FUNDAMENTAL CHEMISTRY WITH MATLAB

Daniele Mazza | Enrico Canuto

ELSEVIER

Fundamental Chemistry With Matlab

Fundamental Chemistry With Matlab

Daniele Mazza
Former Faculty, Politecnico di Torino, Turin, Italy
e-mail: mazzad50@gmail.com

Enrico Canuto
Former Faculty, Politecnico di Torino, Turin, Italy
e-mails: enrico.canuto@formerfaculty.polito.it;
canuto.enrico@gmail.com

ELSEVIER

Elsevier
Radarweg 29, PO Box 211, 1000 AE Amsterdam, Netherlands
The Boulevard, Langford Lane, Kidlington, Oxford OX5 1GB, United Kingdom
50 Hampshire Street, 5th Floor, Cambridge, MA 02139, United States

Notices

Knowledge and best practice in this field are constantly changing. As new research and experience broaden our understanding, changes in research methods, professional practices, or medical treatment may become necessary.

Practitioners and researchers must always rely on their own experience and knowledge in evaluating and using any information, methods, compounds, or experiments described herein. In using such information or methods they should be mindful of their own safety and the safety of others, including parties for whom they have a professional responsibility.

To the fullest extent of the law, neither the Publisher nor the authors, contributors, or editors, assume any liability for any injury and/or damage to persons or property as a matter of products liability, negligence or otherwise, or from any use or operation of any methods, products, instructions, or ideas contained in the material herein.

ISBN: 978-0-323-91341-6

Publisher: Susan Dennis
Acquisitions Editor: Kathryn Eryilmaz
Editorial Project Manager: Franchezca Cabural
Production Project Manager: Bharatwaj Varatharajan
Cover Designer: Miles Hitchen

Typeset by MPS Limited, Chennai, India

Dedication

To our families

Contents

Introduction

Matlab language and environment

Most developers today use the so-called "third-generation languages" such as C, C++, Python, and Java. A third-generation language, a general-purpose language in nature, gives the developer the kind of precise control needed to write exceptionally fast applications that can perform a wide array of tasks. Fourth-generation languages, like Matlab (short for Matrix Laboratory), on the contrary, are designed with a specific purpose in mind, which in the case of Matlab, is *scientific and technical computing*. In this sense, it has been designed for empowering the user to work with heterogeneous collections of data, rather than individual variables, making it easier for the user to focus on the task, instead on the language.

As programming languages progress through generations, they become more intelligible and closer to human abstraction and language. Matlab took advantage since the early developments of the abstraction levels inherent in the mathematical language and operations, thus offering the user a mathematical environment suitable to any hardware and operating system platform. Subsequent developments added graphical environments like that of Simulink devoted to dynamic systems and their automatic control (to be employed in Chapters 3 and 4), graphical user interfaces, document editing, and a rich set of toolboxes covering a wide spectrum of scientific and technical computing and real-time applications in measurements and control.

The chemistry computations proposed in the book will just employ the Matlab core, without any reference to Matlab toolboxes. To this end, we will exploit functions of the following areas addressed by the core: Algebra, Linear algebra (many equations dealing with many unknowns), Calculus, Differential equations, Optimization, Linear regression, Statistics, Curve fitting, and Graphing.

Code listing in this book is specifically designed to work on Matlab platform. However, with a little effort, most of the simple scripts can be exported to Octave, a very similar but free platform. Indeed, if you are looking for an open-source environment close to Matlab in terms of compatibility and computational ability, then Octave is the best alternative. It runs on any operating system without any modifications. Scilab is another open-source option for numerical computing which runs across all the major platforms. Like Octave, it is very similar to Matlab in its implementation, although exact compatibility is not a goal of the project developers. It has the advantage of possessing a graphical interface similar to Simulink, named Xcos [1].

The Matlab scripts

Any computational exercise will be accompanied by a Matlab script, named like `InitChem.m`, where `<.m>` is the extension of Matlab scripts, collecting data from external files, performing computations, and providing results either in a tabular form in the `Command Window` or graphical plots in appropriate windows. No specific introduction to Matlab is provided, except to Simulink in Appendix D (Chapter 14), but the book scripts and the key Matlab functions will be explained in

some detail. The web version of the book will show the default colors which distinguish three kinds of sentences and variables in the script:

1. Black is reserved to statements to be executed.
2. Green is reserved to comments which are introduced by the % character.
3. Bold green is reserved to comments introduced by the %% characters. They split the script into sections which can run separately or step by step. Such comments have been exploited to split the script into sections with different functionalities, for instance, initialization, user data statements, data collection from external excel sheets, main computation, result printout, and 2D and 3D graphical plots.

Fairly all of the scripts designed to produce the book's graphical plots are reported in the book itself. The complete library of scripts (main scripts and functions) of extension <.m>, excel files of extension <.xlsx> and Simulink block diagrams (models) of extension <.slx> can be found in the companion website of the book [2]. Care has been taken to ensure the script execution without warnings and errors, at least as they are reported in the book and on the companion website [2]. Of course, any bug can be reported to the authors via their e-mail addresses.

Simulink models of extension <.slx> could not be included in the book (see Appendix D, Chapter 14), but the image of several block diagrams has been added to Chapters 3 and 4.

Matlab scripts and Simulink models have been developed with an academic license of Matlab/Simulink. No Matlab toolbox has been employed.

The book topics

The 10 chapters of the book will focus on topics that are only partial sections of the vast fields covered by fundamental and general chemistry. The curious reader will be able to spot similarities in adjacent fields of chemistry, not covered yet (but may be in a future treatise), and to adapt codes to this aim. Clues to this are be given below, in the summary of chapter's topics. Numerical data elaboration for advanced techniques in chemistry like X-ray diffraction, nuclear magnetic resonance, molecular dynamics, or lattice energetics, are not covered in this context; the relevant programs are to be found in specific websites or textbooks, being mainly addressed to scholars in the respective sectors.

Book chapters can be arranged into four groups, plus six appendixes of help to specific topics and chapters.

Group one: atomic orbitals

Plotting atomic orbitals with Matlab (Chapter 1)

The mathematical description of the undulatory behavior of electrons in atoms is shortly reviewed. On this basis, electron wave functions are described and plotted in a simplified manner using Matlab `surf` and `fsurf` functions. Some elements of wave functions theory and the Schroedinger equation are recalled in Appendix F. Molecular orbitals are not covered yet but could be in a next edition.

Group two: stoichiometry and kinetics

Balancing chemical reactions with Matlab (Chapter 2)

The algebra underlying the balancing of chemical reactions is highlighted. In such a way, a generic and straightforward solution is devised for simple acid−base or complex redox reactions. This could help solve troublesome issues, being the correct mass balance of utmost relevance in general and educational chemistry. The foundations of the algebraic algorithm employed are reviewed in Appendix A. Only some examples of reactions are given, while the reader will exploit others as well among the many possible.

Chemical kinetics aided by Matlab/Simulink (Chapter 3)

The study of the rate of a chemical reaction in both textbooks and advanced applications requires to deal with differential equations. Their solution, even for simple applications, is here easily enabled by Matlab Ordinary Differential Equations solver and Matlab/Simulink block diagrams. The treatment agrees with the methods of system and control engineering. The foundations are recalled in Appendix B. The two case studies, NO oxidation and ozone decomposition, can be extended to many other known reactions.

More complex kinetics aided by Matlab/Simulink (Chapter 4)

As an example, for those interested in more complex kinetics, like the famous oscillating reactions, here an ample description is to be found, with stunning graphics and plotting, enabled by Matlab. As already mentioned, this topic may be of interest to researchers in system theory and control engineering.

Group three: gases and vapors

Gaseous reactions and equilibria aided by Matlab (Chapter 5)

Dealing with thermodynamics of even simple equilibrium reactions in a varying temperature/pressure environment requires a precise approach. Matlab functions enable to read excel files with the parameters needed to calculate entropy, enthalpy, and free energy and therefrom the equilibrium parameters for some reactions. By exploiting the indicated thermodynamic database, other reactions will be tackled, among the many of interest in environmental issues.

Physical properties of gases and vapors aided by Matlab (Chapter 6)

Electric interaction between molecules is responsible for the phase transformation of gases to liquids. A number of equations can be used for describing the behavior of real gases, in a wide range of temperatures and pressures, which are graphically displayed by Matlab scripts. A gas of choice can be used, as long as its van der Waals parameters are known.

Group four: aqueous solutions

Exploring acid–base equilibria in water with Matlab (Chapter 7)

In general chemistry, this is a relevant topic. Many Matlab functions, among them `fzero`, enables in a few program lines to solve even complex equilibria of this kind. Examples range from monoprotic acids to titration of polyprotic acids. Although a few working examples are given, the reader will be able to insert other acid/base equilibria in solution, even admixtures of acids and bases.

Colligative properties of solutions aided by Matlab (Chapter 8)

These properties only depend on the solute molar concentration and temperature, being independent of the nature of the solute particles. As an example, aqueous solution of NaCl is chosen and discussed, but the conclusion can be extended to all other solutions of this type, including other binary, ternary, or complex salts.

Exploring seawater chemical equilibria with Matlab (Chapter 9)

Seawater is not a simple reservoir of different dissolved salts, like sodium chloride, but is capable of reacting with different substances like the atmosphere's carbon dioxide, by modifying its concentration and buffering the anthropogenic increase. This is one of today most debated topics. The only quantitative way to tackle the issue is by proper solution of chemical equilibrium equations. All this is discussed, explained, and calculated by the simple use of Matlab scripts. As with other scripts of this book, the reader can vary the relevant parameters and discover what happens. The variety of simulations is really amazing, but only some of them are discussed

Prevalence diagrams for some common elements aided by Matlab (Chapter 10)

Any element in aqueous solution is capable of transforming into different chemical forms, according to pH and E_H, the redox potential of the same solution. In this way, we obtain pH/E_H diagrams, known as prevalence (or Pourbaix) diagrams (M. Pourbaix, 1904–98). Matlab graphical capabilities enable an easy representation of these diagrams for some common elements, like Fe, S, and Mn. The methods can be applied as well to other elements, by providing the pertinent electrochemical potentials.

Appendices

Appendix A: Introduction to linear algebra
Appendix B: Introduction to dynamic systems
Appendix C: Introduction to linear regression
Appendix D: Introduction to Matlab Simulink
Appendix E: Table of sea water coefficients
Appendix F: Introduction to the Schroedinger equation

Authorship and acknowledgments

The book was conceived by the first author, as an offspring of his several decade experience as a teacher of fundamental chemistry at Politecnico di Torino, Turin, Italy, and recently as author of two books on current hot topics in environment chemistry [3, 4]. The second author was firstly involved to improve the stoichiometric balance of Chapter 2, taking advantage of the nullspace algorithm. Despite his scholar-level chemistry knowledge, he became passionate of the first author's conception, topics, and algorithms, leading him to a full immersion in the draft revision, enrichment, and completion. He brought a 2-year experience in the conception and writing of the 800 pages of the book referenced in Reference [5] as well as a long-time practice in the field of dynamic system simulation and control aided by Matlab/Simulink environment.

A first draft of the book was written with the Word text editor of Microsoft Office, but soon the authors realized that their collaboration could greatly benefit from a free text editor like Writer of the LibreOffice suite. The only Microsoft Office residuals are the excel tables, which accompany, as data sources, the abundant suite of Matlab scripts presented by the book.

Last but not least, the work of both authors could not have been matured without the patience and support of their respective wives, Maria Teresa and Maria Angela.

Measurement units and universal constants

As a baseline, the SI (International System of Units) measurement units will be employed, although paying attention to technical units still widely employed, but specifying as in Table 1, if they are accepted by the *Bureau International des Poids et Mésures* [8]. However, in specific Chapters, for instance 5 and 6, measurement units will follow those pertinent to tabulated data, even if not strictly adhering to SI standards.

Table 2 reports the universal constants employed in the book.

Table 1 Measurement units employed in the book

No.	Name	Symbol	SI unit symbol	Conversion to SI	Accepted (YES) and not accepted (NO) by SI
1	Standard atmosphere	atm	Pa (Pascal)	101,325	NO, used in Chapters 5, 6
2	Liter	L	dm^3	1	YES
3	Angular degree	°	rad (radian)	$\pi/180$	YES
4	Molar concentration	M	mol/dm^3 (mol/L)	1	NO (may be confused with the prefix M), used in graphical plots of Chapter 9
5	Ångström	Å	nm (nanometer)	0.1	NO, avoided
6	Bar	Bar	Pa (Pascal)	100,000	NO, mentioned in Chapter 9 and 10
7	Degree Celsius	°C	K	K-273.15	SI derived
8	Tonne (metric ton)	t	Mg = 1000 kg	1	YES
9	Part per million of volume	ppmv	cm^3/m^3	1	YES but it must be unambiguous

Table 2 Universal constants employed in the book

No.	Name	Symbol	Unit	Value	Comment
1	Universal gas constant	R	$J/(mol \cdot K)$	8.3144	$cm^3 \cdot MPa/(mol \cdot K)$
2	Elementary electric charge	e	C	0.1602×10^{-18}	
3	Reduced Planck constant	$\overline{h} = h/(2\pi)$	Js	0.1054×10^{-33}	
4	Electron mass	m_e	kg	0.9109×10^{-30}	
5	Vacuum permittivity	ϵ_0	C/(Vm)	8.854×10^{-12}	
6	Avogadro number	N_A	mol^{-1}	0.60221×10^{24}	
7	Faraday constant	$F = e \times N_A$	C/mol	0.16022×10^{-18}	
8	Bohr radius	a_0	m	0.05292×10^{-9}	

Standard conditions for temperature and pressure

Standard temperature and pressure (STP) [6, 7] are sets of conditions to allow comparisons to be made between different sets of data. The most used standards are those of the International Union of Pure and Applied Chemistry (IUPAC), although these are not universally accepted standards.

In chemistry, IUPAC changed the definition of standard temperature and pressure in 1982. Until 1982, STP was defined as a temperature of 273.15K (0°C) and an absolute pressure of exactly 1atm (101.325kPa). Since 1982, STP is defined as a temperature of 273.15K (0°C) and an absolute pressure of exactly 100.0kPa (1bar). Authors' intention was to adhere to IUPAC recommendations, but in some chapters, notably in Chapters 5 and 6, the past standard was followed in order to be coherent with available tabulated data.

STP should not be confused with the *standard state* commonly used in thermodynamic evaluations of the Gibbs energy of a reaction. In chemistry, the *standard state* of a pure substance, mixture or solution, is a reference for calculating the properties under different conditions. A superscript zero is used to denote a thermodynamic variable in the standard state, such as change in enthalpy ΔH^0, change in entropy ΔS^0, or change in Gibbs free energy ΔG^0. The IUPAC recommends using a standard pressure of 100.0kPa (1bar). Strictly speaking temperature is not part of the definition of a standard state, but most tables of thermodynamic quantities (all those used in Chapters 5 and 6) are compiled at 298.15K (25.00°C) as a reference.

Notations

The main book notations have been collected in Table 3.

Abbreviations

The main abbreviations of the book are listed in Table 4.

Table 3 Main notations in the book

No.	Name	Symbol	Unit	Chapter/ Appendix	Comment
1	Concentration of the substance A, initial	$[A], [A]_0$	mol/L	2 ff	Often simplified to A
2	Conserved concentration	W	mol/L	3	Other symbols may be used
3	Vector	$\vec{v} = \vec{OP}$		1, A	Coordinate free
4	Coordinate vector	$v = [x, y, z]$	m	1, A	Cartesian coordinates
5	Stationary wave function	$\psi(v)$		1	
6	Radius	r	m	1	
7	Azimuth	φ	rad	1	
8	Colatitude	θ	rad	1	
9	Elevation	δ	rad	1	
10	Radial wave function	$R(r)$		1	
11	Angular wave function	$Y(\varphi, \theta)$		1	$Y(\varphi, \theta) = \Theta(\theta)\Phi(\varphi)$
12	Quantum numbers	n, l, m		1	
13	Atomic number	Z		1	
14	Imaginary unit	$j = \sqrt{-1}$		1, B	Engineering notation
15	Reaction rate	$r_k, k = 1, 2$	$(\text{mol/L})/\text{s}$	3, 4	
16	Overall rate of a reaction	R	$(\text{mol/L})/\text{s}$		
17	Rate constant	$k_k, k = 1, 2$	$\frac{(\text{mol/L})^{-m+1}}{\text{s}}$	3, 4	m reaction order
18	Time constant (relaxation time)	$\tau_k = 1/k_k$	s	3, 4, B	$m = 1$
19	Current time	t, τ	s	3, 4, B	
20	State vector at time t	$x(t)$		3, 4, B	dim $x = n$ (order)
21	Equilibrium state	$\overline{x}$		3, 4, B	
22	Perturbation from equilibrium	$\delta x = x - \overline{x}$		3, 4, B	δA refers to concentration
23	Steady state, concentration	$x_\infty, [A]_\infty$		3, 4, B	
24	State matrix eigenvalue	$\lambda_k, k = 1, 2$		4, B	It may be a complex number
25	State matrix of LTI system	A		4, B	
26	Entropy and variation	$S, \Delta S$	$\frac{\text{J}}{(\text{mol K})}$	5	
27	Enthalpy and variation	$H, \Delta H$	J/mol	5	
28	Absolute temperature	T	K	5	

(*Continued*)

Table 3 (Continued)

No.	Name	Symbol	Unit	Chapter/ Appendix	Comment
29	Free (Gibbs) energy	$G, \Delta G$	J/mol	5	$\Delta G = \Delta H - T\Delta S$
30	Relative pressure	P	Pa/Pa	5	
31	Transferred heat	q	J/mol	5	
32	Variable at standard state	X^0	Unit	5	Pressure at 100 kPa
33	Variable at standard state and temperature T	X_T^0	Unit	5	
34	Variable at temperature T	X_T	Unit	5	$S_T = S_T^0 - R\log(P)$
35	Reaction quotient	Q		5	
36	Equilibrium constant	K_P, K_{eq}		5, 6	
37	Specific molar heat	$C_P^0(T)$	$\frac{\text{J}}{(\text{mol K})}$	5	Standard pressure and temperature T
38	Reaction coordinate	x		5	
39	Number of moles	n		6	
40	Compressibility factor	Z	J/J	6	Not to be confused with atomic number
41	Kinetic energy	E_c	J	6	
42	Particle absolute velocity	v	m/s	6	
43	Mass	m	kg	6	Not to be confused with quantum number, order of a reaction
44	Mean molecular mass	M	kg/mol	6	
45	Gravity acceleration at sea level	g	m/s^2	6	9.807
46	Altitude from sea level	h	m	6	
47	Temperature gradient	$L = 0.00649$	K/m	6	Altitude on sea level
48	Concentration of hydrogen ion	pH		7	$\text{pH} = -\log_{10}[\text{H}^+]$
49	Ionic product of water	K_w		7	
50	Concentration of hydroxide ion	pOH		7	
51	Ionization (dissociation) constant	K_a		7	
52	Concentration product	K_{sp}		7	
53	Molar conductivity	Λ_m	$\frac{\text{S cm}^2}{\text{mol}}$	7	$= 0.001\text{S/cm/(mol/L)}$
54	van't Hoff coefficient	i_{vH}		8	
55	Density of a substance	ρ	g/m^3	8	

56	Volume of gaseous substance	V	$\mathrm{L} = \mathrm{dm}^3$	8	
57	Limiting conductivity	Λ^0	S/cm	7	At infinite dilution
58	Limiting molar conductivity (ions)	λ_i^0	$\frac{\mathrm{S\ cm}^2}{\mathrm{mol}}$	7	
59	Measurement of y	$\check{y}(k)$	Unit	8	k-th sample
60	Predictor variable of y	$f_\mu(k)$		8	$\mu = 1, \ldots, m$
61	Regression coefficients	$a_\mu, \quad \hat{a_\mu}$		8	Estimate
62	t-statistics (Student's test)	t_μ	Fraction	8, C	Coefficient. a_μ
63	Estimated standard deviation	$\hat{\sigma}_{mu}$		8	Same
64	Root mean squared residual (RMS)	$\hat{\sigma}_{res}$	Unit	8	
65	Explained variance of y	$\hat{s}^2$	Unit^2	8	
66	Data variance	s_y^2	Unit^2	8, C	
67	Residual sum of squares	$\hat{s}_{res}^2$	Unit^2	8	
68	Oversaturation	Ω		9	
69	pH free scale	$\mathrm{pH_F}$		9	Chemical scale of pH
70	pH total scale	$\mathrm{pH_T}$		9	Used in seawater chemistry
71	pH seawater scale	$\mathrm{pH_{SW}}$		9	Same
72	Salinity	S	g/kg	9	
73	Molar volume change	ΔV_i	$\mathrm{cm}^3/\mathrm{mol}$	9	
74	Compressibility change	ΔZ_i	MPa^{-1}	9	
75	F-statistics	F_μ	Fraction	8	Coefficient a_μ
76	Standard electrode potentials	E^0	V	10	Referred to SHE zero potential
77	Electrode potential	E_H	V	10	Solution of the Nernst equation
78	Cell potential difference	E_{cell}	V	10	

Table 4 Main abbreviations

No	Abbreviation	Meaning	Comment
1	2D, 3D	Two, three dimensional	Chapter 1
2	DF	Degree(s) of freedom	Chapter 8
3	DIC	Dissolved inorganic carbon	Chapter 9
4	FGL	Flue gas desulphurization	Chapter 5
5	EMF	Electromotive force	Chapter 10
6	log	Natural logarithm	Adopted by Matlab, WARNING: in most chemistry books the symbol *ln* is adopted
7	$\log_{10}$	Base 10 logarithm	Adopted by Matlab (log10)
8	LTI	Linear time invariant	Refers to state equations, Appendix B
9	LV	Lotka−Volterra	Chapter 4
10	N_A	Avogadro number	
11	nm	Nanometer	1 nm = 10 Ångstrøm, see also Table 1
12	FALSE	Boolean variable value = 0	
13	ODE	Ordinary differential equation	Chapters 3, 4, and Appendix D
14	PDF	Probability density function	Chapters 8 and Appendix C
15	nox	Oxidation number or state	Chapter 2
16	RMS	Root mean square	Chapters 8 and Appendix C
17	RSS	Residual sum of squares	Same
18	ppmv	Part per million of volume	See also Table 1
19	SHE	Standard hydrogen electron	Chapter 10
20	SI	International system of measurement units	From French, Système International, see also Table 1
21	TRUE	Boolean variable value = 1	
22	[Unit]	Denotes a measurement unit	Square brackets are used when isolated or after a literal formula, bracket-free after a number

References

[1] Scilab, *Software, Xcos*, https://www.scilab.org/software/xcos.
[2] https://www.elsevier.com/books-and-journals/book-companion/9780323913416.
[3] D. Mazza, Algorithms in Ocean Chemistry, Youcanprint, Lecce (Italy), 2019.
[4] F. Marino and D. Mazza, *La strega perfetta. Fatti e misfatti della CO2*, Tab Edizioni, Roma (in Italian).
[5] E. Canuto, C. Novara, L. Massotti, D. Carlucci, C. Perez Montenegro, Spacecraft Dynamics and Control: The Embedded Model Control Approach, Butterworth and Heinemann, Oxford, UK, 2018.
[6] Wikipedia, *Standard conditions for temperature and pressure*, https://en.wikipedia.org/wiki/Standard_conditions_for_temperature_and_pressure.
[7] Wikipedia, *Standard state*, https://en.wikipedia.org/wiki/Standard_state.
[8] Bureau international des poids et mésures, Non-SI units that are accepted for use with the SI, Le Système international d'unités (SI)/ The International System of Units (SI), 9th ed., Sèvres, 2019, pp. 145−146.

CHAPTER

Plotting atomic orbitals with Matlab 1

1.1 Wave functions and their factorization

1.1.1 Generalities

Atomic orbitals are mathematical functions that describe the wave nature of electrons (or electron pairs) in an atom.

They offer a way for computing the probability of finding an electron (interpreted as a charged particle) in a specific spatial region around the atom's nucleus. There are four different kinds of orbitals appearing in the chemical elements of the Periodic Table, denoted by the letters *s*, *p*, *d*, and *f*, each one with a different shape. Of the four types, *s* and *p* orbitals are the most common in organic and biological chemistry. An *s*-orbital is spherical with the nucleus at the center, a *p*-orbital is dumbbell-shaped (see Fig. 1.3), four of the five *d*-orbitals are cloverleaf-shaped (see Fig. 1.4). The fifth *d*-orbital is shaped like an elongated dumbbell with a doughnut around its middle (see Fig. 1.2). *f*-orbitals have more complex shapes as in Fig. 1.5. Orbitals are organized into different layers or electron shells.

A useful aid to describe the three-dimensional (3D) probability density distributions of the various orbitals in the hydrogen atom is the Born interpretation of wave functions (M. Born, 1882−1970). Accordingly, the probability densities of the hydrogen atom orbitals can be represented and plotted by spatial surfaces which encompass most of the electron probability.

One should keep in mind that orbitals are formulated as spatial wave functions, mathematical solutions of the Schroedinger equation (E. Schroedinger, 1887−1961, see References [1−3] and Appendix F), which have no immediate physical significance: they are *eigenfunctions* (also eigenstates) of spatial operators describing *energy* and *angular momentum* of the atom's electron orbit. Like in planet orbits, energy, being a scalar, decides the spatial dimension of the orbit, specifically the *radius*. Angular momentum, being a vector, decides the shape, in other terms *eccentricity* and orientation, that is the orbit *normal direction*. Eigenfunctions are countable (the mathematical basis of quantum theory) and therefore associated with integer eigenvalues known as *quantum numbers*. The *principal quantum number n* is associated with energy, the *azimuth* (more precisely *angular momentum*) *quantum number l* is associated with the angular momentum magnitude, and the *magnetic quantum number m* is associated with the angular momentum direction.

The square of the stationary wave function magnitude, denoted by $|\psi|^2$, if normalized to possess a unitary volume, can be interpreted as a probability density function (PDF). The interpretation is justified by the uncertain prediction of the spatial electron position before making observations. Indeed, observations of a population of electrons show the observed electrons to scatter in a waveform. The density $|\psi|^2$ is a 3D function of each spatial point around the atom's nucleus. Each point P is formulated by a vector $\vec{v} = \vec{OP}$ applied to the nucleus center O. The coordinate vector $\boldsymbol{v}$ can be expressed in either

Fundamental Chemistry with Matlab. DOI: https://doi.org/10.1016/B978-0-323-91341-6.00005-7

Cartesian (R. Descartes, 1596−1650) or spherical coordinates (radius r, azimuth φ, colatitude, polar angle, or inclination θ, see Fig. 1.1) as follows

$$\boldsymbol{v} = [x, y, z] = r[\sin\theta\cos\varphi, \sin\theta\sin\varphi, \cos\theta]. \tag{1.1}$$

An alternative triple of spherical coordinates replaces the *polar angle* with the *elevation* $\delta = \pi/2 - \theta$. In the case of the wave function $\psi(\boldsymbol{v})$, spherical coordinates are preferable since they allow $\psi(\boldsymbol{v})$ to be factorized into the product

$$\psi(\boldsymbol{v}) = \psi(r, \varphi, \theta) = R(r)Y(\varphi, \theta) \tag{1.2}$$

of the radial function $R(r)$, which is real, and of the angular function $Y(\theta, \varphi)$, which may be written as a complex number. The normalization of $|\psi|^2$, which provides a PDF, is usually done separately on the radial and angular functions, upon definition of the 3D volume element in spherical coordinates, namely $dV = r^2 dr \sin\theta d\theta d\varphi$:

$$\begin{gathered}\int_0^\infty |R|^2(r)dr \int_0^\pi \int_0^{2\pi} |Y|^2(\varphi, \theta)d\varphi\sin\theta d\theta = 1 \Leftrightarrow \\ \Leftrightarrow \int_0^\infty |R|^2(r)dr = 1, \int_0^\pi \int_0^{2\pi} |Y|^2(\varphi, \theta)d\varphi\sin\theta d\theta = 1\end{gathered}. \tag{1.3}$$

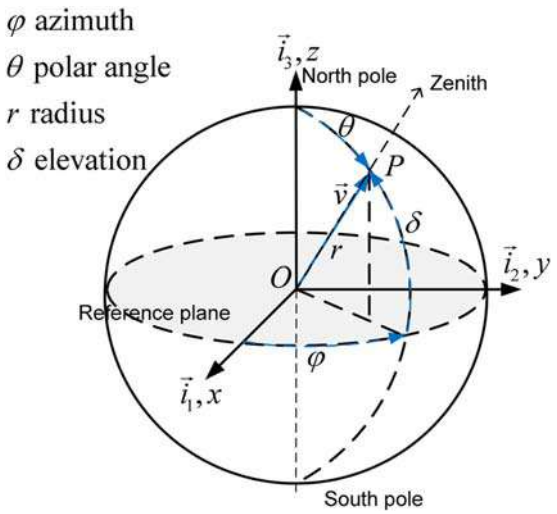


FIGURE 1.1

Cartesian and spherical coordinates on the sphere of radius r.

The infinitesimal quantity

$$dP(\vec{v}) = |\psi|^2 r^2 dr d\varphi \sin\theta d\theta, \tag{1.4}$$

known as *point probability*, is the probability of finding an electron in the volume element dV centered on the tip of the vector $\vec{v}$. The *radial probability*

$$dP(r) = 4\pi r^2 |\psi|^2 dr \tag{1.5}$$

is the probability of finding an electron in an infinitesimal spherical shell of radius r.

1.1.2 Angular wave functions

The angular function also can be written as the product $Y(\theta,\varphi)=\Theta(\theta)\Phi(\varphi)$ of two angular functions each depending on a single coordinate. The function $\Phi(\varphi)$ is the solution of the classical oscillator equation

$$\frac{d^2\Phi(\varphi)}{d\varphi^2}=-m^2\Phi(\varphi),\Phi(0)=\Phi_0, \tag{1.6}$$

where m^2 is the square of the magnetic quantum number. The solutions of Eq. (1.6) are often written using the Euler complex exponential (L. Euler, 1707−83) as follows:

$$\Phi_{\pm}(\varphi)=A\exp(\pm jm\varphi)=A(\cos(m\varphi)\pm j\sin(m\varphi)),A=1/\sqrt{(2\pi)}, \tag{1.7}$$

where $j=\sqrt{(-1)}$ is the imaginary unit (we are employing the electrical engineering notation; the Matlab® notation is `1i`), and the gain A makes unitary the norm of the function. In fact, by recalling that the square magnitude of a complex number equals the product with the conjugate, we can write

$$\left|\Phi_{\pm}(\varphi)\right|^2=A^2\int_{-\pi}^{\pi}exp(jm\varphi)exp(-jm\varphi)d\varphi=A^2 2\pi=1. \tag{1.8}$$

Since each solution in Eq. (1.7) is the conjugate of the other, also the real and the imaginary parts, namely the trigonometric functions $A\cos(m\varphi)$ and $A\sin(m\varphi)$, are solutions of Eq. (1.6). Here, we are interested in them, since they, being real, are suitable to represent orbital shapes.

The derivation of $\Theta(\theta)$, which is associated to the pair $\{l,m\}$ of quantum numbers, is a bit more intricate: the reader is addressed to specialized textbooks (see References [1−3]) and to Appendix F. The overall wave function $Y(\theta,\varphi)$ is referred to as a *spherical harmonic* and indicated, when complex, by $Y_l^m(\theta,\varphi)$. It is the generalization of the harmonic functions defined on the unit circle to the functions $Y_l^m(\theta,\varphi)$ defined on the unit sphere. Harmonic functions depend on a single integer m like $A\cos(m\varphi)$ and $A\sin(m\varphi)$, whereas spherical harmonics depend on the pair $\{l,m\}$. The unit sphere is a surface defined by the coordinates $\{1,\varphi,\theta\}$. The angular functions can be converted into real functions, denoted by $Y_{l,m}(\theta,\varphi)$, by separating real and imaginary parts of $Y_l^m(\theta,\varphi)$.

Given the magnetic quantum number m, for instance $m=\pm1$, a pair of real spherical harmonics $Y_{l,|m|}(\varphi,\theta)=\Theta(\theta)\cos(m\varphi)$ and $Y_{l,-|m|}(\varphi,\theta)=\Theta(\theta)\sin(m\varphi)$, associated with the quantum number m, exist. Their surface representations are each other rotated in the plane $\theta=\pi/2$ by the angle $\pi/(2m)$. Each surface possesses $|m|$ pairs of lobes, one positive and one negative, which correspond to the number of periods of $\cos(m\varphi)$ and $\sin(m\varphi)$ in the range $[0,2\pi]$.

The factor of the angular wave function for an s-orbital $\{n,l=0\}$ is always the same, regardless of the principal quantum number n. As a counterpart, the angular parts of p- and d-orbitals are independent of n, but depend on the angular momentum quantum number l and the magnetic quantum number m. The quantum numbers satisfy the following relationships:

$$n=0,1,2,\ldots,l=0,1,\ldots,n-1,m=0,\pm1,\ldots,\pm l. \tag{1.9}$$

The principal quantum number n subdivides the region around the nucleus into spherical shells of increasing radius. Each shell is subdivided into n spherical subshells. They correspond to the azimuth quantum numbers $l=0,1,\ldots,n-1$ and are denoted with the letters $s,p,d,f,\ldots$. Each subshell l contains $2l+1$ orbitals, which correspond to the magnetic quantum numbers $m=0,\pm1,\ldots,\pm l$.

Table 1.1 Real angular wave functions $Y_{l,m}(\varphi,\theta)$ of hydrogen-like atoms.

	Quantum numbers			Real angular wave functions to be normalized $Y_{l,m}(\varphi,\theta)/N(l,m)$ Eq. (F.27)		
No	l	m	**Orbital name**	**Polar coordinates**	**Cartesian coordinates**	**Normalization $N(l,m)$ Eq. (F.29)**
1	0	0	s	1	1	$\sqrt{\frac{1}{4\pi}}$
2	1	0	$p(z)$	$\cos\theta$	$\frac{z}{r}$	$\sqrt{\frac{3}{4\pi}}$
3	1	1	$p(x)$	$\sin\theta\cos\varphi$	$\frac{x}{r}$	$\sqrt{\frac{3}{4\pi}}$
4	1	−1	$p(y)$	$\sin\theta\sin\varphi$	$\frac{y}{r}$	$\sqrt{\frac{3}{4\pi}}$
5	2	0	$d(z^2)$	$\frac{3\cos^2\theta-1}{2}$	$\frac{2z^2-x^2-y^2}{2r^2}$	$\sqrt{\frac{5}{4\pi}}$
6	2	1	$d(xz)$	$3\sin\theta\cos(\theta)\cos\varphi$	$3\frac{xz}{r^2}$	$\sqrt{\frac{5}{12\pi}}$
7	2	−1	$d(yz)$	$3\sin\theta\cos(\theta)\sin\varphi$	$3\frac{yz}{r^2}$	$\sqrt{\frac{5}{12\pi}}$
8	2	2	$d(x^2-y^2)$	$3\sin^2\theta\cos2\varphi$	$3\frac{x^2-y^2}{r^2}$	$\sqrt{\frac{5}{48\pi}}$
9	2	−2	$d(xy)$	$3\sin^2\theta\sin2\varphi$	$6\frac{xy}{r^2}$	$\sqrt{\frac{5}{48\pi}}$
10	3	0	$f(z^3)$	$\frac{(5\cos^2\theta-3)\cos\theta}{2}$	$\frac{(2z^2-3(x^2+y^2))z}{2r^3}$	$\sqrt{\frac{7}{4\pi}}$
11	3	1	$f(xz^2)$	$\frac{3(5\cos^2\theta-1)\sin\theta\cos\phi}{2}$	$3\frac{(4z^2-x^2-y^2)x}{2r^3}$	$\sqrt{\frac{7}{24\pi}}$
12	3	−1	$f(yz^2)$	$\frac{3(5\cos^2\theta-1)\sin\theta\sin\phi}{2}$	$3\frac{(4z^2-x^2-y^2)y}{2r^3}$	$\sqrt{\frac{7}{24\pi}}$
13	3	2	$f(zx^2-zy^2)$	$15\sin^2\theta\cos\theta\cos2\phi$	$15\frac{z(x^2-y^2)}{r^3}$	$\sqrt{\frac{7}{240\pi}}$
14	3	−2	$f(xyz)$	$15\sin^2\theta\cos\theta\sin2\phi$	$30\frac{xyz}{r^3}$	$\sqrt{\frac{7}{240\pi}}$
15	3	3	$f(x^3-3xy^2)$	$15\sin^3\theta\cos3\phi$	$15\frac{x(x^2-3y^2)}{r^3}$	$\sqrt{\frac{7}{1440\pi}}$
16	3	−3	$f(3yx^2-y^3)$	$15\sin^3\theta\sin3\phi$	$15\frac{y(3x^2-y^2)}{r^3}$	$\sqrt{\frac{7}{1440\pi}}$

The simplest expressions of the real angular wave functions $Y_{l,m}(\varphi,\theta)$ for one-electron hydrogen-like atoms are listed in Table 1.1. In this table, the quantum numbers $l=0,1,\ldots,n-1$ are accompanied by the subshell letters $\{s,p,d,f,\ldots\}$ and the numbers $m=0,\pm1,\pm2,\ldots\pm l$ by an expression of the Cartesian coordinates $\{x,y,z\}$ which indicates the function expression in such coordinates.

Fig. 1.2 shows the surface of the orbital d_{z^2}, written also as $d(z^2)$, which is defined by the angular function $Y_{2,0}(\varphi,\theta)$. It is named (on the top of the figure) from the previous orbital conventions, where the letter d corresponds to the quantum number $l=2$, the letter z denotes the surface axis, and the exponent 2 indicates that there are two node planes where the angular function becomes zero.

