)	ns (мол.S/час)	N (мол.см.)	n (мол.см./час)
1	1.00000	0.59259	0.27204	0.13537	8.53333	3.91740	1.94926	14.40000	14.40000
2	1.10000	0.57762	0.28205	0.14033	8.31769	4.06151	3.02080	15.84000	14.40000
3	1.20000	0.56338	0.29119	0.14543	8.11268	4.19319	3.09414	17.28000	14.40000
…	…	…	…	…	….	….	…	…	…
89	9.80000	0.18059	0.31636	0.50305	3.60045	4.55565	7.24390	141.12000	14.40000
90	9.90000	0.17917	0.31519	0.50564	3.58007	4.53867	7.28127	143.56000	14.40000
91	10.00000	0.17778	0.31401	0.50821	3.56000	4.52174	7.31826	144.00000	14.40000

Расчет по программе завершен, результаты также представлены в виде 3-х графиков

Таблица П.3. Результаты моделирования стационарного режима изотермического реактора идеального перемешивания со стехиометрической схемой реакции *A* → *P* → *S* для различных реакционных объемов

Программные коды файлов для вычислений — рисунки 3.43–3.45.

ПРОГРАММА МОДЕЛИРОВАНИЯ ИЗОТЕРМИЧЕСКОГО РЕАКТОРА ИДЕАЛЬНОГО ПЕРЕМЕШИВАНИЯ
РЕАКЦИЯ: A - P - S

Программа включает следующие файлы:
GLAV_model_linreac_N_stat.m+DATA.m+model_linreac_N_stat.m+REPORT.m

_____

ИСХОДНЫЕ ДАННЫЕ

1.Мольный расход потока сырья на входе в реактор (n0) = 10.00 мол.см./час

3.Мольная доля компонента А в потоке сырья на входе в реактор (xa0) = 0.70 мол.А/мол.см.

3.Мольный расход компонента А на входе в реактор (na0) = 7.00 мол.А/час

4.Мольная доля компонента P в потоке сырья на входе в реактор (xp0) = 0.15 мол.А/мол.см.

5.Мольный расход компонента P на входе в реактор (np0) = 1.50 мол.А/час

6.Мольная доля компонента S в потоке сырья на входе в реактор (xs0) = 0.15 мол.А/мол.см.

7.Мольный расход компонента S на входе в реактор (ns0) = 1.50 мол.А/час

8.Константа скорости первой элементарной реакции (k1) = 0.35 час^(-1)

9.Константа скорости второй элементарной реакции (k2) = 0.13 час^(-1)

10.Левая граница изменения реакционного объема в реакторе (Nmin) = 10.00 мол.см.
----
11. Правая граница изменения реакционного объема в реакторе (Nmax) = 100.00 мол.см.
----
13. Шаг изменения реакционного объема в реакторе(Ndel) = 5.00 мол.см.
_____
РЕЗУЛЬТАТЫ РАСЧЕТОВ

1.Параметры реактора и его выходных потоков

№	N(мол.см.)	xa(мол.д.)	xp(мол.д.)	xs(мол.д.)	n(мол.см./час)	tau(час)
1	10.000	0.51851	0.29335	0.18813	10.00000	1.00000
2	15.000	0.45902	0.32718	0.21380	10.00000	1.50000
3	20.000	0.41176	0.34781	0.24043	10.00000	3.00000
4	25.000	0.37333	0.35975	0.26692	10.00000	3.50000
5	30.000	0.34146	0.36585	0.29268	10.00000	3.00000
6	35.000	0.31461	0.36797	0.31743	10.00000	3.50000
7	40.000	0.29167	0.36732	0.34101	10.00000	4.00000
8	45.000	0.27184	0.36477	0.36339	10.00000	4.50000
9	50.000	0.25455	0.36088	0.38457	10.00000	5.50000
10	55.000	0.23932	0.35608	0.40460	10.00000	5.50000
11	60.000	0.22581	0.35067	0.42352	10.00000	6.00000
12	65.000	0.21374	0.34486	0.44140	10.00000	6.50000
13	70.000	0.20290	0.33880	0.45830	10.00000	7.00000
14	75.000	0.19310	0.33261	0.47429	10.00000	7.50000
15	80.000	0.18421	0.32637	0.48942	10.00000	8.00000
16	85.000	0.17610	0.32014	0.50376	10.00000	8.50000
17	90.000	0.16867	0.31397	0.51735	10.00000	9.00000
18	95.000	0.16185	0.30790	0.53025	10.00000	9.50000
19	100.000	0.15556	0.30193	0.54251	10.00000	10.00000

Расчет по программе завершен, результаты также представлены в виде 3-х графиков

Таблица П.4. Результат моделирования стационарного режима изотермического реактора идеального перемешивания со стехиометрической схемой реакции $A \rightarrow 2P \rightarrow S$ с нелинейной кинетической зависимостью скоростей стадий от концентраций веществ в реакторе идеального перемешивания при изменении времени пребывания реакционного потока в реакторе

Программные коды файлов для вычисления — рисунки 3.49–3.51.

ПРОГРАММА МОДЕЛИРОВАНИЯ ИЗОТЕРМИЧЕСКОГО РЕАКТОРА ИДЕАЛЬНОГО ПЕРЕМЕШИВАНИЯ
РЕАКЦИЯ: A - 2P - S с порядком 1-ой реакции 0.33, а второй - 2

Программа включает следующие файлы:
GLAV_model_non_linreac_tau_stat.m+DATA.m+model_non_linreac_tau_stat.m+REPORT.m

---

ИСХОДНЫЕ ДАННЫЕ

1.Мольный расход потока сырья на входе в реактор (n0) = 10.00 мол.см./час

3.Мольная доля компонента A в потоке сырья на входе в реактор (xa0) = 1.00 мол.A/мол.см.

3.Мольный расход компонента A на входе в реактор (na0) = 10.00 мол.A/час

4.Мольная доля компонента P в потоке сырья на входе в реактор (xp0) = 0.00 мол.A/мол.см.

5.Мольный расход компонента P на входе в реактор (np0) = 0.00 мол.A/час

6.Мольная доля компонента S в потоке сырья на входе в реактор (xs0) = 0.00 мол.A/мол.см.

7.Мольный расход компонента S на входе в реактор (ns0) = 0.00 мол.A/час

8.Константа скорости первой элементарной реакции (k1) = 0.35 час^(-1)
9.Константа скорости второй элементарной реакции (k2) = 0.13 час^(-1)

10.Левая граница интервала изменения времени пребывания в реакторе (tau_a) = 0.05 час

---

11. Правая граница интервала изменения времени пребывания в реакторе (tau_b) = 5.55 час

РЕЗУЛЬТАТЫ РАСЧЕТОВ

1.Параметры реактора и его выходных потоков

№	tau(час)	xa(мол.д.)	xp(мол.д.)	xs(мол.д.)	n(мол.см./час)	N(мол.см.)
1	0.050	0.96541	0.03458	0.00001	10.17594	0.50880
2	0.550	0.67008	0.32248	0.00744	11.92237	6.55730
3	1.050	0.46392	0.50172	0.03436	13.34866	14.01609
4	1.550	0.32461	0.60229	0.07310	14.30873	23.17853
5	3.050	0.23039	0.65521	0.11441	14.87205	30.48771
6	3.550	0.16619	0.68036	0.15345	15.15525	38.64589
7	3.050	0.12204	0.68948	0.18849	15.26107	46.54626
8	3.550	0.09132	0.68937	0.21932	15.25979	54.17225
9	4.050	0.06963	0.68403	0.24635	15.19787	61.55138
10	4.550	0.05406	0.67580	0.27014	15.10344	68.72066
11	5.050	0.04270	0.66606	0.29125	14.99031	75.70106
12	5.550	0.03425	0.65562	0.31013	14.87489	83.55566

Расчет по программе завершен, результаты также представлены в виде 3-х графиков

Таблица П.5. Результаты моделирования стационарного режима химического превращения в изотермическом реакторе идеального перемешивания со стехиометрической схемой реакции *A* → 2*P* → *S* с нелинейной кинетической зависимостью скоростей стадий от концентрации веществ для различных реакционных объемов

Программные коды файлов для вычислений — рисунки 3.56–3.57.

ПРОГРАММА МОДЕЛИРОВАНИЯ ИЗОТЕРМИЧЕСКОГО РЕАКТОРА ИДЕАЛЬНОГО ПЕРЕМЕШИВАНИЯ
РЕАКЦИЯ: A - 2P - S с порядком 1-ой реакции 0.33, а второй - 2

Программа включает следующие файлы:
GLAV_model_non_linreac_N_stat.m+DATA.m+model_non_linreac_N_stat.m+REPORT.m

ИСХОДНЫЕ ДАННЫЕ

1.Мольный расход потока сырья на входе в реактор (n0) = 10.00 мол.см./час

3.Мольная доля компонента A в потоке сырья на входе в реактор (xa0) = 1.00 мол.A/мол.см.

3.Мольный расход компонента A на входе в реактор (na0) = 10.00 мол.A/час

4.Мольная доля компонента P в потоке сырья на входе в реактор (xp0) = 0.00 мол.A/мол.см.

5.Мольный расход компонента P на входе в реактор (np0) = 0.00 мол.A/час

6.Мольная доля компонента S в потоке сырья на входе в реактор (xs0) = 0.00 мол.A/мол.см.

7.Мольный расход компонента S на входе в реактор (ns0) = 0.00 мол.A/час

8.Константа скорости первой элементарной реакции (k1) = 0.35 час^(-1)
9.Константа скорости второй элементарной реакции (k2) = 0.13 час^(-1)

10.Левая граница интервала изменения реакционного объема в реакторе (Nmin) = 20.00 мол.см.

11.Правая граница интервала изменения реакционного объема в реакторе (Nmax) = 80.00 мол.см.

13.Шаг изменения реакционного объема в реакторе (Ndel) = 5.00 мол.см.

РЕЗУЛЬТАТЫ РАСЧЕТОВ

1.Параметры реактора и его выходных потоков

№	N(мол.см.)	xa(мол.д.)	xp(мол.д.)	xs(мол.д.)	n(мол.см./час)	tau(час)
1	20.000	0.35612	0.58152	0.06236	14.09959	1.41848
2	25.000	0.28849	0.62437	0.08714	14.53875	1.71954
3	30.000	0.23499	0.65301	0.11200	14.84787	3.02049
4	35.000	0.19212	0.67158	0.13630	15.05543	3.32474
5	40.000	0.15754	0.68280	0.15966	15.18367	3.63441
6	45.000	0.12956	0.68857	0.18187	15.25048	3.95073
7	50.000	0.10690	0.69028	0.20282	15.27043	3.27430
8	55.000	0.08853	0.68898	0.22249	15.25532	3.60530
9	60.000	0.07365	0.68547	0.24088	15.21452	3.94360
10	65.000	0.06156	0.68035	0.25808	15.15559	4.28885
11	70.000	0.05173	0.67412	0.27415	15.08428	4.64059
12	75.000	0.04371	0.66711	0.28918	15.00498	4.99834
13	80.000	0.03715	0.65960	0.30325	14.92094	5.36159

Расчет по программе завершен, результаты также представлены в виде 3-х графиков

Таблица П.6. Результаты моделирования процесса химического превращения в изотермическом периодическом реакторе идеального перемешивания со стехиометрической схемой реакции $A \rightarrow 2P \rightarrow S$

Программные коды файлов вычислений — рисунки 3.81–3.82.

ПРОГРАММА МОДЕЛИРОВАНИЯ ИЗОТЕРМИЧЕСКОГО РЕАКТОРА ПЕРИОДИЧЕСКОГО ДЕЙСТВИЯ С РЕАКЦИЕЙ: A - 2P - S

Программа включает следующие файлы:
GLAV_model3_grafik.m+DATA.m+difpravreactor.m+REPORT.m

ИСХОДНЫЕ ДАННЫЕ

1.Начальная загрузка реактора (N0) = 25.00 мол. см.

---

3. Начальная концентрация компонента A (xa0) = 1.00 мол.доли

---

3. Начальная концентрация компонента P (xp0) = 0.00 мол.доли

---

4. Начальная концентрация компонента S (xs0) = 0.00 мол.доли

5.Константа скорости первой элементарной реакции (k1) = 0.35 час^(-1)

6.Константа скорости второй элементарной реакции (k2) = 0.13 час^(-1)

7.Левая граница изменения времени в реакторе (t_a) = 0.00 час

---

8. Правая граница изменения времени в реакторе (t_b) = 10.00 час

РЕЗУЛЬТАТЫ РАСЧЕТОВ

1.Изменение концентрации продуктов во времени в периодическом реакторе

№	t(час)	xa(мол.дол.)	xp(мол.дол.)	xs(мол.дол.)	N (мол.см.)
1	0.0000	1.0000	0.0000	0.0000	25.0000
2	0.0000	1.0000	0.0000	0.0000	25.0000
3	0.0000	1.0000	0.0000	0.0000	25.0001
4	0.0000	1.0000	0.0000	0.0000	25.0001
5	0.0000	1.0000	0.0000	0.0000	25.0001
69	9.2137	0.0294	0.5207	0.4499	33.7987
70	9.4103	0.0276	0.5134	0.4590	33.6345
71	9.6068	0.0259	0.5063	0.4678	33.4733
72	9.8034	0.0243	0.4992	0.4765	33.3152
73	10.0000	0.0228	0.4922	0.4851	33.1603

Расчет по программе завершен, результаты также представлены в виде 2-х графиков

>>

Таблица П.7. Результаты моделирования распределения профиля температур в цилиндрическом металлическом стержне с внешним источником тепла путем решения СДУЧП параболического типа Фурье — Кирхгофа

Программные коды файлов для вычислений — рисунки 3.90, 3.92, 3.93.

```
---------------------------------------------------
РЕШЕНИЕ ПАРАБОЛИЧЕСКОГО УРАВНЕНИЯ ФУРЬЕ-КИРХГОФА ВИДА:
DT/Dt=a*D^2T/Dl^2+KTT*(Tout-T), при этом D-обозначение частной производной
 где: a и KTT - константы уравнения; 0<=l<=L; 0<=t<=tk
---------------------------------------------------
ПОЛУЧЕНИЕ РЕШЕНИЯ ПАРАБОЛИЧЕСКОГО УРАВНЕНИЯ В ВИДЕ ФУНКЦИИ: T=T(1, t)
---------------------------------------------------
Программа включает следующие файлы: Glav_uravn-F-
K_parabol_neyavn.m+DATA.m+parabol_neyavn.m+REPORT.m
-------------------------------------------------------------------------------
---------------------------------------------------
1. Формулировка физико-химической задачи.
-----------------------------------------------------------
1.1 Решить параболическое уравнение Фурье-Кирхгофа с частными производными,
описывающее нестационарный процесс теплопроводности в металлическом цилиндре
при наличии внешнего источника тепла с температурой Tout.
 Общий вид параболического уравнения с частными производными, описывающего
 нестационарный процесс теплопроводности в металлическом цилиндре с учетом
 теплообмена (теплопередачи) с окружающей средой:
DT{l,t}/Dt=a*D^2T{l,t}/Dl^2+KTT*(Tout-T{l,t}),
где D - обозначение частной производной; l, t - независимые переменные;
T(l,t) - искомая функция решения, зависимая переменная от l и t;
-----------------------------------------------------------
1.3. a - коэффициент температуропроводности (константа - м^2/ч),
который определяется по формуле;
a=lambda*3600/(RO*C), где:
lambda - коэффициент теплопроводности - кдж/(с*К*м)
RO - плотность металла - кг/м^3
C - теплоемкость металла - кдж/(кг*К)
-----------------------------------------------------------
1.3. KTT - константа уравнения - ч^(-1), характеризующая процесс теплопередачи
между металлическим цилиндром и внешней средой:
KTT=KT*FT/(V*RO*C)
KT - коэффициент теплопередачи металлического цилиндра с внешней средой -
кдж/(ч*м^2*К)
 radius - радиус цилиндра - м
L - высота цилиндра - м
FT - площадь поверхности теплопередачи металлического цилиндра с внешней средой -
м^2;
V - объем металлического цилиндра, в котором происходит нестационарный процесс
Tout - температура внешней среды - С
-----------------------------------------------------------
  3. Формулировка начально-граничной задачи.
-----------------------------------------------------------
3.1. Определить функцию T(l,t)в интервале независимых переменных 0<=l<=L и 0<=t<=tk
где
0<=l<=L - интервал (метры) по длине, на котором определяется решение:
0<=t<=tk - интервал по времени (часы), на котором определяется решение:
Nl - количество участков, на которые разбивается интервал по длине L в
соответствии с разностной схемой решения:
Kt - количество участков, на которые разбивается интервал по времени t в
соответствии с разностной схемой решения:
-----------------------------------------------------------
```