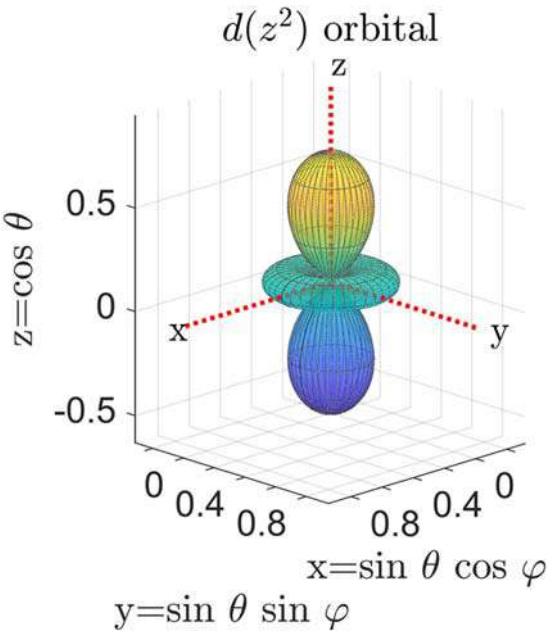


FIGURE 1.2

The orbital $d(z^2)$.

The pair $l = 1, m = \pm 1$ is replaced by the pair $\{p_x, p_y\}$ [also $p(x), p(y)$] to indicate that the main axes are x and y, in the reference plane of Fig. 1.1.

Fig. 1.3 shows the surfaces of the orbitals $\{p_x, p_y\}$ defined by the angular functions $Y_{2,\pm 1}(\varphi, \theta) = \{\Theta(\theta)\cos(\varphi), \Theta(\theta)\sin(\varphi)\}$. They are named (on the top of the figure) from the previous orbital conventions, where the letter p corresponds to the quantum number $l = 1$, and the surface orientation in the reference plane of Fig. 1.1 is indicated by the axis symbol. The plane orthogonal to the axis, either yz (see left in the figure) or xz (see right in the figure), is the *node plane* where the wave function goes to zero (null probability). For $|m| > 1$, the number of lobes increases together with the number and position of the node planes.

The pair $\{d_{xz}, d_{yz}\}$ (not in the figure, but it can be plotted with the script of Section 1.3.2 by selecting cases 6 and 7) corresponds to $l = 2, m = \pm 1$ and indicates that the axes are the bisectors of the planes xz and yz, respectively. A similar indication is employed by the orbital d_{xy} in Fig. 1.4, left, corresponding to one of the two orbitals $l = 2, m = \pm 2$, which implies that the two axes are the bisectors of the plane xy. The second orbital for $l = 2, m = \pm 2$ is denoted by $d_{x^2-y^2}$ (Fig. 1.4, right) and indicates that the two axes of symmetry are x and y.

Two of the complex orbitals f, namely $f\left(xz^2\right)$ and $f\left(x^3 - 3xy^2\right)$, which possess three axes of symmetry, are shown in Fig. 1.5.

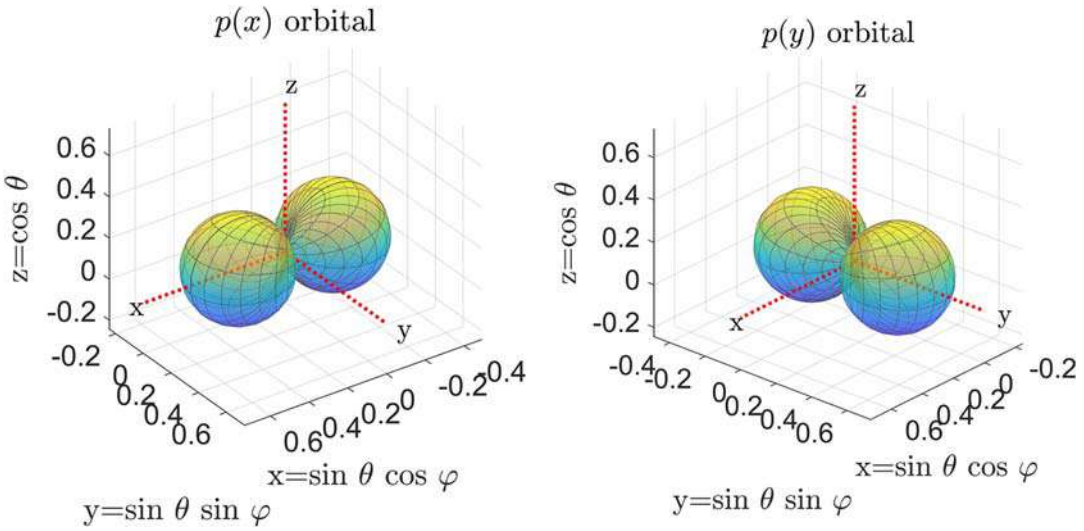


FIGURE 1.3

The orbitals p_x (left) and p_y (right).

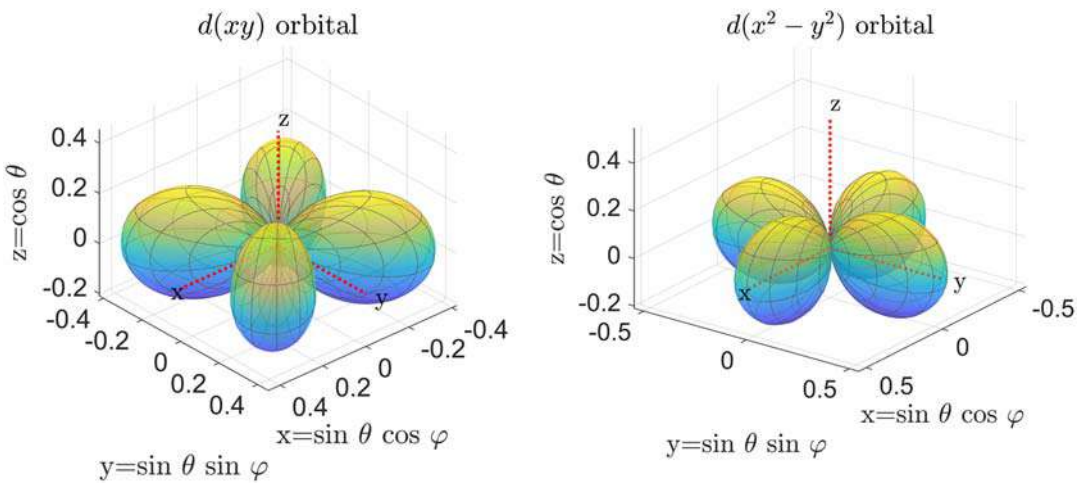


FIGURE 1.4

(Left) The orbital $d(xy)$ and (right) the orbital $d(x^2 - y^2)$.

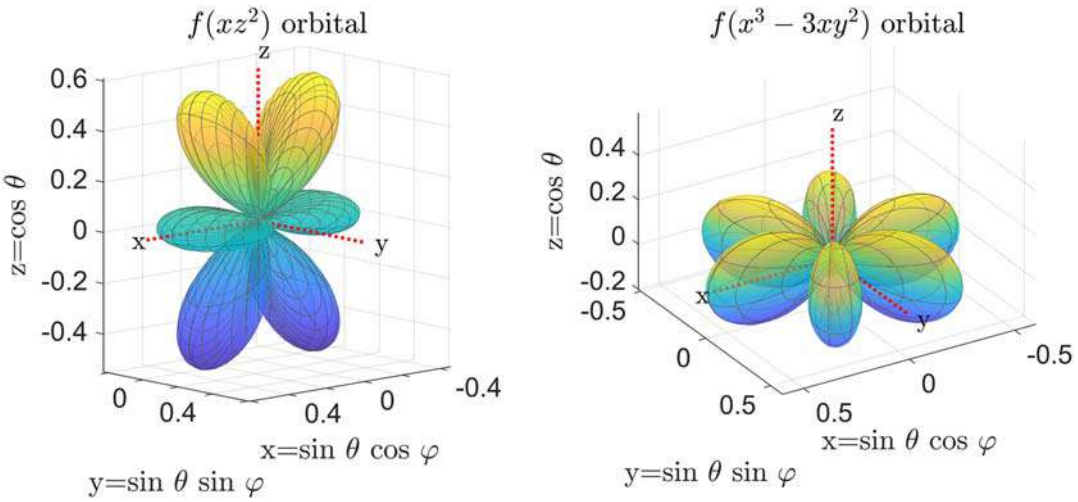


FIGURE 1.5

(Left) Orbital $f(xz^2)$ and (right) Orbital $f(x^3 - 3xy^2)$.

Orbitals of multielectron atoms possess similar forms to hydrogen orbitals. They are in some way reduced in size due to the greater attraction of the positively charged nucleus, but their shapes remain similar. Atomic orbitals in multielectron atoms are filled up with pairs of electrons (one with spin-up and the companion with spin-down) in a process known with the German word *Aufbau*, meaning over-construction. The electron spin is the analog of the angular momentum of a rigid body around a rotation axis passing through the center of mass, to be distinguished from the orbital angular momentum due to the translational motion of the center of mass around the local orbit center [4]. The electron orbital angular momentum corresponds to the orbit around the nucleus center, whose quantum number is denoted by l (the azimuth quantum number). The electron spin assumes only two eigenstates, spin-up and spin-down, corresponding to the quantum numbers $\pm 1/2$ [3].

Table 1.1 lists quantum numbers, orbital names, and angular and radial wave functions of the low degree orbitals, $l \leq 3$, of a hydrogen-like atom. They are exact solutions of the Schroedinger wave equation (see References [1−3] and Appendix F). The ordinal number in the first column corresponds to the ordinal to be selected in the script of Section 1.3.2, devoted to the graphical plot of angular wave functions. Let us recall that angular wave functions do not depend on the principal quantum number n, which, therefore, does not appear in Table 1.1 and in the orbital names. In other words, $Y_1^0(\varphi, \theta)$, for instance, applies to orbitals $1p(z), 2p(z), \ldots, np(z), \ldots$.

Table 1.1 reports only the real angular wave functions, indicated by $Y_{l,m}$, which, in the case of $|m| > 0$, are obtained from the complex functions $Y_l^{\pm |m|}$ (see References [3] and [5]) through the identities

$$Y_{l,|m|} = \frac{1}{\sqrt{2}}\left(Y_l^{-|m|} - Y_l^{|m|}\right), Y_{l,-|m|} = \frac{j}{\sqrt{2}}\left(Y_l^{-|m|} + Y_l^{|m|}\right). \tag{1.10}$$

The previous identities imply that the complex scale factor must be multiplied by $\sqrt{2}$. Let us remark that usually, as in References [3] and [5], the scale factor includes the normalization factor

$N(l,m)$ appearing in Eq. (F.29) of Appendix F and a common factor of the associated Legendre functions to be normalized (see Eq. (F.27) of Appendix F). In Table 1.1 the functions to be normalized (in spherical and Cartesian coordinates) and the normalization factor $N(l,m)$ are reported in separate columns.

1.1.3 Table of the radial functions

The expression of the radial functions for the hydrogen atoms is derived in Appendix F, Section F.4. The graphical algorithm of the *radial probability density* in Eq. (1.19) and the resulting profiles can be found in Section 1.4. Here, we anticipate the table of the expressions for $n \leq 4$.

Table 1.2 Radial functions.

	Quantum numbers			
No	n	l	**Orbital name**	**Expression**
1	1	0	$1s$	$R(1s) = 2\left(\frac{Z}{a_0}\right)^{3/2}\exp(-\sigma)$
2	2	0	$2s$	$R(2s) = \frac{1-\sigma}{\sqrt{2}}\left(\frac{Z}{a_0}\right)^{3/2}\exp(-\sigma)$
3	2	1	$2p$	$R(2p) = \frac{\sigma}{\sqrt{6}}\left(\frac{Z}{a_0}\right)^{3/2}\exp(-\sigma)$
4	3	0	$3s$	$R(3s) = \frac{2-4\sigma+\frac{4}{3}\sigma^2}{3\sqrt{3}}\left(\frac{Z}{a_0}\right)^{3/2}\exp(-\sigma)$
5	3	1	$3p$	$R(3p) = \frac{4(2-\sigma)\sigma}{9\sqrt{6}}\left(\frac{Z}{a_0}\right)^{3/2}\exp(-\sigma)$
6	3	2	$3d$	$R(3d) = \frac{4\sigma^2}{9\sqrt{30}}\left(\frac{Z}{a_0}\right)^{3/2}\exp(-\sigma)$
7	4	0	$4s$	$R(4s) = \frac{3-9\sigma+6\sigma^2-\sigma^3}{12}\left(\frac{Z}{a_0}\right)^{3/2}\exp(-\sigma)$
8	4	1	$4p$	$R(4p) = \sigma\frac{5-5\sigma+\sigma^2}{4\sqrt{15}}\left(\frac{Z}{a_0}\right)^{3/2}\exp(-\sigma)$
9	4	2	$4d$	$R(4d) = \sigma^2\frac{3-\sigma}{12\sqrt{5}}\left(\frac{Z}{a_0}\right)^{3/2}\exp(-\sigma)$
10	4	3	$4f$	$R(4f) = \frac{\sigma^3}{12\sqrt{35}}\left(\frac{Z}{a_0}\right)^{3/2}\exp(-\sigma)$

The expression of the radial functions in Table 1.2 includes the generic atomic number denoted by Z, since these functions apply to any single-electron atom, that is, to hydrogen atoms or hydrogen-like ions with $Z > 1$. The parameter σ appearing throughout Table 1.2 is given by:

$$\sigma(r,n) = \frac{Zr}{na_0}, \tag{1.11}$$

where r is the distance of the electron from the nucleus and a_0 is known as the Bohr radius (N. Bohr, 1885−1962, see Appendix F, Section F.4). The Bohr radius is related to other constants of

the Schroedinger equation, namely the Planck's constant h (M. Planck, 1858−1947), the electron mass m_e, the electron charge e, and the vacuum permittivity ε_0 as follows:

$$a_0 = \frac{\varepsilon_0 \quad h^2}{\pi m_e e^2} \quad \text{[m]}. \tag{1.12}$$

To obtain the complete wave function of a particular orbital, the radial part must be multiplied by the angular part. For instance, with the help of the expressions in Table 1.1, row 1, in Table 1.2, row 1 and $Z = 1$, the wave function of the 1*s*-orbital of the hydrogen atom becomes

$$\psi(r,\theta,\varphi) = R(r,1s)Y(\theta,\varphi,1s) = 2\left(\frac{1}{a_0}\right)^{3/2}\exp\left(\frac{-r}{a_0}\right)\frac{1}{\sqrt{4\pi}}\int^{\pi} \tag{1.13}$$

The *point probability density* [free of the infinitesimal volume in Eq. (1.4)] holds

$$f_p(r,\varphi,\theta) = |\psi|^2 = |R(r,1s)|^2|Y(\theta,\varphi,1s)|^2 = \frac{1}{\pi}\left(\frac{1}{a_0}\right)^3\exp\left(\frac{-2}{a_0}r\right), \tag{1.14}$$

whereas the *radial probability density* [free of the infinitesimal shell in Eq. (1.5)] holds

$$f_r(r) = |\psi|^2 4\pi r^2 = R(r,1s)^2 r^2 = 4r^2\left(\frac{1}{a_0}\right)^3\exp\left(\frac{-2}{a_0}r\right) \tag{1.15}$$

The reader is asked to prove that the volume of both probability densities is equal to one, as expected.

1.2 Graphical plot of the wave functions

Since wave functions are defined over a 3D space, their graph requires a four-dimensional space. We adopt the following way.

1. *Radial and angular functions are plotted separately*. The radial function $R(r)$ requires a 2D graph, the abscissa being equal to radius r, whereas the angular function $Y(\varphi,\theta)$, scaled by $R(r_0) = R_0$ may pose some problems (see the point 2 below), since it may be written as a complex function of the azimuth φ as shown in Eq. (1.7). The 2D plot of the radial function is presented in Section 1.4.
2. *Real and imaginary parts of the angular function are plotted separately*, as already mentioned when discussing expression (1.7).
3. *The angular function is plotted versus Cartesian axes, instead of representing it on a spherical surface*. In principle, $R(r_0)Y(\varphi,\theta)$ is a spherical wave to be represented like the height (positive and negative) of a stationary ocean or land surface on the Earth's sphere of radius r_0. Ocean and land surfaces or the Earth's magnetic field intensity with respect to a spherical reference (actually with respect to the Earth's ellipsoid) are usually represented by 2D contour lines. This way may be an alternative, but the 3D shape of $R_0Y(\varphi,\theta)$ would be lost. The common way is to plot the surface $R_0Y(\varphi,\theta)$ parameterized by azimuth and polar angles in the Cartesian coordinates defined by the triple $\{\varphi,\theta,|R_0Y(\varphi,\theta)|\}$. $R(r_0)$ is selected so that the volume of $|\psi(r\le r_0)|^2$ covers the 90% of the probability density $|\psi(r,\varphi,\theta)|^2$. The Cartesian coordinates of each point $\vec{v}$ of the surface holds

$$\boldsymbol{v} = [x,y,z] = R_0|Y(\varphi,\theta)|[\sin\theta\cos\varphi, \sin\theta\sin\varphi, \cos\theta]. \tag{1.16}$$

The tangent planes where the surface vector becomes zero, namely $\vec{v}=0$, are the *node planes* of the angular wave. When the planes pass through the origin, they correspond to the great circles which are intersections of the planes with the sphere. For instance, the node plane in Fig. 1.3, left, is the plane yz and corresponds to the polar great circle of azimuth $\varphi=\pi/2$. By varying the scale factor R_0, the surface of the angular function describes a *3D contour* of the four-dimensional graph of the wave function ψ at different distances from the center of the atom's nucleus. As already said, the contour $R(r)=R_0$ is chosen such that

$$\int_0^{r_0}|R|^2(r)r^2dr\int_0^{\pi}\int_0^{2\pi}|Y|^2(\varphi,\theta)d\varphi\sin\theta d\theta=0.9. \tag{1.17}$$

Fig. 1.6 shows an example of the angular functions listed in Table 1.1, namely p_z with $R_0=1$. The red circle (out of scale) denotes the sphere equator orthogonal to the pole z. Azimuth and polar angles are indicated together with the vectorial magnitude of the surface $R_0|Y|$. Since, from Table 1.1, we find $Y(p_z)=\sqrt{0.75/\pi}\cos\theta$, the upper lobe corresponds to $\cos\theta\geq 0, |\theta|\leq\pi/2$ (positive spherical wave), whereas the lower lobe corresponds to $\cos\theta\leq 0, \pi/2\leq|\theta|<\pi$ (negative spherical wave). In the literature, *negative lobes* are usually plotted with different colors. If we consider the square magnitude

$$\left|Y(p_z)\right|^2=\frac{0.75}{\pi}\cos^2\theta=\frac{0.75}{\pi}\frac{1+\cos2\theta}{2}\geq 0, \tag{1.18}$$

we obtain a nonnegative wave (interpreted as a probability density) over the entire sphere.

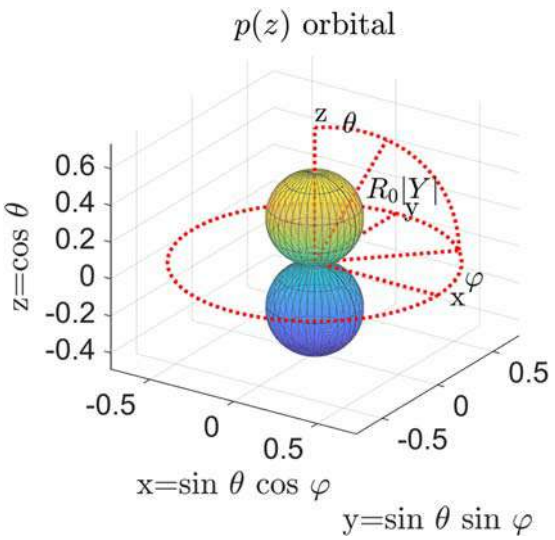


FIGURE 1.6

An orbital example, p_z, plotted together with Cartesian and spherical coordinates.

1.3 Graphical algorithm

1.3.1 Description

The graphical plot of the angular functions is obtained by fixing the scale factor to the unit, $R_0 = 1$. The Matlab plotting function `fsurf` is capable of plotting a parameterized surface $Y(\varphi, \theta)$ in the 3D space defined by three Cartesian axes. The function requires the Cartesian vector $\boldsymbol{v}$ defined in Eq. (1.16) and the range of the parameters, that is, $0 \leq \varphi < 2\pi, 0 \leq \theta < \pi$. Spherical to Cartesian transformation in Matlab employs the elevation angle $\delta = \pi/2 - \theta$ instead of the polar angle θ, and the order of the input spherical coordinates is $\{\varphi, \delta, r = Y\}$, as in the following statement:

```
[x,y,z] = sph2cart(phi,pi/2 - theta,R);
```

The function call is the following:

```
fsurf(x,y,z,[0 pi 0 2*pi],':','Meshdensity',10)
```

where the square brackets contain the parameter range. The algorithm works in the symbolic Matlab environment, which requires to define azimuth and polar angle as symbols as follows:

```
syms theta phi
```

The reader must select the number of the orbital to be plotted according to the ordinal number of the first column of Table 1.1. The function expression is accompanied by the azimuth and magnetic quantum numbers, necessary for computing the scale factors of the angular waves which are reported in Table 1.1.

```
% scale factor
sFactor=sqrt((2*l+1)*factorial(l-m)/(4*pi*factorial(l+m)));
Y=abs(Y)*sFactor;
```

The second part of the plot statements adds Cartesian axes and labels to the graph as in Fig. 1.6. The resulting figure can be rotated using the rotation key in the Matlab pane, so as to align the axes as desired. The initial script `InitChem` can be found in Appendix D.

1.3.2 Matlab script

`InitChem` invokes the Latex interpreter of the text appearing in the titles and legends of the graphical plots. Mathematical expressions and words to be converted by the Latex interpreter must be included between a pair of $.

```
%% Angular wave functions
%% Initialization
InitChem;
syms theta phi
%% Choosing orbital type
orbital =16; % <<< user
switch orbital
case 1; Y = 1;
   T1 ='$s$ orbital';l=0; m=0;
case 2; Y = cos(theta);
   T1 ='$p(z)$ orbital';l=1;m=0;
case 3; Y = sin(theta)*cos(phi);
   T1 ='$p(x)$ orbital';l=1;m=1;
case 4; Y = sin(theta)*sin(phi);
   T1 ='$p(y)$ orbital';l=1;m=1;
case 5; Y = (3*cos(theta)^2 - 1)/2;
   T1 ='$d(z^2)$ orbital';l=2; m=0;
case 6; Y = 3*sin(theta)*cos(theta)*cos(phi);
   T1 ='$d(xz)$ orbital';l=2; m=1;
case 7; Y = 3*sin(theta)*cos(theta)*sin(phi);
   T1='$d(yz)$ orbital';l=2; m=1;
case 8; Y = 3*sin(theta)^2*cos(2*phi);
   T1 ='$d(x^2-y^2)$ orbital';l=2;m=2;
case 9; Y = 3*sin(theta)^2*sin(2*phi);
   T1 ='$d(xy)$ orbital';l=2;m=2;
case 10; Y = (5*cos(theta)^2 - 3)*cos(theta)/2;
   T1 ='$f(z^3)$ orbital';l=3;m=0;
case 11; Y = 3*(5*cos(theta)^2 - 1)*sin(theta)*cos(phi)/2;
   T1 ='$f(xz^2)$ orbital';l=3;m=1;
case 12; Y = 3+(5*cos(theta)^2 - 1)*sin(theta)*sin(phi)/2;
   T1 ='$f(yz^2)$ orbital'; l=3;m=1;
case 13; Y =15* sin(theta)^2*cos(theta)*sin(2*phi);
   T1 ='$f(xyz)$ orbital'; l=3;m=2;
case 14; Y =15* sin(theta)^2*cos(theta)*cos(2*phi);
   T1 ='$f(zx^2 - zy^2)$ orbital' ; l=3;m=2;
case 15; Y = 15*sin(theta)^3*cos(3*phi);
   T1='$f(x^3 - 3xy^2)$ orbital'; l=3;m=3;
case 16; Y = 15*sin(theta)^3*sin(3*phi);
   T1='$f(3yx^2 - y^3)$ orbital';l=3;m=3;
end
% scale factor
sFactor=sqrt((2*l+1)*factorial(l-m)/(4*pi*factorial(l+m)));
if m>0, sFactor=sFactor*sqrt(2); end
Y=abs(Y)*sFactor;
```

```
% Transformation of spherical to Cartesian coordinates
% elevation angle = 90°-theta, azimuth angle = phi.
[x,y,z] = sph2cart(phi,pi/2 - theta,Y);
%% Surface plot
h=fsurf(x,y,z,[0 pi 0 2*pi],':','Meshdensity',20); hold on;
h.FaceAlpha = 0.6; % transparency degree
% WARNING: limits refer to function Y symbolic coordinates
xlabel('x=sin $\theta$ cos $\varphi$');
ylabel('y=sin $\theta$ sin $\varphi$');
zlabel('z=cos $\theta$');
axis equal
view(120,20);title(T1); % <<< to be arranged by user
% axes drawing
maxLine=sFactor*1.5; % <<< user, to be fitted to plot
aLine=[0 maxLine]; hold on;
line(aLine,[0 0],[0 0],'Color','r','LineStyle',':','LineWidth',2);
line([0 0],aLine,[0 0],'Color','r','LineStyle',':','LineWidth',2);
line([0 0],[0 0],aLine,'Color','r','LineStyle',':','LineWidth',2);
text(maxLine*1.1,0,0,'x','FontSize',16);
text(0,maxLine*1.1,0,'y','FontSize',16);
text(0,0,maxLine*1.1,'z','FontSize',16);
hold off;
%% Case 2 axes and notations
if orbital==2
   c=cos(pi/4);
   c1=cos(pi/6);
   s1=sin(pi/6);
   Nline=50;
   % axes
   hold on;
   line(c*aLine,c*[0 maxLine],[0 0],'Color','r','LineStyle',':',...
        'LineWidth',2);
   line(s1*c*aLine,s1*c*aLine,c1*aLine,'Color','r','LineStyle',':',...
        'LineWidth',2);
   % circle
   th=linspace(0,2*pi,Nline);
   plot3(maxLine*cos(th),maxLine*sin(th),zeros(Nline,1),'r',...
        'LineStyle',':','LineWidth',2);
```

```
  thm=linspace(0,pi/2,Nline);
  plot3(c*maxLine*cos(thm),c*maxLine*cos(thm),maxLine*sin(thm),'r',...
       'LineStyle',':','LineWidth',2);
  text(maxLine,maxLine/3,0,'${\varphi}$','interpreter','latex',...
       'FontSize',16)
  text(0.1,0.1,maxLine,'${\theta}$','interpreter','latex',...
       'FontSize',16);
  text(maxLine/4, maxLine/4,maxLine/2,'${R_0|Y|}$','interpreter',...
       'latex','FontSize',16)
  view(30,20);hold off;
end
hold off;
```

1.4 Graphical plot of the radial probability density for *s*-type orbitals

1.4.1 Description

The plot of the radial part of the wave function $R(r)$ (see Appendix F), as reported in Table 1.2, requires a 2D graph, the abscissa being equal to radius r. The radial function decays exponentially to zero as r increases. This allows each orbital to be assigned a size as shown below. In general, the larger the value of n, the larger the orbital size.

The radial function crosses the horizontal axis $n-l-1$ times, hence the *s*-type orbital with $l=0$ crosses the axis $n-1$ times before decaying to zero. The radial values where the radial function crosses the horizontal axis are known as the *radial nodes*.

The radial functions of the orbital types $p, d, f, \ldots$, with the exception of the *s*-type orbitals, have zero value at the atom's nucleus center, corresponding to $r=0$. Instead, $R(ns)$ reaches the maximum at the center defined by $r=0$

As already said in Section 1.1, the square $f_p(r,\varphi,\theta)=|\psi|^2=R^2(r)|Y(\varphi,\theta)|^2$ represents the probability density of finding an electron (interpreted as a particle) in a point of coordinate $\{r,\varphi,\theta\}$ (*point probability density*). This is different from the probability density $f_r(r,\varphi,\theta)=4\pi r^2|\psi|^2$ of finding an electron on the surface of a sphere of radius r (*radial probability density*). In the case of *s*-type orbitals, due to a constant angular function, the latter density only depends on the radius as in Eq. (1.15) and holds

$$f_r(r,ns)=|\psi|^2 4\pi r^2=R(r,ns)^2 r^2. \tag{1.19}$$

In the following script, the radial probability density of the ns orbitals with $n=1,2,3$ is computed and plotted versus the radial distance r. Although the radial function $R(r,ns)$ is nonzero for $r=0$, $f_r(r,ns)$ in Eq. (1.19) becomes zero at the atom's center. The density functions in Fig. 1.7 possess n maxima and $n+1$ minima equal to zero. One of the minima occurs for $r\to\infty$.

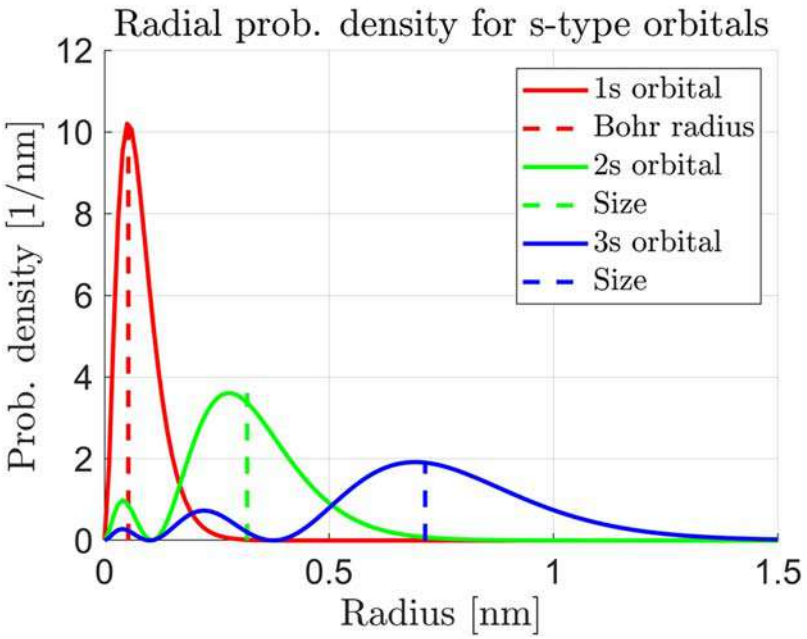


FIGURE 1.7

Radial probability density functions of the 1*s*, 2*s*, and 3*s* orbitals.

The radius $r_{max}(n)$ of the largest maximum is close to but not coincident with the orbital size $\overline{r}_{nl}$, defined as the mean radius of the probability density $f_r(r, ns)$ in Eq. (1.19). The orbital size in general and for the *s*-type orbitals ($l = 0$) holds

$$\begin{aligned} \overline{r}_{nl} &= \frac{n^2 a_0}{2Z}\left(3 - \frac{l(l+1)}{n^2}\right), \quad \text{general case} \\ r_{max}(n) \approx \overline{r}_n &= \frac{3n^2 a_0}{2Z}, \quad \text{s} - \text{type orbital} \end{aligned} \qquad (1.20)$$

Orbital sizes are indicated in Fig. 1.7 by vertical dashed lines. For $n = 1$, the size corresponds to the Bohr radius.

1.4.2 Matlab script and graphical results

The script follows.

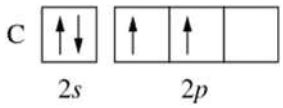
```
%% Radial probability density, s-type orbital
%% Initialization
InitChem;
%% Parameters
r = (0:0.01:1.5).'; % radius in nanometer
dimr=length(r);
Z = 1; % nuclear charge in hydrogenoid atoms
n = [1; 2; 3]; % smallest quantum numbers 'n' of s-type orbitals (1s 2s 3s)
a0 = 0.0529; % Bohr radius in nanometer
Za0=(Z/a0)^(3/2);
surface = 4*pi*r.^2;
%% Wave and probability functions
Y = sqrt(1/4/pi); % angular function
sigma=zeros(dimr,3);
fradial=zeros(dimr,3);
maxf=zeros(3,1);
psi=zeros(dimr,3);
for i=1:3
    sigma(:,i) = r/(a0*n(i));
    psi(:,i)=Y*Za0*exp(-sigma(:,i));
end
psi(:,1)=psi(:,1)*2;  %1s orbital
psi(:,2)=psi(:,2).*(1-sigma(:,2))/sqrt(2); % 2s orbital
psi(:,3)=psi(:,3)*2.*(1-2*sigma(:,3)+(2/3)*sigma(:,3).^2)/(3*sqrt(3));
for  i=1:3
    fradial(:,i)=surface.*psi(:,i).^2; %radial probability
    maxf(i)=max(fradial(:,i));
end
```

```
%% Probability density plot
x=ones(3,2)*a0;
x(2,:)=x(2,:)*3*n(2)^2/2;
x(3,:)=x(3,:)*3*n(3)^2/2;
y=[0 1 ;0 1 ; 0 1];
for i=1:3, y(i,:)=y(i,:)*maxf(i); end
figure ('name','Radial prob.'); hold on; grid on;
for i=1:3
   plot(r,fradial(:,i),color(i));
   line (x(i,:),y(i,:), 'color', color(i),'LineStyle','--');
end
xlabel('Radius [nm]');
ylabel('Prob. density [1/nm]')
title('Radial prob. density for s-type orbitals');
legend('1s orbital','Bohr radius', '2s orbital', 'Size','3s orbital ', ...
   'Size','FontSize',16);
```

1.5 Hybrid orbitals

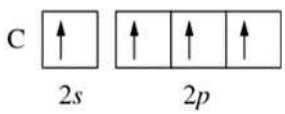
One of the most successful methods of describing the formation of chemical bonds is the simple overlap of atomic orbitals of the two atoms engaged in bond formation. In most of the cases, however, the description of the molecular geometry based on a simple overlap of unmodified atomic orbitals does not conform to the observed molecular geometry [1].

The stable, lowest energy configuration in an atom is called the ground state [6] (see Appendix F, Section F.4). On the basis of this configuration, the valence shell of the carbon, which is the ensemble of electrons capable of forming chemical bonds [7], can be represented as follows:



By employing only half-filled orbitals, we expect the existence of a molecule with the formula CH_2 and a bond angle of 90 degrees. Only half-filled atomic orbitals can overlap to form a single covalent bond, in such a way that the resulting molecular orbital is filled with two electrons.

On the contrary, the simplest hydrocarbon observed under normal laboratory conditions is CH_4, the methane. This is a stable molecule with a molecular formula consistent with the octet rule of the Lewis theory [1] (G.N. Lewis, 1875–1946). In order to obtain this molecular formula, we need four orbitals with four unpaired electrons on the carbon atom, just before the formation of four C − H bonds, so that orbital overlapping leads to four bonds. To get such a diagram, let us imagine that one of the electrons in the 2*s*-orbital of a carbon atom absorbs energy and is promoted to the empty orbital.



The resulting electron configuration is that of an excited state, namely a configuration with a higher energy with respect to the ground state. The electron configuration of this excited state suggests a molecule with three mutually perpendicular bonds based on the orbitals of the carbon atom having bond angles of 90 degrees. This description, however, does not agree with experimental bond angles, since all the four angles are found to be 109.5 degrees, pointing toward the vertices of a tetrahedron.

To comply with experimental geometry, the valence orbitals of the carbon atom are mathematically transformed into a new set of orbitals, more appropriate for bonding in a tetrahedral arrangement. This transformation involves replacing four atomic orbitals with four *new* orbitals, having 109.5 degree angles between them.

The appropriate linear combinations of the wave functions representing one 2*s* and three 2*p*-orbitals can be found in specialized textbooks or websites [8]. To obtain four new orbitals, four different algebraic combinations are required, each combination representing one of the new orbitals. This mathematical process, called *hybridization*, was first introduced by L. Pauling (1901–94) with the aim of transforming pure atomic orbitals into reformulated atomic orbitals suited to bonded atoms (*hybrid orbitals*). It has remained up to the present day a key concept in chemical bond description.

The newly formed hybrid orbitals are still atomic orbitals, but only exists at the time of bond formation, and since they are excited states (with higher energy) in the isolated atom, they remain unoccupied. Their shape becomes deformed when the bond is formed, due to the presence of the atomic orbitals of the bonded atom and the consequent electric interactions.

Hybrid orbitals experience interelectronic repulsion among themselves, being filled with electrons. Since their description in the isolated atom is rather useless, approximated, more elongated shapes are depicted in all the general chemistry textbooks [1].

As an instance, if we combine the wave functions of $2s$ and $2p_z$ orbitals in order to obtain the hybrid orbital sp^3, we are forced to use appropriate coefficients, so that the resulting wave function becomes

$$\psi_{sp3} = \frac{1}{2}\psi_{2s} + \sqrt{\frac{3}{4}}\psi_{2p(z)}. \tag{1.21}$$

By plotting the angular part of ψ_{sp3}, according to the above combination of the $2s$ and $2p_z$-orbital angular functions in Table 1.1, the resulting sp^3 hybrid orbital, being parallel to z-axis, has a shape of a bulky sphere as in Fig. 1.8, which is unfit for describing bond formation and is also unrealistic if we consider the presence of the other three hybrid orbitals and their repulsive interactions. As a consequence, and according to the majority of general chemistry textbooks [1], we adopt a simplified representation of the hybrid orbital, in the form of an elongated ellipsoid.

To plot the parameterized surface of a hybrid orbital, the Matlab function `cylinder` is used, shaped by the empirical function of the surface radius $r = 0.2\left(t - 0.03t^6\right), 0 \leq t \leq 2$, where t is the coordinate of the surface axis.

```
t = 0:0.1:2;
r = 0.2*(t - 0.03*t.^6);
[X,Y,Z] = cylinder(r);
```

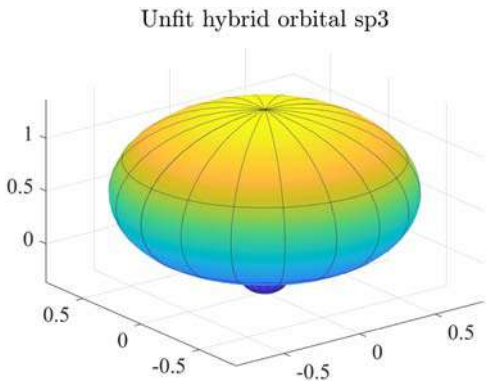


FIGURE 1.8

sp^3 hybrid orbital, unfit for describing bond formation.

The graphical object is then repeatedly rotated the angles required by the hybridization type. As a rule, hybrid orbitals are located in order to obey a maximum repulsion. In the following, hybrid orbital formation is treated starting from 4, 5, and 6 atomic orbitals, respectively, in sp^3, dsp^3, and d^2sp^3 hybrid formation.

1.5.1 Hybrid orbitals sp^3

The four hybrid orbitals in Fig. 1.9 are directed from the axis center to the vertices of a regular tetrahedron. They are obtained by combining one *s*-type and three *p*-type orbitals.

The script follows.

```
InitChem
t = 0:0.1:2;
r = 0.2*(t - 0.03*t.^6);
[X,Y,Z] = cylinder(r);
% Plotting a cylinder with the base centered at the origin
surf(X,Y,Z)
hold on
Y1 = Y.*(-0.334) + Z.*0.943;
Z1 = Z.*(-0.334) - Y.*0.943;
surf(X,Y1,Z1)
X2 = X.*(-0.5) + Y1.*0.866;
Y2 = Y1.*(-0.5) - X.*0.866;
surf(X2,Y2,Z1)
X3 = X.*(-0.5) + Y1.*(-0.866);
Y3 = Y1.*(-0.5) - X.*(-0.866);
surf(X3,Y3,Z1)
title('sp3 hybrid Orbitals')
```

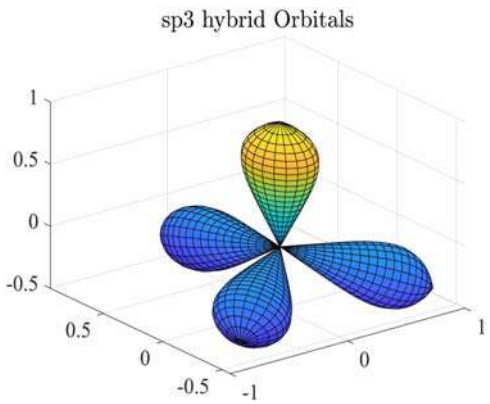


FIGURE 1.9

Four sp^3 hybrid orbitals.

1.5.2 Hybrid orbitals dsp^3

The five hybrid orbitals in Fig. 1.10 are directed from the axis center to the vertices of a triangular bipyramid. They are obtained by combining one *s*-type, three *p*-type, and one *d*-type orbitals.

The script follows.

```
InitChem
t = 0:0.1:2;
r = 0.2*(t - 0.03*t.^6) ;
[X,Y,Z] = cylinder(r);
% Plot the cylinder with the base centered at the origin.
surf(X,Y,Z)
hold on
surf(X,Y,-Z)
Y1 = Z;
Z1 = -Y;
surf(X,Y1,Z1)
X2 = X.*(-0.5) + Y1.*0.866;
Y2 = Y1.*(-0.5) - X.*0.866;
surf(X2,Y2,Z1)
X3 = X.*(-0.5) + Y1.*(-0.866);
Y3 = Y1.*(-0.5) - X.*(-0.866);
surf(X3,Y3,Z1)
title('dsp3 Hybrid Orbitals')
```

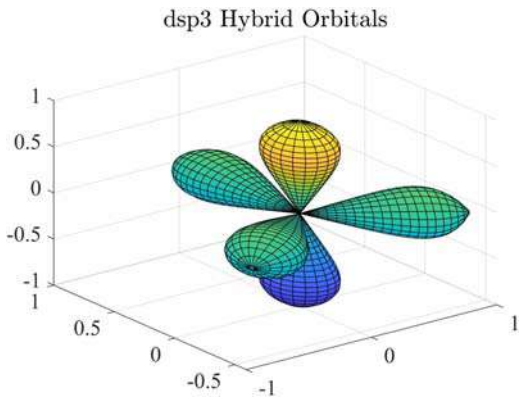


FIGURE 1.10

The dsp^3 orbital shape.

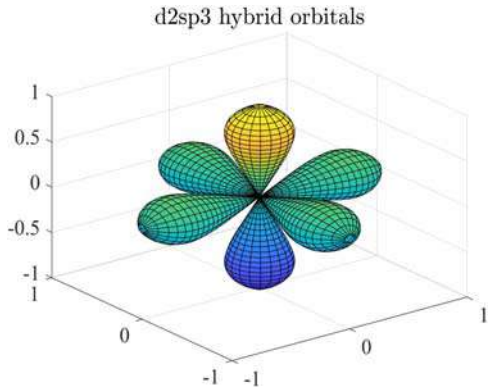


FIGURE 1.11

The d^2sp^3 orbital shape.

1.5.3 Hybrid orbitals d^2sp^3

The six hybrid orbitals in Fig. 1.11 are directed from the axis center to the vertices of an octahedron. They are obtained by combining one *s*-type, three *p*-type, and two *d*-type orbitals.

The script follows.

```
InitChem
t = 0:0.1:2;
r = 0.2*(t - 0.028*t.^6) ;
[X,Y,Z] = cylinder(r);
% Plot the cylinder with the base centered at the origin.
surf(X,Y,Z)
title('d2sp3 hybrid orbitals');hold on
surf(Z,X,Y)
surf(Y,Z,X)
surf(X,Y,-Z)
surf(-Z,X,Y)
surf(Y,-Z,X)
```

References

[1] R.H. Petrucci, F.G. Herring, J.D. Madura, C. Bissonette, General Chemistry Principles and Modern Applications, 10th Edition, Pearson, Canada, 2011.

[2] J.R. Chelikowsky, Introductory Quantum Mechanics with Matlab for Atoms, Molecules, Clusters, and Nanocrystals, Wiley, 2019.

[3] D.J. Griffiths, D.F. Schroeter, Introduction to Quantum Mechanics, Third Edition, Cambridge University Press, 2018.

[4] E. Canuto, C. Novara, L. Massotti, D. Carlucci, C. Perez Montenegro, Spacecraft Dynamics and Control: The Embedded Model Control Approach, Butterworth-Heinemann, 2018.
[5] Wikipedia, *Table of spherical harmonics*, https://en.wikipedia.org/wiki/Table_of_spherical_harmonics.
[6] Wikipedia, *Ground State of a quantum-mechanical system*, https://en.wikipedia.org/wiki/Ground_state.
[7] Wikipedia, *Valence electrons/Valence shell* https://en.wikipedia.org/wiki/Valence_electron#Valence_shell.
[8] K. Geriche, Theoretische Chemie:Construction of hybrid orbitals, Uni. Braunschweig, http://www.pci.tu-bs.de/aggericke/PC4e/Kap_II/Hybride/Konstruktion.htm.

CHAPTER

Balancing chemical reactions with Matlab

2

2.1 Introduction

A balanced chemical equation is a condensed statement that expresses, by symbols and formulas, both qualitative and quantitative information about a specific chemical reaction. It tells us, in terms of molecular ratios, the specific relationships which exist between and among the initial reactants and the chemical products into which reactants are being converted. The numeric coefficients assigned to a balance equation express the relative amounts of atoms or molecules which take part to the specific reactions. More precisely tells us the proportions of moles involved therein. This fact derives directly from *Proust law* (J.L. Proust, 1750−1826) or *law of definite proportions*. It states that a given chemical compound always contains its component elements in a fixed ratio, irrespectively of the preparation route.

The quantitative aspect, dealing with mass and volume relations among reactants and products, is termed stoichiometry [1].

A mole is defined as the amount of matter that contains as many objects (atoms, molecules, electrons, protons, or whatsoever objects we are considering) as the same number of atoms in exactly 12 g of ^{12}C. This number is also known as Avogadro's number (N_A), a constant equal to 6.023×10^{23} molecules/mole (A.C. Avogadro, 1776−1856). Thus one mole of an entity contains N_A particles of that entity.

As a consequence, the molar mass is the mass (in gram) of 1 mole of a substance; it is known as the molecular weight as well. Let us clarify these concepts with a simple example, the hydrogen combustion. The reaction is represented by the following balanced chemical equation:

$$2H_2(g) + O_2(g) \Rightarrow 2H_2O(g) (g = \text{gaseous state}) \tag{2.1}$$

1. Molecular relative ratio: two molecules of H_2 react with one molecule of O_2 forming two molecules of H_2O vapor:

$$2 \text{ molecules} + 1 \text{ molecule} \Rightarrow 2 \text{ molecules} \tag{2.2}$$

2. Mole relative ratio: 2 mol of H_2 react with 1 mol of O_2 forming 2 mol of H_2O vapor:

$$\begin{gathered} 2 \text{ mol} + 1 \text{ mol} \Rightarrow 2 \text{ mol} \\ \left(2 \times 6.023 \times 10^{23} \text{ molecules}\right) + \left(1 \times 6.023 \times 10^{23} \text{ molecules}\right) \Rightarrow \\ \Rightarrow \left(2 \times 6.023 \times 10^{23} \text{ molecules}\right) \end{gathered} \tag{2.3}$$

3. Weight ratio: 4 g of H_2 react with 32 g of O_2 forming 36 g of H_2O vapor:

$$4 \text{ g} + 32 \text{ g} \Rightarrow 36 \text{ g} \tag{2.4}$$

Fundamental Chemistry with Matlab. DOI: https://doi.org/10.1016/B978-0-323-91341-6.00015-X

Therefore the coefficients in a balanced chemical reaction can be interpreted as the relative number of moles, molecules, or volumes (if reactants are gases) involved in the reaction. These coefficients are known as *stoichiometrically equivalent quantities*; they are expressed by *minimal integer numbers.*

It follows that the coefficients of a properly balanced equation determine the quantitative aspects of a chemical reaction. However, in our immediate concern with a balanced equation which provides the ultimate stoichiometric relationships, we seek a procedure that correctly establishes the coefficients of the equation; that is, the arithmetical multipliers of the different chemical units therein expressed by minimal integers.

To be stoichiometrically valid, a chemical equation must fulfill the following conditions:

1. The equation must express what experimentally occurs from reactants into products. The chemical identities and chemical formulas of all species participating in the change must be known and written in their exact elemental, molecular, or ionic identities. Frequently, little is known about the precise mechanism by which a particular substance undergoes alteration of its electronic configuration. The intermediate changes in the chemical identities of species as they pass from the reactants to the ultimate products and their time evolution are the difficult derivations sought by the chemical kinetics. Essentially, the net balanced equation merely reports the disappearance or depletion of initial reactants and the formation or maintenance of final products. In the following illustration,

$$A + B \Rightarrow AB(\text{intermediate}); \; AB + B \Rightarrow AB_2, \tag{2.5}$$

the final net reaction does not account for the AB intermediate species and appears as

$$A + 2B \Rightarrow AB_2 \quad (\text{net}) \tag{2.6}$$

The fact that species A has not combined with species B directly to form AB_2, but indirectly, through the intermediate formation of AB, has not altered the overall net reaction of change which is being sought in the balanced chemical equation.

2. The equation must be balanced atom by atom. In conformity with the law of conservation of mass, equal numbers of the same kinds of all atoms must appear on both sides of the arrow.
3. If the reaction comprises ionic (charged) species among reactant and products, equation must be balanced electronically. In conformity with the law of conservation of electric charge, the total net charge, apparent or real, of the reacting materials specified on the left side must equal the total net electrical charge of the products specified on the right side.

Balancing of electron charges—losses and gains—which result from breaking old and making new chemical bonds can be rather complicated. Appropriate algebraic procedures have been developed for ensuring and satisfying the law of mass conservation, in other terms, for ensuring that the number of atoms of a type must be the same among reactants and products.

Sometimes, reactions in a solution (mainly water) can be written in ionic form, thus dismissing all the ions which do not directly participate to the reaction (the spectator ions) and thus only showing those ions directly participating to the process.

For instance, the oxidation of ferrous sulfate ($FeSO_4$) with potassium permanganate in an acid medium is written in molecular form as:

$$10FeSO_4 + 2KMnO_4 + 8H_2SO_4 \Rightarrow 2MnSO_4 + 5Fe_2(SO_4)_3 + 1K_2SO_4 + 8H_2O \tag{2.7}$$

and, in ionic form, it becomes

$$5Fe^{2+} + MnO_4^- + 8H^+ \Rightarrow 5Fe^{3+} + Mn^{2+} + 4H_2O. \quad (2.8)$$

In this case, a second equation should be added in order to express the conservation of the charge, which amounts to 17 positive charges on the left (18 positive minus one negative) balanced by 17 positive charges on the right.

Ionic equations can be solved with the proposed algorithm; however, they will not be treated directly. As a matter of fact, leaving aside spectator ions induces a loss of information on how to carry out reactions in practice. For example, reaction (2.8) does not specify which acid should be used. A weak acid would not bring reaction to be complete. Chloridric acid, too, though being a strong acid, will be oxidized by permanganate, becoming useless.

2.2 Nonredox and redox reactions

There are two broad categories within which all chemical reactions can be classified as *nonredox* and *redox* (a contraction of reduction and oxidation [2]). Differences between them are not based upon the chemical identities of the involved substances but rather upon the algebraic involvements of the specific subatomic particles—electrons—which numerically determine the chemical characteristics of all identifiable matter.

2.2.1 Nonredox reactions (or metatheses)

In a metathesis, as a net effect, no electrons are actually lost or gained by the substances undergoing mutual changes. The involved substances merely exchange chemical partners, and each of the partners retains the full complement of electrons and chemical bonds in its electronic structure which existed before the exchange. This fact may be envisaged by the following exchange reaction:

$$\text{silver nitrate} + \text{sodium chloride} \Rightarrow \text{silver chloride} + \text{sodium nitrate} \quad (2.9)$$

which is symbolized by

$$AgNO_3 + NaCl \Rightarrow AgCl + NaNO_3. \quad (2.10)$$

Other types of nonredox reactions include among others, precipitation, association or dissociation double exchange, metal complex formation, and acid−base neutralization.

2.2.2 Redox reactions

In this class of reactions, the oxidation number/state (briefly *nox*) of at least two elements undergoes a variation from reactants to products. First, we should clarify the meaning of the oxidation state of an element. It is defined as the charge (real or imaginary) which an atom appears to have when combined with other atoms. In the case of atomic ions, the oxidation number of the element is the same as the charge of the ion. It represents the number of electrons lost or gained by an element during its change from the free state to the compound state, in other terms, the extent of

oxidation or reduction of an element during its change. Oxidation number is given positive sign if electrons are lost and negative sign if electrons are gained.

An oxidation number represents real charges in case of ionic compounds. In covalent compounds, it represents an imaginary charge.

2.2.3 Stoichiometry and nonredox reactions

The usual method of assigning proper coefficients to a nonredox reaction is a stepwise procedure in which single elements are balanced on the left and right sides of the reaction by temporarily assigning coefficients to the compounds that contain such elements.

The procedure becomes a little easier if invariant groups of atoms are found, for instance, sulfate, SO_4^{2-}, and phosphate, PO_4^{3-}, groups. In this case, these groups can be balanced directly (by inspection) without counting the single elements in the overall summation. In the previous groups, oxygen, sulfur, and phosphor can be excluded, for instance.

2.2.4 Stoichiometry and redox reactions

In general chemistry textbooks, balancing of redox reactions is obtained by the *oxidation state method*. The method exploits the fact that the number of electrons gained during reduction must be equal to the number of electrons lost during oxidation. The necessary steps for balancing redox equations are as follows.

1. Write the skeleton equation (if not given, frame it) representing the chemical change.
2. With the help of the oxidation number of elements, find out which atom is undergoing oxidation/reduction and write separate equations for the atom undergoing oxidation/reduction.
3. Add the respective electrons on the right side for oxidation equations and on the left side for reduction equations. Let us remark that the net charge on the left and right sides should be equal.
4. Multiply oxidation and reduction reactions by suitable integers so that total electrons lost in one reaction is equal to the total electrons gained by the other reaction.
5. Transfer the coefficients of oxidizing and reducing agents and their products which have been found in the above steps to the concerned molecule or ion.
6. By inspection, supply the proper coefficient to the other substance formulas, which do not undergo oxidation and reduction, in order to balance equation. This is typically the case of water, which is added (among reactants or products) in order to fulfill the need of equating H and O, at the end of the balancing process.

2.3 General method

2.3.1 The balance equation

The lack of a general method in the reaction balance process causes a widespread use of iterative numerical procedures, their goal being to reach the final set of coefficients by balancing in each step a selected element or functional group.

A general method can be found with the aid of *linear algebra* (see Appendix A), and many reactions, notably redox reactions, readily lend themselves to a convenient algebraic algorithm based on the *nullspace concept*. The algorithm bypasses both the oxidation number procedure and iterative methods and applies to complex reactions. Let us introduce the relationship of this concept with the balancing process that yields the stoichiometric equation.

The principle of mass balance is based on the law of conservation of mass, that is, the number of atoms of an element remains constant in a chemical reaction (we exclude nuclear reactions, where new elements may be originated).

Balancing each element in a reaction imposes a set of simple equations in which the unknowns are the stoichiometric coefficients of each substance in reactants and products. To this end, the chemical formula of each compound must be well established with integer atomic indices, like H_2SO_4.

Let us consider a reaction with one reactant S_1 and two products S_2 and S_3, where the symbol S_k stands for the k-th substance chemical formula (either reactant or product), and the subscript k counts from left to right:

$$S_1 \Rightarrow S_2 + S_3. \tag{2.11}$$

Conservation of mass and mole implies the following linear equation:

$$x_1 M_1 = x_2 M_2 + x_3 M_3, \tag{2.12}$$

where the symbol $M_k, k = 1, 2, 3$ denotes the substance mole and the unknown coefficients $x_k, k = 1, 2, 3$ must ensure equality of both sides, a result which is referred to as *reaction balance*. The equation can be written by subtracting the right side from the left side, thus obtaining a homogeneous equation, with the known term in the right side equal to zero:

$$x_1 M_1 - (x_2 M_2 + x_3 M_3) = 0 \Leftrightarrow \alpha(x_1 M_1 - x_2 M_2 - x_3 M_3 = 0). \tag{2.13}$$

It is straightforward to prove that the linear relationship between the moles M_k of the substances does not change if the whole equation is multiplied by an arbitrary nonzero scalar α (scale factor), as in the right side of Eq. (2.13). Eq. (2.13) can be extended to an arbitrary number n of substances by writing

$$\begin{gathered} s_1 x_1 M_1 + \ldots + s_k x_k M_k + \ldots + s_n x_n M_n = 0 \Leftrightarrow \boldsymbol{M}^T S \boldsymbol{x} = 0 \\ \boldsymbol{M} = [M_1, \ldots, M_k, \ldots, M_n], \boldsymbol{x} = [x_1, \ldots, x_k, \ldots, x_n], S = \mathrm{diag}(s_1, \ldots, s_k, \ldots, s_n) \end{gathered}. \tag{2.14}$$

The right equation in the top row has been written in a vectorial form, where $\boldsymbol{M}$ and $\boldsymbol{x}$ are vectors, and S is a diagonal matrix of signs, $s_k = \pm 1$, the sign being positive for reactants and negative for products.

Mole conservation of reactants and products is the result of the mole conservation of each element $E_j, j = 1, \ldots, m$ entering the reaction substances. This implies that m balance equations must be written, one for each element. Summation of these equation must provide Eq. (2.14). To this end, the mole M_k of a generic substance $S_k = E_1 a_{1k} E_2 a_{2k} \ldots E_j a_{jk} \ldots E_m a_{mk}$, where E_j is a generic element with a_{jk} atoms (integer number) in the compound S_k, can be written as the linear combination of the element moles N_j as follows

$$\begin{gathered} M_k = N_1 a_{1k} + \ldots + N_j a_{jk} + \ldots + N_m a_{mk} \Leftrightarrow \boldsymbol{N}^T \boldsymbol{a}_k \\ \boldsymbol{N} = \left[N_1, \ldots, N_j, \ldots, N_m\right], \boldsymbol{a}_k = [a_{1k}, \ldots, a_{jk}, \ldots, a_{mk}] \end{gathered}, \tag{2.15}$$

where some integer coefficients may be zero, $a_{jk} = 0$, since the vector $\boldsymbol{N}$ must include all the reaction elements.

Replacement of Eq. (2.15) into (2.14) provides the following vectorial equation

$$\boldsymbol{N}^T A S \boldsymbol{x} = 0$$

$$A = [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_k, \ldots, \boldsymbol{a}_n] = \begin{bmatrix} \boldsymbol{b}_1 \\ \ldots \\ \boldsymbol{b}_j \\ \ldots \\ \boldsymbol{b}_m \end{bmatrix}, \tag{2.16}$$

where A is an $m \times n$ known matrix. The k-th column $\boldsymbol{a}_k$ refers to the k-th substance, and the j-th row of A denoted by $\boldsymbol{b}_j$ refers to the j-th element. The sign is accounted for by matrix S. Mole conservation of a single element is expressed by setting to zero the j-th row of $AS\boldsymbol{x}$, if premultiplied by the mole N_j. Since the latter scalar plays the role of a scale factor, it can be dropped, implying, as already told, that balancing can be obtained without knowing the element moles.

As a conclusion, the m element conservation equations (*balance equations*) can be written in a matrix form as a system of m linear homogeneous equations with n unknowns collected into the vector $\boldsymbol{x}$, which is the vector of the *stoichiometric coefficients*:

$$AS\boldsymbol{x} = 0. \tag{2.17}$$

Since the matrix AS is integer and we look for an integer solution, Eq. (2.17) is known as a *linear homogeneous Diophantine equation* [3].

2.3.2 Example

The different elements in reactants and products are balanced by inspection, thus obtaining a set of m equations, each for any single element. In the following example of a redox reaction, oxidation of silver sulfide with nitric acid:

```
a Ag2S + b HNO3 + c HCl ==> d S + e AgCl + f NO(gas) + g H2O
```

the six elements Ag, Cl, H, S, N, and O are balanced by obtaining a set of six linear equations with the seven stoichiometric coefficients a, b, c, d, e, f and g as unknowns.

```
InitChem;
% Find the coefficient of a redox chemical reaction.
disp ('a Ag2S + b HNO3 + c HCl ==> d S + e AgCl + f NO(gas) + g H2O');
% Oxidation of silver sulphide with nitric acid.
syms a b c d e f g
eqn1 = 2*a - e == 0;  % balance Ag
eqn2 = c - e == 0;  % balance Cl
eqn3 = b + c - 2*g  == 0;  % balance H
eqn4 = a - d == 0;  % balance S
eqn5 = b - f == 0;  % balance N
eqn6 = 3*b - f - g == 0; % balance O
```

2.4 Solution of the homogeneous system of linear equations

2.4.1 The nullspace method

As a solution of Eq. (2.14), we look for a positive integer vector $\hat{\boldsymbol{x}}$ of minimum norm which satisfies the equation. That is, the greatest common divisor of the components $\hat{x}_k$ must be the unit. Since the entries of the matrix AS are integers, positive, zero, or negative, a nontrivial solution, if it exists, will be a rational vector r. The necessary and sufficient condition for the solution existence is that

$$\text{rank}(AS) < n. \tag{2.18}$$

In other terms, the m rows of AS must not completely span the rational vector space $\mathcal{Q}^n$, but they must only span a subspace $\mathcal{Q}_A$ of $\mathcal{Q}^n$, having dimension equal to $\rho = \text{rank}(AS)$. Moreover, Eq. (2.14) tells us that the solution $\hat{\boldsymbol{x}}$ must be orthogonal to the rows of AS and therefore to the subspace $\mathcal{Q}_A$. The subspace $\mathcal{Q}_N$ of $\mathcal{Q}^n$ containing all the vectors orthogonal to $\mathcal{Q}_A$, is known as the *nullspace* of AS, and its dimension holds $\nu = n - \rho$. Thus $\hat{\boldsymbol{x}}$ must be searched in $\mathcal{Q}_N$ (see Fig. 2.1).

We assume that the components of a chemical reaction, namely reactants and products, are properly written, such as no element is missing and chemical formulas respect the valence rules. In this case, the integer stoichiometric coefficients are defined so that the nullspace possesses minimal dimension, formally $\nu = n - \rho = 1$, in which case the nullspace reduces to a line passing through the origin as in Fig. 2.1. Moreover, we assume the following value m for the rows of AS.

1. Most of redox equations: $m = n - 1$ and $\nu = n - \rho = 1$.
2. Exceptions of redox equations: $m = n$ and $\nu = n - \rho = 1$.
3. Nonredox equations: $m \geq n$ and $\nu = n - \rho = 1$.

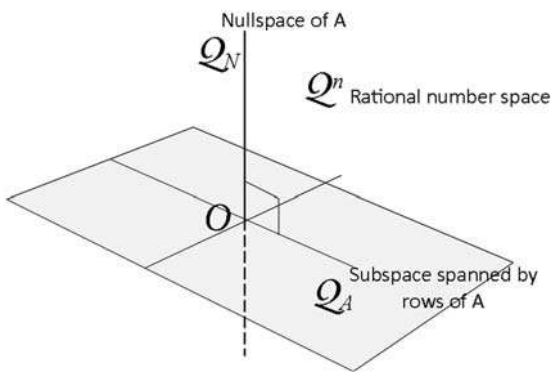


FIGURE 2.1

Sketch of the subspace spanned by the rows of A and the relevant nullspace.

2.4.2 Example

By continuing the previous example of the silver sulfide oxidation, the six element equations are combined in a symbolic matrix `A`, by using the `equationsToMatrix` function in Matlab®. Thereafter, the matrix `A` is transformed into a double precision array:

```
[A] = equationsToMatrix([eqn1,eqn2,eqn3,eqn4,eqn5,eqn6]);
[A] = double([A]);
```

As a second step, a vector in the nullspace of the matrix *A* is found by the function `null()`, and a rational expression is calculated by the function `rat()`:

```
Z=null(A,'r');
[nZ,dZ]=rat(Z);
```

Finally, the least common multiple of `dZ` is extracted into the vector `intZ`, which contains the integer stoichiometric coefficients to be printed out.

```
intZ=Z*double(lcm(sym(dZ)));
T1=  [num2str(intZ(1)),'   Ag2S  +  ',num2str(intZ(2)),'   HNO3   +
',num2str(intZ(3)),' HCl  ==> ',num2str(intZ(4)),' S  + '];
T1 =  [T1,num2str(intZ(5)),' AgCl  + ',num2str(intZ(6)),' NO(gas) +
',num2str(intZ(7)),' H2O'];
disp('when balanced looks like : ');
disp(T1);
```

The printed results in the `command window` appear as follows:

```
a Ag2S + b HNO3 + c HCl ==> d S + e AgCl + f NO(gas) + g H2O
when balanced looks like :
3 Ag2S + 2 HNO3 + 6 HCl ==> 3 S + 6 AgCl + 2 NO(gas) + 4 H2O
```

2.4.3 Balancing algorithm from chemical formulas

The explicit setup of a system of linear, homogeneous equations may be time consuming and a source of mistakes. As an improvement, reactants and products may be coded directly and interpreted by Matlab through the symbolic function `children`. The Matlab script starts with the symbolic chemical reactions which must be coded in the following way:

1. *Symbols of the elements* are listed after `syms` and then collected into the vector `s` as follows:

```
% Elements
syms Fe S O K Mn H
s=[Fe;S;O;K;Mn;H];
```

2. *Writing the symbols of the reaction substances*: the multiplicity of each element, say 2, must be written as an exponent with the caret symbol, like `H^2`, and the elements must be separated by `*` (Matlab multiplication symbol), like `H^2*O`. Reactants are collected into the vector `Fleft` and products into `Fright`. The whole substance vector is `F=[FLeft;Fright]`. The order of the elements may be any, since symbolic Matlab reorders them in a lexicographic way.

```
% Left side of the chemical equation
% Warning: number of atoms preceded by caret ^
FLeft=[Fe*S*O^4;K*Mn*O^4;H^2*S*O^4];
dimFLeft=length(FLeft);
% Right side
FRight=[Mn*S*O^4;Fe^2*S^3*O^12;K^2*S*O^4;H^2*O];
% Vector of molecules
F=[FLeft;Fright];
```

3. *The balance equation solution* is found by the function `stoichiometry`

```
% Function call
intCoeff=stoichiometry(s,F,dimFLeft,text,logVersion);
```

4. The first three input variables of the function have been already defined. The last two variables, `text` and `logVersion` allow the name of the reaction to be entered:

```
text='Redox reaction 2 - oxidation of hydrogen peroxide';
```

and the script to fit Matlab 2020b and successive versions:

```
V2020b='2020b';
logVersion=0;
if V2020b==version('-release')
   logVersion=1;
end
```

The reason of the previous statements is the output of the function `children(expression)` to be employed by the function `stoichiometry`. Before Matlab 2020b version, the output was a vector with the subexpressions as entries. Since the 2020b version, the output is a cell array, which is converted into a vector as shown in Section 2.5.1.

5. *Details of the function* `stoichiometry` see Section 2.5.1. In the first part, using symbolic Matlab, the balance matrix `A`, corresponding to the matrix *AS* in Eq. (2.14), is extracted from the substance vector `F`. In the second part, the `null` Matlab function finds the rational vector of the nullspace of `A`, which is then converted into the minimum norm integer `intZ`.

2.5 Nullspace algorithm for balancing chemical reactions

The main script applies to balancing the oxidation of iron(II) sulfate with potassium permanganate in an aqueous solution of sulfuric acid. Iron is oxidized from *oxidation number* (or *oxidation state*), nox $+2$ to $+3$, and, necessarily, another element is reduced. In this case, manganese shifts from nox $+3$ to $+2$. In most redox reactions, only two elements vary their oxidation number, one increasing it (the element being oxidized) and the other decreasing it (the element being reduced).

```
%% Chemistry with Matlab - Stoichiometry (Chapter 2)
% Stoichiometry matrix and balance from reaction formula
%% 1) Example
% text='Name of the reaction';
% syms Fe SO4 H O  % list of elements and radical
% s=[ Fe; SO4; H; O  ]; % vector of previous elements
% % Left side of chemical reaction
% % 1) * between elements
% % 2) number of atoms preceded by caret ^
% FLeft=[Fe*SO4; H^2*O^2; H^2*SO4];
% % Right side
% FRight=[Fe^2*SO4^3 ; H^2*O ];
% % Balancing function
%intCoeff=stoichiometry(s,[FLeft;FRight],length(FLeft),text,logVersion);
% % Printout: R =reagent, P=product
%% 2) Initialization
InitChem;
% Matlab release version
V2020b='2020b';
logVersion=0;
if V2020b==version('-release'), logVersion=1; end
%% 3) Redox reaction 1 - Plug-in code
% Finds the coefficient of a redox chemical reaction.
% Oxidation of iron(II)sulphate with potassium permanganate.
text='Redox reaction 1 - Oxidation of iron sulphate';
syms Fe S O K Mn H
s=[Fe;S;O;K;Mn;H];
FLeft=[Fe*S*O^4;K*Mn*O^4;H^2*S*O^4];
FRight=[Mn*S*O^4;Fe^2*S^3*O^12;K^2*S*O^4;H^2*O];
intZ=stoichiometry(s,[FLeft;Fright],length(FLeft),text,logVersion);
%% Other plug-in codes
StoichiometryNSPlugin;
```

The script `StoichiometryNSPlugin.m`, available in the companion website of the book [5] contains the sequence of all the plug-in codes reported in the following sections. The *plug-in code* in the main script above is specific of the oxidation of iron sulfate by potassium permanganate. It must be replaced by the plug-in code of the reaction (redox or nonredox) to be balanced, as

explained in the following. Each plug-in code ends by calling the function `stoichiometry`, which prints the results in the `command window`. The function must be saved as a `function` script, under the name `stoichiometry.m` in the same folder of the main script.

2.5.1 Nullspace function stoichiometry

As already said, the nullspace function first builds the balance matrix by interpreting the chemical formula of the involved substances, then finds the rational nullspace vector of the matrix, and converts it into a minimum integer vector. As already mentioned, the output type of the function `children()` depends on the Matlab version. Since the 2020b version, the output is a cell array which the following function converts into a vector.

```
%% Function Stoichiometry
function intZ=stoichiometry(s,F,dimFLeft,text,logVersion)
% Initialization
   disp(' --- Minimum balance stoichiometry ---');
   disp(text);
   dimVar=length(s);
   dimF=length(F);
% Type
   syms R P
   type(1:dimFLeft)=R;
   type(dimFLeft+1:dimF)=P;
% Elements and substances
   fprintf('Element No.=%3d  Substance No.=%3d\n',dimVar,dimF);
% Redox/non-redox dimensions
   logDim= dimF==dimVar+1;
   if logDim ==0
        disp('WARNING: Non-redox reaction, with exceptions');
   end
% Balance matrix construction
   A=zeros(dimVar,dimF);
   for i=1:dimF
        sa=1;
        if i>dimFLeft, sa=-1; end
% Single element like O^2 is separated from multiple like H^*O
        varF=symvar(F(i));
        F1=F(i);  % single element
% Multiple element
        if length(varF)>1
           F1=children(F(i)); % before 2020b F1=vector, now cell array
% conversion of cell array into vector
           if logVersion==1, F1=[F1{:}]; end
        end
```

```
% Matrix entries
       for j=1:length(F1)
           F2=children(F1(j)); % before 2020b F1=vector, now cell array
% conversion of cell array into vector
           if logVersion==1, F2=[F2{:}]; end
           a=sa;
           if length(F2)==2, a=a*double(F2(2)); end
           for k=1:dimVar
                if F2(1)==s(k), A(k,i)=a; end
           end
       end
  end
% Finding balance integer coefficients (least amount)
  Z=null(A,'r');
  [nZ,dZ]=rat(Z);
  intZ=Z*double(lcm(sym(dZ)));
  fprintf('      Substance   Amount  Type\n');
  for i=1:dimF
       fprintf('%15s  %5d  %4s\n',F(i),intZ(i),type(i));
  end
end
```

2.5.2 Results of the plug-in code example

The balance results are shown in the `command window` as follows.

```
 --- Minimum balance stoichiometry ---
Redox reaction  - Oxidation of iron sulphate
Element No.=  6  Substance No.=  7
      Substance   Amount  Type
       Fe*O^4*S      10     R
       K*Mn*O^4       2     R
      H^2*O^4*S       8     R
       Mn*O^4*S       2     P
  Fe^2*O^12*S^3       5     P
      K^2*O^4*S       1     P
          H^2*O       8     P
```

2.6 Catalog of the reactions and of their plug-in codes

2.6.1 Oxidation of hydrogen peroxide by potassium permanganate

1. *Plug-in code*. In this reaction, the interest concerns the oxygen that appears both as atomic O and molecular O_2. This fact is peculiar to a very few redox reactions where a part of the oxygen varies the oxidation number from -1 to 0, whereas the remaining oxygen remains stable to nox -2. The symbol O_2 indicates oxygen atoms that change nox, whereas the symbol O denotes those oxygen atoms which remain unchanged. Disregarding this fact, the unaware chemist will never be able to balance this reaction.

```
%% 4) Redox reaction 2
text='Redox reaction 2 - oxidation of hydrogen peroxide';
syms H O2 K Mn O S
s=[H;O2; K; Mn; O;S];
FLeft=[H^2*O2;K*Mn*O^4;H^2*S*O^4];
FRight=[H^2*O;O2;Mn*S*O^4;K^2*S*O^4];
intZ=stoichiometry(s,[FLeft;FRight],length(FLeft),text,logVersion);
```

2. *Results* in the `command window`

```
--- Minimum balance stoichiometry ---
Redox reaction 2 - oxidation of hydrogen peroxide
Element No.= 6  Substance No.=  7
      Substance   Amount  Type
         H^2*O2       5     R
        K*Mn*O^4      2     R
       H^2*O^4*S      3     R
           H^2*O      8     P
              O2      5     P
        Mn*O^4*S      2     P
       K^2*O^4*S      1     P
```

2.6.2 Oxidation of silver sulfide by aqua regia

1. *Plug-in code*. The highly insoluble silver sulfide can be readily dissolved by a mixture of nitric and chloridric acid. The mixture is known as *aqua regia* being capable of dissolving noble metals like gold and platinum. Products of the reaction are gaseous NO, sulfur, and the insoluble salt silver chloride.

```
%% 5) Redox reaction 3
text='Redox reaction 3 - oxidation of silver sulphide by aqua regia';
syms Ag S H N O Cl
s=[Ag; S; H; N; O; Cl];
FLeft=[Ag^2*S;H*N*O^3;H*Cl];
FRight=[S;Ag*Cl; N*O; H^2*O];
intZ=stoichiometry(s,[FLeft;FRight],length(FLeft),text,logVersion);
```

2. *Results*

```
--- Minimum balance stoichiometry ---
Redox reaction 3 -
Element No.=  6  Substance No.=  7
      Substance   Amount  Type
         Ag^2*S       3     R
        H*N*O^3       2     R
           Cl*H       6     R
              S       3     P
          Ag*Cl       6     P
            N*O       2     P
          H^2*O       4     P
```

2.6.3 Oxidation of bromidric acid by potassium dichromate

1. *Plug-in code*. $K_2Cr_2O_7$, the potassium dichromate, is a strong oxidizing agent in an acid solution. In this case, the acid is HBr, the hydrogen bromide, which acts as a reducing agent as well. Only a part of Br is oxidized to nox $+3$, the remaining stays at nox -1. Unlike the reaction in Section 2.6.1, there is no need to differentiate the two types of bromine atoms.

```
%% 6) Redox reaction 4
text='Redox reaction 4 - Oxidation of bromidric acid by potassium dichromate';
syms K Cr O Br H
s=[ K; Cr; O; Br; H];
FLeft=[K^2*Cr^2*O^7; H*Br];
FRight=[Cr*Br^3; Br^2;K*Br;H^2*O];
intZ=stoichiometry(s,[FLeft;FRight],length(FLeft),text,logVersion);
```

2. *Results*

```
Redox reaction 4 -Oxidation of bromidric acid by potassium dichromate
No elements=  5  No molecules=  6
      Substance   Amount  Type
   Cr^2*K^2*O^7      1     R
          Br*H      14     R
       Br^3*Cr       2     P
          Br^2       3     P
          Br*K       2     P
         H^2*O       7     P
```