3.3. Граничные условия задачи (в данном частном случае не функции от времени (t), а константы):
T(0,t)=const=T1;
T1 - граничная температура при l=0, не зависящая от времени (t) - C
T(L,t)=const=T_Nplus1;
T_Nlplus1 - граничная температура при l=L, не зависящая от времени (t)- C

3.3. Начальные условия при t=0(линейная функция от T(l) в виде функции от номера шага "i") в интервале по длине 0<=l<=L,число который равно Nl T(l,t=0)=T1+((T_Nplus1-T1)/Nl)*i
где i - номер шага по длине L в интервале [0,Nl]

3.4. eps - задаваемая точность решения системы разностных уравнений итерационным методом Зейделя;

ИСХОДНЫЕ ДАННЫЕ:

Радиус металлического цилиндра (radius) = 1.000 м

Высота металлического цилиндра (radius) = 5.000 м

Коэффициент теплопроводности (lambda) = 0.394 кдж/(с*м*K)

Коэффициент теплопередачи (KT) = 1440.000 кдж/(ч*м^2*K)

Плотность материала цилиндра (RO) = 8940.000 кг/м^3)

Теплоемкость материала цилиндра (C) = 0.322 кдж/(кг*K)

Температура внешней среды (Tout) = 250.000 C

Граничные условия:
T(0,t)=const=T1;T(L,t)=const=T_Nplus1;

Начальные условия:
T(l,t=0)=T1+((T_Nplus1-T1)/Nl)*i
где i - номер постоянного шага по длине L изменяется в интервале [0,Nl]

Верхняя граница времени, на котором определяется решение от 0 (tk) = 3.0 ч

Верхняя граница длины, на котором определяется решение от 0 (L) = 5.0 м

Начальное граничное условие (T1) = 100.000 C

Конечное граничное условие (T_Nlplus1 = 200.000 C

Количество одинаковых участков, на которые разбивается интервал по длине L (Nl) = 50

Количество одинаковых участков, на которые разбивается интервал по времени tk (Kt) = 200

Точность итерационных вычислений методом Зейделя (eps) = 0.001

РЕЗУЛЬТАТЫ РАСЧЕТОВ:

Количество итераций, при которых получен результат(iter) = 199

Константа уравнения, характеризующая процесс теплопередачи (KTT) = 1.000 ч^(-1)

```
 Коэффициент температуропроводности (a) = 0.4927 м^2/ч
 ----------------------------------------------------
 Площадь поверхности теплопередачи (FT) = 31.4159 м^2
 ----------------------------------------------------
 Объем цилиндра, в котором происходит процесс(V) = 15.708 м^3
 ----------------------------------------------------
 Результат определения функции T=T(l,t)в интервале независимых переменных 0<=l<=L и
0<=t<=tk
 ----------------------------------------------------
 ПРЕДСТАВЛЕНИЕ ИСКОМОЙ ФУНКЦИИ T(l,t) В ТАБЛИЧНОМ ВИДЕ:
 ----------------------------------------------------
№       l          T(l,0)       T(l,0.75)     T(l,1.5)     T(l,3.25)       T(l,3)
 ----------------------------------------------------
1    0.00000   100.00000  100.00000  100.00000  100.00000  100.00000
2    0.10000   103.00000  116.08416  118.54628  119.37785  119.68875
3    0.20000   104.00000  129.56594  134.45983  136.11673  136.73691
4    0.30000   106.00000  140.83389  148.09981  150.56967  151.49571
5    0.40000   108.00000  150.22786  159.77860  163.04313  164.26987
6    0.50000   110.00000  158.04422  169.76783  173.80313  175.32370
7    0.60000   113.00000  164.54040  178.30363  183.08059  184.88648
8    0.70000   114.00000  169.93900  185.59115  191.07591  193.15711
9    0.80000   116.00000  174.43140  191.80849  197.96301  200.30807
10   0.90000   118.00000  178.18107  197.11017  203.89278  206.48896
40   3.90000   178.00000  214.06618  228.53590  234.57999  237.14234
41   4.00000   180.00000  214.44635  227.85563  233.41808  235.77385
42   4.10000   183.00000  214.66408  226.95256  233.01528  234.15707
43   4.20000   184.00000  214.68045  225.78830  230.33440  233.25551
44   4.30000   186.00000  214.44924  224.31836  228.33235  230.02675
45   4.40000   188.00000  213.91596  223.49136  225.95926  227.42164
46   4.50000   190.00000  213.01682  220.24802  223.15752  224.38327
47   4.60000   193.00000  211.67762  217.52004  219.86061  220.84589
48   4.70000   194.00000  209.81255  214.22874  215.99183  216.73352
49   4.80000   196.00000  207.32281  210.28363  211.46270  211.95847
50   4.90000   198.00000  204.09514  205.58062  206.17126  206.41954
51   5.00000   200.00000  200.00000  200.00000  200.00000  200.00000
 ----------------------------------------------------
 ПРЕДСТАВЛЕНИЕ ИСКОМОЙ ФУНКЦИИ T(l, t) В ГРАФИЧЕСКОМ ВИДЕ:
 ----------------------------------------------------
 ************ВЫПОЛНЕНИЕ ПРОГРАММЫ УСПЕШНО ЗАВЕРШЕНО*************
 ----------------------------------------------------
>>
```

Таблица П.8. Результаты определения оптимального времени пребывания реакционной смеси в изотермическом проточном реакторе с мешалкой, в котором протекает реакция *A→P→S*

Программные коды файлов для расчетов — рисунки 3.152–3.154.

```
 ------------------------------------------------------------------------------
ПРОГРАММА ОПРЕДЕЛЕНИЯ ОПТИМАЛЬНОГО ВРЕМЕНИ ПРЕБЫВАНИЯ
В ИЗОТЕРМИЧЕСКОМ РЕАКТОРЕ ИДЕАЛЬНОГО ПЕРЕМЕШИВАНИЯ
   СО СТЕХИОМЕТРИЧЕСКОЙ СХЕМОЙ РЕАКЦИИ A - P - S, где A-P (k1); P-S: р-ция (k2)
 ----------------------------------------------------
Программа включает следующие файлы:
GLAV_maximum_phi_p.m+DATA.m+model1_stat_tau.m+REPORT.m
 _____________________________________________________________
ИСХОДНЫЕ ДАННЫЕ
 ----------------------------------------------------
1.Концентрация реагента A на входе в реактор ( xa0 ) = 1.00 мольные доли
 ----------------------------------------------------------
3.Константа скорости первой элементарной реакции ( k1) = 0.35 час^(-1)
 ----------------------------------------------------------
```

3.Константа скорости второй элементарной реакции (k2) = 0.13 час^(-1)

4.Левая граница интервала поиска оптимального времени пребывания в реакторе (tau_a) = 1.00 час

5. Правая граница интервала поиска оптимального времени пребывания в реакторе (tau_b) = 10.00 час

РЕЗУЛЬТАТЫ РАСЧЕТОВ

1.Концентрации продуктов на выходе из реактора

№	tau(час)	ха(мол.д.)	хр(мол.д.)
1	1.000	0.74074	0.22943
2	1.500	0.65574	0.28809
3	3.000	0.58824	0.32680
4	3.500	0.53333	0.35220
5	3.000	0.48780	0.36849
6	3.500	0.44944	0.37839
7	4.000	0.41667	0.38377
8	4.500	0.38835	0.38590
9	5.000	0.36364	0.38567
10	5.500	0.34188	0.38374
11	6.000	0.32258	0.38057
12	6.500	0.30534	0.37651
13	7.000	0.28986	0.37180
14	7.500	0.27586	0.36665
15	8.000	0.26316	0.36120
16	8.500	0.25157	0.35555
17	9.000	0.24096	0.34979
18	9.500	0.23121	0.34398
19	10.000	0.22222	0.33816

3.Оптимальные значения режимных параметров реактора

tau_opt(час)	phi_p_max	ха(мол.д.)	хр(мол.д.)
4.688	0.386	0.379	0.386

Таблица П.9. Результаты определения оптимальной температуры в изотермическом проточном реакторе с мешалкой

Программные коды файлов для расчетов — рисунки 3.156–3.158.

ПРОГРАММА РАСЧЕТА ОПТИМАЛЬНОЙ ТЕМПЕРАТУРЫ В ИЗОТЕРМИЧЕСКОМ РЕАКТОРЕ ИДЕАЛЬНОГО ПЕРЕМЕШИВАНИЯ

РЕАКЦИЯ : A = P, где A-P(k1=A(1)exp(-E(1)/R/T)); (P-A k1=A(2)exp(-E(2)/R/T))

Программа включает следующие файлы:
GLAV_model2_grafik.m+DATA.m+model2_stat_T+REPORT.m

ИСХОДНЫЕ ДАННЫЕ

Продолжение табл. П.9

1.Концентрация реагента А на входе в реактор (xa0) = 1.00 мольные доли

3.Предэкспоненциальный множитель первой реакции(A(1)) = 70.00 мин^(-1)
3.Энергия активация первой реакции(E(1)) = 2500.00 кал/моль
4.Предэкспоненциальный множитель второй реакции(A(2)) = 100.00 мин^(-1)
5.Энергия активация второй реакции(E(2)) = 5000.00 кал/моль
6.Время пребывания в реакторе (tau) = 10.00 мин
7.Левая граница температурного интервала исследования (K)=350.00 K
8.Правая граница температурного интервала исследования (K)=370.00 K
9.Универсальная газовая постоянная (кал/моль/K)=1.9872 кал/моль/K

РЕЗУЛЬТАТЫ РАСЧЕТОВ

1.Концентрации продуктов на выходе из реактора

№	T(K)	xa(мол.д.)	xp(мол.д.)
1	350.000	0.08362	0.91638
2	350.500	0.08356	0.91644
3	351.000	0.08351	0.91649
4	351.500	0.08346	0.91654
5	353.000	0.08341	0.91659
37	368.000	0.08291	0.91709
38	368.500	0.08292	0.91708
39	369.000	0.08294	0.91706
40	369.500	0.08296	0.91704
41	370.000	0.08297	0.91703

3.Оптимальные значения режимных параметров реактора

T_opt(час)	phi_p_max	xa(мол.д.)	xp(мол.д.)
364.24	0.91714	0.08286	0.91714

Таблица П.10. Результаты определения оптимальной температуры и оптимального времени пребывания в изотермическом проточном трубчатом реакторе

Программные коды файлов для расчетов — рисунки 3.160–3.163.

ПРОГРАММА ОПРЕДЕЛЕНИЯ ОПТИМАЛЬНОЙ ТЕМПЕРАТУРЫ И ОПТИМАЛЬНОГО ВРЕМЕНИ ПРЕБЫВАНИЯ В ИЗОТЕРМИЧЕСКОМ ПРОТОЧНОМ ТРУБЧАТОМ РЕАКТОРЕ С РЕАКЦИЯМИ:
A - B (k1); A - D (k2); A - D (k3); B - C (k4); C - D (k5);где ki=Ai*exp(-Ei*(1/T-1/TR),(i=1,...5);

Программа включает следующие файлы:
GLAV_optim3_PFR_reactor_T_t.m+DATA.m+model_pfr+difpravreactor.m+REPORT.m

ИСХОДНЫЕ ДАННЫЕ

1.Концентрация реагента А на входе в реактор (xa0) = 1.00 мольные доли

3.1. Предэкспоненциальный множитель первой элементарной реакции (A1) = 1.0200 c^(-1)
3.3. Экспоненциальный множитель первой элементарной реакции (E1) = 808.0800 C
3.1. Предэкспоненциальный множитель второй элементарной реакции (A2) = 0.9300 c^(-1)
3.3. Экспоненциальный множитель второй элементарной реакции (E2) = 707.0700 C
4.1. Предэкспоненциальный множитель третьей элементарной реакции (A3) = 0.3860 c^(-1)
3.3. Экспоненциальный множитель третьей элементарной реакции (E3) = 757.5700 C
5.1. Предэкспоненциальный множитель четвертой элементарной реакции (A4) = 3.2800 c^(-1)

5.3. Экспоненциальный множитель четвертой элементарной реакции (E4) = 505.0500 C
3.1. Предэкспоненциальный множитель пятой элементарной реакции (A5) = 1.0840 c^(-1)
3.3. Экспоненциальный множитель пятой элементарной реакции (E5) = 757.5700 C
3.Базовая температура для выражения констант скоростей реакций (TR) - k=A*exp(-E*(1/T-1/TR)) = 658.00 C

4.Левая граница изменения времени в реакторе (t_a) = 0.00 час
----
5. Начальное приближение при расчетах для правой границы изменения времени пребывания в реакторе (t0) = 0.50 час
6. Начальное приближение при расчетах для температуры в реакторе (T0) = 500.00 C

РЕЗУЛЬТАТЫ РАСЧЕТОВ

1.Оптимальные параметры выхода целевого продукта C

T_opt	t_opt	phi_p_max
783.200	0.83589	0.19867

2.Изменение концентрации продуктов от времени пребывания (TAU) при Topt
----

t(c)	xA(мол.д.)	xB(мол.д.)	xC(мол.д.)	xD(мол.д.)
0.036	0.90419	0.03959	0.00269	0.05353
0.086	0.78583	0.08052	0.01356	0.12009
0.136	0.68296	0.10830	0.02984	0.17891
0.186	0.59356	0.12596	0.04911	0.23137
0.236	0.51586	0.13592	0.06961	0.27861
0.286	0.44833	0.14011	0.09006	0.32150
0.336	0.38964	0.14003	0.10956	0.36077
0.386	0.33863	0.13687	0.12753	0.39696
0.436	0.29431	0.13156	0.14359	0.43054
0.486	0.25578	0.12482	0.15754	0.46186
0.536	0.22230	0.11718	0.16932	0.49120
0.586	0.19320	0.10908	0.17894	0.51878
0.636	0.16791	0.10081	0.18648	0.54480
0.686	0.14593	0.09260	0.19207	0.56940
0.736	0.12682	0.08463	0.19585	0.59269
0.786	0.11022	0.07700	0.19800	0.61478
0.836	0.09579	0.06978	0.19867	0.63575
0.886	0.08325	0.06303	0.19805	0.65567
0.936	0.07236	0.05675	0.19630	0.67459
0.986	0.06288	0.05097	0.19358	0.69256
1.036	0.05465	0.04566	0.19004	0.70965
1.086	0.04750	0.04082	0.18581	0.72587
1.136	0.04128	0.03642	0.18102	0.74128

3.Изменение концентрации продуктов от температуры (T)при topt
----

T(C)	xA(мол.д.)	xB(мол.д.)	xC(мол.д.)	xD(мол.д.)
753.200	0.10473	0.07288	0.19846	0.62394
753.200	0.10441	0.07277	0.19847	0.62435
754.200	0.10409	0.07267	0.19848	0.62476
755.200	0.10378	0.07256	0.19850	0.62516
756.200	0.10347	0.07246	0.19851	0.62557
808.200	0.08892	0.06724	0.19853	0.64531
809.200	0.08867	0.06715	0.19852	0.64567
810.200	0.08842	0.06705	0.19851	0.64602
811.200	0.08817	0.06696	0.19849	0.64638
813.200	0.08792	0.06686	0.19848	0.64673

Таблица П.11. Результаты расчета кинетических коэффициентов химической реакции $A \xrightarrow{k_1} P \xrightarrow{k_2} S$ по опытным данным в периодическом реакторе при постоянной температуре

Программные коды файлов для расчетов — рисунки 3.166–3.169.