2.6.4 Oxidation of mercury by nitric and chloridric acid

1. *Plug-in code*. Mercury is oxidized from nox 0 (the metallic state) to nox +2 by a mixture of nitric and chloridric acid (*aqua regia*, already mentioned). As a result, mercuric chloride is obtained.

```
%% 7) Redox reaction 5
text='Redox reaction 5- Oxidation of mercury by nitric and chloridric acid';
syms Hg H N O Cl
s=[ Hg ; H; N ; O; Cl];
FLeft=[Hg;H*N*O^3;H*Cl];
FRight=[Hg*Cl^2; N*O;H^2*O];
intZ=stoichiometry(s,[FLeft;FRight],length(FLeft),text,logVersion);
```

2. *Results*

```
Redox reaction 5 - Oxidation of mercury by nitric and chloridric acid
No elements=  5  No molecules=  6
      Substance   Amount  Type
             Hg       3     R
       H*N*O^3        2     R
          Cl*H        6     R
       Cl^2*Hg        3     P
           N*O        2     P
         H^2*O        4     P
```

2.6.5 Oxidation of chromium(II) bromide by sodium bromate

1. *Plug-in code*. In this reaction, a part of bromine among reactants is reduced from nox +5 to −1. Unlike the reaction in Section 2.6.1, there is no need to differentiate the two types of bromine atoms.

```
%% 8) Redox reaction 6
text='Redox reaction 6 - Oxidation of Chromium (II) bromide by sodium bromate';
syms Co Br Na O H
s=[ Co; Br ;Na; O; H];
FLeft=[Co*Br^2; Na*Br*O^3; Na*O*H];
FRight=[Co^2*O^3;Na*Br; H^2*O];
intZ=stoichiometry(s,[FLeft;FRight],length(FLeft),text,logVersion);
```

2. *Results*

```
Redox reaction 6 - Oxidation of chromium (II) bromide by sodium bromate
No elements=  5  No molecules=  6
      Substance   Amount  Type
Br^2*Co       6     R
      Br*Na*O^3       1     R
         H*Na*O      12     R
       Co^2*O^3       3     P
          Br*Na      13     P
          H^2*O       6     P
```

2.6.6 Zinc oxidation by silver arseniate

1. *Plug-in code*. In this reaction, one element, Zn, is oxidized, whereas two elements are simultaneously reduced: Ag from nox +1 to 0 and As from nox +5 to −3.

```
%% 9) Redox reaction 7
text='Redox reaction 7 - Zinc oxidation by silver arseniate';
syms Ag As O Zn H S
s=[ Ag; As; O; Zn ;H ;S];
FLeft=[Ag^3*As*O^4; Zn; H^2*S*O^4];
FRight=[Ag; As*H^3;Zn*S*O^4; H^2*O];
intZ=stoichiometry(s,[FLeft;FRight],length(FLeft),text,logVersion);
```

2. *Results*

```
Redox reaction 7 -Zinc oxidation by silver arseniate
No elements=  6  No molecules=  7
      Substance   Amount  Type
    Ag^3*As*O^4      2     R
             Zn     11     R
      H^2*O^4*S     11     R
             Ag      6     P
         As*H^3      2     P
       O^4*S*Zn     11     P
          H^2*O      8     P
```

2.6.7 Precipitation of an insoluble salt (AgCl)

1. *Plug-in code*. This case is an example of a *nonredox reaction*, where all the elements keep their oxidation state. It can be formally classified as a double exchange reaction. Chemically speaking, it is a *precipitation* of an insoluble salt, AgCl, by admixing two solutions, silver nitrate and calcium chloride.

```
%% 10) Non redox equation 1
text='Non-redox reaction 1 - Precipitation of an insoluble salt (AgCl)';
syms Ag N O Ca Cl
s=[ Ag; N; O; Ca; Cl];
FLeft=[Ag*N^3*O^3; Ca*Cl^2];
FRight=[Ag*Cl; Ca*N^6*O^6];
intZ=stoichiometry(s,[FLeft;FRight],length(FLeft),text,logVersion);
```

2. *Results*

```
Non-redox reaction 1 - - Precipitation of an insoluble salt, AgCl
No elements=  5  No molecules=  4
WARNING: Non-redox reaction
      Substance   Amount  Type
     Ag*N^3*O^3      2     R
        Ca*Cl^2      1     R
          Ag*Cl      2     P
     Ca*N^6*O^6      1     P
```

2.6.8 Neutralization of carbonic acid with sodium hydroxide

1. *Plug-in code*. This case is an example of a classical nonredox reaction of *neutralization*, where the acid and base pair yield salt and water.

```
%% 11) Non redox equation 2
text='Non-redox reaction 2 - Neutral. of carbonic acid with sodium hydroxide';
syms H C O Na
s=[ H; C; O; Na ];
FLeft=[H^2*C*O^3; Na*O*H];
FRight=[Na^2*C*O^3;H^2*O];
intZ=stoichiometry(s,[FLeft;FRight],length(FLeft),text,logVersion);
```

2. *Results*

```
Non-redox reaction 2 - Neutral.of carbonic acid with sodium hydroxide
No elements=  4  No molecules= 4
WARNING: Non-redox reaction
      Substance   Amount  Type
      C*H^2*O^3       1     R
         H*Na*O       2     R
     C*Na^2*O^3       1     P
          H^2*O       2     P
```

2.6.9 Hydrolysis of sodium carbonate with nitric acid

1. *Plug-in code*. Formally, it is a double exchange reaction of nonredox type. Chemically, it may be classified as a hydrolysis reaction, where the sodium carbonate salt derives from a weak acid, H_2CO_3, and a strong base, NaOH. Experimental conditions lead to the decomposition of the obtained acid, H_2CO_3, into gaseous carbon dioxide, CO_2, and water.

```
%% 12) Non redox equation 3
text='Non-redox reaction 3 - Hydrolysis of sodium carbonate with nitric acid';
syms  Na C O H N
s=[ Na; C ;O; H; N];
FLeft=[Na^2*C*O^3; N*O^3*H];
FRight=[C*O^2;Na*N*O^3;H^2*O];
intZ=stoichiometry(s,[FLeft;FRight],length(FLeft),text,logVersion);
```

2. *Results*

```
Non-redox reaction 3 - Hydrolysis of sodium carbonate with nitric acid
No elements=  5  No molecules=  5
WARNING: Non-redox reaction
      Substance   Amount  Type
     C*Na^2*O^3       1     R
        H*N*O^3       2     R
          C*O^2       1     P
       N*Na*O^3       2     P
          H^2*O       1     P
```

2.6.10 Oxidation of sodium sulfite to sulfate by potassium permanganate

1. *Plug-in code*. This case is an example of reactions occurring in a strong basic medium like KOH, the potassium hydroxide. As a consequence, Mn is reduced from nox +7 to +4 by forming insoluble manganese dioxide [4].

```
%% 13) Redox reaction 8
text='Redox reaction 8 - Oxidation of sodium sulphite to sulphate';
syms K Mn O Na H S
s=[ K; Mn; O; Na; H; S];
FLeft=[K*Mn*O^4; Na^2*S*O^3; H^2*O];
FRight=[Mn*O^2; Na^2*S*O^4; K*O*H ];
intZ=stoichiometry(s,[FLeft;FRight],length(FLeft),text,logVersion);
```

2. *Results*

```
Redox reaction 8 - Oxidation of sodium sulphite to sulphate
Element No.=  6  Substance No.=  6
WARNING: Non-redox reaction, with exceptions
      Substance   Amount  Type
       K*Mn*O^4       2     R
     Na^2*O^3*S       3     R
          H^2*O       1     R
         Mn*O^2       2     P
     Na^2*O^4*S       3     P
          H*K*O       2     P
```

Let us remark the WARNING statement in the previous result table. Actually, the reaction is a redox reaction but the number of elements and substances (molecules) is the same, contrary to standard cases.

2.6.11 Oxidation of iron(II) sulfate by hydrogen peroxide in acid solution

1. *Plug-in code*. Hydrogen peroxide, a strong oxidizing agent, can easily oxidize a Fe^{2+} salt to $+3$ oxidation state in ferric sulfate. The reaction is carried out in acidic medium (sulfuric acid).

```
%% 14) Redox reaction 9
text='Redox reaction 9 - Oxid.of iron(II) sulphate by hydrogen peroxide';
syms Fe SO4 H O
s=[ Fe; SO4; H; O];
FLeft=[Fe*SO4; H^2*O^2; H^2*SO4];
FRight=[Fe^2*SO4^3 ; H^2*O ];
intZ=stoichiometry(s,[FLeft;FRight],length(FLeft),text,logVersion);
```

2. *Results*

```
Redox reaction 9 - Oxid. of iron(II) sulphate by hydrogen peroxide
Element No.=  4  Substance No.=  5
      Substance   Amount  Type
         Fe*SO4       2     R
        H^2*O^2       1     R
        H^2*SO4       1     R
     Fe^2*SO4^3       1     P
          H^2*O       2     P
```

2.6.12 Oxidation of manganese(II) chloride to potassium permanganate by sodium bromate

1. *Plug-in code. Ubi maior minor cessat*. In other words, when a stronger oxidizer like sodium bromate is present, the permanganate does not act as an oxidizing agent. On the contrary, Mn is oxidized to permanganate, in a chloridric acid solution.

```
%% 15) Redox reaction 10
text='Redox reaction 10 - Oxid.of manganese(II) chloride to K permanganate';
syms H Cl Mn Na Bi O K
s=[ H; Cl;Mn; Na; Bi; O; K];
FLeft=[H*Cl; Mn*Cl^2; Na*Bi*O^3; K*Cl];
FRight=[K*Mn*O^4; Bi*Cl^3 ;H^2*O; Na*Cl];
intZ=stoichiometry(s,[FLeft;FRight],length(FLeft),text,logVersion);
```

2. *Results*

```
Redox reaction 10 - Oxid. of manganese(II) chloride to K permanganate
Element No.=  7  Substance No.=  8
      Substance   Amount  Type
           Cl*H      14     R
         Cl^2*Mn      2     R
      Bi*Na*O^3       5     R
           Cl*K       2     R
        K*Mn*O^4      2     P
         Bi*Cl^3      5     P
           H^2*O      7     P
           Cl*Na      5     P
```

References

[1] E.J. Margolis, Formulation and Stoichiometry. A Review of Fundamental Chemistry, Appleton Century Crofts, New York, 1968.

[2] Wikipedia, Redox, https://en.wikipedia.org/wiki/Redox.

[3] W. Givens, Parametric solution of linear homogeneous Diophantine equations, Bull. Am. Math. Soc. 53 (8) (1947) 780–783.

[4] A. Lowe, Chemische Reaktionstechnik mit MATLAB und SIMULINK, Wiley Verlag GmbH, Kapitel 3.1 Stochiometrische Gleichungen.

[5] Companion Website of the Book. Elsevier, 2021. Available from: https://www.elsevier.com/books-andjournals/book-companion/9780323913416.

CHAPTER

Chemical kinetics aided by Matlab/Simulink

3

3.1 Introduction

Chemical reactions are processes in which a substance or substances, known as reactants, are transformed into other substances, known as products (see References [1−3]). When this change directly occurs, a complete description of the reaction mechanism is straightforward. However, complex processes in which substances undergo a series of step-wise changes (each constituting a reaction in its own right) are much more common.

A simple case of the former reaction is a single-step irreversible reaction, in which products cannot be converted back to the original reactants. In this case, the rate of the forward reaction decreases until all the reactants have been consumed, and the reaction terminates.

On the contrary, in a single-step reversible reaction, the apparent rate of the forward reaction decreases together with the accumulation of the reaction products until the condition of dynamic equilibrium is finally established. At equilibrium, forward and backward reactions proceed at equal rates.

Time evolution of reactant and product concentrations requires an appropriate model of the reactions in terms of ordinary differential equations (ODE). Only simple linear equations (and in few cases also nonlinear equations) can be explicitly integrated so as to obtain an expression relating well-established thermodynamic parameters of the reactions (like activation energy, equilibrium constants) to time-varying concentrations. The alternative and generic approach is the numerical integration by means of appropriate ODE solvers. Matlab® offers a rich set of ODE solvers for integrating complex ODE [4]. Matlab/Simulink (see Appendix D) does the same job, by converting differential equations into block diagrams of the state space equations together with their feedback loops, and integrates them with the available Matlab ODE solvers, with the possibility of observing the graphical profile of the solution during its temporal progress.

Approximate solutions can often be found by using some simplifying modeling assumptions. In this way, the differential model can be used as a basis to describe the process. Quite often, with a tiny increase in programming efforts and further refinement, approximations can be eliminated, and considerable improvements can be achieved.

In kinetics, the parameters of interest are the quantities of reactants and products and their rate of change. Starting with a single-irreversible reaction, expressions for reactants are given a negative sign as the reactants are used up during a reaction. Product amounts increase, and their rate of change is therefore positive. As they are not constants, the rates must be written as differentials. Thus, in terms of a general reaction,

$$aA + bB + \ldots \Rightarrow cC + dD + \ldots, \tag{3.1}$$

Fundamental Chemistry with Matlab. DOI: https://doi.org/10.1016/B978-0-323-91341-6.00010-0

the *reaction rates* $r_k, k = 1, 2, \ldots$, for the individual components are:

$$r_1 = -\frac{1}{a}\frac{d[A]}{dt}, \quad r_2 = -\frac{1}{b}\frac{d[B]}{dt}, \quad r_3 = \frac{1}{c}\frac{d[C]}{dt}, \quad r_4 = \frac{1}{d}\frac{d[D]}{dt} \tag{3.2}$$

The coefficients $a, b, c, d, \ldots$ are the *stoichiometric coefficients*, which are necessary to balance reactants and products. Their inverse divides the first-order derivative of the concentrations, so that $r_1 = r_2 = r_3 = r_4 = R$, R being the *overall rate of the reaction*.

The square brackets denote concentrations in [mol/L]. Instead, in seawater calculations (see Chapter 9), concentrations are given in [mol/kg], thereby avoiding variations due to increase in water pressure and a resulting decrease in volume (in the depth of the oceans).

In addition, unless otherwise indicated, only closed, homogeneous systems are considered, in which there is no gain or loss of material during the reaction. Reactions are considered to proceed isothermally, so that temperature can be treated as an independent variable.

The rate R of a reaction at a fixed temperature is proportional to the concentration of reactants as can be seen in the following identities

$$R = r_1 = r_2 = r_3 = r_4 = k[A]^{\alpha}[B]^{\beta} \tag{3.3}$$

The proportionality constant k in Eq. (3.3) is known as the *rate constant*. The k unit can be deduced by the examination of the rate expression. It can be written as

$$\text{unit} = \frac{1}{(\text{mol/L})^{m-1}\text{s}}, m = \alpha + \beta + \cdots \tag{3.4}$$

where m, the sum of all the exponents of the concentrations, defines the order of reaction, whereas α is the order of reaction with respect to A, and β is the order of reaction with respect to B, and so on. The reader must pay attention that the term *order* also applies to differential equations and specifically to their versions in terms of *state equations* (see Appendix B), where order refers to the number of the state variables and equations. The order of reactions and of their differential equations is generically different.

When $n = 1$, the reaction is known as *first order*, and the inverse $\tau = 1/k$ is a time interval which is known in the *dynamic system theory* (see Appendix B) as *time constant*. In the case of irreversible reactions, it denotes the time interval in which the unitary initial concentration of the reactant reduces to $exp(-1) = 0.369$ before converging to zero steady state. The concept can be extended to nonzero steady states (reversible reactions) and complex reactions by means of the eigenvalues of the state matrix in the case of linear time invariant systems (see Appendix B). In Physics and Chemistry, time constants are also known as *relaxation times*. It means that the decaying quantities become negligible (with an error less than 1% of the initial concentration) about 5τ after the reaction starting time, and the product quantities reach the steady state in the same interval, with the same percentage error, but referred to the difference between initial and final concentration.

In the case of an irreversible reaction, the reaction order for each reacting compound should be determined experimentally since it cannot be predicted from the equations describing the reaction. The exponents may be positive integers (as it is usual for simple reactions) or fractions (when a reaction requires intermediate steps). Apart from simple reactions, they do not have to be equal to the stoichiometric coefficients of the reactant in the net reaction. We shall see that this does not apply to reversible reactions, where one must also consider the inverse reaction from reactants to products and the corresponding kinetic laws. However, when a reversible reaction reaches equilibrium, the stoichiometric coefficients can be inserted directly in the expression for the mass-conservation law (see Chapters 5, 7, and 9 and the relevant thermodynamic explanations).

3.2 First-order irreversible kinetics

3.2.1 State equation construction

The first-order irreversible reaction, where $m = 1$ denotes the order, is represented by the following formula:

$$A \to B(\text{rate constant} = k). \tag{3.5}$$

The rate of a first-order reaction is proportional to the concentration of the single reactant through the *rate constant* $k\left[\mathrm{s}^{-1}\right]$. The corresponding time constant is $\tau = 1/k[\mathrm{s}]$, with unit in seconds.

The conversion of Eq. (3.5) into differential equation assumes that the amount $d[A(t)]$, which undergoes chemical change in the short time interval $t, t + dt$, only depends on the concentration of A at time t, if one assumes no change in volume, temperature, or any other factors that could affect the reaction rate. Thus the rate expression which describes the reaction is

$$d[A(t)]/dt = -k[A], [A(t)](0) = [A]_0, \tag{3.6}$$

where $[A]_0$ is the initial concentration. As it can be found in chemistry textbooks, Eq. (3.6) is a first-order differential equation which can be integrated in time given the initial condition $[A]_0$. Let us remark that the order $n >= 1$ of the differential equation (here always converted to become state space equations) does not coincide with the order $m = 1$ of the reaction. In fact, what about the product concentration $[B]$? Irreversible reaction implies that the product concentration rate is the opposite of the reactant, that is

$$d[B(t)] = -d[A(t)] \Rightarrow d[B(t) + [A(t)]] = dW(t) = 0, \tag{3.7}$$

where $W(t)$ is the total concentration, which remains constant. Thus we must express this *conservation property* with another differential equation provided by (3.7), that is

$$dW(t)/dt = 0, \quad W(0) = [A]_0 + [B]_0. \tag{3.8}$$

Integration of Eq. (3.8) is immediate and provides the expected *conservation relation:*

$$W(t) = W(0) = [A]_0 + [B]_0, \quad t \geq 0 \Rightarrow [B(t)] = [B]_0 + [A]_0 - [A(t)]. \tag{3.9}$$

The pair of first-order differential Eqs. (3.6) and (3.8) prove that the underlying *dynamic system* is second order, namely $n = 2$, since it describes the time evolution of two concentrations. This result is well formulated by the method of the *state space equations* (not to be confused with thermodynamic state equations) which possess a rich volume of theory and applications for studying and solving state equations of any kind (see Appendix B). The previous second-order state equation ($n = 2$, linear and time invariant) can be written in matrix form as follows:

$$\begin{gathered} \boldsymbol{dx}(t)/dt = \boldsymbol{F}\boldsymbol{x}(t), \quad \boldsymbol{x}(0) = \boldsymbol{x}_0 \\ \boldsymbol{x} = \begin{bmatrix} [A] \\ W = [B] + [A] \end{bmatrix}, \quad F = \begin{bmatrix} -k & 0 \\ 0 & 0 \end{bmatrix}. \end{gathered} \tag{3.10}$$

The Matlab graphical application Simulink allows to graphically represent Eq. (3.10) and integrate the equation in time starting from the given initial conditions. Before doing this, we manually solve Eq. (3.10), which can be done separately since the matrix F being diagonal, tells us that the pair of equations can be independently integrated.

The second-row equation has been already integrated in Eq. (3.9). The first equation, corresponding to Eq. (3.6), has the well-known decaying solution $[A](t) = \exp(-kt)[A]_0$. Combination

with Eq. (3.9) provides the complete solution of the irreversible first-order kinetics Eq. (3.5):

$$\begin{matrix}[A](t) = \exp(-kt)[A]_0 \\ [B](t) = [B]_0 + (1-\exp(-kt))[A]_0\end{matrix}. \tag{3.11}$$

The solution, as expected, has the following properties.

1. The reactant concentration tends to zero, namely $\lim_{t\to\infty}[A](t)\to 0$.
2. The product concentration tends to the initial total concentration, namely $\lim_{t\to\infty}[B](t)\to[A]_0+[B]_0$.

3.2.2 Simulink graphical representation

The key element of state space equations (briefly state equations) is the pure integrator describing the elementary differential equation $dx/dt = u(t), x(0) = x_0$, where $u(t)$ is an input which in the first equation of Eq. (3.10) holds $u(t) = -k[A](t)$ and in the second one is zero.

The block diagram in Fig. 3.1 splits into three parts. The top diagram is the higher-level diagram in which any single-state equation is represented by a block: reactant $[A]$ and conservation blocks. The upper part of the diagram shows the clock, the time display, and the block `out.t` for recording time. The left bottom diagram details the first-order equation of the reactant. The right bottom diagram details the conservation equation.

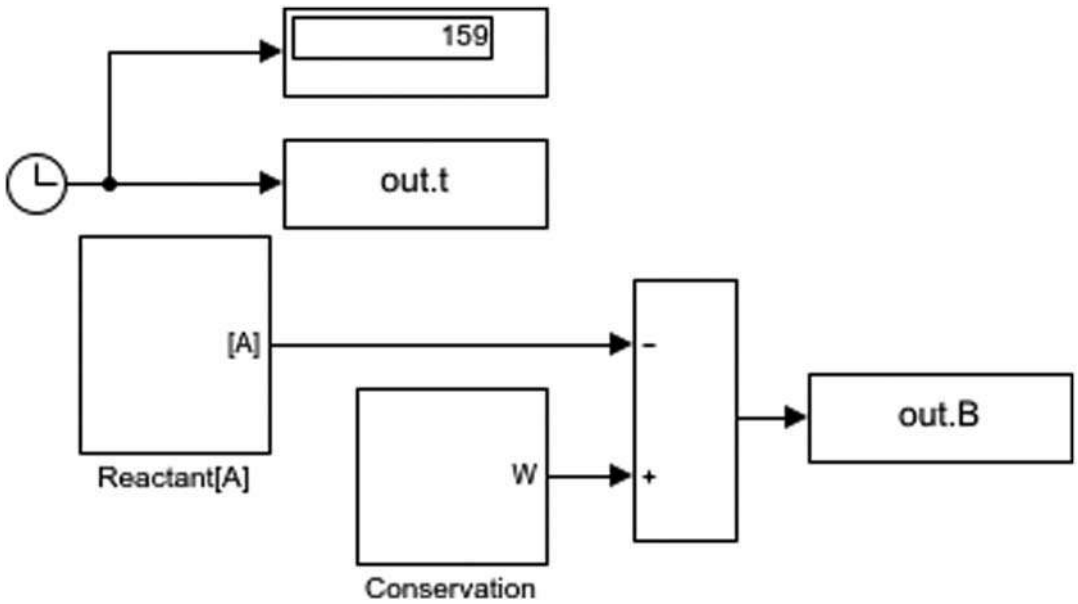


FIGURE 3.1

(Top) Simulink block diagram of the state space Eq. (3.10). (Left bottom) The first-order equation of the reactant $[A]$. (Right bottom) Conservation equation.

The block diagram in Fig. 3.1 includes two integrators denoted by $1/s$ (the Laplace transform of integration in time, see Appendix B) driven by initial conditions. The left bottom integrator of the reactant $[A]$ receives a negative feedback. The external input to both integrators is set equal to zero. Time t and the

output concentrations [*A*] and [*B*] are recorded by boxes `out.t`, `out.A`, and `out.B`. The `save format` of the data boxes has been set to `Array`. The Simulink block diagram must be accompanied by a Matlab script fixing parameter values and initial conditions, calling the ODE solver and graphically plotting output data. The type of ODE solver, here the fixed-step `ode4` (`Runge Kutta`, see Appendix D), and its parameters like initial and final time (`tInit` and `tEnd` in the script) and the sampling time `Ts` can be specified in different ways: for instance, through the `Configuration pane` which is opened by the button `Model Settings` in the Simulink bar (see Appendix D).

3.2.3 The Matlab script

The initial script `InitChem` can be found in Appendix D.

```
%% Chemistry with Matlab - Kinetics
%% Irreversible first Order Kinetics
%% 1) Initialization
InitChem;
%% 2) Parameters
% Kinetic rate
k=0.06;
tau=1/k; % Time constant
% Initial conditions
A0=0.05; % mol/dL reactant
B0=0.025; % Product
W0=A0+B0; % Total
%% 3) Timing and simulation
Ts=1; % Ts<< tau  sampling time
tInit=0; % Init time
tEnd=10*tau; % End time
inttEnd=floor(tEnd/Ts); % Integer multiple of Ts
tEnd=inttEnd*Ts;
out=sim('IrrevFirstOrderSim');
%% 4) Plotting time profiles
t=out.t;A=out.A;B=out.B;
InitTan=A(1)*(1-t*k); %Initial  tangent
dimt=length(t);
W=ones(dimt,1)*W0;As=zeros(dimt,1);  % Steady state
IndShort=find(t<=tau*1.1);  % Tangent interval
figure('name','Output');hold on; grid on;
plot(t,A,'b');plot(t,As,'b--');plot(t,B,'r');
plot(t,W,'k--');
plot(t(IndShort),InitTan(IndShort),'g--');
xlabel('Time [s]');ylabel('Concentration [mol/L]');
title('Irreversible first-order kinetics');
legend('Reactant: [A] ','[A] steady state','Product: [B]', ...
    'Total Concentration','Initial tangent','FontSize',18, ...
    'Location','best');
text(tau/3,-0.005,'$\tau$','FontSize',18);
```

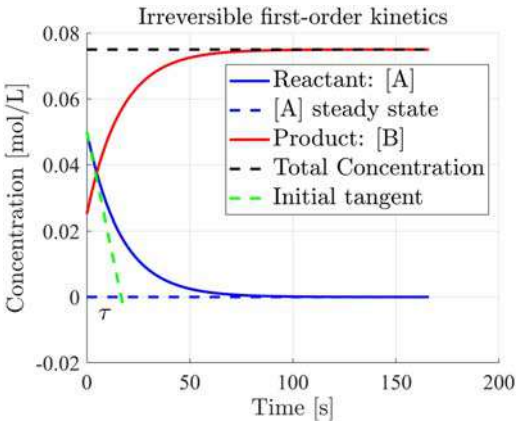


FIGURE 3.2

Time profiles of reactant and product of an irreversible first-order kinetics.

Fig. 3.2 shows the time profiles obtained by the previous script. The reactant profile converges to zero (the black dashed line), whereas the product profile converges to the total concentration indicated by the top horizontal black dashed line. The time constant is equal to $\tau = 1/k = 16.6$ s and can be estimated/checked in the plotted profiles by the intersection of the tangent to the initial profile of the reactant (the green dashed line) and the zero steady state, as in Fig. 3.2. It is left to the reader to write the initial tangent expression.

3.2.4 A simpler alternative Matlab script

Simulink block diagrams and the relevant script may be avoided by directly integrating the differential equation of the reactant [*A*] as in the following Matlab script, through the variable-step `ode45()` solver (see Appendix D).

```
InitChem;
k = 0.06; % Rate constant of reaction
timespan = (0:30);
A0 = 0.05;
first = @(t,A) - k*A;
[t,A_calc] = ode45(first,timespan,A0);
plot(t,A_calc,'LineWidth',2);grid on
xlabel('Time, arbitrary units')
ylabel('[A] concentration')
title('First order irreversible reaction')
```

3.3 First-order reversible reaction

3.3.1 State equation construction

The chemical reaction is the following:

$$\begin{array}{c} A \Leftrightarrow B, \text{ reversible reaction} \\ A \Rightarrow B, k_1 \\ A \Leftarrow B, k_2 \end{array} \tag{3.12}$$

Two state equations must be written, one in the reactant and one in the product, since the reaction is reversible:

$$\begin{array}{c} d[A]/dt = -k_1[A] + k_2[B], [A](0) = [A]_0 \\ d[B]/dt = k_1[A] - k_2[B], [B](0) = [B]_0 \end{array}. \tag{3.13}$$

Eq. (3.13) can be simplified by replacing the second equation with a conservation equation, where $W = [A] + [B]$ is conserved, and by rearranging the first equation as follows:

$$\begin{array}{c} d[A]/dt = -k[A] + k_2 W, \quad [A](0) = [A]_0, \quad k = k_1 + k_2 \\ dW/dt = 0, \quad W(0) = [A]_0 + [B]_0 \end{array}. \tag{3.14}$$

The first-state equation shows that the time constant holds $\tau = (k_1 + k_2)^{-1}$. In other terms, the reactant rate constant $k = k_1 + k_2$ is the sum of the direct and reverse constants.

We want to solve the first equation which is linear, but unlike Eq. (3.6), is forced by the constant input $W = W_0$. The expression of $[A]$ is still in closed form, but is the sum of two terms:

1. The *free response* $[A]_{\text{free}} = \exp(-kt)[A]_0$, already encountered in Eq. (3.11) but with a different parameter in the exponent, decays to zero since $k > 0$.
2. The *forced response* is expressed through the *convolution integral*

$$[A]_{\text{forced}} = k_2 \int_0^t \exp(-k(t-\tau)) W_0 d\tau = \frac{k_2}{k}(1 - \exp(-kt)) W_0. \tag{3.15}$$

The convolution integral (see Appendix B) is just the sum of infinitesimal free responses which start at the current time τ and end at t. They are *driven* by the infinitesimal initial condition $W_0 d\tau$, which, in this case, is imposed by the constant input value W_0. The sum of free and forced responses provides—the reader is asked to prove it—the total response of reactant and product:

$$\begin{array}{l} [A] = \dfrac{1}{k}\exp(-kt)\left(k_1[A]_0 - k_2[B]_0\right) + \dfrac{k_2}{k}\left([A]_0 + [B]_0\right) \\ [B] = -\dfrac{1}{k}\exp(-kt)\left(k_1[A]_0 - k_2[B]_0\right) + \dfrac{k_1}{k}\left([A]_0 + [B]_0\right) \end{array}. \tag{3.16}$$

For $t \to \infty$, we obtain the steady-state concentrations

$$\begin{array}{l} [A]_\infty = \dfrac{k_2}{k}\left([A]_0 + [B]_0\right) \\ [B]_\infty = \dfrac{k_1}{k}\left([A]_0 + [B]_0\right) \end{array}, \tag{3.17}$$

whose sum equals the conserved total concentration W.

3.3.2 Simulink graphical representation and results

The Simulink block diagram in Fig. 3.3 repeats Eq. (3.14). Unlike Fig. 3.1, the reactant diagram receives as input the total concentration W multiplied by the gain k_2.

The Matlab script, not reported here, is similar to the script in Section 3.2.3. The time profiles of the reactant $[A]$ and product $[B]$ are shown in Fig. 3.4. Unlike the irreversible case in Fig. 3.2, where the product concentration $[B]$ converges to the total concentration, here the product converges to the reduced concentration $[B]_\infty$ in Eq. (3.17), since $[B]$ in turn acts as the reactant of the reverse reaction. The total concentration W is conserved, being the sum of the steady-state concentrations. The time constant is equal to $\tau = (0.004+0.008)^{-1} = 83.3$ s and can be estimated/checked in the graphical profiles by the intersection of the tangent to the initial reactant profile and the nonzero steady-state dashed line. The reader is asked to prove that the reactant initial tangent holds

$$A_{init}(t) = [A_0](1 - tk_1) + tk_2[B_0], \quad t \geq 0. \tag{3.18}$$

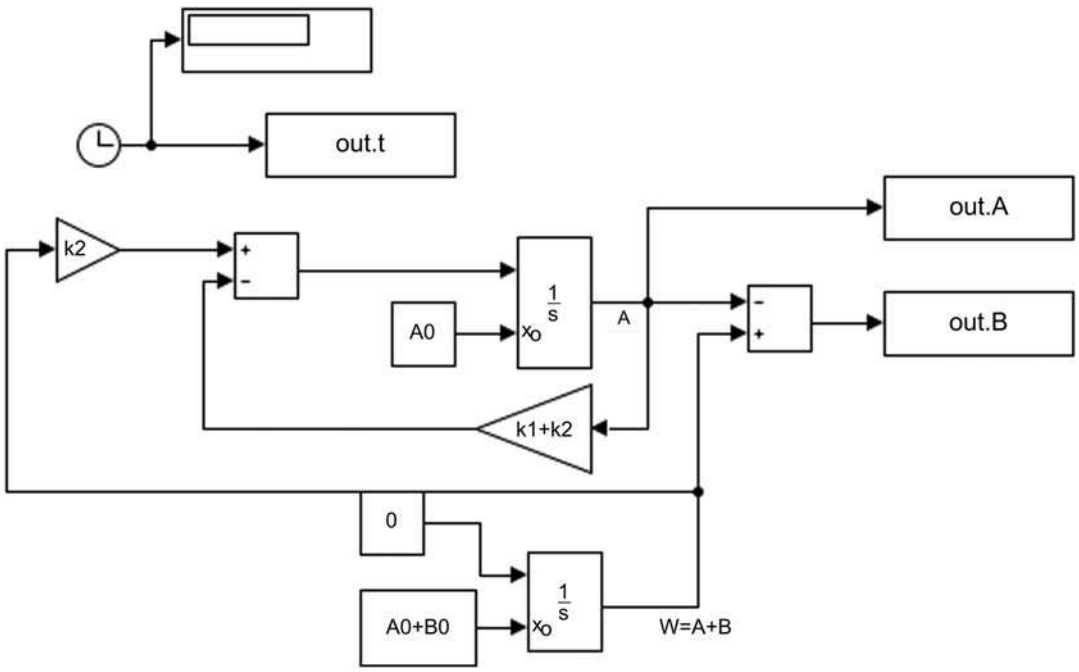


FIGURE 3.3

Block diagram of first-order reversible kinetics.

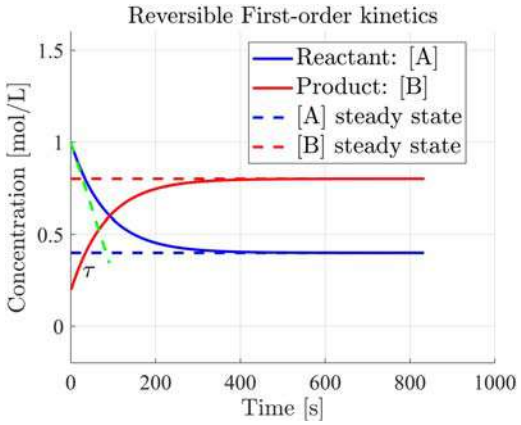


FIGURE 3.4

Time profiles of reactant and product of a first-order reversible reaction.

3.3.3 Alternative Matlab script

The alternative Matlab script to Simulink block diagram, calling the solver `ode45()`, is the following.

```
%% Reversible Reactions
% A <==> B where A ==>B ,k1, B ==> A  ,k2
InitChem;
% Define initial concentrations.
C0 = [1,0.2];
% Define time span.
tspan = [0,480];
% Run ODE solver.
[t, y] = ode45(@reversible_A_B, tspan, C0);
% Plot
c1 = y(:,1);c2 = y(:,2);
plot(t/60,c1,'b',t/60,c2,'r','LineWidth',2);grid on;
xlabel('time arbitrary unit');
ylabel('concentration');legend('[A]','[B]');
%% Function
function dC = reversible_A_B(t, C)
    Rate constants.
   k1 = 0.008;
   k2 = 0.004; % therefore Keq = k1/k2 = 2
   % Rate laws.
   r1 = k1*C(1); % C(1) = [A]
   r2 = k2*C(2); % C(2) = [B]
   % Mass balances.
   dCA  = -r1 + r2;
   dCB = - r2 + r1;
   % Assign output variables.
   dC(1,:) = dCA;
   dC(2,:) = dCB;
end
```

3.4 Second-order reversible reaction

3.4.1 State equations

The chemical reaction is the following:

$$\begin{array}{c} A+B \Leftrightarrow C \\ A+B \Rightarrow C, k_1, \text{direct reaction} \\ A+B \Leftarrow C, k_2, \text{inverse reaction} \end{array}. \tag{3.19}$$

The first form of state equations is found to be:

$$\begin{array}{ll} d[A]/dt = -k_1[A][B]+k_2[C], & [A](0)=[A]_0 \\ d[B]/dt = -k_1[A][B]+k_2[C], & [B](0)=[B]_0 \\ d[C]/dt = -k_2[C]+k_1[A][B], & [C](0)=[C]_0 \end{array} \tag{3.20}$$

Eq. (3.20) is *nonlinear* due to the term $k_1[A][B]$. As in sections 3.2.1 and 3.3.1, we aim to replace part of equations with one or more conservation equations. It is immediate to check that the following algebraic sums are equal to zero

$$\begin{array}{l} \left(d[A]/dt - d[B]/dt\right)/2 = 0 \\ \left(d[A]/dt + d[B]/dt\right)/2 + d[C]/dt = 0 \end{array}, \tag{3.21}$$

which sums imply two conservation equations:

$$\begin{array}{l} dD/dt = 0, D_0 = \left([A]_0 - [B]_0\right)/2 \\ dW/dt = 0, W_0 = \left([A]_0 + [B]_0\right)/2 + [C]_0 \end{array}. \tag{3.22}$$

Concentrations which are conserved have been denoted by $W=([A]+[B])/2+[C]$ and $D=([A]-[B])/2$. We define also the mean reactant concentration $S=([A]+[B])/2$ and the relationships $[A]=S+D$ and $[B]=S-D$. The reader is asked to prove the following state equations which replace Eq. (3.20):

$$\begin{array}{c} dS(t)/dt = -k_1\left(S^2-D^2\right)-k_2S+k_2W \\ dD/dt=0, \quad D(0)=D_0 \\ dW/dt=0, \quad W(0)=W_0 \end{array} \Leftrightarrow \begin{array}{l} dS(t)/dt = -k_1S^2-k_2S+U_0, S(0)=S_0 \\ \\ \end{array}, \tag{3.23}$$

where $D(t)=D_0, W(t)=W_0$. As a result, the original Eq. (3.20) has been simplified into the first-order quadratic equation

$$dS(t)/dt = -k_1S^2-k_2S+U_0, \quad S(0)=S_0, \tag{3.24}$$

in terms of the state variable S and driven by the constant term

$$U_0 = k_1D^2{}_0 + k_2W_0. \tag{3.25}$$

The original concentrations, acting as the output variables of the state equation, hold

$$[A]=S+D, \quad [B]=S-D, \quad [C]=W-S. \tag{3.26}$$

Eq. (3.24) is known as a *Riccati equation with constant coefficients* and possesses a closed form solution to be found in Section 3.4.2.