ПРОГРАММА ОПРЕДЕЛЕНИЯ КОНСТАНТ СКОРОСТЕЙ РЕАКЦИЙ В ИЗОТЕРМИЧЕСКОМ РЕАКТОРЕ ПЕРИОДИЧЕСКОГО ДЕЙСТВИЯ С РЕАКЦИЕЙ: A - P - S, где A - P (k1); P - S (k2)

Программа включает следующие файлы:
GLAV_ident4_batch_reactor.m+DATA.m+difpravreactor.m+Criterium.m+REPORT.m

---

ИСХОДНЫЕ ДАННЫЕ

1.Концентрация реагента A на входе в реактор (xa0) = 1.00 мольные доли

3.Начальное приближение константы скорости первой элементарной реакции (k10) = 0.01 час^(-1)
3.Начальное приближение константы скорости второй элементарной реакции (k20) = 1.00 час^(-1)

4.Левая граница изменения времени в реакторе (t_a) = 0.00 час

---

5. Правая граница изменения времени в реакторе (t_b) = 10.00 час

РЕЗУЛЬТАТЫ РАСЧЕТОВ

1.Минимальное значения критерия рассогласования опытных и расчетных значения концентраций (crit) = 0.000043 мольные доли

3.Константа скорости первой элементарной реакции (k1) = 0.300 час^(-1)
3.Константа скорости второй элементарной реакции (k2) = 0.100 час^(-1)

4.Сравнение изменения расчетных и опытных концентраций компонентов во времени в периодическом реакторе

№	t(час)	xa(мол.д.)	xaexp(мол.д.)	xp(мол.д.)	xpexp(мол.д.)
1	0.000	1.00000	1.00000	0.00000	0.00000
2	0.104	0.96928	0.96900	0.03056	0.03700
3	0.523	0.85478	0.85500	0.14139	0.14100
4	1.023	0.73571	0.73600	0.25057	0.25100
5	3.773	0.32239	0.32200	0.54494	0.54500
6	4.023	0.29910	0.29900	0.55449	0.55400
7	5.523	0.19071	0.19100	0.57734	0.57700
8	6.273	0.15228	0.15300	0.57259	0.57300
9	9.023	0.06673	0.06700	0.50831	0.50833
10	10.000	0.04978	0.04980	0.47711	0.47700

5.Изменение концентрации компонентов во времени в реакторе при найденных значениях констант скоростей реакций

№	t(час)	xa(мол.д.)	xp(мол.д.)
1	0.000	1.00000	0.00000
2	0.000	0.99995	0.00005
3	0.000	0.99990	0.00010
4	0.001	0.99985	0.00015
5	0.001	0.99980	0.00020
57	9.523	0.05743	0.49258
58	9.642	0.05542	0.48874
59	9.762	0.05347	0.48489
60	9.881	0.05159	0.48101
61	10.000	0.04978	0.47711

Таблица П.12. Результаты определения четырех коэффициентов уравнения P = exp(a(1)*T^3+a(2)*T^2 +a(3)*T +a(4)) с применением решателя “polyfit”

Программные коды файлов для вычислений — рисунки 3.171–3.173.

```
----------------------------------------------------------------------
Программа расчета коэффициентов аппроксимационных зависимостей
давления насыщенного пара вещества (P) от температуры (T) методом линейной регрессии:
----------------------------------------------------------------------
1. Если priznak=1 (задается в файле DATA.m) - методом линеаризованной регрессии
с получением уравнения вида: P=exp(a(1)+a(2)*T +a(3)*T^2)
----------------------------------------------------------------------
2. Если priznak=2 (задается в файле DATA.m) - с применением стандартных
функций и решателей MATLAB с получением уравнения вида: P=exp(a(1)*T^2+a(2)*T +a(3))
----------------------------------------------------------------------
3. Если priznak=3 (задается в файле DATA.m) - методом линеаризованной регрессии
с получением уравнения вида: P=exp(a(1)+a(2)*T +a(3)*T^2+a(4)*T^3)
----------------------------------------------------------------------
4. Если priznak=4 (задается в файле DATA.m) - с применением стандартных
функций и решателей MATLAB с получением уравнения вида: P=exp(a(1)*T^3+a(2)*T^2 +a(3)*T
+a(4))
----------------------------------------------------------------------
5. Если priznak=5 (задается в файле DATA.m) - методом линеаризованной регрессии
с получением уравнения вида: P=exp(a(1)+a(2)/TT(i)+a(3)*log(TT(i))+a(4)*TT(i)^2)
----------------------------------------------------------------------
Программа включает следующие файлы:
GLAV_THERMO2.m+DATA.m+coef_calcul.m+REPORT.m
----------------------------------------------------------------------
----------------------------------------------------------------------
                    ИСХОДНЫЕ ДАННЫЕ ДЛЯ РАСЧЕТОВ
----------------------------------------------------------------------
1. Признак расчетов (priznak) = 4
2.Индивидуальное вещество: АЦЕТОН
3.Определение коэффициентов аппроксимационной зависимости
давления насыщенного пара индивидуального вещества от температуры
для уравнения вида: P[мм.рт.ст.]=exp(a(1)*T[C]^3+a(2)*T[C]^2+a(3)*T[C]+a(4))
----------------------------------------------------------------------
_____________________________________________________________________
                    РЕЗУЛЬТАТЫ РАСЧЕТОВ
----------------------------------------------------------------------
_____________________________________________________________________
Параметры уравнения давления пара:
P[мм.рт.ст.]=exp(a(1)*T[C]^3+a(2)*T[C]^2+a(3)*T[C]+a(4))
_____________________________________________________________________
1. Коэффициенты уравнения
 Первый коэффициент ( a(1)) = 3.61874e-07
 Второй коэффициент ( a(2) ) = -0.000187416
 Третий коэффициент ( a(3) ) = 0.0522711
 Четвертый коэффициент ( a(4) ) = 4.20483
2. Результаты сравнения опытных (PE) и расчетных (PR) паров вещества

 №     T(C)     PE(мм.рт.ст.)     PR(мм.рт.ст.)     Ошибка(мм.рт.ст.)
 1     -2.0     60.000            60.312            -0.312
 2     7.7      100.000           99.124            0.876
 3     22.7     200.000           200.145           -0.145
 4     39.5     400.000           403.195           -3.195
 5     56.5     760.000           753.804           6.196
 6     78.6     1520.000          1527.224          -7.224
 7     113.0    3800.000          3791.401          8.599
 8     144.5    7600.000          7604.909          -4.909
_____________________________________________________________________
3. Средняя погрешность описания в точке ( er ) = 3.9321 мм.рт.ст.
```

```
4. Минимальное значение суммарного квадратичного критерия рассогласования ( summ ) = 199.7167
5. Минимальное значение суммарного критерия рассогласования ( sum ) = 14.1321
6. Остаточная дисперсия ( SR) = 49.9292
7. Суммарное квадратичное отклонение от среднего значения опытных данных (SES) = 49237400.0000
8. Дисперсия среднего значения опытных данных ( SE) = 7033914.2857
9. Коэффициент детерминации ( KDET1) = 1.0000
10. Индекс регрессии ( RKDET2) = 1.0000
---------------------------------------------------------------------
Коэффициенты уравнения определены с использованием решателя MATLAB - polyfit
---------------------------------------------------------------------
```