The steady state (or equilibrium) concentrations are obtained from the equilibrium equations, by setting to zero the concentration time derivatives on the left side of Eq. (3.23):

$$S_\infty^2 + \frac{k_2}{k_1} S_\infty - \frac{U_0}{k_1} = 0 \Leftrightarrow S_\infty = 0.5 \frac{k_2}{k_1} \left(-1 + \sqrt{1 + \frac{4U_0 k_1}{k_2^2}} \right) \simeq 0.493 \text{mol/L}$$
$$[C]_\infty = W_0 - S_\infty = 0.407 \text{mol/L}$$
$$[A]_\infty = S_\infty + D_0 = 0.693 \text{mol/L} \quad . \tag{3.27}$$
$$[B]_\infty = S_\infty - D_0 = 0.293 \text{mol/L}$$

The Simulink block diagram in Fig. 3.5 repeats the second form in Eq. (3.23). Let us remark that the diagram of S shows two negative feedback links and an input which is combination of the two conservation equations. The latter ones are described by pure integrators without feedback.

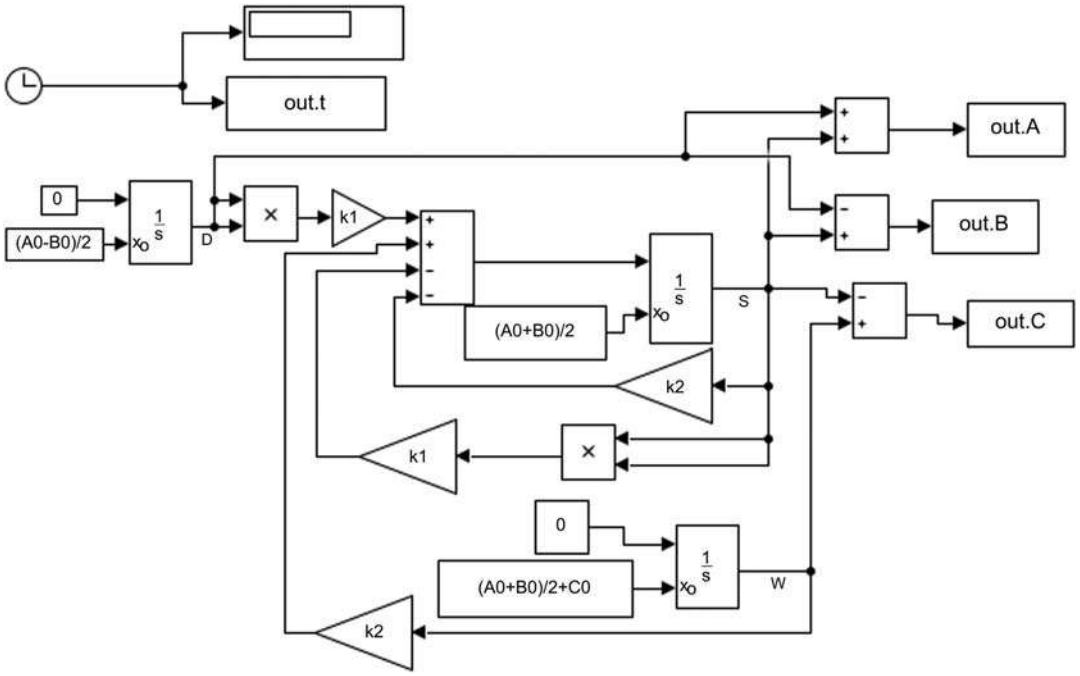


FIGURE 3.5

Block diagram of second-order reversible kinetics.

Fig. 3.6 shows the time profiles and the steady-state values of reactants and product. Due to the nonlinearity of Eq. (3.23), it is of interest to find an approximate time constant which justifies decay and rise times of concentrations in Fig. 3.6. This can be done by approximating the *quadratic equation* in Eq. (3.23) with a *linear perturbation equation* (see Appendix B) around a steady-state value, when the latter is not too far from the initial value. To this end, we define the *perturbation* $\delta S = S - S_\infty$, which implies $d\delta S/dt = dS/dt$ since S_∞ is constant. The right-side term is expanded

into a Taylor series up to the first-order term (the order zero term goes to zero since it defines the steady state), which provides the linear equation

$$\begin{array}{c} d\delta S/dt = -k_1 S_\infty^2 - 2k_1 S_\infty \delta S - k_2 S_\infty - k_2 \delta S + U_0 \Leftrightarrow \\ \Leftrightarrow d\delta S/dt = -(2k_1 S_\infty + k_2)\delta S, \quad \delta S(0) = \delta S_0 \end{array}, \tag{3.28}$$

and the time constant

$$\tau = (2k_1 S_\infty + k_2)^{-1} = 84.1 \text{ s}. \tag{3.29}$$

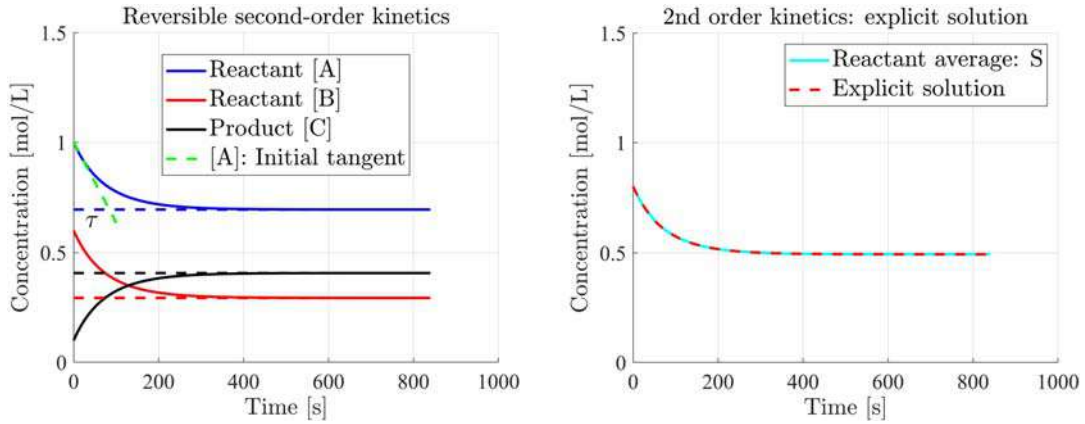


FIGURE 3.6

(Left) Time profiles of reactants and product of a second-order reversible reaction. (Right) Check of the explicit solution.

The same value can be estimated from the Fig. 3.6, by plotting the initial tangent of the reactant $[A]$ (the green line), whose expression the reader is asked to prove:

$$A_{Init}(t) = [A]_0 - kt(S_0 - S_\infty). \tag{3.30}$$

The same time constant will be found in the exact solution (3.32) of the Riccati equation to be derived in section 3.4.2.

3.4.2 Solution of the Riccati equation

The solution assumes the inequality $S_0 > S_\infty$, which follows from $S = 0.5([A] + [B])$ (the reactant average). The explicit solution requires the rate constant $k = k_2\sqrt{(1 + 4U_0 k_1/k_2^2)} \simeq 0.019 \quad \text{s}^{-1}$, which is the inverse of the time constant τ in (3.29), and the constants $\overline{S}$ and s_0 defined by

$$\overline{S} = \frac{k_2}{2k_1}\left(1 + \sqrt{1 + \frac{4U_0 k_1}{k_2^2}}\right), \; s_0 = \frac{S_0 - S_\infty}{S_0 + \overline{S}}. \tag{3.31}$$

Separation of variables in Eq. (3.24) and integration of the left and right sides, lead to the following expression

$$\int_{S_0}^{S(t)} \frac{dS}{k_1 S^2 + k_2 S - U_0} = -(t - t_0) \Rightarrow S(t) = \frac{S_\infty + \bar{S} s_0 exp(-k(t - t_0))}{1 - s_0 exp(-k(t - t_0))}, \tag{3.32}$$

which provides the same time profile of the Simulink integration, as shown by Fig. 3.6, right.

3.4.3 Alternative Matlab script

The following Matlab script, which calls the solver `ode45()`, is alternative to the Simulink block diagram of Fig. 3.6.

```
%% Second order Reversible Reactions
% A + B <==> C, k1, direct reaction A + B ==> C,k2, inverse A + B <== C
InitChem;
% Defines initial concentrations
C0 = [1,0.6,0.1]; % Defines initial concentrations.
tspan = [0,480]; % Defines time span.
% Run ODE solver.
[t, y] = ode45(@reversible_A_B_C, tspan, C0);
% Plot
c1 = y(:,1);c2 = y(:,2);c3 = y(:,3);
plot(t/60,c1,'b',t/60,c2,'r',t/60,c3,'g','LineWidth',2);grid on;
xlabel('Time,arbitrary unit');ylabel('Concentration');legend('[A]','[B]','[C]');
title('$A + B <==> C$ reversible reaction');
%% Function
function dC = reversible_A_B_C(t, C)
    % Rate constants.
    k1 = 0.008;
    k2 = 0.004; % Therefore Keq = k1/k2 = 2
    % Rate laws.
    r1 = k1*C(1)*C(2); % C(1) = [A]    C(2) = [B]
    r2 = k2*C(3);      % C(3) = [C]
    % Mass balances.
    dCA = -r1 + r2;
    dCB = -r1 + r2;
    dCC = -r2 + r1;
    % Assign output variables.
    dC(1,:) = dCA;
    dC(2,:) = dCB;
    dC(3,:) = dCC;
end
```

3.5 Consecutive irreversible reactions

3.5.1 State equations

The chemical reaction is the following:

$$\begin{array}{l} A \Rightarrow B, k_1 \text{ direct reaction} \\ B \Rightarrow C, k_2 \text{ direct reaction} \\ C \Rightarrow D, k_3 \text{ direct reaction} \end{array}. \tag{3.33}$$

The state equations have the following expression:

$$\begin{array}{l} d[A]/dt = -k_1[A], \quad [A](0) = [A]_0 \\ d[B]/dt = k_1[A] - k_2[B], \quad [B](0) = [B]_0 \\ d[C]/dt = k_2[B] - k_3[C], \quad [C](0) = [C]_0 \\ d[D]/dt = k_3[C], \quad [D](0) = [D]_0 \end{array}. \tag{3.34}$$

The unique conservation equation is obtained by replacing the fourth equation with

$$dW/dt = 0, W_0 = [A]_0 + [B]_0 + [C]_0 + [D]_0, \tag{3.35}$$

where $W = [A] + [B] + [C] + [D]$ remains constant. The reader is asked to rewrite Eq. (3.34) in a matrix form (see Appendix B) and to find the free response to the initial conditions $[A]_0 = 1\text{M}, [B]_0 = [C]_0 = [D]_0 = 0$, with the help of symbolic Matlab. The state matrix A is lower triangular, implying that each concentration only depends on the upstairs reactants (see also the block diagram in Fig. 3.7).

The Simulink block diagram in Fig. 3.7 shows that the blocks of the first-order kinetics are connected in cascade as implied by an irreversible consecutive equation. Each block contains a first-order state equation defined by integrator and negative feedback.

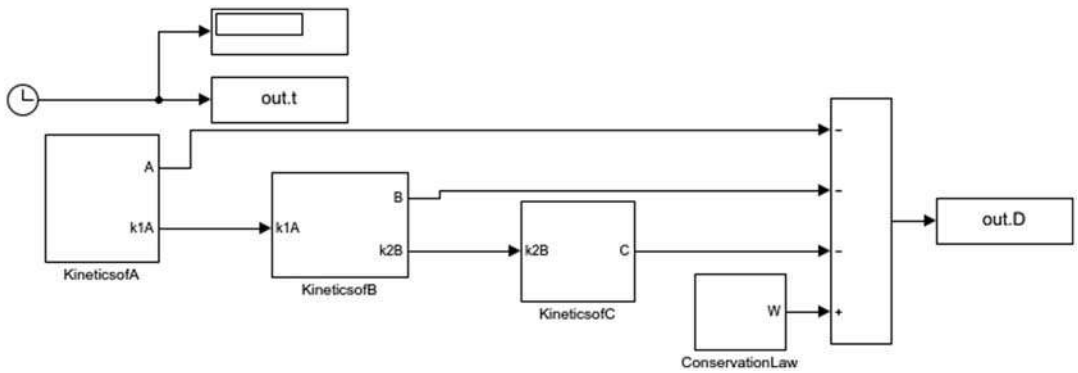


FIGURE 3.7

Block diagram of irreversible consecutive kinetics.

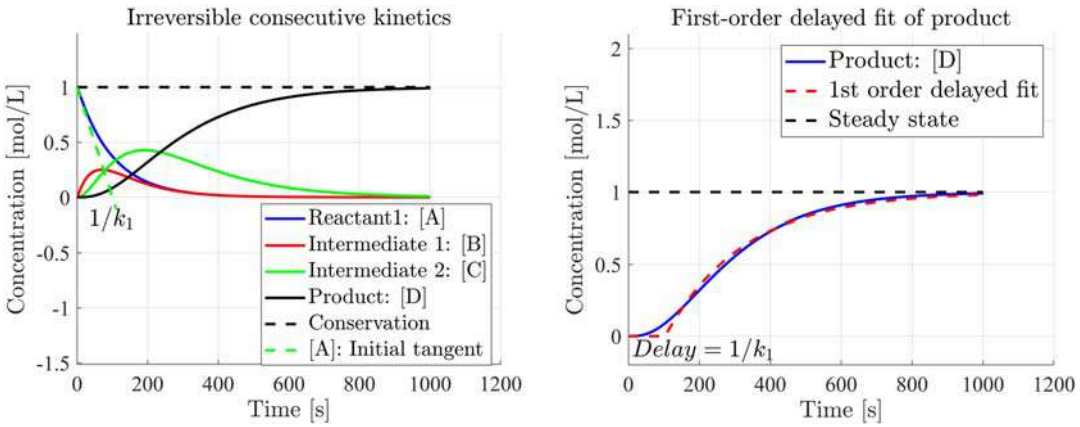


FIGURE 3.8

(Left) Time profile of reactant, intermediate substances, and product of an irreversible consecutive reaction. (Right) Product profile and first-order delayed approximation.

Fig. 3.8, left, shows the time profiles of the reactant, intermediate substance, and product concentrations. The reactant $[A]$ decays with the time constant $\tau_1 = 1/k_1 = 100\text{s}$, which can be estimated/checked by the intersection of the initial tangent (the green line) with the zero-ordinate line. The first intermediate substance $[B]$ grows with the time constant $\tau_2 = 1/k_2 = 50\text{s}$ and then decays with the time constant τ_1. The product $[D]$ increases to the total concentration W_0 with the time constant $\tau_D = 225\text{s} \simeq \left(1/k_2 + 1/k_3\right)$, but after a delay of the order of $\tau_1 = 100\text{s}$. The delay expresses the fact that the last reaction waits for the production of intermediate substances. Fig. 3.8, right, compares the simulated profile of the product $[D]$ with a delayed first-order response which has been obtained by fitting the following expression

$$[D]_{\text{Fit}}(t) = \begin{cases} 0, t < \tau_1 \\ [D]_\infty \left(1 - exp\left(-\dfrac{t-\tau_1}{\tau_D}\right)\right), t \geq \tau_1 \end{cases} \tag{3.36}$$

with the help of the Matlab function `fminsearch`.

3.5.2 Alternative Matlab script

The following Matlab script, which calls the solver `ode45()`, is alternative to the Simulink block diagram in Fig. 3.7.

```
% Consecutive Irreversible Reactions
% A ==> B,k1      B ==> C,k2     C ==> D,k3
InitChem;
% Defines initial concentrations
C0 = [1,0,0];
% Defines time span.
tspan = [0,480];
% Run ODE solver.
[t, y] = ode45(@consecutive_A_B_C, tspan, C0);
% Plotting
c1 = y(:,1);c2 = y(:,2);c3 = y(:,3);
plot(t/60,c1,'b',t/60,c2,'r',t/60,c3,'g',t/60,1-c1-c2-c3,'c');grid on;
xlabel('time, arbitrary units');ylabel('[A] concentration')
title('A => B => C => D consecutive reactions')
function dC = consecutive_A_B_C(t, C)
    % Rate constants.
    k1 = 0.01;
    k2 = 0.02;
    k3 = 0.006;
    % Rate laws.
    r1 = k1*C(1); % C(1) = [A]
    r2 = k2*C(2); % C(2) = [B]
    r3 = k3*C(3); % C(3) = [C]
    % Mass balances.
    dCA = -r1;
    dCB = r1 - r2;
    dCC = r2 - r3;
    % Assigns output variables.
    dC(1,:) = dCA;     dC(2,:) = dCB;     dC(3,:) = dCC;
end
```

3.6 Two-stage NO to NO_2 oxidation

3.6.1 State equations

The chemical reaction is the following:

$$\begin{gathered} 2\mathrm{NO(g)} \Leftrightarrow \mathrm{N_2O_2(g)}, k_1 \text{ direct reaction, } k_2 \text{ reverse reaction} \\ \mathrm{N_2O_2(g)} + \mathrm{O_2(g)} \Rightarrow 2\mathrm{NO_2(g)}, k_3 \\ [A] = [\mathrm{NO}],\ [B] = [\mathrm{N_2O_2}],\ [C] = [\mathrm{NO_2}],\ [D] = [\mathrm{O_2}] \end{gathered} . \tag{3.37}$$

The state equations are found to be:

$$\begin{gathered} d[A]/dt = -2k_1[A]^2 + 2k_2[B], \quad [A](0) = [A]_0 \\ d[B]/dt = k_1[A]^2 - k_2[B] - k_3[B][D], \quad [B](0) = [B]_0 \\ d[C]/dt = 2k_3[B][D], \quad [C](0) = [C]_0 \\ d[D]/dt = 0, \quad [D](0) = [D]_0 \end{gathered} , \tag{3.38}$$

where the molecular oxygen concentration $[D]$ is assumed to remain constant (first-conservation equation). The second-conservation equation is given by the sum of the other concentrations

$W = [A] + 2[B] + [C]$, which implies the identities:

$$[C] = W - ([A] + 2[B]), \quad dW/dt = 0, \quad W(0) = W_0. \tag{3.39}$$

Eq. (3.38) is now rewritten in terms of $[D]$, W, the sum $S = [A] + 2[B]$, and $[A]$. It is left to the reader to prove the following version of the original state equations:

$$\begin{array}{c} d[A]/dt = -2k_1[A]^2 - k_2[A] + k_2 S, \quad [A](0) = [A]_0 \\ dS/dt = -k_3 S[D] + k_3[A][D], \quad S(0) = S_0 \\ dW/dt = 0, \quad W(0) = W_0 \\ d[D]/dt = 0, \quad [D](0) = [D]_0 \end{array}, \tag{3.40}$$

where

$$k_1 = 0.08\left(\text{mol/L}\right)^{-1}\text{s}^{-1},\ k_2 = 0.04\ \text{s}^{-1},\ k_3[D] = 0.002\ \left(\text{mol/L}\right)\text{s}^{-1}. \tag{3.41}$$

We remark that the first equation is nonlinear (quadratic), but unlike the first equation in Eq. (3.23), is driven by the time-varying input $S(t)$.

Fig. 3.9 shows the time profiles of the reactant concentrations $[A]$ and $[B]$, which converge to zero, and of the product $[C]$ which reaches the total conserved concentration. The constant $[D]_0 = [O_2]$ acts as a scale factor of the rate constant $k_3 = 0.01\left((\text{mol/L}) \times \text{s}\right)^{-1}$ and is plotted with a dashed line. The left figure shows the long-term profiles which are dominated by the direct reaction time constant $\tau_3 = \left(k_3[D]_0\right)^{-1} = 500\text{s}$. The right figure zooms on the initial transient which is dominated by the reversible reaction time constant.

The latter constant can only be approximated since the relevant state equation is quadratic. We adopt the same method of Section 3.4.1, but the steady state concentration of $[A]$ in Eq. (3.40) cannot be taken as the steady state for $t \to \infty$, since the concentration converges to zero. We want to find a *short-term time constant* to be estimated in Fig. 3.9, right. Therefore, we compute the equilibrium of the *reversible equation*, but decoupled from the successive direct reaction (see the relevant dashed lines in Fig. 3.9, right). The short-term steady-state equations are as follows:

$$\begin{array}{c} A^2_{\text{ShortTerm}} + \dfrac{k_2}{2k_1}A_{\text{ShortTerm}} - \dfrac{k_2}{2k_1}S_0 = 0 \Leftrightarrow A_{\text{ShortTerm}} = 0.39\text{mol/L} \\ [B]_{\text{ShortTerm}} = (S_0 - A_{\text{ShortTerm}})/2 = 0.30\text{mol/L} \\ k_1 = 0.08\left((\text{mol/L}) \times \text{s}\right)^{-1}, k_2 = 0.04\text{s}^{-1}, S_0 = 1\text{mol/L} \end{array}. \tag{3.42}$$

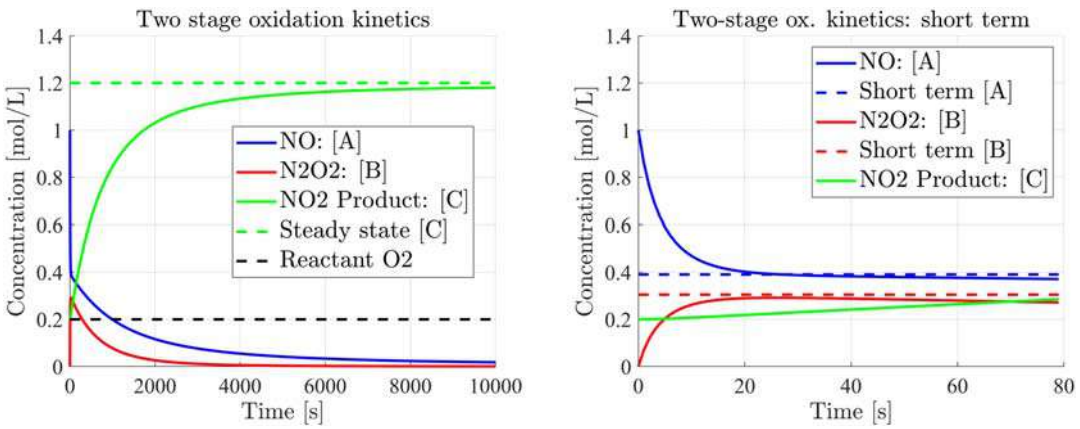


FIGURE 3.9

Time profiles of reactants and product of two-stage oxidation. (Left) Long-term plot. (Right) Short-term plot.

We convert the quadratic equation into a linear perturbation equation around the steady-state value $A_{\text{ShortTerm}}$, like in Eq. (3.28):

$$\frac{d\delta A(t)}{dt} = -(4k_1 A_{\text{ShortTerm}} + k_2)\delta A(t), \quad \delta A = A - A_{\text{ShortTerm}}\,. \tag{3.43}$$

The above equation allows us to find the approximated short-term time constant:

$$\tau_{\text{ShortTerm}} = (4k_1 A_{\text{ShortTerm}} + k_2)^{-1} = 6.1\text{s}, \tag{3.44}$$

which is very small with respect the long-term time constant. In Appendix B, Section B.3.1, the complete perturbed equation of Eq. (3.35) around the short-term steady state (equilibrium) will be derived as an example.

3.6.2 Alternative Matlab script

The following Matlab script, which calls the solver `ode45()`, is alternative to Simulink block diagram. The latter can be found in the companion website of the book [6].

```
%% Reversible reaction:2NO(g)<==>N2O2(g) k1=direct reaction k2 = reverse
%% Followed by irreversible:N2O2(g)+O2(g) ==>2NO2(g) k3=direct reaction
% Defines initial concentrations.
InitChem;
C0 = [1,0,0.2];
tspan = [0,480];   % Define time span.
[t, y] = ode45(@reversible_A_B, tspan, C0);   % Run ODE solver.
c1 = y(:,1);c2 = y(:,2);c3 = y(:,3);           % Plot
plot(t/60,c1,'b',t/60,c2,'r',t/60,c3,'g','LineWidth',2);grid on;
xlabel('time, minutes');ylabel('concentrations');
title('Nitric oxide ==> Nitrogen dioxide two steps reaction');
legend('[NO]','[N2O2]','[NO2]');
%% Function
function dC = reversible_A_B(t, C)
    % Rate constants.
    k1 = 0.08;
    k2 = 0.04;
    k3 = 0.01;C_O2 = 0.2;
    % Rate laws.
    r1 =k1*C(1)^2;     % C(1) = [NO]
    r2 = k2*C(2);       % C(2) = [N2O2]
    r3 = k3*C(2)*C_O2; % C(3) = [NO2]
    % Mass balances.
    dCNO  = -2*r1 + 2*r2;
    dCN2O2 = - r2 + r1 - r3;
    dCNO2 = 2*r3;
    % Assigns output variables.
    dC(1,:) = dCNO;
    dC(2,:) = dCN2O2;
    dC(3,:) = dCNO2;
end
```

3.7 Ozone decomposition into oxygen

3.7.1 State equations

The chemical reaction is the following:

$$\begin{array}{c} O_3(g) \Leftrightarrow O(g) + O_2(g),\ k_1 \text{ direct reaction},\ k_2 \text{ reverse reaction} \\ O(g) + O_3(g) \Rightarrow 2O_2(g),\ k_3 \text{ direct reaction} \\ [A] = [O_3],\ [B] = [O],\ [C] = [O_2] \text{ product},\ [D] = [O_2] \text{ reactant} \end{array} . \tag{3.45}$$

The state equations are as follows:

$$\begin{array}{c} d[A]/dt = -k_1[A] + k_2[B][D] - k_3[A][B], \quad [A](0) = [A]_0 \\ d[B]/dt = k_1[A] - k_2[B][D] - k_3[A][B], \quad [B](0) = [B]_0 \\ d[C]/dt = 2k_3[A][B], \quad [C](0) = [C]_0 \\ d[D]/dt = 0, \quad [D](0) = [D]_0 \end{array} . \tag{3.46}$$

The molecular oxygen concentration $[D]$ is assumed to remain constant when acting as a reactant (first conservation equation). The second conservation equation is given by the sum of the other concentrations $W = [A] + [B] + [C]$, which implies

$$[C] = W - ([A] + [B]), \quad dW/dt = 0, \quad W(0) = W_0. \tag{3.47}$$

Eq. (3.46) is rewritten in terms of $[D]$, W, the sum $S = ([A] + [B])/2$, and the difference $V = ([A] - [B])/2$. It is left to reader to prove the following version of the original state equations:

$$\begin{array}{c} dV/dt = -(k_1 + k_2[D])V + (-k_1 + k_2[D])S, \quad V(0) = V_0 \\ dS/dt = -k_3\left(S^2 - V^2\right), \quad S(0) = S_0 \\ dW/dt = 0, \quad W(0) = W_0 \\ d[D]/dt = 0, \quad [D](0) = [D]_0 \end{array} . \tag{3.48}$$

The parameter values are as follows

$$\begin{array}{c} k_1 = 0.08\ s^{-1},\ k_2 = 0.04\ (mol/L)^{-1}s^{-1},\ k_3 = 0.2\ (mol/L)^{-1}s^{-1} \\ [D]_0 = 0.2\ mol/L,\ [A]_0 = 0.1\ mol/L,\ [B]_0 = [C]_0 = 0 \end{array} . \tag{3.49}$$

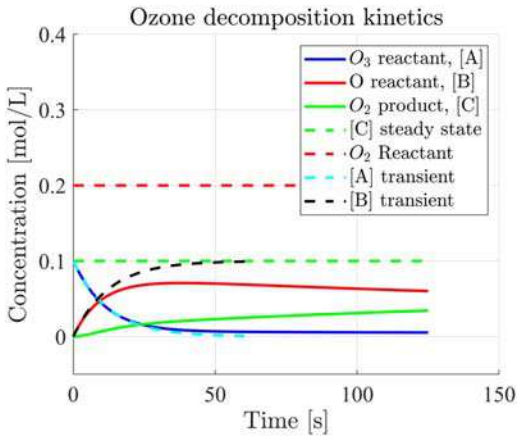


FIGURE 3.10

Time profiles of reactants and product for the ozone decomposition into oxygen.

Fig. 3.10 shows the time profile of the ozone and atomic oxygen (the reactants) and of the molecular oxygen (the product). The latter enters the reaction as a reactant too, but with constant concentration (the dashed red line).

Fig. 3.10 shows also the approximated short-term transients of $[A]$ (the dashed cyan line) and $[B]$ (the dashed black line), which depend on the time constant $\tau_1 = 12.5$s. The transient of $[A]$ follows from the first equation of (3.46), by assuming $[B] = [B]_0 = 0$. It holds

$$[A](t) \simeq \exp(-k_1 t)[A]_0. \tag{3.50}$$

The transient of $[B]$ follows from Eq. (3.47) by assuming $[C] = [C]_0 = 0$, which leads to $[B] \simeq W_0 - [A](t)$. The approximation ceases to be valid as soon as $[C](t)$ becomes significantly non-zero. Long-term approximations (not reported here) are more complex, but follow from the third equation of Eq. (3.46).

3.7.2 Alternative Matlab script

The following Matlab script, which calls the solver `ode45()`, is alternative to Simulink block diagram. The latter can be found in the companion website of the book [6].

```
%% Revers.reaction: O3(g)<==>O(g)+O2(g) k1=direct reaction k2= reverse
%% followed by irreversible: O(g)+O3(g)==>2O2(g) k3 = direct reaction
% Defines initial concentrations.
InitChem;
C0 = [0.1,0];
tspan = [0,480];   % Define time span.
[t, y] = ode45(@reversible_A_B, tspan, C0);    % Run ODE solver.
c1 = y(:,1);c2 = y(:,2);                        % Plot
plot(t/60,c1,'b',t/60,c2,'r','LineWidth',2);grid on;
xlabel('time, minutes');ylabel('concentrations');
title('Ozone decomposition into oxygen');
legend('[O3]','[O]');
%% Function
function dC = reversible_A_B(t, C)
    % Rate constants.
    k1 = 0.08;
    k2 = 0.04;
    k3 = 0.2;C_O2 = 0.2;
    % Rate laws.
    r1 = k1*C(1);       % C(1) = [O3]
    r2 = k2*C(2)*C_O2; % C(2) = [O]
    r3 = k3*C(1)*C(2); % C_O2 = [O2]
    % Mass balances.
    dC_O3  = -r1 + r2 - r3;
    dC_O = r1 - r2 - r3;
    % Assigns output variables.
    dC(1,:) = dC_O3;
 dC(2,:) = dC_O;
end
```

3.8 Irreversible A → B reaction with linear temperature increase

The reactions studied in the sections 3.2 to 3.7, have been assumed to occur under isothermal conditions. If, on the contrary, temperature is assumed to vary, also the specific rate constant k varies according to the Arrhenius equation (S.A. Arrhenius 1859−1927), which states:

$$k = A_{st}\exp\left(-E_{act}/(RT)\right), \tag{3.51}$$

where T is the absolute temperature in Kelvin [K], $R = 8.314$ J/(mol × K), A_{st} is the steric factor, and E_{act} is the activation energy in Joule/mole [J/mol]. The steric factor is related to the number of collisions among molecules with favorable orientation, and the activation energy is the energy needed by colliding molecules to form the activated complex. This latter constitutes a saddle point on the potential energy surface between reactants and products, where old bonds are breaking and new ones are forming [5]. The fraction of these molecules over the total is given by $\exp\left(-E_{act}/(RT)\right)$. Once the activated complex is formed, it can decay to form the products of reaction, as depicted in Fig. 3.11.

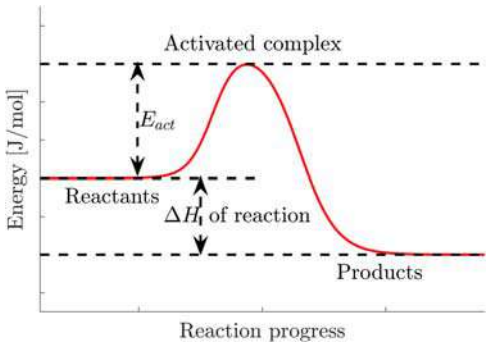


FIGURE 3.11

Energy diagram of a chemical reaction.

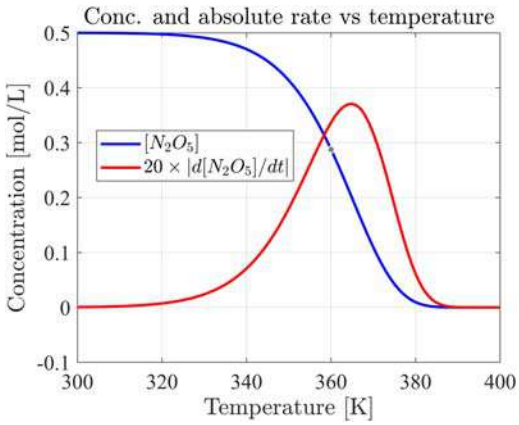


FIGURE 3.12

Temperature effect on first-order reaction kinetics.

The following example refers to the decomposition of dinitrogen pentaoxide into carbon tetrachloride according to the irreversible single-step reaction [3]:

$$N_2O_5 \Longrightarrow N_2O_4 + \frac{1}{2}O_2. \tag{3.52}$$

It is interesting to plot the reactant concentration and the concentration rate as well, which latter in Fig. 3.12, shows, unlike isothermal reactions, a maximum followed by a steep decay.

```
%% Reaction is N2O5(in CCl4) ==> N2O4 + 1/2 O2
% Activation energy Eact=1.06e5 Jo: Steric factor Ast=1.38e14: A0=0.5
% Heating rate 1 K/sec from 300 to 400 K
InitChem;
Eact = 1.06e5;R = 8.314;
Ast = 1.38e14;A0 = 0.5;
timespan = (0:100);
first = @(t,A) - Ast*exp(-Eact/R/(t+300))*A;
[t,A_calc] = ode45(first,timespan,A0);
A1_calc = diff(A_calc);
A1_calc(101) = 0;
A1_calc = -20*A1_calc;
plot(t+300,A_calc,'k',t+300,A1_calc,'k--');grid on
xlabel('Temperature [K]');ylabel('Concentration [mol/L]');
legend('$[N_2 O_5]$','$20 \times \vert d[N_2 O_5]/dt \vert$');
title (' Conc. and absolute rate vs temperature')
```

References

[1] P.L. Brezonik, Chemical Kinetics and Process Dynamics in Aquatic Systems, Lewis Publisher, a CRC Press Company, 1993.

[2] C. Capellos, B.H.J. Bielski, Kinetic Systems—Mathematical Description of Chemical Kinetics in Solution, Wiley-Interscience, 1969.

[3] R.H. Petrucci, F.G. Herring, J.D. Madura, C. Bissonnette, General Chemistry Principles and Modern Applications, 10th edition, Pearson, Canada, 2011.

[4] S.C. Chapra, Applied Numerical Methods with MATLAB for Engineers and Scientists, Fourth Edition, Mc Graw-Hill, 2018.

[5] Wikipedia, *Activated complex*, https://en.wikipedia.org/wiki/Activated_complex

[6] Companion Website of the Book. Elsevier, 2021. Available from: https://www.elsevier.com/books-andjournals/book-companion/9780323913416

CHAPTER

More complex kinetics aided by Matlab/Simulink

4

4.1 Introduction

The chapter extends the chemical kinetics of Chapter 3 to more complex chemical reactions, partly discovered in the second half of the 20th century, being of significance not only for chemistry but also for general science and dynamic systems. This fact suggests that the treatment employs methods and concepts of dynamic system theory as partly done in Chapter 3. In particular, an extensive use of linear state equations, their state matrix, and the relevant eigenvalues will be done in order to assess stability properties of the chemical kinetics under study. To this end, a short introduction to dynamic systems is provided in Appendix B, with focus on the topics of the book. A more extended introduction can be found in Reference [1].

We start from the well-known class of Michaelis–Menten reactions in biochemistry. The German biochemist L. Michaelis (1875–1949) and the Canadian physician M. Menten (1879–1960) developed, in the early 20th century, the work of the French-Russian chemist V. Henri (1872–1940), who had found in 1901 that enzyme reactions were initiated by a bond between enzyme and substrate. In 1913 they proposed the mathematical model of the reaction to be studied in Section 4.2. It is a fourth-order nonlinear state equation, which reduces to second order by separating conservation equations. The equation is characterized by different time scales: the short-time enzyme bonding to substrate and the long-term substrate transformation into product.

The second reaction in Section 4.3 is the striking iodine clock, in which the production of triiodide changes the solution color into dark blue. In this case also, the relevant nonlinear equation is characterized by different time scales, which allow us to find an approximate solution by fitting the simulated time profiles.

The second part of this chapter is devoted to oscillating chemical reactions. The underlying mechanism is that of the classical second-order Lotka–Volterra equations (LV, A.J. Lotka, 1880–1949, V. Volterra, 1860–1940) [2], which are recalled in Section 4.4 without reference to real chemical reactions. The modeled phenomenon is that of a population/species (prey, activator in chemistry) whose growth is limited by the predator population (inhibitor species in chemistry). Let us remark the peculiar behavior of the chemical activator which is at the same time reactant and product, with amplified concentration as in population dynamics. As a result, both populations become oscillating around the equilibrium population. In terms of system theory, both populations stay on a closed orbit which is the free response to a particular perturbation from equilibrium. The orbit corresponds to a new type of conservation equation, analog to a linear second-order oscillator with imaginary eigenvalues. Both examples of chemical oscillating equations, Briggs–Rauscher (Section 4.5) and Belousov–Zhabotinsky (BZ) (Section 4.6) are richer in their free response mode than LV equations, as they are sensible to variations of stoichiometric coefficients. Depending on some key coefficients, their equilibrium concentrations may change from asymptotic stability to instability as soon as the perturbed equation eigenvalues cross the imaginary axis

Fundamental Chemistry with Matlab. DOI: https://doi.org/10.1016/B978-0-323-91341-6.00008-2

(Hopf bifurcation [3], H. Hopf, 1894–1971). In the unstable case, the perturbed trajectories are captured by a stable limit cycle, giving rise to steady oscillations. Stability conditions are derived and shown graphically. The BZ reaction in a continuous-flow stirred-tank reactor may as well exhibit chaos [4], unlike the Oregonator model [5] studied in this book, which has no chaotic solutions. The topic of deterministic chaos has been omitted.