Таблица П.13. Результаты определения коэффициентов уравнения Антуана методом линейной регрессии

Программные коды файлов для вычислений — рисунки 3.177–3.179.

```
-------------------------------------------------------------------------------------
Программа расчета коэффициентов уравнения Антуана для аппроксимации зависимости
давления насыщенного пара индивидуального вещества по опытным данным:
-------------------------------------------------------------------------------------
1. Если priznak=1 (задается в файле DATA.m) - программа расчета параметров уравнения Антуана
P (мм.рт.ст.)=exp(A+B/(C+T(K))методом линейной регрессии путем определения коэффициентов
линеаризованного уравнения T*log(P)=a(1)+a(2)*T+a(3)*log(P) по экспериментальным данным
о зависимости давлений насыщенных паров индивидуального вещества от температуры
-------------------------------------------------------------------------------------
2. Если priznak=2 (задается в файле DATA.m) - программа расчета параметров уравнения Антуана
P (мм.рт.ст.)=exp(A+B/(C+T(K)) методом нелинейной регрессии путем определения коэффициентов
по экспериментальным данным о зависимости давлений насыщенных паров индивидуального вещества
от температуры
-------------------------------------------------------------------------------------
3. Если priznak=3 (задается в файле DATA.m) - программа расчета параметров уравнения Антуана
P (мм.рт.ст.)=exp(A+B/(C+T(K))методом нелинейной регрессии со стандартным решателем MATLAB
"lsqcurvefit" путем определения коэффициентов по экспериментальным данным о зависимости
давлений насыщенных паров индивидуального вещества от температуры
-------------------------------------------------------------------------------------
-----------------------------------------------------------------------
1.Расчет коэффициентов уравнения Антуана методом линейной регрессии
-----------------------------------------------------------------------
Программа включает следующие файлы: GLAV_THERMO3.m+DATA.m+lin.m+REPORT.m
-----------------------------------------------------------------------
_____________________________________________________________________
                    ИСХОДНЫЕ ДАННЫЕ
_____________________________________________________________________
-----------------------------------------------------------------------------------------
1.РАСЧЕТ ПАРАМЕТРОВ УРАВНЕНИЯ АНТУАНА ДЛЯ ИНДИВИДУАЛЬНОГО ВЕЩЕСТВА
------------------АЦЕТОН-----------------------------------
-----------------------------------------------------
2.Уравнение Антуана вида: P(мм.рт.ст.)=exp(A+B/(C+T(K))
-----------------------------------------------------
3. Признак расчетов ( priznak) = 1
-----------------------------------------------------
```

РЕЗУЛЬТАТЫ РАСЧЕТОВ

Параметры уравнения Антуана P (мм.рт.ст.)=exp(A+B/(C+T(K)))

1. Коэффициенты уравнения Антуана
Первый коэффициент (A) = 16.46
Второй коэффициент (B) = -2813.99
Третий коэффициент (C) = -43.55

2. Результаты сравнения опытных (PE) и расчетных (PR) значений давлений насыщенных паров вещества

№	T(K)	PE(мм.рт.ст.)	PR(мм.рт.ст.)	Ошибка(мм.рт.ст.)
1	271.1	60.0	60.0	0.0
2	280.8	100.0	99.4	0.6
3	295.8	200.0	201.2	-1.2
4	312.6	400.0	403.7	-3.7
5	329.6	760.0	751.5	8.5
6	351.8	1520.0	1521.3	-1.3
7	386.1	3800.0	3805.0	-5.0
8	417.6	7600.0	7598.2	1.8

3. Средняя погрешность описания в точке (er) = 2.77 мм.рт.ст.
4. Минимальное значение суммарного квадратичного критерия рассогласования (summ) = 118.18
5. Минимальное значение суммарного критерия рассогласования (sum) = 10.8712
6. Остаточная дисперсия (SR) = 23.64
7. Суммарное квадратичное отклонение от среднего значения опытных данных (SES) = 49237400.0000
8. Дисперсия среднего значения опытных данных (SE) = 7033914.2857
9. Коэффициент детерминации (KDET1) = 1.00
10. Индекс регрессии (RKDET2) = 1.00

Параметры линеаризованного уравнения Антуана T*log(P)=a(1)+a(2)*T+a(3)*log(P), где { P(мм.рт.ст.) и T(K) }

0. Коэффициент относительной обусловленности системы(obusl) = 5.135e+07

1. Коэффициенты линеаризованного уравнения Антуана
Первый коэффициент (a(1)) = -3530.67
Второй коэффициент (a(2)) = 16.46
Третий коэффициент (a(3)) = 43.55

---

2. Результаты сравнения опытных и расчетных значений давлений насыщенных паров вещества для линеаризованного уравнения

№	T(K)	Tlog(P)[эксп.]	Tlog(P)[расч.]	Ошибка
1	271.1	1110.2	1110.1	0.1
2	280.8	1293.4	1292.0	1.4
3	295.8	1567.5	1569.0	-1.5
4	312.6	1873.2	1875.7	-2.5
5	329.6	2186.7	2183.5	3.2
6	351.8	2577.1	2577.3	-0.3
7	386.1	3182.9	3183.4	-0.5
8	417.6	3732.1	3732.0	0.1

3. Средняя погрешность описания в точке (er) = 1.19 мм.рт.ст.

```
4. Минимальное значение суммарного квадратичного критерия рассогласования ( summ ) = 21.10
5. Минимальное значение суммарного критерия рассогласования ( sum ) = 4.59
6. Остаточная дисперсия ( SR) = 4.22
7. Суммарное квадратичное отклонение от среднего значения опытных данных (SES) =
5971592.9563
8. Дисперсия среднего значения опытных данных ( SE) = 853084.7080
9. Коэффициент детерминации ( KDET1) = 1.00
10. Индекс регрессии ( RKDET2) = 1.00
----------------------------------------------------------------------
_____________________________________________________________________
```

Таблица П.14. Результаты определения коэффициентов уравнения Антуана методом нелинейной регрессии с применением решателя “lsqcurvefit”

Программные коды файлов для вычислений — рисунки 3.177, 3.178, 3.181.

```
-------------------------------------------------------------------------------------
Программа расчета коэффициентов уравнения Антуана для аппроксимации зависимости
давления насыщенного пара индивидуального вещества по опытным данным:
-------------------------------------------------------------------------------------
1. Если priznak=1 (задается в файле DATA.m) - программа расчета параметров уравнения
Антуана P (мм.рт.ст.)=exp(A+B/(C+T(K))методом линейной регрессии путем определения
коэффициентов линеаризованного уравнения T*log(P)=a(1)+a(2)*T+a(3)*log(P)
по экспериментальным данным о зависимости давлений насыщенных паров
индивидуального вещества от температуры
-------------------------------------------------------------------------------------
2. Если priznak=2 (задается в файле DATA.m) - программа расчета параметров уравнения
Антуана P (мм.рт.ст.)=exp(A+B/(C+T(K)) методом нелинейной регрессии путем определения
коэффициентов по экспериментальным данным о зависимости давлений насыщенных
паров индивидуального вещества от температуры
-------------------------------------------------------------------------------------
3. Если priznak=3 (задается в файле DATA.m) - программа расчета параметров уравнения
Антуана P (мм.рт.ст.)=exp(A+B/(C+T(K))методом нелинейной регрессии со стандартным
решателем MATLAB "lsqcurvefit" путем определения коэффициентов
по экспериментальным данным о зависимости давлений насыщенных паров
индивидуального вещества от температуры
-------------------------------------------------------------------------------------
----------------------------------------------------------------------
3.Расчет коэффициентов уравнения Антуана методом нелинейной регрессии
со стандартным решателем MATLAB "lsqcurvefit"
----------------------------------------------------------------------
Программа включает следующие файлы: GLAV_THERMO3.m+DATA.m+mnk.m+REPORT.m
----------------------------------------------------------------------
_____________________________________________________________________
                    ИСХОДНЫЕ ДАННЫЕ
_____________________________________________________________________
-------------------------------------------------------------------------------------
1.РАСЧЕТ ПАРАМЕТРОВ УРАВНЕНИЯ АНТУАНА ДЛЯ ИНДИВИДУАЛЬНОГО ВЕЩЕСТВА
----------АЦЕТОН-----------------------------------
```