The Simulink block diagram of a few equations is reported in the book. Others can be found together with their Matlab scripts in the companion website of the book [6].

4.2 Michaelis–Menten kinetics

4.2.1 State equations, equilibrium, and stability

It is the standard model of enzyme kinetics. It involves an enzyme E, which binds to a substrate S so as to constitute a complex ES, which in turn releases a product P and regenerates the original enzyme. The underlying reaction consists of two steps: (1) the first one is a reversible reaction with forward rate constant k_f and reverse constant k_r and (2) the second and final one is a direct reaction with rate constant k_c, as follows

$$E + S \Leftrightarrow ES \Rightarrow E + P. \tag{4.1}$$

The corresponding state equations in terms of the substance concentrations hold:

$$\begin{aligned} \frac{d[E]}{dt} &= -k_f[E][S] + (k_r + k_c)[ES], \quad [E](0) = [E]_0 \\ \frac{d[S]}{dt} &= -k_f[E][S] + k_r[ES], \quad [S](0) = [S]_0 \\ \frac{d[ES]}{dt} &= k_f[E][S] - (k_r + k_c)[ES], \quad [ES](0) = [ES]_0 \\ \frac{d[P]}{dt} &= k_c[ES], \quad [P](0) = [P]_0 \end{aligned} \tag{4.2}$$

The first step aims to simplify the fourth-order Eq. (4.2) by discovering the conservation equations. The first one involves the whole combination W of the involved substances and holds:

$$\begin{aligned} \frac{dW(t)}{dt} &= 0, W(0) = W_0 \\ W &= [E] + [S] + [P] + 2[ES] \end{aligned} \tag{4.3}$$

The second conservation equation tells us that the enzyme is a catalyst and therefore is conserved throughout the process. Thus free and combined enzymes are conserved, and we can write

$$\begin{aligned} \frac{dK(t)}{dt} &= 0, K(0) = K_0 \\ K = [E] + [ES] &\Rightarrow [ES] = K - [E] \rightarrow [S] + [P] = W - 2K + [E] \end{aligned} \tag{4.4}$$

We define the differences $\Delta E = [E] - K_0$ and $\Delta P = [P] - (W_0 - K_0)$ as new state variables. Using them, the reader is asked to prove that the other two equations hold:

$$\begin{aligned} \frac{d\Delta E(t)}{dt} &= -k_f\Delta E^2 - \left(k_f K_0 + k_r + k_c\right)\Delta E + k_f\Delta P(\Delta E + K_0), \quad \Delta E(0) = \Delta E_0 \\ \frac{d\Delta P(t)}{dt} &= -k_c\Delta E, \quad \Delta P(0) = \Delta P_0 \end{aligned} \tag{4.5}$$

The equilibrium is obtained by zeroing the derivatives in Eq. (4.5), which provides

$$\begin{array}{c} \Delta E_\infty = 0,\ \ \Delta P_\infty = 0 \Rightarrow E_\infty = K_0,\ \ P_\infty = W_0 - K_0 \\ S_\infty = 0,\ \ [ES]_\infty = 0 \end{array}. \tag{4.6}$$

We can now derive a linear perturbation equation around the above equilibrium, upon definition of the perturbations $\delta E = [E] - E_\infty$, $\delta P = [P] - P_\infty$. We have found:

$$\begin{array}{c} \dfrac{d\delta P}{dt} = -k_c \delta E, \delta P(0) = \delta P_0 \\ \dfrac{d\delta E}{dt} = -k_f S_\infty \delta E - k_f K_0(\delta E - \delta P) - (k_r + k_c)\delta E = -\left(k_f K_0 + k_r + k_c\right)\delta E + k_f K_0 \delta P, \delta E(0) = \delta E_0 \end{array} \tag{4.7}$$

The eigenvalues of the state matrix of Eq. (4.7) (the derivation is left to the reader) are obtained by solving the second-order algebraic equation:

$$\begin{array}{c} det\begin{bmatrix} \lambda + a + b + c & -a \\ c & \lambda \end{bmatrix} = \lambda^2 + (a+b+c)\lambda + ac = 0 \\ \lambda_{12} = -\dfrac{a+b+c}{2} \pm \dfrac{\sqrt{(a+b+c)^2 - 4ac}}{2} \end{array}, \tag{4.8}$$

with the parameters $a = k_f K_0$, $b = k_r$, $c = k_c$. It is straightforward to prove that both eigenvalues are real and negative for any value of the rate constants, which implies *asymptotic stability* and justifies the previous equilibrium. These findings tell us that the equilibrium, if reached, is conserved in time; but they do not reveal whether the equilibrium could be reached whichever be the initial conditions. In other terms, they do not reveal whether the nonlinear equations in Eq. (4.7) are asymptotically stable in the large, an issue to be better studied in the section 4.2.2.

4.2.2 The Michaelis–Menten equation of the production rate

Convergence conditions to equilibrium can be obtained under the following practical assumptions [7].

1. The forward rate constant k_f is larger than the reverse constant k_r, and the latter is larger than the catalytic constant k_c. This implies that the enzyme binds with substrate very fast [reversible first reaction in Eq. (4.1), see Fig. 4.5, right], whereas catalytic production of the final product is very slow [irreversible second reaction in Eq. (4.1)]. During the second reaction, the substrate concentration $[S]$ progressively reduces to yield the product $[P]$, whereas the free and bounded enzyme concentrations remain practically constant. Formally:

$$k_f \gg k_r \gg k_c \Leftrightarrow \tau_f = 1/k_f \ll \tau_r = 1/k_r \ll \tau_c = 1/k_c. \tag{4.9}$$

2. The substrate initial concentration S_0 is larger than the enzyme E_0, which implies that during enzyme binding [first equation in Eq. (4.2)], we have $[S(t)] \simeq [S_0]$. Under this assumption and recalling the conservation law $[ES] = K_0 - [E]$, the first state equation in Eq. (4.2) can be simplified into the following linear and asymptotically stable equation:

$$\frac{d[E](t)}{dt} \simeq -\left(k_f\left([S]_0 - [E]_0\right) + k_r\right)[E](t) + k_r K_0,\ \ [E](0) = [E]_0. \tag{4.10}$$

The relevant equilibrium value holds [the numerical value comes from the values in Eq. (4.13)]

$$E_{\text{Binding}} \simeq \frac{k_r K_0}{k_f\left([S]_0 - [E]_0\right) + k_r} \simeq \frac{k_r K_0}{k_f [S] + k_r} \simeq 0.011 \text{ mol/L}, \tag{4.11}$$

where the last approximation descends from the inequality $[S(t)] \gg [E_0]$, which is valid during enzyme binding.

3. As soon as the enzyme is bounded to the substrate, $ES_{\text{Binding}} \simeq K_0 - E_{\text{Binding}}$ remains fairly constant, which implies a constant increasing rate of the product [fourth equation in Eq. (4.2)]:

$$\frac{d[P]}{dt} = k_c ES_{\text{Binding}}, \quad [P](0) = [P]_0. \tag{4.12}$$

By replacing Eq. (4.12) in Eq. (4.11) and by recalling that $K_0 = [E]_0$, we obtain the well-known *Michaelis–Menten equation* of the production rate:

$$\frac{d[P]}{dt} \simeq k_c [E]_0 \frac{[S]}{[S] + k_r/k_f} = V_{max} \frac{[S]}{[S] + k_r/k_f}, \tag{4.13}$$

where $V_{max} = k_c [E]_0$ is the maximum reaction rate.

4. As soon as $[S]$ becomes negligible, the bounded enzyme $[ES]$ converges to zero [third equation in Eq. (4.2)] and the product rate [fourth equation in Eq. (4.2)] tends to zero, implying steady-state achievement and convergence to equilibrium.

The simulated results, to be shown below, have been obtained with the following simple values, close, for what concerns the rate constants, to the case of chymotrypsin as an enzyme and *N*-acetyl-valine ethyl ester as a substrate [6]:

$$\begin{gathered} k_f = 10\big((\text{mol/L}) \times \text{s}\big)^{-1}, \; k_r = 1\text{s}^{-1}, \; k_c = 0.1 k_r \\ E_0 = 0.1 \text{ mol/L}, \; S_0 = 10 E_0, \; S_0 = E S_0 = 0 \end{gathered}. \tag{4.14}$$

The initial concentrations have been chosen to obtain neat graphical results.

Fig. 4.1 shows short-term and long-term time profiles of the product $[P]$ and the rate $d[P]/dt$, together with the profile of the Michaelis–Menten production rate in Eg. (4.13). The latter rate in [(mol/L)/s] has been multiplied by the factor 20.

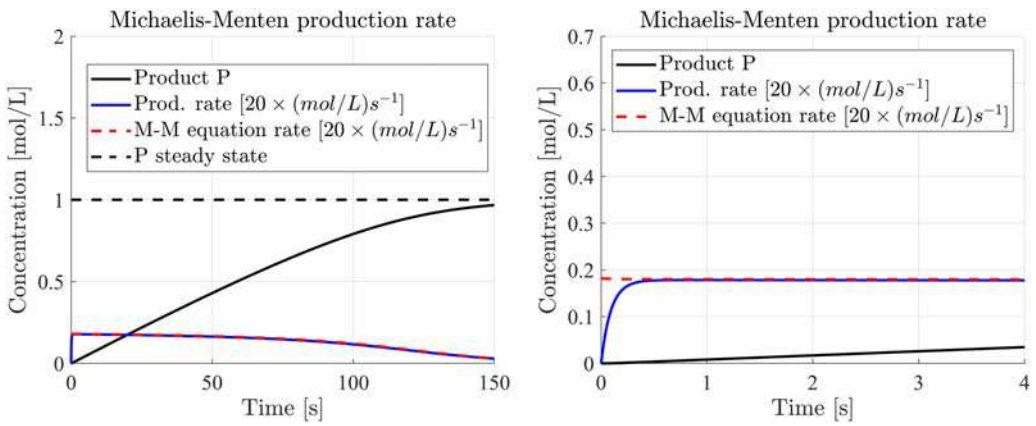


FIGURE 4.1

(Left) Simulated Michaelis–Menten production rate and the approximate long-term profile from Eq. (4.13) (dashed red). (Right) Short-term profiles.

4.2.3 Script, block diagram and graphical results

Time simulation is launched by the following script organized into four parts: initialization, parameter values (rate constants and initial concentrations), timing with the simulation launch command, and graphical results.

```
%% Michaelis-Menten kinetics
%% Initialization
InitChem;
disp('Michaelis-Menten kinetics')
%% 1) Parameters
disp('1) Parameters');
% Rate constants
kf=10; kr=1.0;kc=0.1;tau=1/kc;
% Initial concentration [mol/dm^3]
E0=0.1; ES0=0.0; S0=1;P0=0.0;
W0=E0+S0+P0+2*ES0; K0=E0+ES0;% conserved
% Reverse reaction equilibrium
Einf=K0;Pinf=W0-K0;
%% 2) Time simulation
disp('2) Time simulation');
Ts=0.02;  %<< tau=1/kf
tInit=0;tEnd=15*tau;
inttEnd=floor(tEnd/Ts);tEnd=inttEnd*Ts;
out=sim('MichaelisMentenSim');
%% 3) Graphical results
disp('3) Graphical results');
% Variables to be plotted
t=out.t;dimt=length(t);
E=out.E;ES=out.ES;S=out.S;P=out.P;
dotP=out.dotP;
% Steady state
Ess=ones(dimt,1)*K0;
Pss=ones(dimt,1)*(W0-K0);
% Figures 1) Output
% Long term
figure('name','Output');hold on; grid on;
plot(t,E,'b');plot(t,Ess,'b--');
plot(t,ES,'r');plot(t,S,'g');
plot(t,P,'k');plot(t,Pss,'k--');
xlabel('Time [s]');
ylabel('Concentration [mol/L]');
title('Michaelis-Menten reaction');
legend('Enzyme E ','E steady state','Complex ES ',...
   'Substrate S','Product P', 'P steady state');
```

```
% Short term
figure('name','Output ST');hold on; grid on;
plot(t,E,'b');plot(t,Ess,'b--');
plot(t,ES,'r');plot(t,S,'g');
plot(t,P,'k');plot(t,Pss,'k--');
xlabel('Time [s]');
ylabel('Concentration [mol/L]');
title('Michaelis-Menten reaction');
legend('Enzyme E ','E steady state','Complex ES ',...
   'Substrate S','Product P', 'P steady state');
xlim([0 4]);
% 2) Derivative of P and MM rate
% Long term
scaleRate=20;
MMEq=kc*E0*S./(S+kr/kf); %
figure('name','Prod. rate');hold on; grid on;
plot(t,P,'k');plot(t,dotP*scaleRate,'b');
plot(t,MMEq*scaleRate,'r--');plot(t,Pss,'k--');
xlabel('Time [s]');
ylabel('Concentration [mol/L]');ylim([0 1.2]);
ylim([ 0 2]);
title('Michaelis-Menten production rate');
legend('Product P','Prod. rate $[20 \times (mol/L)s^{-1}]$',...
   'M-M equation rate $[20 \times (mol/L)s^{-1}]$',...
   'P steady state', 'location', 'best','FontSize',16);
% Short term
figure('name','Prod. rate ST');hold on; grid on;
plot(t,P,'k');plot(t,dotP*scaleRate,'b');
plot(t,MMEq*scaleRate,'r--');
xlabel('Time [s]');
ylabel('Concentration [mol/L]');ylim([0 1.2]);
ylim([ 0 0.7]);
xlim([0 4] );
title('Michaelis-Menten production rate');
legend('Product P','Prod. rate $[20 \times (mol/L)s^{-1}]$',...
   'M-M equation rate $[20 \times (mol/L)s^{-1}]$',...
   'location', 'best','FontSize',16);
```

Fig. 4.2 shows the higher-level Simulink block diagram including four blocks which describe the four equations in Eq. (4.3) (conservation), in Eq. (4.4) (conservation), and in Eq. (4.5) (product and free enzyme).

Figs. 4.3 and 4.4 show the details of the free enzyme and product blocks.

The most complex diagram pertains to the free enzyme. Two negative feedback links are reported, one linear (the upper loop) and one mixed (quadratic and linear, the lower loop) because of the product $[E][S]$. The free enzyme $[E]$ also enters the diagram of $[P]$ as a negative linear feedback. The overall dynamics contains a loop constituted by the cascade of two integrators and a negative feedback. In general, a loop of this kind may oscillate and also become unstable, but the above analysis has shown that under practical assumptions the overall equation converges to equilibrium concentrations.

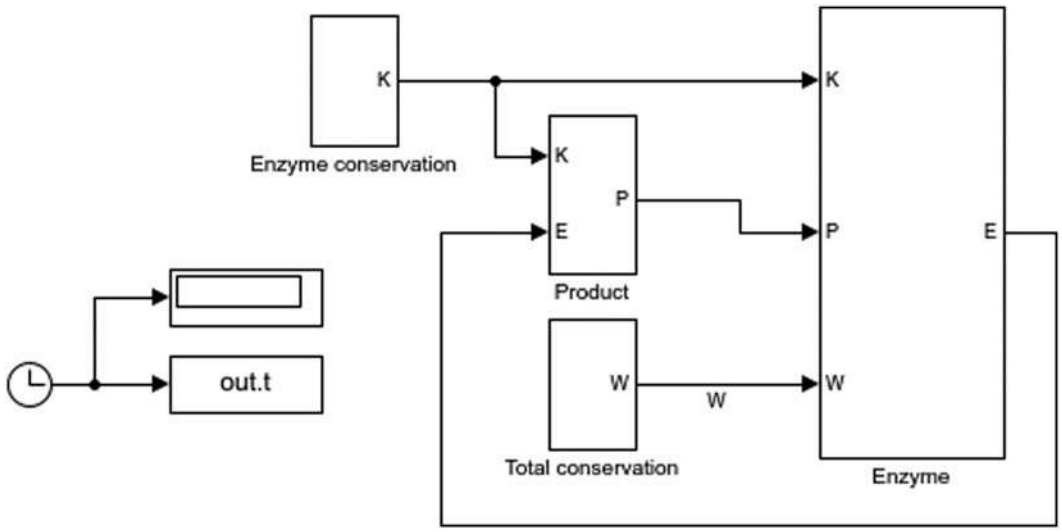


FIGURE 4.2

Higher-level block diagram of the Michaelis-Menten state equations.

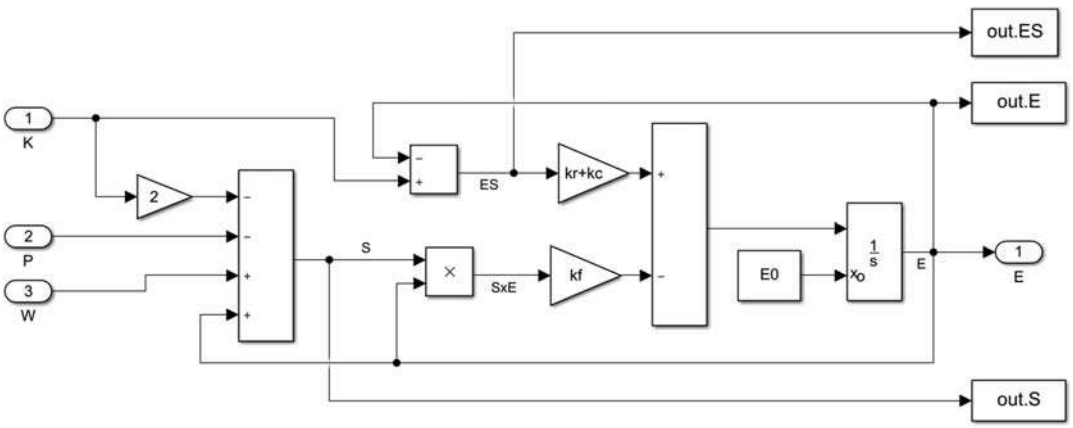


FIGURE 4.3

Details of the free enzyme block.

Fig. 4.5 shows long-term and short-term time profiles. On the left side, you can see the intermediate linear progression (decay and growth) of the substrate (green line) and product (black line) concentrations, before equilibrium achievement. The right side focus on the initial transient corresponding to the forward first reaction in Eq. (4.1), when the free enzyme (blue line) binds with the substrate, which latter is slightly decreasing to the purpose, altogether with the free enzyme. The free enzyme is recovered [reverse first reaction in Eq. (4.1)] as soon as the substrate is transformed into the product (see the left side after $t = 100$s when the free enzyme reaches the steady-state).

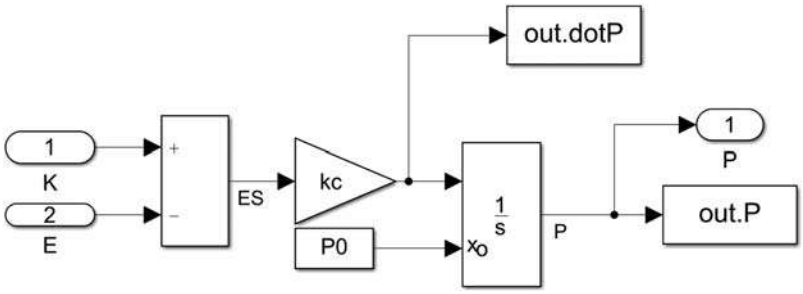


FIGURE 4.4

Details of the product block.

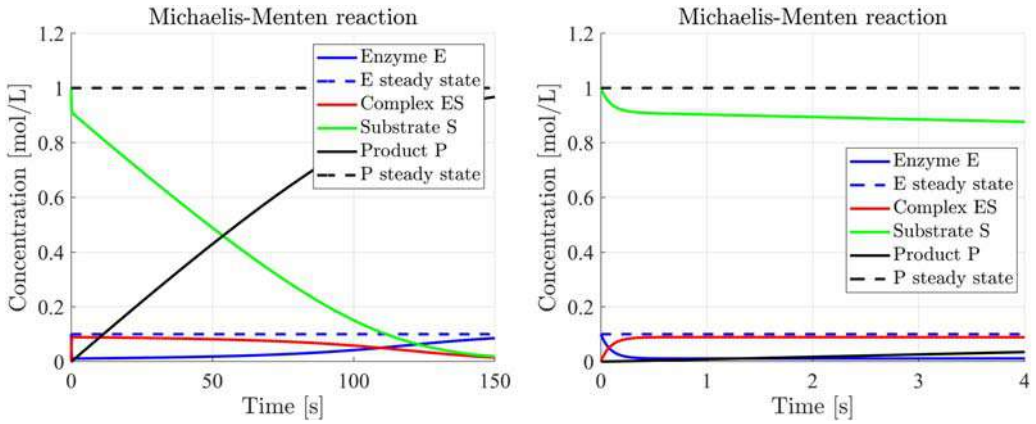


FIGURE 4.5

(Left) Long-term profiles. (Right) Short-term profiles.

4.3 The iodine clock reaction

4.3.1 State equations

The iodine clock reaction, discovered by Landolt in 1886 (H.H. Landolt, 1831−1910), allows us to visualize how different rate constants in consecutive reactions affect the concentration of species.

In this case, iodine anions I^- are colorless. When they are reacted with hydrogen peroxide and protons, triiodide I_3^- is formed showing a dark blue color. The reaction is a series of the two consecutive irreversible reactions [8]:

$$\begin{array}{l}(H_2O_2) + 3I^- + 2H^+ \Rightarrow I_3^- + 2H_2O \quad (\text{rate constant } k_1)\\ I_3^- + 2SO_3^{2-} \Rightarrow 3I^- + S_2O_6^{2-} \quad (\text{rate constant } k_2)\end{array}, \tag{4.15}$$

where $k_1 \ll k_2$. Let us introduce the following concentration symbols:

$$[A] = \left[I_3^-\right],\ [B] = [I^-],\ [C] = \left[S_2O_3^{2-}\right]. \tag{4.16}$$

The state equations are as follows:

$$\begin{array}{l}\dfrac{d[A]}{dt} = k_1[B]^3 - k_2[A][C]^2,\ [A](0) = [A]_0\\ \dfrac{d[B]}{dt} = -k_1[B]^3 + 3k_2[A][C]^2,\ [B](0) = [B]_0\\ \dfrac{d[C]}{dt} = -k_2[A][C]^2,\ [C](0) = [C]_0\end{array}, \tag{4.17}$$

with the parameter values:

$$\begin{array}{l}k_1 = 0.005\ (\text{mol/L})^{-2}\text{s}^{-1}, k_2 = 5\ (\text{mol/L})^{-2}\text{s}^{-1}\\ [A]_0 = 0\ \text{mol/L}, [B]_0 = 1\ \text{mol/L}, [C]_0 = 0.5\ \text{mol/L}\end{array}. \tag{4.18}$$

The conservation law is found to be:

$$W = [A] + [B] + 2[C], \frac{dW}{dt} = 0, W(0) = W_0. \tag{4.19}$$

Thus, the set of equations in Eq. (4.17) simplifies into:

$$\begin{array}{l}\dfrac{d[B]}{dt} = -k_1[B]^3 + 3k_2\left(W_0 - [B] - 2[C]\right)[C]^2,\ [B](0) = [B]_0\\ \dfrac{d[C]}{dt} = -k_2\left(W_0 - [B] - 2[C]\right)[C]^2,\ [C](0) = [C]_0\end{array}. \tag{4.20}$$

Equations (4.19) and (4.20) tell us that the equilibrium concentrations hold:

$$[A]_\infty = W_0 = [A]_0 + [B]_0 + [C]_0,\ [B]_\infty = 0,\ [C]_\infty = 0. \tag{4.21}$$

The higher-level Simulink block diagram is shown in Fig. 4.6. The reader is asked to build up the diagram of each block.

Asymptotic stability can be proved by solving Eq. (4.20) in an approximate way. Let us start with the third equation in Eq. (4.17) and assume that the nonzero though negligible $[A]$ is equal to a constant average $[A] \simeq \overline{A}$. Separation of variables leads to the following approximate solution:

$$[C](t) \simeq \begin{cases}[C]_0, t < t_0\\ \dfrac{[C]_0}{1 + k_2\overline{A}[C]_0 t}, t \geq t_0\end{cases}. \tag{4.22}$$

The second equation of Eq. (4.17) is more complex. The solution splits into two parts, under the assumption $k_1 \ll k_2$, which leads to $[B](t) \simeq [B]_0 + 3k_2\overline{A}\int_0^{t_1}[C]^2(\tau)d\tau$, where t_1 roughly corresponds

to the result of $\arg\max_t[B](t)$. After this time instant, the solution is approximated by the free response of $[\dot{B}](t) = -k_1[B]^3, [B](t_1) = [B]_1$, which holds

$$[B](t) \simeq \frac{[B]_1}{\sqrt{1 + k_1[B]_1^2(t - t_1)}}, \quad t > t_1. \tag{4.23}$$

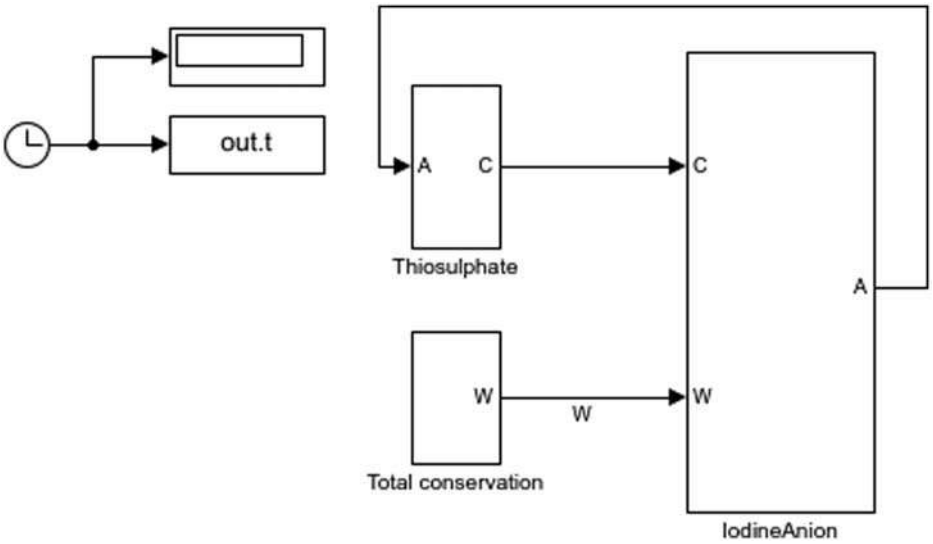


FIGURE 4.6

Higher-level block diagram of the simplified equations (4.19) and (4.20).

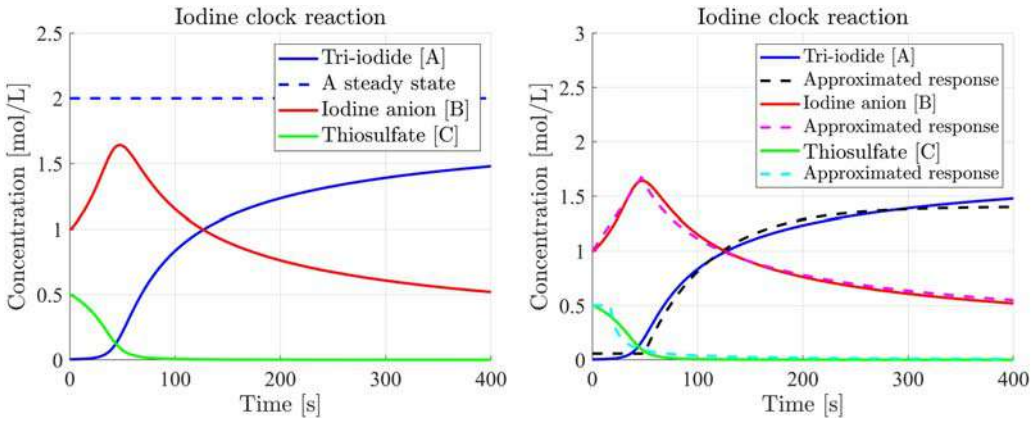


FIGURE 4.7

(Left) Iodine clock time profiles with triiodide steady state (dashed line). (Right) Approximate analytic solutions compared to simulated ones.

The solution of the first equation in Eq. (4.17) can be approximated by

$$[A](t) \simeq \begin{cases} \overline{A}, t < t_2 \\ \int_{t_2}^{t} \exp\Big(- k_2 \overline{C}^2 (t-\tau)\Big) k_1 [B]^3(\tau) d\tau, \quad t \geq t_2 \end{cases}. \tag{4.24}$$

Simulated profiles of the concentrations are shown in Fig. 4.7, left. The approximate analytic solutions are shown in Fig. 4.7, right. Their parameters $\overline{A}, [B]_1, t_0, t_1, t_2, k_1$ have been tuned by minimizing the deviation from the simulated profiles with the help of the Matlab function `fminsearch`.

4.4 Oscillating kinetics: introduction

4.4.1 The Lotka–Volterra model

Chemical oscillating reactions have in common a rather unusual reaction occurring in homogeneous means, for instance aqueous solutions, which we refer to as *breeding reaction* and is also known as *auto-catalytic reaction* [9]. Since actually no catalysis is actuated by reaction products, the authors prefer the former expression.

One of the reactants, say A, with the help of other reactants (say, food denoted by F), *generates* itself as a *product* with growing concentration, say $k[A], k > 1$integer, as in the following simple example:

$$\begin{gathered} [F] + [A] \Rightarrow 2[A], \ \text{ rate constant } \ k \\ \frac{d[S]}{dt} = k[F][A] - \cdots, \ \ [A](0) = [A]_0 \end{gathered}, \tag{4.25}$$

where the positive rate $k[F][A]$, if moderated by other reactions using A as a reactant, may give rise to *oscillations*.

A well-known model which can be interpreted as a set of hypothetical chemical reactions, though not occurring in practice, is the *LV model* [2]. In terms of population dynamics, a living being A fed by the unlimited resource F generates other living beings by doubling for instance its concentration. Moderation is provided by another living being B which upon consuming A doubles its concentration. The concentration of B is moderated by natural death. The model can be expressed in terms of the following reactions:

$$\begin{gathered} [F] + [A] \Rightarrow 2[A], \text{ rate constant } k_1, \text{ breeding reaction 1} \\ [A] + [B] \Rightarrow 2[B], \text{ rate constant } k_2, \text{ breeding reaction 2} \\ [B] \Rightarrow [P], \text{ rate constant } k_3, \text{ death reaction} \end{gathered} \tag{4.26}$$

The corresponding state equations have the form of the LV equations:

$$\begin{gathered} \frac{d[A]}{dt} = k_1 [F]_0 [A] - k_2 [A][B], \ [A](0) = [A]_0 \\ \frac{d[B]}{dt} = -k_3 [B] + k_2 [A][B], \ [B](0) = [B]_0 \end{gathered}, \tag{4.27}$$

where the food constant concentration $[F]_0$ may be included into k_1. The Simulink block diagram, being rather elementary, is not reported. The above equations are known to have an oscillating solution around the *nonzero equilibrium point* $\{\overline{A}, \overline{B}\}$, with the amplitude established by initial

conditions. A pair of equilibriums exist (the reader is asked to prove it), but only the nonzero equilibrium is of interest and is known as the *critical point*:

$$\begin{gathered}\overline{A}=\frac{k_3}{k_2},\ \overline{B}=\frac{k_1[F]_0}{k_2}\\ \overline{A}_0=0,\ \overline{B}_0=0\end{gathered}. \tag{4.28}$$

In view of other oscillating reactions, it is of interest to find the stability properties of the critical point, through the usual perturbation equation, which is written in matrix form and in terms of the perturbations from the equilibrium $\delta A=[A]-\overline{A},\ \delta B=[B]-\overline{B}$:

$$\begin{bmatrix}\dfrac{d\delta A(t)}{dt}\\ \dfrac{d\delta B(t)}{dt}\end{bmatrix}=\begin{bmatrix}0 & -k_2\overline{A}=-k_3\\ k_2\overline{B}=k_1[F]_0 & 0\end{bmatrix}\begin{bmatrix}\delta A\\ \delta B\end{bmatrix}(t),\ \begin{bmatrix}\delta A\\ \delta B\end{bmatrix}(0)=\begin{bmatrix}\delta A_0\\ \delta B_0\end{bmatrix} \tag{4.29}$$

Stability is revealed by the state matrix eigenvalues (the reader is asked to prove them):

$$\lambda_{12}=\pm j\sqrt{k_3k_1[F]_0}. \tag{4.30}$$

In this case, it can be proved that any small perturbation from the critical point *generates a closed trajectory (closed orbit)* around it, in agreement with the imaginary eigenvalues of Eq. (4.30). The eigenvalues indicate that the free response of Eq. (4.29) is oscillating with a period $P_{small}=2\pi/\sqrt{k_3k_1[F]_0}$. Actually, this period only applies to small perturbations. As soon as perturbations increase, the period will depend in a rather complicated way on initial conditions [2].

The fundamental question is: *what is conserved*? In the previous kinetic equations, the total amount of the initial concentrations was conserved, sometimes together with other concentrations. Here, the conservation concept is subtler and can be described from two standpoints.

1. A function $V([A],[B])$ of the concentrations is conserved, which can be considered as the *energy* of the closed loop trajectory described by the concentration pair (see Fig. 4.8, right). We can write $V([A],[B])=V\left([A]_0,[B]_0\right)$ [2].
2. The closed loop trajectory is conserved, given the initial conditions $\{[A]_0,[B]_0\}$.

Fig. 4.8 shows the time profiles and the 2D trajectories of the LV kinetics, with different initial conditions. The rate constant values are as follows

$$k_1=0.2\frac{\text{L/mol}}{\text{s}},\ k_2=0.3\frac{\text{L/mol}}{\text{s}},\ k_3=0.4\frac{1}{\text{s}},\ [F]_0=1\ \text{mol/L}. \tag{4.31}$$

Initial conditions have been perturbed from the critical point defined by $\overline{A}=1.33\ \text{mol/L}$ and $\overline{B}=0.67\ \text{mol/L}$, which lies at the center of the small red ellipse in Fig. 4.8, right. By increasing the perturbation magnitude, the period slightly increases above $P_{small}=222\text{s}$ and the orbit deforms with respect to the small perturbation ellipse (in red line, Fig. 4.8, right). The reader is asked to build up the Simulink block diagram of Eq. (4.27).

The script, in charge of launching the simulation and of providing graphical results, contains the loop of the different closed orbits in Fig. 4.8, right, generated by the perturbation vectors `dA` and `dB` of the equilibrium pair `Abar,Bbar`. The values of the perturbation vectors can be modified by the readers. The time profile in Fig. 4.8, left, refers to the largest orbit (Orbit 5).

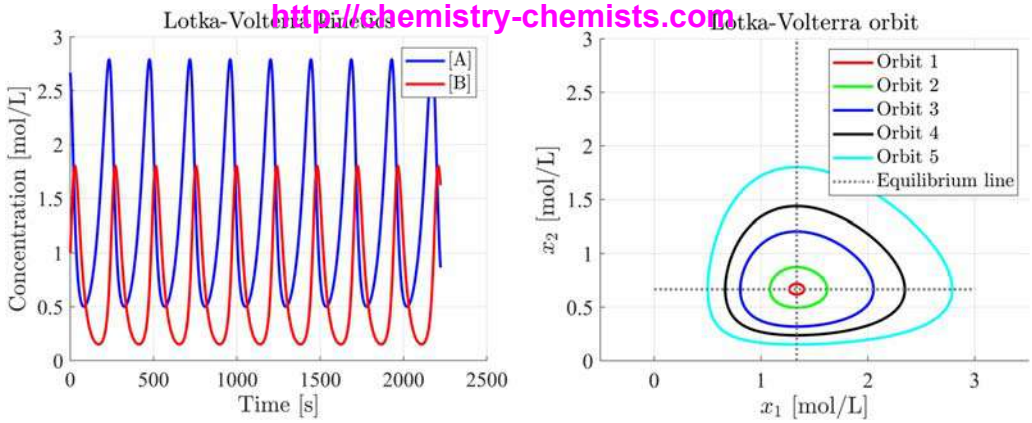


FIGURE 4.8

Lotka–Volterra kinetics. (Left) Time profiles of the largest trajectory in the right side. (Right) Closed orbits of different initial conditions.

```
%% Chemistry with Matlab: Lotka Volterra kinetics (Chapter 4)
%% Initialization
InitChem;
disp('Lotka Volterra kinetics');
%% 1) Parameters
disp('1) Parameters');
k1=0.2; k2=0.3;k3=0.4;F0=1;
OmSmall=sqrt(k1*F0*k3);PSmall=2*pi/OmSmall;
Abar=k3/k2;Bbar=k1*F0/k2; % Equilibrium
%% 2) Timing and simulation
disp('2) Monte Carlo simulation');
Ts=0.1;
tInit=0;tEnd=10*PSmall;
inttEnd=floor(tEnd/Ts);tEnd=inttEnd*Ts;
MC=5; % Monte Carlo simulation
dA=zeros(MC,1);dB=zeros(MC,1);
dA=[0.05 ;0.2; 0.5; 0.7; 1];
dB=[0.05;0.2; 0.5; 0.7; 1]/2;
dimt=inttEnd+1;
A=zeros(dimt,MC);B=zeros(dimt,MC);
for i=1:MC
    A0=Abar*(1+dA);      B0=Bbar*(1+dB);
    out=sim('LotkaVolterraSim');
    % Plotting time profiles
    t=out.t;  dimt=length(t);
    A(:,i)=out.A(:,i);  B(:,i)=out.B(:,i);
    if i==MC
        figure('name','Output');hold on; grid on;
        plot(A(:,i),'b');plot(B(:,i),'r');
        xlabel('Time [s]');
        ylabel('Concentration [mol/L]');ylim([0 3]);
        title('Lotka-Volterra kinetics');
        legend(' [A] ',' [B]');
    end
```

```
end
%% 3) Graphical results: orbit
disp('3) Graphical results: orbit');
figure('name','Orbit');hold on; grid on;
% Equilibrium lines
xr=[0 Abar;3 Abar]; yr=[Bbar 0;Bbar 3];
for i=1:MC
   plot(A(:,i),B(:,i),color(i));
end
line(xr,yr,'color',grey,'LineStyle',':');
xlabel('$x_1$ [mol/L]');
ylabel('$x_2$ [mol/L]');
title('Lotka-Volterra orbit');
legend('Orbit 1','Orbit 2','Orbit 3','Orbit 4', 'Orbit 5',...
   'Equilibrium line');axis equal;
```

4.5 Oscillating kinetics of Briggs–Rauscher

4.5.1 Simplified reaction scheme

An oscillating chemical reaction which includes the pattern of the LV kinetics in Eq. (4.26) is the Briggs–Rauscher reaction, discovered in 1972. The reaction, also known as the oscillating clock, is one of the most common demonstrations of a chemical oscillator reaction. The general scheme can be simplified as follows:

$$IO_3^- + 2H_2O_2 + CH_2(COOH)_2 + H^+ \Rightarrow I - CH(COOH)_2 + 2O_2 + 3H_2O. \tag{4.32}$$

The reaction begins when three colorless solutions are mixed together. They are

1. potassium iodate KIO_3 in sulfuric acid H_2SO_4,
2. manganese sulfate $MnSO_4$ and malonic acid $CH_2(COOH)_2$ in water, and
3. aqueous solution of hydrogen peroxide H_2O_2.

The color of the resulting mixture oscillates between clear, amber, and deep blue for about 3–5 min. The solution ends up into a blue and black mixture. A small amount of starch is usually added, so as to enhance the color of the I^- ions.

Substrates for the reaction are IO_3^- ions (from potassium iodate KIO_3), hydrogen peroxide H_2O_2, and malonic acid $CH_2(COOH)_2$ as a reducing agent. Mn^2 ions from the manganese sulfate $MnSO_4$ act as catalysts.

The reaction comprises different consecutive steps. A simplified model was proposed in 1976 [10] and consists of four different steps. The concentration of the previous substrate reactants is assumed to remain constant throughout the whole reaction cycle and is indicated with the letter *C*. On the contrary, three intermediate species, namely *hypoiodous acid* (*X*), HIO, *iodine* (*Y*), I_2,

and the *iodide ion* (Z), I^-, are highly variable in their concentration. They are mutually reacting according to the equations:

$$\begin{array}{c}[C]\Rightarrow[X],\ k_1,\ \text{zero order kinetics}\\ [C]+2[X]\Rightarrow 2[Y],\ k_2\\ [Y]+[Z]\Rightarrow 2[Z],\ k_3,\ \text{breeding reaction as in the LV pattern}\\ [X]+[Z]\Rightarrow[P],\ k_4,\ \text{moderating reaction}\end{array}. \tag{4.33}$$

The product P is the iodo-malonic acid $ICH(COOH)_2$. The above four steps can be associated with the following four reactions, even if each step actually consists of more than one consecutive reactions:

1. Production of *hypoiodous acid* (X), HIO:

$$[C]\rightarrow[X]\Leftrightarrow IO_3^- + 2H_2O + H^+ \rightarrow HIO + 2O_2 + 2H_2O. \tag{4.34}$$

2. Production of molecular *iodine* (Y), I_2:

$$[C]+2[X]\rightarrow 2[Y]\Leftrightarrow 2IO_3^- + 6H_2O + 2HIO + 2H^+ \rightarrow + 2I_2 + 6O_2 + 8H_2O. \tag{4.35}$$

3. Production of *iodide ion* (Z), I^-, the *breeding reaction*:

$$[Y]+[Z]\rightarrow 2[Z]\Leftrightarrow CH_2(COOH)_2 + I_2 + I^- \rightarrow 2I^- + H^+ + ICH(COOH)_2. \tag{4.36}$$

4. Production of iodo-malonic acid (P, end product):

$$[X]+[Z]\rightarrow[P]\Leftrightarrow HIO + I^- + H^+ + 2CH_2(COOH)_2 \rightarrow 2ICH(COOH)_2 + H_2O.$$

4.5.2 State equations and stability analysis

The simplified reactions in Eq. (4.33) can be converted into state equations, by assuming $[C]=[C]_0=$ constant and including $[C]_0$ into the rate constants $k_1=\kappa_1[C]_0$ and $k_2=\kappa_2[C]_0$. The state equations of Eq. (4.33) are as follows

$$\begin{array}{l}\dfrac{d[X]}{dt}=k_1-2k_2[X]^2-k_4[X][Z],[X](0)=[X]_0\\ \dfrac{d[Y]}{dt}=2k_2[X]^2-k_3[Y][Z],[Y](0)=[Y]_0\\ \dfrac{d[Z]}{dt}=k_3[Y][Z]-k_4[X][Z],[Z](0)=[Z]_0\end{array}. \tag{4.37}$$

The higher-level block diagram is shown in Fig. 4.9. Lower-level block diagrams are left to readers.

In view of the following treatment, we simplify the previous notations into $x=[X], y=[Y], z=[Z]$, and we will use the time derivative notation $dx(t)/dt=\dot{x}(t)$. Looking for equilibrium points, we write the following equations:

$$\begin{array}{c}\dot{x}(t)+\dot{y}(t)+\dot{z}(t)=0\rightarrow k_1-2k_4\overline{x}\overline{z}=0\Rightarrow\overline{x}=\dfrac{1}{2}\sqrt{\dfrac{k_1}{k_2}}\\ \dot{y}(t)+\dot{z}(t)=0\rightarrow 2k_2\overline{x}-k_4\overline{z}=0\Rightarrow\overline{z}=\dfrac{2k_2}{k_4}\overline{x}=\dfrac{1}{k_4}\sqrt{k_1k_2}\\ \dot{z}(t)=0\rightarrow k_3\overline{y}-k_4\overline{x}=0\Rightarrow\overline{y}=\dfrac{k_4}{k_3}\overline{x}=\dfrac{k_4}{2k_3}\sqrt{\dfrac{k_1}{k_2}}\end{array}. \tag{4.38}$$

Unlike LV equations, Eq. (4.38) provides a single nonzero equilibrium point (the critical point).

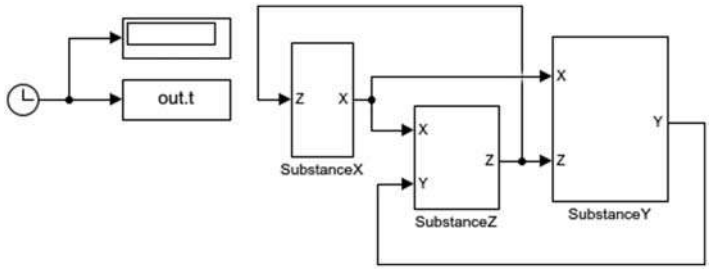


FIGURE 4.9

Higher-level block diagram of Eq. (4.37).

The second step is the study of the perturbation stability around the equilibrium, in terms of the perturbations $\delta x = x - \overline{x}, \delta y = y - \overline{y}, \delta z = z - \overline{z}$. The following third-order LTI equation, written in matrix form, is found:

$$\delta\dot{\boldsymbol{x}}(t) = \begin{bmatrix} -4k_2\overline{x} - k_4\overline{z} & 0 & -k_4\overline{x} \\ 4k_2\overline{x} & -k_3\overline{z} & -k_3\overline{y} \\ -k_4\overline{z} & k_3\overline{z} & -k_4\overline{x} + k_3\overline{y} \end{bmatrix} \delta\boldsymbol{x}(t), \delta\boldsymbol{x}(0) = \delta\boldsymbol{x}_0. \tag{4.39}$$

Stability is decided by the eigenvalues $\{\lambda_1, \lambda_2, \lambda_3\}$ of the state matrix in Eq. (4.39). Asymptotic stability occurs when all the three eigenvalues possess negative real part, that is $\mathrm{Re}\lambda_k < 0, k = 1, 2, 3$. When eigenvalues have a complex expression (not shown) as in the present case, stability conditions can be derived by applying the Routh criterion to the characteristic polynomial $\lambda^3 + a_2\lambda^2 + a_1\lambda + a_0$, here written in the generic form (see Appendix B). Asymptotic stability asks that the following three inequalities are satisfied:

$$a_2 > 0, a_0 > 0, a_2a_1 > a_0. \tag{4.40}$$

In the present case, they reduce to $a_2a_1 > a_0$, since the other two inequalities are satisfied for $k_j > 0$, $j = 2, 3, 4$, $\overline{x} > 0$, $\overline{y} > 0$, $\overline{z} > 0$. The inequality $a_0 > 0$ implies that at least one real root λ_1 exists with negative real part. Thus, if $a_2a_1 < a_0$ holds, namely that condition Eq. (4.40) is not respected, a pair of eigenvalues exist, either real or complex, having positive real part. In our case, it can be proved that $a_2a_1 < a_0$ only depends on the ratios $\alpha = k_2/k_4$ and $\beta = k_3/k_4$, and that the *instability condition* writes as follows

$$\begin{gathered} f(\alpha, \beta) = \beta^2(1 + 6\alpha) - 2(1 - 9\alpha)\beta - 3 < 0 \\ \alpha = k_2/k_4 > 0, \beta = k_3/k_4 > 0 \end{gathered}. \tag{4.41}$$

The instability region in the plane $\{\alpha, \beta\}$ is the colored area in Fig. 4.10.

4.5.3 Behavior of the state equations

The upper boundary of the instability region in Fig. 4.10 separates two different behaviors of Eq. (4.37).

1. *Stable behavior*: any perturbation converges to the equilibrium point of Eq. (4.38), whose value depends on the rate constants (simulated case 1).

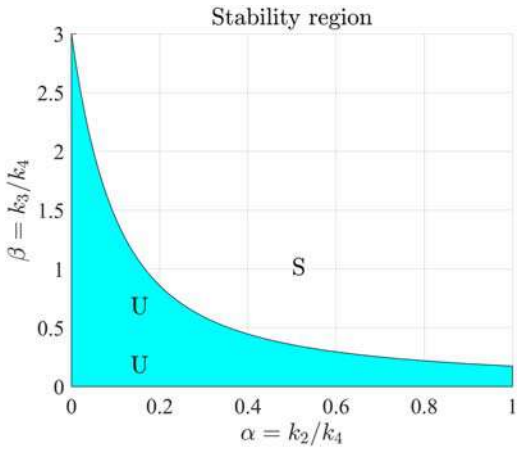


FIGURE 4.10

Instability region: U (unstable) and S (stable) denote the rate constant ratios of the three simulated cases.

2. *Unstable equilibrium* and *oscillating behavior:* any perturbation converges from inside (unstable equilibrium) and from outside to a *stable limit cycle* whose size, shape, and period depend on the rate constants and not on the initial conditions (simulated cases 2 and 3).

Table 4.1 Parameters of the simulated cases.

No.	Parameter	Symbol	Unit	Case 1	Case 2	Case 3
1	Rate	k_1	$(\mathrm{mol/L})\mathrm{s}^{-1}$	0.06	0.02	0.02
2	Rate constant	k_2	$(\mathrm{mol/L})^{-1}\mathrm{s}^{-1}$	0.03	0.008	0.008
3	Rate constant	k_3	$(\mathrm{mol/L})^{-1}\mathrm{s}^{-1}$	0.06	0.04	0.01
4	Rate constant	k_4	$(\mathrm{mol/L})^{-1}\mathrm{s}^{-1}$	0.06	0.06	0.06
5	Initial condition	X_0	mol/L	1	0.79 + 0.01	0.79 + 0.01
6	Initial condition	Y_0	mol/L	0.1	1.19 + 0.01	4.74 + 0.01
7	Initial condition	Z_0	mol/L	0.1	0.21 + 0.01	0.21 + 0.01

Rate constants and initial conditions are listed in Table 4.1. In the simulated cases 2 and 3, initial conditions correspond to the equilibrium point which is slightly perturbed of 0.01.

In the case 1, the time profiles in Fig. 4.11 converge to the equilibrium point $[1, 1, 1]/\sqrt{2}$. The reader should pay attention that the time unit is kilosecond.

In the case 2, the equilibrium point is unstable, and any small perturbation as in Fig. 4.12 triggers increasing oscillations that converge to a *stable limit cycle*, which is reached also from outside. Fig. 4.12, left, shows the limit cycle (the outermost) and the unstable spiral (see Appendix B) leading from the equilibrium point to the limit cycle. The spiral is traveled counterclockwise. The convergence

to a limit cycle can be checked by computing the so-called *Lyapunov exponents* (see Appendix B). One of them, the largest, must converge to zero, whereas the other two exponents must be negative.

The case 3 is similar to case 2, but the limit cycle in Fig. 4.13, left, becomes very narrow along the X axis and the periodic waves in Fig. 4.13, right, approach square waves (hypoiodous acid) and saw-tooth waves (iodine) due to the small rate constant $k_3 = 0.01\left((\text{mol/L}) \times \text{s}\right)^{-1}$.

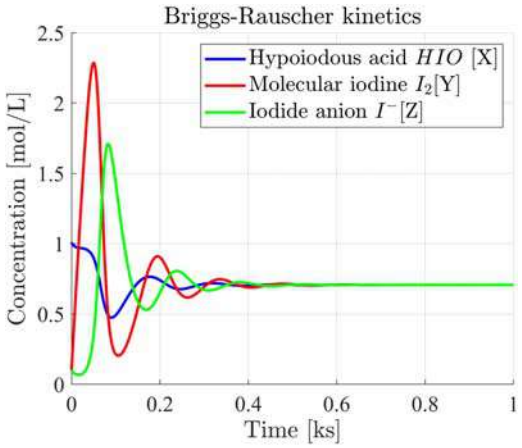


FIGURE 4.11

Time profiles of the asymptotically stable case 1.

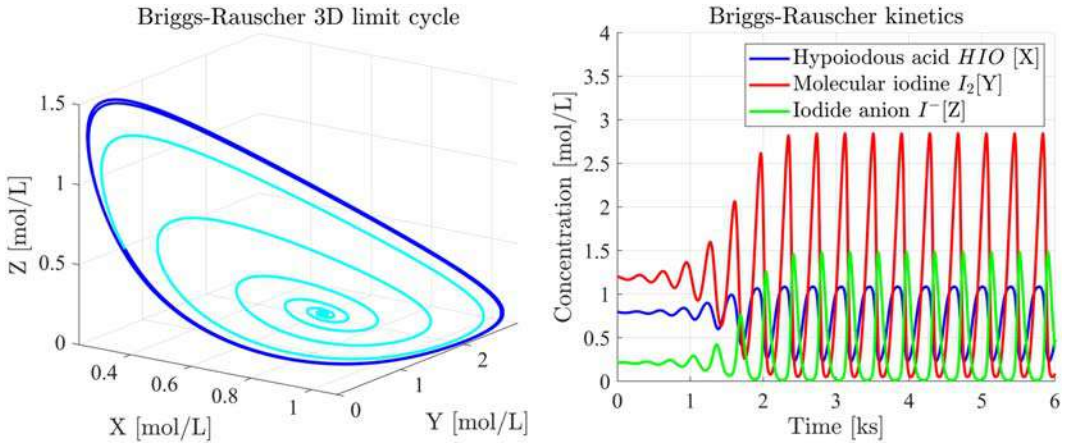


FIGURE 4.12

(Left) Limit cycle. (Right) Time profiles of the oscillating case 2.

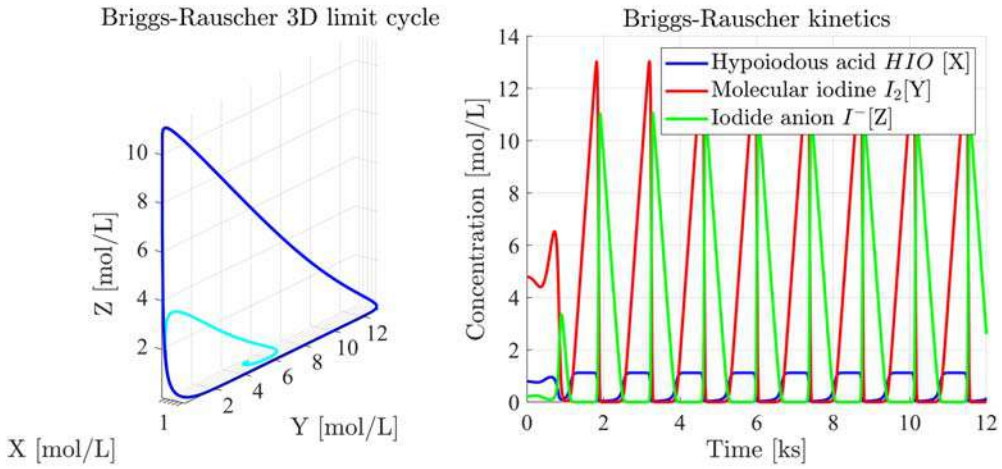


FIGURE 4.13

Case 3. (Left) Limit cycle. (Right) Time profiles.

4.6 Introduction to Belousov–Zhabotinsky kinetics

4.6.1 State equations

According to Reference [11], the BZ reaction is the prototype system for nonlinear chemical dynamics. It was accidentally discovered in 1950 by the Russian chemist B.P. Belousov (1893–1970), but it remained unknown partly because of a traditional skepticism about oscillating reactions, until one decade later A.M. Zhabotinksy (1938–2008) succeeded in publishing his results about several variants of the Belousov's reaction. In the early 1970s, R.J. Field, E. Körös (1927–2002), and R.M. Noyes (1919–2007), known also under the acronym FKN, proposed a detailed mechanism of the BZ reaction and were able to numerically simulate the resulting rate equations, proving the oscillatory behavior [5]. The FKN mechanism was further reduced to a much simpler model, the Oregonator (from the Oregon University where it was proposed, see Reference [12]). The standard Oregonator model is the sequence of five irreversible reactions written as follows:

$$\begin{array}{c} A+Y \Rightarrow X+P,\ r_1=k_1\,[A][Y] \\ X+Y \Rightarrow 2P,\ r_2=k_2\,[X][Y] \\ A+X \Rightarrow 2X+2Z,\ r_3=k_3\,[A][X] \\ 2X \Rightarrow A+P,\ r_4=k_4\,[X]^2 \\ B+Z \Rightarrow \dfrac{1}{2}fY,\ r_5=k_5\,[B][Z] \end{array} . \tag{4.42}$$

The species of the above set of reactions are:

$$\begin{array}{c} A=\mathrm{BrO_3^-}, B=\text{organic species such as the malonic acid } \mathrm{CH_2(COOH)_2} \\ P(\text{product})=\mathrm{HOBr} \text{ or } \mathrm{BrCH(COOH)_2} \\ X=\mathrm{HBrO_2}\ (\text{the activator species}), Y=\mathrm{Br^-} \\ Z=\text{the inhibitor species, oxidized form of the catalyst, Ce(IV)} \end{array} . \tag{4.43}$$

It is usually assumed that $[A]$ and $[B]$ are constant and denoted by A and B, respectively. Finally, f is an adjustable stoichiometric coefficient, and k_5 is an adjustable rate constant. The relevant state equations are as follows:

$$\begin{gathered}\frac{d[X]}{dt}(t) = k_1\,[A][Y] - k_2\,[X][Y] + k_3\,[A][X] - 2k_4[X]^2,\ [X](0) = [X]_0 \\ \frac{d[Y]}{dt}(t) = -k_1[A][Y] - k_2[X][Y] + \frac{f}{2}k_5[B][Z],\ [Y](0) = [Y]_0 \\ \frac{d[Z]}{dt}(t) = 2k_3[A][X] - k_5B[Z],\ [Z](0) = [Z]_0 \\ \frac{d[P]}{dt}(t) = k_1[A][Y] + 2k_2[X][Y] + k_4[X]^2,\ [P](0) = [P]_0 \\ \frac{d[A]}{dt}(t) = 0,\ [A]\,(0) = A \\ \frac{d[B]}{dt}(t) = 0,\ [B]\,(0) = B\end{gathered} \tag{4.44}$$

The typical parameter values are as follows:

$$\begin{gathered}A = 0.06\ \text{mol/L},\ B = 0.02\ \text{mol/L} \\ k_1 = 1.25\ (\text{mol/L})^{-1}\text{s}^{-1},\ k_2 = s_2\ 24\times10^3\ (\text{mol/L})^{-1}\text{s}^{-1},\ s_2 \geq 1 \\ k_3 = 100/3\ (\text{mol/L})^{-1}\text{s}^{-1},\ k_4 = 2.4\times10^3\ (\text{mol/L})^{-1}\text{s}^{-1} \\ k_5 = s_5\ (\text{mol/L})^{-1}\text{s}^{-1}, f > 0\end{gathered} \tag{4.45}$$

The previous values of $\{k_1, k_2, k_3, k_4\}$ derive from experimental data [12]. In the following, only the scale factor f will be varied for finding different behaviors of Eq. (4.44), whereas $s_2 = s_5 = 1$. The higher-level block diagram of the first three equations is shown in Fig. 4.14.

Of the three lower-level diagrams, only `OregonX` is shown in Fig. 4.15. The nonlinear combination of the different state variables in Eq. (4.44) is operated by the function `oregonx`, whose statements are below.

```
function u= oregonx(x,z,y,par)
    f=par(1);
    q=par(2);
    u=x*(1-x)+(q-x)*y;
end
```

The other two lower-level diagrams are left to the reader.

The following analysis is limited to the former three equations in Eq. (4.44), since the fourth one (the product rate) is just depending on their state variables and the last two ones are conservation equations. Analysis and solution are simplified by the following dimensionless state variables and time transformation typical of the literature [13]:

$$x = \frac{2k_4}{k_3A}X,\ y = \frac{k_2}{k_3A}Y,\ z = \frac{k_4k_5B}{(k_3A)^2}Z,\ \tau = k_5Bt. \tag{4.46}$$

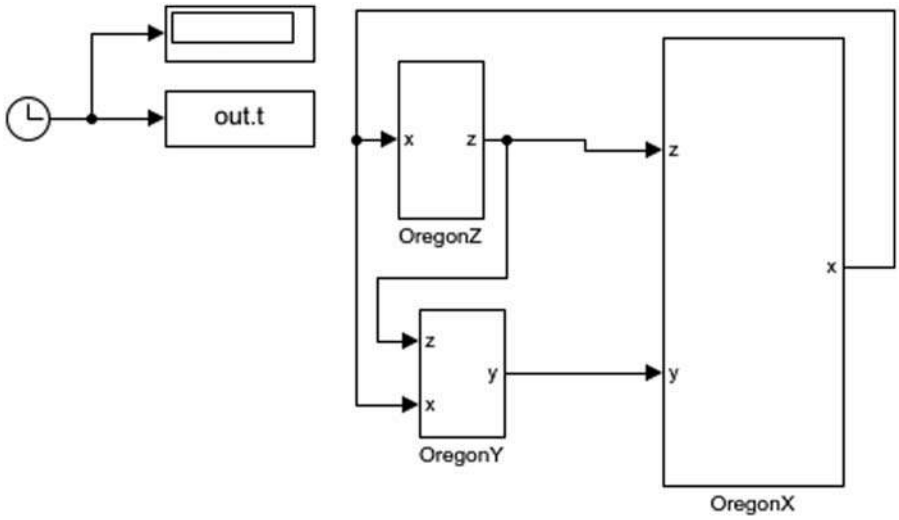


FIGURE 4.14

Higher-level block diagram of the first three equations in Eq. (4.44).

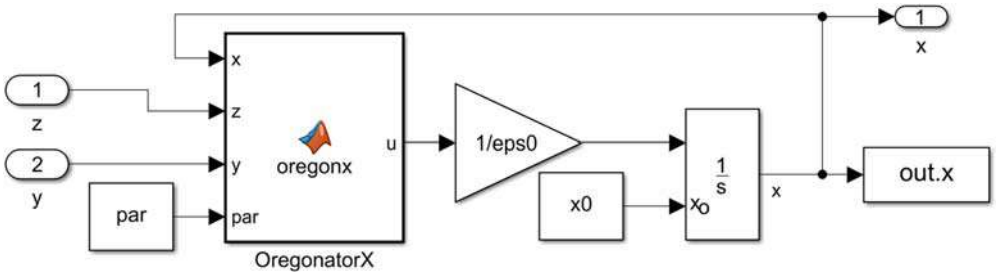


FIGURE 4.15

Lower-level diagram in Fig. 4.14.

The previous transformations lead to the dimensionless state equations:

$$\begin{aligned}\frac{dx}{d\tau}(t) &= \epsilon^{-1}((q-x)y + x(1-x)), x(0) = x_0\\ \frac{dy}{d\tau}(t) &= \eta^{-1}(-(q+x)y + fz), y(0) = y_0\\ \frac{dz}{d\tau}(t) &= x - z, z(0) = z_0\end{aligned} \quad . \tag{4.47}$$

The expression of the three dimensionless parameters $\{q, \epsilon, \eta\}$ is the following

$$q = \frac{2k_4k_1}{k_3k_2}, \epsilon = \frac{k_5B}{k_3A}, \eta = \frac{2k_4k_5B}{k_2k_3A}. \tag{4.48}$$

Their values are found from Eq. (4.45) to hold:

$$q = 0.0075/s_2,\ \epsilon = 0.01 s_5,\ \eta = 0.002\, s_5/s_2. \tag{4.49}$$

4.6.2 Equilibrium and stability analysis

Eq. (4.47) is third order like the Briggs–Rauscher Eq. (4.37), but here one may take the advantage of the small value of η in order to neglect $\eta\dot{y}(t) \simeq 0$ and solve the second equation for $y = (q+x)^{-1} fz$. The method, known as the *singular perturbation method* (see Appendix B), applies to state equations with very different time constants. It consists in treating equations with a high common factor (like here η^{-1}) as if they had already reached equilibrium, which implies to set the relevant state derivatives to zero. In fact, the equilibrium point of Eq. (4.47) holds:

$$\begin{aligned} \dot{x}(t) = 0 \quad &\Rightarrow \overline{z} = \frac{\overline{x}(\overline{x}-1)(q+\overline{x})}{f(q-\overline{x})} \\ \dot{y}(t) = 0 &\Rightarrow \overline{y} = \frac{f}{q+\overline{x}}\ \overline{x} \\ \dot{z}(t) = 0 &\Rightarrow \overline{z} = \overline{x} \end{aligned} \quad . \tag{4.50}$$

The first identity in Eq. (4.50) has not yet been solved for $\overline{x}$ since the first and third identities define the zero rate curves (*nullclines*) in the state plane $\{x, z\}$ (also known as the *phase plane*). The equilibrium point in the phase plane is the nullcline intersection. The nullclines for $f = 1, s_2 = 1$ are shown in Fig. 4.16. The three branches (decreasing branch close to the origin, increasing intermediate branch, and decreasing right branch) of the $\dot{x}$ nullcline have been marked with AS, instability, and AS, respectively, as well as with $\{f_{min}, f_{max}\}$, to be clarified below.

By replacing y into the first equation of Eq. (4.47), the reduced second-order state equation to be studied, becomes:

$$\begin{aligned} \epsilon \frac{dx}{d\tau}(t) &= (1-x)x + \frac{q-x}{q+x} fz, \quad x(0) = x_0 \\ \frac{dz}{d\tau}(t) &= x - z, \quad z(0) = z_0 \end{aligned} \quad . \tag{4.51}$$

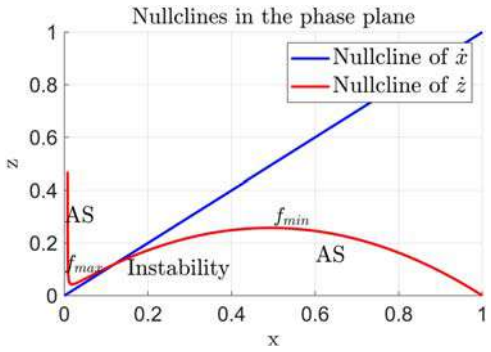


FIGURE 4.16

The first and third nullclines defined by Eq. (4.50).

We start by deriving the stability conditions of the equilibrium point by means of the perturbed equation of Eq. (4.51), the same method that was adopted in Section 4.5:

$$\begin{bmatrix} \delta\dot{x} \\ \delta\dot{z} \end{bmatrix}(t) = \begin{bmatrix} a(f,\bar{x})/\epsilon & b(f,\bar{x})/\epsilon \\ 1 & -1 \end{bmatrix} \begin{bmatrix} \delta x \\ \delta z \end{bmatrix}(t), \begin{bmatrix} \delta x \\ \delta z \end{bmatrix}(0) = \begin{bmatrix} \delta x_0 \\ \delta z_0 \end{bmatrix}. \tag{4.52}$$

The expression of the parameter pair $\{a, b\}$ is left to readers (it can be found in the Matlab script of Section 4.6.4). The parameters have been written as a function of f, since stability conditions will be found for a variable $f > 0$. Let us write the second-degree characteristic polynomial of the state matrix in Eq. (4.52) as follows:

$$\det(sI - A) = s^2 + a_1(f)s + a_0(f). \tag{4.53}$$

The Routh's necessary and sufficient conditions for AS (see Appendix B) require that

$$a_1(f) > 0, a_0(f) > 0. \tag{4.54}$$

The profile of both coefficients versus $f > 0$ is shown in Fig. 4.17. We find that $a_0(f) > 0$, whereas the sign of $a_1(f)$ changes twice, and becomes negative in the intermediate region defined by $f_{min} < f < f_{max}$, which therefore corresponds to *instability* conditions, whereas the left and right regions correspond to AS. The two boundary points $\{f_{min}, f_{max}\}$ correspond to the so-called *Hopf bifurcation*, where AS eigenvalues change into unstable eigenvalues because of a positive real part.

It can be proved that the three regions of f correspond to the three regions of the $\dot{x}$ nullcline (red line) in Fig. 4.16, in the sense that the equilibrium point (the intersection between the nullclines) will belong to left AS branch for $f > f_{max}$, to the intermediate unstable branch for $f_{max} > f > f_{min}$, and to the right AS branch for $f < f_{min}$. Simulated results will show free-response behaviors of the same kind as in Section 4.5. Specifically:

1. *Asymptotic stability*: perturbations from the equilibrium generate a trajectory returning to the equilibrium point.
2. *Unstable equilibrium*: perturbations from the equilibrium approach a bounded limit cycle where they remain confined.

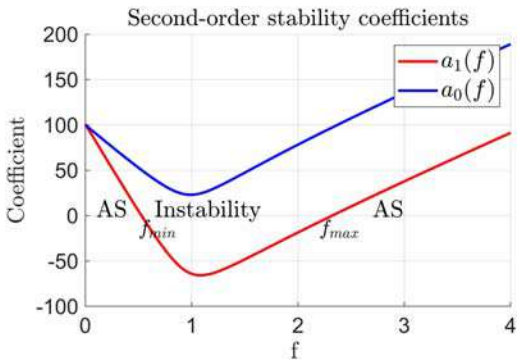


FIGURE 4.17

Coefficients of the characteristic polynomial in Eq. (4.53) versus f.

4.6.3 Simulated results

Three simulated results will be shown, corresponding to the three stability regions defined in section 4.6.2.

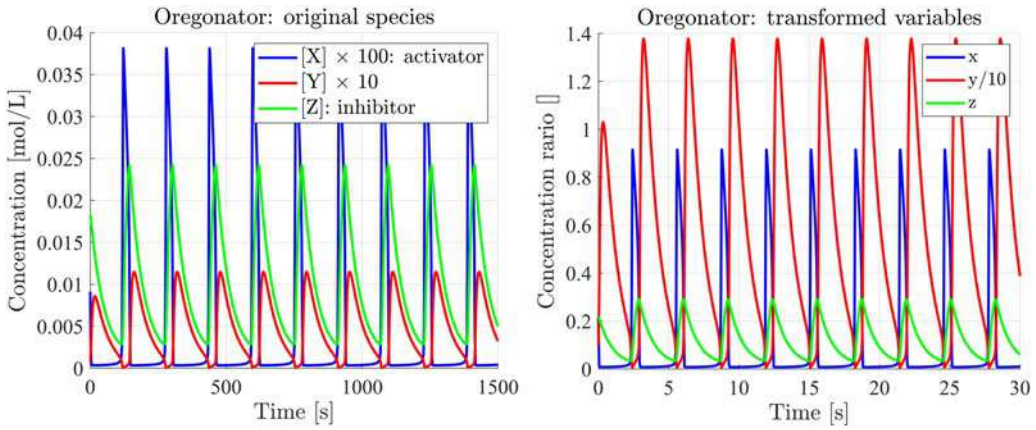


FIGURE 4.18

Case 1. Limit cycle time profiles reached from an unstable equilibrium. (Left) Original species concentrations and (Right) transformed state variables.

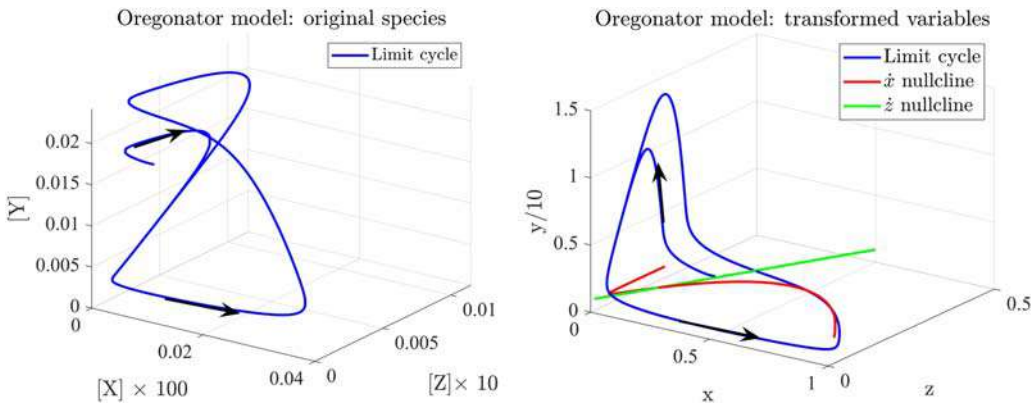


FIGURE 4.19

Case 1. Limit cycle reached from unstable equilibrium. (Left) Original species concentrations (unit: scaled $[\mathrm{mol/L}]$). (Right) Transformed variables. The $\dot{x}$ nullcline is in red color.

1. *Case 1, unstable equilibrium point:* the trajectory moves to a stable limit cycle, under $f_{min} < f < f_{max}$, in this case $f = 1$ and $s_5 = 1$ (see Figs. 4.18 and 4.19). Fig. 4.19, right,

shows the path direction (black arrows) and the $\{\dot{x}, \dot{z}\}$ nullclines in the xz plane. The starting point is close to an unstable equilibrium point on the intermediate branch of the $\dot{x}$ nullcline.

The perturbations of the transformed equilibrium are $\delta x = \delta y = \delta z = 0.1$. The three *Lyapunov exponents*, which confirm the existence of a stable limit cycle, are plotted in Fig. B.6, right, of Appendix B.

2. *Case 2, AS equilibrium point* for $f = 2.35$: the value is close to f_{max}, namely to the left Hopf bifurcation, which is the minimum point of the $\dot{x}$ nullcline between left and intermediate branches. Equilibrium has been slightly perturbed and the convergence to equilibrium occurs through damped oscillations (see Fig. 4.20) since eigenvalues are complex with negative real part. Perturbations from the transformed equilibrium in Eq. (4.50) are $\delta x = \delta y = \delta z = 0.005$.

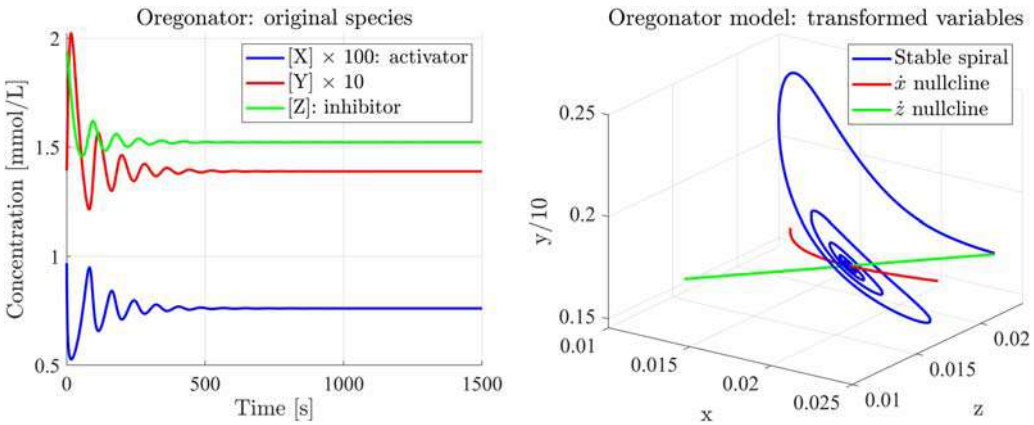


FIGURE 4.20

Case 2. (Left) Time profiles of the original species converging to scaled equilibrium concentrations (ordinate in mmol/L). (Right) The converging trajectory (3D logarithmic spiral) of the transformed state variables.

3. *Case 3, AS equilibrium point* for $f = 0.52$: the value is close to f_{min}, namely to the right Hopf bifurcation, which is the maximum point of the $\dot{x}$ nullcline between intermediate and right branches. Equilibrium has been strongly perturbed and the convergence to equilibrium occurs through a large excursion around the equilibrium point (see Fig. 4.21), which, close to the nullcline maximum, is reached after some damped oscillations. Equilibrium perturbations are $\delta x = \delta y = \delta z = 0.1$.