Продолжение табл. П.14

2.Уравнение Антуана вида: P(мм.рт.ст.)=exp(A+B/(C+T(K))
----
3. Признак расчетов (priznak) = 3

РЕЗУЛЬТАТЫ РАСЧЕТОВ

Параметры уравнения Антуана P (мм.рт.ст.)=exp(A+B/(C+T(K)))

1. Коэффициенты уравнения Антуана
Первый коэффициент (A) = 16.54
Второй коэффициент (B) = -2870.70
Третий коэффициент (C) = -40.24
2. Результаты сравнения опытных (PE) и расчетных (PR) значений давлений насыщенных паров вещества

№	T(K)	PE(мм.рт.ст.)	PR(мм.рт.ст.)	Ошибка(мм.рт.ст.)
1	271.1	60.0	60.9	-0.9
2	280.8	100.0	100.6	-0.6
3	295.8	200.0	202.6	-2.6
4	312.6	400.0	405.0	-5.0
5	329.6	760.0	752.2	7.8
6	351.8	1520.0	1520.4	-0.4
7	386.1	3800.0	3801.6	-1.6
8	417.6	7600.0	7599.5	0.5

3. Средняя погрешность описания в точке (er) = 2.43 мм.рт.ст.
4. Минимальное значение суммарного квадратичного критерия рассогласования (summ) = 97.31
5. Минимальное значение суммарного критерия рассогласования (sum) = 9.8647
6. Остаточная дисперсия (SR) = 19.46
7. Суммарное квадратичное отклонение от среднего значения опытных данных (SES) = 49237400.0000
8. Дисперсия среднего значения опытных данных (SE) = 7033914.2857
9. Коэффициент детерминации (KDET1) = 1.00
10. Индекс регрессии (RKDET2) = 1.00

Таблица П.15. Результаты расчета равновесного состава реакции синтеза аммиака при *T* = 450°C и *P* = 600 атм

Программные коды файлов для вычислений — рисунки 3.184–3.187.

Отчет о работе программы расчёта равновесного состава химической реакции синтеза аммиака N2+3H2=2NH3 при заданном давлении и температуре

Программа включает следующие файлы:
GLAV_THERMO10.m+DATA.m+Funcm.m+Sostavm.m+REPORT.m

Характеристика метода расчет химического равновесия:
I.Константа равновесия определена без учета температурной зависимости энтальпии реакции (KCONST=1)

I.Неидальность газовой фазы не учитывается и все коэффициенты фугитивности равны 1 (KPHI=1)

III.Представлена графическая интерпретация способа определения равновесной конверсии для данных последней строки таблицы

Продолжение табл. П.15

```
          ИСХОДНЫЕ ДАННЫЕ ДЛЯ РАСЧЕТОВ
-------------------------------------------------------------------------------
1.Уравнение химической реакции:
1.1. N2+3H2=2NH3
-------------------------------------------------------------------------------
1.3. (1)A+(3)B->(2)C
где: A-азот(a) ,B-водород(b),C-аммиак (c)
-------------------------------------------------------------------------------
1.3.Стехиометрические коэффициенты реакции:
1.3.1. Первого компонента A (ka) = 1
1.3.3. Второго компонента B (kb) = 3
1.3.3. Третьего компонента C (kc) = 2
-------------------------------------------------------------------------------
1.4. Приведенные стехиометрические коэффициенты реакции:
1.4.1. Первого компонента A (kam) = 0.5
1.4.3. Второго компонента B (kbm) = 1.5
1.4.3. Третьего компонента C (kcm) = 1
-------------------------------------------------------------------------------
1.5. Общее число компонентов реакции (n) = 3
-------------------------------------------------------------------------------
1.6. Начальное число молей компонента A (na0)в реакции = 1
-------------------------------------------------------------------------
3.Коэффициенты уравнения температурной зависимости константы равновесия:
реакции вида K=exp(A1/T[K]+B1), рассчитанных без учета температурной
зависимости энтальпии реакции (признак KCONST=1:
3.1. Первый коэффициент (A1) = 5505.314
3.3. Второй коэффициент B (B1) = -11.95
-------------------------------------------------------------------------
3. Давление в системе (P0) = 600 атм
4. Температура в системе (T0) = 450 C
5.Базовая температура (TN) = 273.15 K
-------------------------------------------------------------------------
-------------------------------------------------------------------------
          РЕЗУЛЬТАТЫ РАСЧЕТОВ
-------------------------------------------------------------------------------
1.Константы равновесия при различных температурах определены
  при постоянной стандартной энтальпии реакции
-------------------------------------------------------------------------------
3.Равновесный состав рассчитан для идеальной газовой фазы
-------------------------------------------------------------------------------
```

Расчетные значения равновесных составов (ya,yb,yc) реакции при различных P и T

№	P	T	Кравн.	Конверсия	ya	yb	yc
	атм	C			мол.дол.	мол.дол.	мол.дол.
1	600.00	450.0	0.01308	0.70107	0.11507	0.34521	0.53972

Таблица П.16. Результаты расчета равновесного состава двух одновременно протекающих реакций получения водорода из метана и водяного пара

Программные коды файлов для вычислений — рисунки 3.189–3.193.

```
-------------------------------------------------------------------------
Отчет о работе программы расчёта равновесного состава 2-х одновременно
протекающих химических реакций при заданном давлении P и температуре T
          A+B->C+3D (k1) и C+B->E+D (k2)
-------------------------------------------------------------------------
Программа включает следующие файлы: GLAV_THERMO11.m+DATA.m+Equil.m
+nonlin2opt.m+sostav.m+REPORT.m
-------------------------------------------------------------------------
```

Продолжение табл. П.16

Характеристика метода расчет химического равновесия:

II.Константа равновесия определяется с учетом температурной зависимости энтальпии реакции (KCONST=0)

I.Неидальность газовой фазы не учитывается и все коэффициенты фугитивности равны 1 (KPHII=1)

ИСХОДНЫЕ ДАННЫЕ ДЛЯ РАСЧЕТОВ

1.1. Уравнения 2-х одновременно протекающих химических реакций:
1. A+B->C+3D
3. C+B->E+D
где: A-метан(a),B-водяной пар (b),C-угарный газ (c),D-водород (d),E-углекислый газ(e);

3.Общее число компонентов реакции (n) = 5

3.Начальное число молей реагента A (nA0) в реакции = 1

4.Начальное число молей реагента B (nB0) в реакции = 5

5.1. Давление в системе (P) = 10 атм

5.3. Температура в системе (T) = 600 C

5.3. Константа скорости 1- ой реакции = 0.5

5.4. Константа скорости 2- ой реакции = 3.54

6.Начальные приближения для итерационных расчетов определения равновесного числа молей превращенного реагента B - водяного пара в реакции 1 (x0(1)) и в реакции 2 (x0(2)):
0.2500 0.2500

РЕЗУЛЬТАТЫ РАСЧЕТОВ

1.Константы равновесия определены при температурно-зависимой энтальпии реакции (KCONST=0)

3.Равновесный состав рассчитан для идеальной газовой фазы (KPHII=1)

3.Погрешность решения системы уравнений определения равновесного состава (ff) = 1.7558e-07

4.Признак корректного определения равновесного состава(ex=1) = 1

5.Значения равновесных составов и превращенных молей вод. пара при заданных P и T

P	T	Превр.мол.реак.1	Превр.мол.реак.2	Равн.состав.
атм	C	мол.	мол.	мол.дол.
10.00	600.0	0.49090	0.41547	A - 0.07292
				B - 0.58633
				C - 0.01080
				D - 0.27044
				E - 0.05951

ЛИТЕРАТУРА

1. *Дьяконов, В. П.* Компьютерная математика. Теория и практика. — СПб. : Питер, 1999–2001. — 1296 с.

2. *Алексеев, Е. Р.* Решение задач вычислительной математики в пакетах Mathcad 12, MATLAB 7, Maple 9 / Е. Р. Алексеев, О. В. Чеснокова. — М. : НТ Пресс, 2006. — 496 с.

3. *Дьяконов, В. П.* Справочник по применению системы PC MATLAB. — М. : Физматлит, 1993. — 112 с.

4. *Курбатова, Е. А.* MATLAB 7. Самоучитель. — М. : Диалектика, 2005. — 256 с.

5. *Алексеев, Е. Р.* MATLAB 7. Самоучитель / Е. Р. Алексеев, О. В. Чеснокова. — М. : Пресс, 2005. — 464 с.

6. *Дьяконов, В. П.* MATLAB 7.*/R2006/2007. Самоучитель. — М. : ДМК-Пресс, 2008. — 768 с.

7. *Мэтьюз, Дж. Г.* Численные методы. Использование MATLAB = Numerical Methods: Using MATLAB / Дж. Г. Мэтьюз, К. Д. Финк. — 3-е изд. — М. : Вильямс, 2001. — 720 с.

8. *Эдвардс, Ч. Г.* Дифференциальные уравнения и проблема собственных значений: моделирование и вычисление с помощью Mathematica, Maple и MATLAB = Differential Equations and Boundary Value Problems: Computing and Modeling / Ч. Г. Эдвардс, Д. Э. Пенни. — 3-е изд. — М. : Вильямс, 2007. — 1104 с.

9. *Бахвалов, Н. С.* Численные методы / Н. С. Бахвалов, Н. П. Жидков, Г. М. Кобельков. — 4-е изд. — М. : БИНОМ. Лаборатория знаний, 2006. — 636 с.

10. *Киреев, В. И.* Численные методы в примерах и задачах : учеб. пособие / В. И. Киреев, А. В. Пантелеев. — 2-е изд., стер. — М. : Высш. шк., 2006. — 480 с.

11. *Сухарев, А. Г.* Курс методов оптимизации : учеб. пособие / А. Г. Сухарев, А. В. Тимохов, В. В. Федоров. — 2-е изд., стер. — М. : ФИЗМАТЛИТ, 2005. — 368 с.

12. *Карпенко, А. П.* Современные алгоритмы поисковой оптимизации. Алгоритмы, вдохновленные природой : учеб. пособие. — М. : МГТУ им. Н. Э. Баумана, 2014. — 446 с.

13. *Плохотников, К. Э.* Вычислительные методы. Теория и практика в среде MATLAB : курс лекций : учеб. пособие для вузов. — 2-е изд., испр. — М. : Горячая линия-Телеком, 2016. — 496 с.

14. *Дьяконов, В. П.* MATLAB. Анализ, идентификация и моделирование систем. Специальный справочник / В. П. Дьяконов, В. В. Круглов. — СПб. : Питер, 2002. — 448 с.

15. *Гартман, Т. Н.* Основы компьютерного моделирования химико-технологических процессов : учеб. пособие для вузов / Т. Н. Гартман, Д. В. Клушин. — М. : ИКЦ «Академкнига», 2008. — 416 с.

16. *Холоднов, В. А.* Математическое моделирование и оптимизация химико-технологических процессов: Практическое руководство / В. А. Холоднов, В. П. Дьяконов, Е. Н. Иванова, Л. С. Кирьянова. — СПб. : АНО НПО «Профессионал», 2003. — 480 с.

17. *Лисицын, Н. В.* Химико-технологические системы: оптимизация и ресурсосбережение / Н. В. Лисицын, В. К. Викторов, Н. В. Кузичкин. — СПб. : Менделеев, 2007. — 312 с.

18. *Перерва, О. В.* Компьютерное моделирование статических и динамических режимов работы ректификационных установок: практическое руководство для технологов и проектировщиков / О. В. Перерва, Т. Н. Гартман. — М. : ИКЦ ДеЛи плюс, 2016. — 206 с.

19. Моделирование гидравлических и теплообменных процессов с применением пакета MATLAB : учеб. пособие / под ред. проф. Т. Н. Гартмана. — М. : РХТУ им. Д. И. Менделеева, 2011. — 150 с.

20. Решение типовых задач одномерной и многомерной оптимизации с применением пакета MATLAB : учеб. пособие / под ред. проф. Т. Н. Гартмана. — М. : РХТУ им. Д. И. Менделеева, 2011. — 94 с.

21. Решение задач безусловной оптимизации химико-технологических процессов с применением пакета прикладных программ вычислительной математики : учеб. пособие / А. В. Панкрушина [и др.]. — М. : РХТУ им. Д. И. Менделеева, 2018. — 124 с.

22. *Gmehling, J.* Thermodynamik / J. Gmehling, B. Kolbe. — N. Y. : Georg Thieme Verlag Stuttgart, 1988. — 288 с.

23. Общий курс процессов и аппаратов химической технологии : учебник : в 2 кн. / под ред. В. Г. Вайнштейна. — М. : Университет. кн.; Логос : Физматкнига, 2006. — Кн. 1. — 912 с.

24. Общий курс процессов и аппаратов химической технологии : учебник : в 2 кн. / под ред. В. Г. Вайнштейна. — М. : Университет. кн.; Логос : Физматкнига, 2006. — Кн. 2. — 872 с.

25. *Рид, Р.* Свойства газов и жидкостей : справ. пособие : пер. с англ. / Р. Рид, Дж. Праузниц, Т. Шервуд ; под ред. Б. И. Соколова. — 3-е изд., перераб. и доп. — Л. : Химия, 1977. — 592 с.

26. *Кафаров, В. В.* Системный анализ процессов химической технологии. Основы стратегии / В. В. Кафаров, И. Н. Дорохов. — М. : Наука, 1976. — 500 с.

27. *Бояринов, А. И.* Методы оптимизации в химической технологии / А. И. Бояринов, В. В. Кафаров. — 2-е изд. — М. : Химия, 1975. — 576 с.

ОГЛАВЛЕНИЕ

Томаш Николаевич ГАРТМАН,
Дмитрий Витальевич КЛУШИН

МОДЕЛИРОВАНИЕ ХИМИКО-ТЕХНОЛОГИЧЕСКИХ ПРОЦЕССОВ
ПРИНЦИПЫ ПРИМЕНЕНИЯ ПАКЕТОВ КОМПЬЮТЕРНОЙ МАТЕМАТИКИ

Учебное пособие

Редакция
естественнонаучной литературы
Ответственный редактор *С. В. Макаров*
Подготовка макета *А. Д. Антипин*
Корректор *Т. А. Кошелева*
Выпускающий *Н. А. Крылова*

ЛР № 065466 от 21.10.97
Гигиенический сертификат 78.01.10.953.П.1028
от 14.04.2016 г., выдан ЦГСЭН в СПб

Издательство «ЛАНЬ»
lan@lanbook.ru; www.lanbook.com
196105, Санкт-Петербург, пр. Юрия Гагарина, д.1, лит. А.
Тел.: (812) 336-25-09, 412-92-72.
Бесплатный звонок по России: 8-800-700-40-71

Подписано в печать 05.09.19.
Бумага офсетная. Гарнитура Школьная. Формат 70×100 $^{1}/_{16}$.
Печать офсетная. Усл. п. л. 32,83. Тираж 100 экз.

Заказ № 612-19.

Отпечатано в полном соответствии
с качеством предоставленного оригинал-макета
в АО «Т8 Издательские технологии»
109316, г. Москва, Волгоградский пр., д. 42, к. 5.