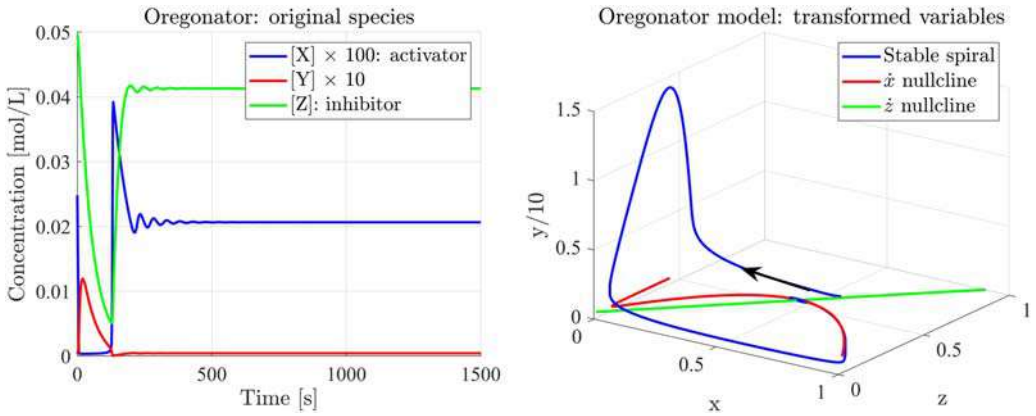


FIGURE 4.21

Case 3. (Left) Time profiles of the original species converging to scaled equilibrium concentrations. (Right) The converging trajectory of the transformed state variables.

4.6.4 Matlab script

The script includes (1) the computation of the $\dot{x}, \dot{z}$ nullclines, and their graphical plot as in Fig. 4.16, (2) the stability coefficients of the perturbation Eq. (4.52) and their graphical plot in Fig. 4.17, (3) the 3D plot of the limit cycle overlapping the $\{\dot{x}, \dot{z}\}$ nullclines as in Fig. 4.19, right, Fig. 4.20, right, and Fig. 4.21, right. The complete script `OregonatorW.m`, available in the companion website of the book [13], includes a section for plotting two figures of Appendix B, namely Fig. B.3, left and Fig. B.4. The selection of f and the perturbation vector δx, according to the three above cases, is done in the parameter section.

```
% --- Case selection
Case=3; % 1 2, 3
Chapter=4; % 4 and 12 (OregonatorW.m)
% scale parameter
fv=[1 ;2.35; 0.52]; % unstable , AS, AS  <<<< user
f0=fv(Case);
% Initial state perturbation of transformed concentrations
dx=[0.1; 0.005; 0.1];
% case 1 and 3, perturbationfrom transformed equilibrium
if Chapter==12, dx(1)=0.005;end
% --- End of case selection
```

The overall script follows.

```
%% Belousov-Zhabotinsky reaction (Chapter 4)
%% Oregonator model
InitChem;
disp('Belousov-Zhabotinsky kinetics: Oregonator model')
%% 1) Parameters
disp('1) Parameters');
% --- Case selection
Case=3; % 1 2, 3
Chapter=4; % 4 and  12 (OregonatorW.m)
% scale parameter
fv=[1 ;2.35; 0.52]; % unstable , AS, AS  <<<< user
f0=fv(Case);
% Initial state perturbation of transformed concentrations
dx=[0.1; 0.005; 0.1];
% case 1 and 3, perturbation from transformed equilibrium
if Chapter==12,  dx(1)=0.005;end
% --- End of case selection
% Rate constant and scales
s2=1;
k1=1.25;k2=2.4e4*s2;
k3=100/3;k4=2400;k5=1;
A=0.06; B=0.02;
q0=2*k4*k1/(k3*k2);
eps0=k5*B/(k3*A);
eta0=2*k4*k5*B/(k2*k3*A);
% Parameter vector
par(1)=f0;par(2)=q0;par(3)=eps0;
% Equilibrium:
xbar=0.5*(1-f0-q0 +sqrt((1-f0-q0)^2+4*q0*(1+f0)));
ybar=f0*xbar/(xbar+q0);
z0=xbar+dx(Case);x0=xbar+dx(Case);y0=ybar+dx(Case);
% Initial state  %% 2) Nullclines
disp('2) Nullclines');
x=0:0.00025:1;z1=x;
z2=x.*(x-1).*(x+q0)./(f0*(q0-x));indz2=find(z2>0);
figure('name', 'Nullclines');hold on; grid on;
plot(x,z1,'b');
plot(x(indz2),z2(indz2),'r');
xlabel('x');ylabel ('z');
legend('Nullcline of $\dot x$',
Nullcline of $\dot z$', 'FontSize',18);
```

```
title ('Nullclines in the phase plane');
text (0.15, 0.1,'Instability', 'FontSize',18);
text (0.0, 0.3,'AS', 'FontSize',18);
text (0.6, 0.15,'AS', 'FontSize',18);
text (0.0, 0.12,'$f_{max}$', 'FontSize',16);
text (0.5, 0.3,'$f_{min}$', 'FontSize',16);
%% 3) Stability analysis
disp('3) Stability analysis');
f=0:0.01:4;
% Equilibrium: versus f
xbarv=0.5*(1-f-q0 +sqrt((1-f-q0).^2+4*q0*(1+f)));
ybarv=f0*xbarv./(xbarv+q0);
% Perturbation equation parameters
av=(1-2*xbarv.*(1+q0*f./(q0+xbarv).^2))/eps0;
bv=(1/eps0)*f.*(q0-xbarv)./(q0+xbarv);
% Stability coefficients versus f
a1v=-av+1;a0v=-av-bv;
figure ('name', 'Stability');hold on; grid on;
plot(f,a1v,'r');plot(f,a0v,'b');
xlabel ('f ');
ylabel('Coefficient');
legend(' $a_1(f)$','$a_0(f)$','FontSize',18);
title('Second-order stability coefficients');
text (0.65, 5,'Instability', 'FontSize',18);
text (0.1, 5,'AS', 'FontSize',18);
text (2.7, 5,'AS', 'FontSize',18);
text (0.5, -15,'$f_{min}$', 'FontSize',16);
text (2.2, -15,'$f_{max}$', 'FontSize',16);
%% Time simulation
disp('4) Time simulation');
Ts=0.001/s2;
tInit=0;tEnd=30;
inttEnd=floor(tEnd/Ts);tEnd=inttEnd*Ts;
out=sim('OregonatorSim');
%% 5) Graphical results: time profiles
disp('5) Graphical results: time profiles');
t=out.t;dimt=length(t);
x=out.x;y=out.y;z=out.z;
% Transformation to original species
X=k3*A*x/(2*k4);
Y=k3*A*y/k2;
Z=(k3*A)^2*z/(k4*k5*B);
told=t/(k5*B);
```

```
% Transformed variables
figure('name','Output');hold on; grid on;
plot(t,x,'b');
plot(t,y/(10*s2),'r');
plot(t,z,'g');
xlabel('Time [s]');
ylabel('Concentration rario []');
title('Oregonator: transformed variables');
legend('x','y/10','z');hold off
% Original species
scale=1;
if Case==2, scale=1000; end
figure('name','Original output');hold on; grid on;
plot(told,X*100*scale,'b');
plot(told,Y*10*scale,'r');
plot(told,Z*scale,'g');
xlabel('Time [s]');
ylabel('Concentration [mol/L]');
if Case==2
   ylabel('Concentration [mmol/L]');
end
title('Oregonator: original species');
legend('[X] $\times$ 100: activator ','[Y] $\times$  10',...
   '[Z]: inhibitor','FontSize', 16);hold off
%% 6) Graphical results: limit cycle
disp('6) Graphical results: limit cycle');
scaley=10*s2;
% Nullclines
z1=x;z2=x.*(1-x).*(x+q0)./(f0*(x-q0));
ybar3=ybar*ones(dimt,1)/scaley;
ys=y/scaley;
figure('name', 'LimitCycle: transformed');hold on; grid on;
plot3(x,z,ys,'b');
plot3(x,z2,ybar3,'r');
plot3(x,z1,ybar3,'g');
xlabel('x');ylabel('z');zlabel('y/10');
title('Oregonator model: transformed variables');
legend('Limit cycle', '$\dot x$ nullcline',...
   '$\dot z$ nullcline' );
```

```
switch Case
case 1
   ylim([0 0.5]);
   annotation('arrow',[0.27 0.26],[0.46 0.60]);
   annotation('arrow',[0.30 0.45],[0.22 0.18]);
   legend('Limit cycle', '$\dot x$ nullcline',...
        '$\dot z$ nullcline' ,'FontSize', 16 );
case 2
   legend('Stable spiral', '$\dot x$ nullcline',...
        '$\dot z$ nullcline' ,'FontSize', 16 );
case 3
   annotation('arrow',[0.533 0.41], [0.31 0.36]);
   legend('Stable spiral' ,'$\dot x$ nullcline',...
        '$\dot z$ nullcline' ,'FontSize', 16);
end
view(35,25);
% Original species
figure('name', 'Limit cycle: original');hold on; grid on;
plot3(X*100,Y*10,Z,'b');
xlabel('[X] $\times$ 100 ');
ylabel('[Z]$\times$ 10');
zlabel('[Y]');
if Case ==1
   annotation('arrow',[0.25 0.345],[0.64 0.68]);
   annotation('arrow',[0.31 0.45],[0.265 0.23]);
end
title('Oregonator model: original species');
legend('Limit cycle' );view(35,25);
```

References

[1] E. Canuto, C. Novara, L. Massotti, D. Carlucci, C. Perez Montenegro, Spacecraft Dynamics and Control: The Embedded Model Control Approach, Butterworth-Heinemann (Elsevier), 2018.

[2] E. Canuto, D. Mazza, Introduction to population dynamics and resource exploitation, in ArXiv.org > q-bio, doc. ArXiv:2102.01205, January 2021.

[3] Wikipedia, *Hopf bifurcation*, https://en.wikipedia.org/wiki/Hopf_bifurcation.

[4] L. Gyorgyi, R.J. Field, A three-variable model of deterministic chaos in the Belousov-Zhabotinsky reaction, Nature 355 (1992) 808–810.

[5] R.J. Field, E. Körös, R.M. Noyes, Oscillations in chemical systems II. Thorough analysis of temporal oscillations in the Ce-BrO_3-malonic acid system, J. Am. Chem. Soc. 94 (1972) 8649–8664.

[6] Companion Website of the Book. Elsevier, 2021. Available from: https://www.elsevier.com/books-and-journals/book-companion/9780323913416.

[7] D. Voet, J.G. Voet, Biochemistry, J. Wiley & Sons, 2011. Chapter 14.

[8] M. Tuckerman, Oscillating reactions, LibreTexts, Chemistry, 2020. Chapter 9.

[9] Wikipedia, *Autocatalysis*, https://en.wikipedia.org/wiki/Autocatalysis, 2020.
[10] O. Gurel, D. Gurel, Oscillations in chemical reactions, Topics in Current Chemistry 118, Springer Verlag, Berlin, 1983.
[11] F. Sagués, I.R. Epstein, Nonlinear chemical dynamics, Dalton Trans. (2003) 1201–1217.
[12] R.J. Field, *Oregonator*, Scholarpedia, 2007, http://www.scholarpedia.org/article/Oregonator.
[13] J.J. Tyson, A quantitative account of oscillations, bistability, and traveling waves in the Belousov-Zhabotinsky reaction, in: R.J. Field, M. Burger (Eds.), Oscillations and Traveling Waves in Chemical Systems, J. Wiley & Sons, New York, 1985.

CHAPTER

Gaseous reactions and equilibria aided by Matlab

5

5.1 The second law of thermodynamics

In dealing with chemical transformations, the first question to be addressed is whether, under the transformation circumstances, the process is spontaneous. The criterion of the entropy increase does not apply directly, and, as established by J.W. Gibbs (1839−1903), one should account for two entropy changes to be simultaneously considered: the change of the system itself, ΔS_{sys}, and the change of the surroundings ΔS_{surr}, their sum being the entropy variation of the whole, generally speaking, of the universe:

$$\Delta S_{univ} = \Delta S_{sys} + \Delta S_{surr} > 0. \tag{5.1}$$

Expression (5.1) provides a basic criterion to discriminate spontaneous changes. It is indeed one way of stating the second law of thermodynamics. In other words, all the spontaneous processes produce an increase in the entropy of the universe.

According to expression (5.1), if a process produces positive entropy changes in both the system and the surroundings, the process is surely spontaneous. If both entropy changes are negative, the process occurs to be nonspontaneous. If one of the entropy changes is positive and the other negative, whether the sum is positive or negative depends on the relative magnitude of the pair of changes.

Unfortunately, the evaluation of the entropy change for the universe is difficult, and a criterion to be applied to the system itself is preferable, without having to worry about changes in the surroundings. An entropy change, formally ΔS, is based on two measurable quantities: heat and temperature; they both affect the availability of energy levels to the microscopic particles of a system. This is stated by the same entropy definition in the second principle of thermodynamics [1,2], namely that

$$\Delta S = \frac{q_{rev}}{T}, \tag{5.2}$$

where q_{rev} is the heat transferred from surrounding to system (positive sign) or from system to surrounding (negative sign) in a reversible manner. In a chemical isothermal transformation when heat is exchanged from the system to the surrounding, Eq. (5.2) becomes $\Delta S_{surr} = -\Delta H_{sys}/T$. By substituting the last definition in Eq. (5.1) and multiplying both terms by $-T$, we obtain an equation, which is valid for an isothermal transformation:

$$-T\,\Delta S_{univ} = \Delta H_{sys} - T\,\Delta S_{sys}. \tag{5.3}$$

The right side of Eq. (5.3) only involves the system, whereas the left side shows the term corresponding to the criterion of spontaneous process, like chemical reactions. Eq. (5.3) is usually

Fundamental Chemistry with Matlab. DOI: https://doi.org/10.1016/B978-0-323-91341-6.00017-3

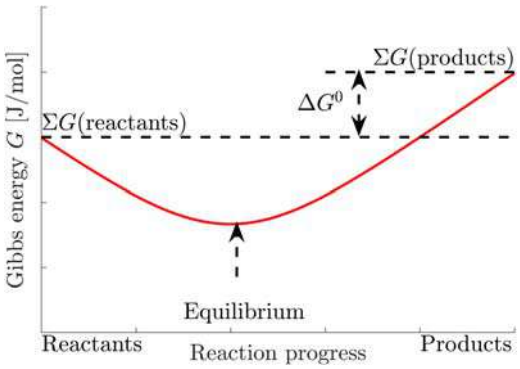


FIGURE 5.1

Chemical reactions reach an equilibrium state at a minimal Gibbs energy.

modified by introducing a new thermodynamic function, known as *free* (*or Gibbs*) *energy*, denoted by G. It is defined by the identity

$$\Delta G = -T\,\Delta S_{univ}. \tag{5.4}$$

At the end, one obtains the identity

$$\Delta G = \Delta H - T\,\Delta S, \tag{5.5}$$

where, since all the terms refer to system changes, the subscript *sys* has been omitted. Now, by observing that ΔG is negative when ΔS_{univ} is positive, we can establish the final criterion for spontaneous change, only based on the properties of the system itself.

For a process occurring at constant T and P, the following statements hold TRUE.

1. If $\Delta G < 0$ (negative), the process is spontaneous.
2. If $\Delta G > 0$ (positive), the process is nonspontaneous.
3. If $\Delta G = 0$ (zero), the process is reversible and the system has reached an equilibrium.

At equilibrium, there is no driving force of the reaction, so there is no further reaction and the Gibbs energy must be at a minimum, as shown in Fig. 5.1.

5.2 Application of the Gibbs energy criterion to chemical reactions

The Gibbs energy variation of a system is useful for predicting the spontaneity of changes under the condition of constant T and P, a condition which is frequently encountered, for instance in phase transitions and chemical reactions. In such situations, the Gibbs energy variation arises from changes in the amount of substances, that is, from a change in the system composition. In a chemical reaction, composition changes because certain amounts of reactants are converted into products.

Before discussing the variation of the chemical functions, H and G, it is necessary to define their standard states. The entropy S, according to the third law of thermodynamics [2], is zero at $T = 0$ K, so it is possible to define the entropy content of whatever amount of substance according to entropy increase by supposing to heat the same in a reversible manner from 0 K to T, that is, $S = \int_0^T \delta q/T$. If $T = 289.15$K (or 25°C) and the pressure is $P = 101.325$ kPa $= 1$ atm, we are in the *standard state* at room temperature (see the section about 'Standard conditions of temperature and pressure' in the book Introduction) and S will be denoted by S^0. If not otherwise specified, S^0 refers to 1 mol of substance.

On the contrary, for H and G, it would be arbitrary to spot a condition for which $H = 0$ or $G = 0$; therefore, in chemical thermodynamics, H^0 is defined as the enthalpy variation under T, P standard conditions during the formation of 1 mol of the compound from its constituent elements, all of them being in their standard state. For instance, H^0 for water refers to the reaction in which 1 mol of O_2 reacts with 2 mol of H_2 to obtain 1 mol of liquid H_2O, at 25°C and $P = 1$ atm. H^0 is usually referred to as the standard heat of formation of a substance. If the substance is an element, then $H^0 = 0$.

From Eq. (5.5), it follows that it is possible to define G^0 as the standard free energy (or the Gibbs energy) of the formation for any amount of substance as follows:

$$G^0 = H - T\,\Delta S^0, \tag{5.6}$$

where ΔS^0 is the entropy difference between substance and the constituent elements. Usually, G^0 refers to 1 mol of a certain substance, at T, P standard conditions.

As a consequence, by considering a generic chemical reaction, the *standard Gibbs energy of reaction*, indicated by $\Delta_{(r)}G^0$, is defined as the Gibbs energy change in a process which involves the complete conversion of the stoichiometric amounts of pure, unmixed reactants in their standard states, to stoichiometric amounts of pure, unmixed products in their standard states. The specific change $\Delta_{(r)}G^0$ of a process can be obtained from tabulated thermochemical data of pure substances in their standard states [1]. For a generic reaction, like $aA + bB \rightarrow cC + dD$, it holds

$$\Delta_{(r)}G^0 = c\Delta G^0(C) + \Delta G^0(D) - a\Delta G^0(A) - b\Delta G^0(B), \tag{5.7}$$

where ΔG^0 is the *molar Gibbs energy of formation* of a pure substance, like A, B, C, D. The Gibbs energies of formation refers to 1 mol of substance, and therefore they are to be multiplied by the proper stoichiometric coefficients $\{a, b, c, d\}$ as in Eq. (5.6).

This simple scheme allows to predict whether a reaction occurs spontaneously from left to right. But what occurs, if reactants and products are not pure substances in their standard states, but vary in concentrations? To predict direction and amount of spontaneous chemical changes, we need an equation that relates a change of G to a change in composition. With the help of this equation, we could decide whether G will increase or decrease when system composition changes by any amount.

Let us consider the system of the gaseous substances (g), N_2(g), H_2(g), and NH_3(g), treated as ideal gases, and let the triple $\{n_1, n_2, n_3\}$ denote the initial amounts, in moles, of each gas. Suppose that $\{n_1, n_2, n_3\}$ are not yet the equilibrium values, hence the system composition will change, at constant T and constant P, because the ammonia reaction $N_2 + 3H_2 \rightarrow 2NH_3$ advances in order to

reduce $\Delta G_{(r)}$ by a certain amount until a minimum is reached. The reaction, at least in the initial stage, is therefore spontaneous, as shown in Table 5.1, where $x > 0$.

The variable x denotes the molar amount of the chemicals which are transformed due the reaction progress and is known in chemistry as the *reaction coordinate*.

Table 5.1 Progress of the ammonia reaction.

State	N_2 (gas)	H_2 (gas)	NH_3 (gas)	Total moles
Initial state	n_1	n_2	n_3	$n_1 + n_2 + n_3$
Reacted amount	$-x$	$-3x$	$+2x$	$-2x$
Final state	$n_1 - x$	$n_2 - 3x$	$n_3 + 2x$	$n_1 + n_2 + n_3 - 2x$

The next key point is to relate the reaction coordinate to the transformation ΔG. The relevant relation should provide a quantitative criterion for detecting to which degree the reaction spontaneously occurs ($\Delta G < 0$) and for discovering when the equilibrium state ($\Delta G = 0$) has been reached.

The expression of ΔG is obtained by the following steps. First, we must recall [2] that the entropy of an (ideal) gas depends on the pressure as follows

$$S_T = S_T^0 - R \log\left(\frac{P}{P_{\text{standard}} = 0.1 \text{ MPa}}\right) = S_T^0 - R \log(P), \tag{5.8}$$

where $R = 8.314 \text{ JK}^{-1}\text{mol}^{-1}$ is the gas constant, and the term S_T^0 refers to standard pressure at temperature T and S_T refers to $\{T, P\}$ general conditions. By applying Eq. (5.8) to each component of the ammonia reaction in Table 5.1, we obtain (the deponent T has been omitted for clarity);

$$S_{NH3} = S_{NH3}^0 - R \log(P_{NH3}), S_{N2} = S_{N2}^0 - R \log(P_{N2}), S_{H2} = S_{H2}^0 - R \log(P_{H2}). \tag{5.9}$$

If we substitute the above expressions in $\Delta S_{reaction} = 2S_{NH3} - S_{NH3} - 3S_{NH3}$, we find:

$$\Delta S_{reaction} = 2S_{NH3}^0 - S_{NH3}^0 - 3S_{NH3}^0 - 2R \log(P_{NH3}) + R \log(P_{N2}) + 3\log(P_{H2}) \tag{5.10}$$

and

$$\Delta S = \Delta S^0 + R \log\left(\frac{P_{N2} P_{H2}^3}{P_{NH3}^2}\right) = \Delta S^0 - R \log\left(\frac{P_{NH3}^2}{P_{N2} P_{H2}^3}\right). \tag{5.11}$$

The Gibbs equation for standard pressure holds $\Delta G_T^0 = \Delta H_T^0 - T\Delta S_T^0$.

If pressures are no more standard, the last expression turns into $\Delta G_{T,P} = \Delta H_T^0 - T\Delta S_{T,P}$, where, by substituting Eq. (5.5), we obtain:

$$\begin{aligned} \Delta G_{T,P} &= \Delta H_T^0 - T\Delta S_T^0 + RT \log\left(\frac{P_{NH3}^2}{P_{N2} P_{H2}^3}\right) = \\ &= \Delta G^0 + RT \log\left(\frac{P_{NH3}^2}{P_{N2} P_{H2}^3}\right) = \Delta G^0 + RT \log(Q) \end{aligned}, \tag{5.12}$$

where Q is known as the reaction quotient.

Eq. (5.6) holds for every value of pressure of reactants and products, thus obtaining a variety of ΔG values. When the pressure is exactly equal to the equilibrium value, $\Delta G = 0$, as it will be recalled in Section 5.3.

5.3 Relationship of ΔG with the equilibrium constant K_P

We encounter an interesting situation when we apply Eq. (5.6) to a generic reaction at equilibrium. We have learned that $\Delta G = 0$ at equilibrium. When a generic reaction like $aA + bB \Leftrightarrow cC + dD$ is at equilibrium, we can write $Q = K_P$, where K_P is *the equilibrium constant* expressed in terms of partial pressures:

$$K_P = \frac{P_C^c P_D^d}{P_A^a P_B^b}, \tag{5.13}$$

or in the case of ammonia synthesis:

$$K_p = \frac{P_{\mathrm{NH3}}^2}{P_{\mathrm{N2}} P_{\mathrm{H2}}^3}. \tag{5.14}$$

Therefore, at equilibrium, the identity $\Delta G_{T,P} = \Delta G_T^0 + RT \log(K_P) = 0$ implies

$$\Delta G_T^0 = -RT \log(K_P) \Rightarrow K_P = \exp\left(\frac{-\Delta G_T^0}{RT}\right). \tag{5.15}$$

It follows that, if we know ΔG^0, we can use Eq. (5.15) to compute the equilibrium constant K_P. This means that standard thermodynamic data can be used as a direct source of equilibrium constants at any temperature.

Since the values of S_T^0, H_T^0, G_T^0 markedly vary with the temperature, particularly the entropy, it is not possible to use their values at the standard temperature say, $S_{25}^0, H_{25}^0, G_{25}^0$. As we shall see in Section 5.4, the values at the standard temperature can be extrapolated to other temperatures with proper coefficients, like the NASA CEA (Chemical Equilibrium with Applications) interpolation coefficients [3] which are reported in the table of Fig. 5.2.

NASA Coefficients

Ar	1	1000	6000	2.011E+01	-5.993E-02	2.500E+00	-3.992E-08	1.205E-11	-1.819E-15	1.079E-19	-7.450E+02	4.379E+00
CaCO3	1	500	1603	-2.584E+05	0.000E+00	1.197E+01	3.264E-03	0.000E+00	0.000E+00	0.000E+00	-1.497E+05	-5.961E+01
CaO	1	500	3172	-1.459E+05	0.000E+00	7.174E+00	-1.960E-03	1.291E-06	-2.077E-10	0.000E+00	-7.892E+04	-3.659E+01
CaSO4	1	500	1473	-2.984E+05	0.000E+00	1.360E+01	5.857E-03	0.000E+00	0.000E+00	0.000E+00	-1.778E+05	-6.802E+01
CaSO4	2	1473	1733	0.000E+00	0.000E+00	1.984E+01	0.000E+00	0.000E+00	0.000E+00	0.000E+00	-1.798E+05	-1.045E+02
CaSO4	3	1733	6000	0.000E+00	0.000E+00	1.984E+01	0.000E+00	0.000E+00	0.000E+00	0.000E+00	-1.765E+05	-1.026E+02
CH4	1	200	1000	-1.767E+05	2.786E+03	-1.203E+01	3.918E-02	-3.619E-05	2.027E-08	-4.977E-12	-2.331E+04	8.904E+01
CH4	2	1000	6000	3.730E+06	-1.384E+04	2.049E+01	-1.962E-03	4.727E-07	-3.729E-11	1.624E-15	7.532E+04	-1.219E+02
CO	1	200	1000	1.489E+04	-2.922E+02	5.725E+00	-8.176E-03	1.457E-05	-1.088E-08	3.028E-12	-1.303E+04	-7.859E+00
CO	2	1000	6000	4.619E+05	-1.945E+03	5.917E+00	-5.664E-04	1.399E-07	-1.788E-11	9.621E-16	-2.466E+03	-1.387E+01
CO2	1	200	1000	4.944E+04	-6.264E+02	5.302E+00	2.504E-03	-2.127E-07	-7.690E-10	2.850E-13	-4.528E+04	-7.048E+00
CO2	2	1000	6000	1.177E+05	-1.789E+03	8.292E+00	-9.223E-05	4.864E-09	-1.891E-12	6.330E-16	-3.908E+04	-2.653E+01
H2	1	200	1000	4.078E+04	-8.009E+02	8.215E+00	-1.270E-02	1.754E-05	-1.203E-08	3.368E-12	2.682E+03	-3.044E+01
H2	2	1000	6000	5.608E+05	-8.372E+02	2.975E+00	1.252E-03	-3.741E-07	5.937E-11	-3.607E-15	5.340E+03	-2.203E+00
H2O	1	200	1000	-3.948E+04	5.756E+02	9.318E-01	7.223E-03	-7.343E-06	4.955E-09	-1.337E-12	-3.304E+04	1.724E+01
H2O	2	1000	6000	1.035E+06	-2.413E+03	4.646E+00	2.292E-03	-6.837E-07	9.426E-11	-4.822E-15	-1.384E+04	-7.978E+00
H2Ocl	1	200	273	-4.027E+05	2.748E+03	5.738E+01	-8.268E-01	4.413E-03	-1.054E-05	9.694E-09	-5.530E+04	-1.903E+02
H2Ocl	2	273	373	1.326E+09	-2.448E+07	1.879E+05	-7.679E+02	1.762E+00	-2.151E-03	1.093E-06	1.102E+08	-9.780E+05
H2Ocl	3	373	600	1.264E+09	-1.680E+07	9.278E+04	-2.722E+02	4.479E-01	-3.919E-04	1.426E-07	8.113E+07	-5.134E+05
N2	1	200	1000	2.210E+04	-3.818E+02	6.083E+00	-8.531E-03	1.385E-05	-9.626E-09	2.520E-12	7.108E+02	-1.076E+01
N2	2	1000	6000	5.877E+05	-2.239E+03	6.067E+00	-6.140E-04	1.492E-07	-1.923E-11	1.062E-15	1.283E+04	-1.587E+01
NH3	1	200	1000	-7.681E+04	1.271E+03	-3.893E+00	2.146E-02	-2.184E-05	1.317E-08	-3.332E-12	-1.265E+04	4.366E+01
O2	1	200	1000	-3.426E+04	4.847E+02	1.119E+00	4.294E-03	-6.836E-07	-2.023E-09	1.039E-12	-3.391E+03	1.850E+01
O2	2	1000	6000	-1.038E+06	2.345E+03	1.820E+00	1.268E-03	-2.188E-07	2.054E-11	-8.193E-16	-1.689E+04	1.739E+01
SO2	1	200	1000	-5.311E+04	9.090E+02	-2.357E+00	2.204E-02	-2.511E-05	1.446E-08	-3.369E-12	-4.114E+04	4.046E+01
SO2	2	1000	6000	-1.128E+05	-8.252E+02	7.616E+00	-1.999E-04	5.656E-08	-5.454E-12	2.918E-16	-3.351E+04	-1.656E+01
SO3	1	200	1000	-3.953E+04	6.209E+02	-1.438E+00	2.764E-02	-3.145E-05	1.793E-08	-4.126E-12	-5.184E+04	3.391E+01
SO3	2	1000	6000	-2.167E+05	-1.301E+03	1.096E+01	-3.837E-04	8.467E-08	-9.705E-12	4.498E-16	-4.398E+04	-3.655E+01

FIGURE 5.2

Table of the NASA CEA thermochemical coefficients (see Section 5.5).

Once we have calculated a value for ΔG_T^0 by employing the above parameterization, the equilibrium is directly solved, thus providing the equilibrium partial pressures (and the concentrations as well). All we need is to substitute the values of the equilibrium pressures and solve a unique algebraic equation with a single unknown. By continuing on the ammonia reaction in Table 5.1 and by recalling Dalton's law, the partial pressure of a single component in a gaseous mixture equals the mole fraction multiplied by total pressure P, as follows

$$P_{\mathrm{NH3}} = P_{TOT}\frac{n_3+2x}{n_T},\quad P_{\mathrm{N2}} = P_{TOT}\frac{n_1-x}{n_T},\quad P_{\mathrm{H2}} = P_{TOT}\frac{n_2-3x}{n_T}, \tag{5.16}$$

where x is the *reaction coordinate* to be found, already encountered in Table 5.1.

From Eqs. (5.12) and (5.15), we obtain the following expression of the equilibrium constant for the ammonia reaction:

$$K_P = \frac{P_{\mathrm{NH3}}^2}{P_{\mathrm{N2}}P_{\mathrm{H2}}^3} = \frac{\left(\frac{n_3+2x}{n}\right)^2}{\frac{n_1-x}{n}\left(\frac{n_2-3x}{n}\right)^3}P_{TOT}^{-2},\ n = n_1+n_2+n_3-2x. \tag{5.17}$$

If we assume that the initial composition consists of 1 mol N_2, 3 mol H_2, and 0 mol NH_3, the total mole n amounts to $n = 4-2x$. Under this assumption, the molar ratio H_2/N_2 is equal to the stoichiometric ratio 3. By rewriting Eq. (5.17) with $n_1 = 1, n_2 = 3$, and $n_3 = 0$, we obtain the following algebraic equation in the unknown x:

$$27K_PP^2(1-x)^4 - 16(x(2-x))^2 = 0. \tag{5.18}$$

In the Matlab® scripts to be detailed in Section 5.6 and the following up to Section 5.13, Eq. (5.18) is solved by exploiting the symbolic function `vpasolve()` as follows:

```
y1=Kp*P^2*27*(1-x)^4-16*(x*(2-x))^2;
S = double(vpasolve(y1 == 0, x, [0 1]));
```

5.4 The value of ΔS, ΔH, and ΔG as a function of temperature

A thermodynamic database for chemical substances usually refers to standard conditions, but also offers the possibility, by proper interpolation, to study the equilibrium conditions of chemical reactions at different temperatures, even higher or very higher (for instance up to 6000K). In order to perform this task, we can use the NASA CEA thermodynamic database, a collection of interpolated coefficients which are available for thousands of substances (see Section 5.5) [4]. They allow us to extrapolate enthalpy and entropy in a wide range of temperatures and thereafter to calculate ΔG for chemical reactions.

More precisely, the nine coefficients of the database (the last nine columns in the table of Fig. 5.2) express C_P (the specific molar heat) of each substance as a function of the absolute temperature in a given range (the third and fourth columns in Fig. 5.2) [3]. The database was developed to extrapolate the thermodynamic properties of individual species starting from their properties at standard conditions. The NASA CEA computer code uses these data (see Fig. 5.2), as documented in Reference [3].

Seven out of nine coefficients, namely $a_1, \ldots, a_7$, interpolate C_P as follows:

$$C_P^0(T)/R = a_1 T^{-2} + a_2 T^{-1} + a_3 + a_4 T + a_5 T^2 + a_6 T^3 + a_7 T^4, \tag{5.19}$$

where T is the absolute temperature in K and $R = 8.314\ \mathrm{JK^{-1}mol^{-1}}$ is the universal gas constant in the international system of units (SI).

Enthalpy and entropy are obtained from Eq. (5.19) by integrating the $C_P^0(T)$ and $C_P^0(T)/T$ expressions with respect to T, the last two coefficients a_8, a_9 being the respective integration constants. We obtain, respectively,

$$H_T^0/R = -a_1 T^{-1} + a_2 \log(T) + a_3 T + a_4 \frac{T^2}{2} + a_5 \frac{T^3}{3} + a_6 \frac{T^4}{4} + a_7 \frac{T^5}{5} + a_8, \tag{5.20}$$

and

$$S_T^0/R = -a_1 \frac{T^{-2}}{2} - a_2 T^{-1} + a_3 \log(T) + a_4 T + a_5 \frac{T^2}{2} + a_6 \frac{T^3}{3} + a_7 \frac{T^4}{4} + a_9. \tag{5.21}$$

The background of Eqs. (5.20) and (5.21) derives from the thermodynamic relationships between the specific heat C_P and the difference of H and S with respect to their reference values H_{25}^0 and S_{25}^0, which are defined at $T_0 = T_{\text{standard}}$ and $P_0 = P_{\text{standard}}$, (see [1], pp. 112−114):

$$\Delta H_T^0 = H_T^0 - H_{25}^0 = \int_{25}^{T} C_P(T)\delta T,\ \Delta S_T^0 = S_T^0 - S_{25}^0 = \int_{25}^{T} C_P(T)\delta T/T. \tag{5.22}$$

The identities in Eqs. (5.20) and (5.21) have been found to provide a reasonable approximation of the thermodynamic quantities over a wide range of temperatures.

In the Matlab scripts to be explained starting from Section 5.6, the nine coefficients $\{a_k, k = 1, \ldots, 9\}$ are read from an excel data sheet and then employed to compute ΔH and ΔS at any given temperature for each reactant and product. $\Delta G(T)$ is computed at the temperature T for each species by means of Eq. (5.12). For instance, if the reaction is $aA + bB \Rightarrow cC + dD$, then $\Delta G^0(T) = cG_c + dG_d - aG_a - bG_b$.

The value of $\Delta G^0(T)$ is inserted in Eq. (5.15), and the equilibrium constant K_P is computed. $\Delta G^0(T)$ denotes the reaction ΔG at the standard pressure of 1 atm for each species, but at a temperature $T \neq T_0$. The effect of the pressure is accounted for by solving an algebraic equation in terms of the reaction coordinate x like in Eq. (5.18), which implies that the results are valid for whatsoever temperature and pressure.

5.5 The table of the NASA CEA thermochemical coefficients

With reference to the table of Fig. 5.2, the column description is as follows.

1. The first column contains the substance symbol, where H_2O stands for water gas and H_2O_{cl} stands for water ice, liquid, and gas phases.
2. The second column contains the progressive row number of the same substance, and each row refers to a different temperature range.
3. The third and fourth columns contain the minimum and maximum values of the temperature range $T_{min} \leq T < T_{max}$ [K] where coefficients apply.

4. Columns 5−13 contain the nine coefficients $\{a_k, k = 1, \ldots, 9\}$ of the appropriate temperature functions $f_k(T)$ in Eqs. (5.20) and (5.21). Only four significant figures are reported. The complete excel file `NineCoeffs.xlsx` can be found in the companion web site of the book [5]. Readers can complete the table at their will, by respecting the above rules.

5.6 Introduction to Matlab scripts

Fig. 5.2 shows the NASA CEA thermochemical coefficients to be used in the scripts of this chapter.

5.6.1 Organization of the Matlab scripts

The following statements are taken from the script treating hydrogen production at high and low temperature in Section 5.10.

The scripts of this chapter include, as a baseline, four parts, each part starting with a comment headed by %% (in green color).

1. *Initialization*: the common script `InitChem` (in Appendix D) is called, common parameters are defined including the boolean `logDisp` which regulates the result printout (zero value = short summary).
2. *User data:* reaction parameters and temperature ranges can be defined by users according to the following script taken from Section 5.10.2, where two different cases are treated, being denoted with the dimension `dimc = 2`. For each case the total number of substances and the number of reactants must be defined in the dimension vectors `dims = [4;4]` and `dimr = [2;2]`.Each case requires the substance names and their stoichiometry in `sr = cell(dimc,4)` and `cr = zeros (dimc,4)`. Finally a single temperature range and step must be given, like `rTemp = [0;1500;50]`.

```
%% Reaction parameters and temperature
disp('Reaction 1:  CH4 + H2O <==> CO + 3 H2 (high temp.)');
disp('Reaction 2:   CO + H2O <==> CO2 + H2 (low temp.)');
% --- To be filled by user -------
% 1) water vapor synthesis
dimc=2; % case No.
dims=[4; 4]; % total substances
dimr=[2 ;2]; % reactant No
sr=cell(dimc,4); % substance name
cr=zeros(dimc,4); % stoichiometry
sr(1,:)={ 'CH4', 'H2O','CO', 'H2'};
cr(1,:)=[ 1,1, 1,3];
% 2) Heat content of gaseous reaction (fumes)
sr(2,:)={'CO', 'H2O', 'CO2', 'H2'};
cr(2,:)=[ 1,1, 1,1];
% Temp. range
rTemp=[0; 1500; 50];  % Celsius degree, same interval
% --- End of user data
```

3. *Reading NASA table:* the following statements are common to all the scripts.

```
%% Reading NASA table and building data base
[row,upperTemp,Tcoef,logHalt]=...
    NASAdata(sr,TempMax, TempMin,dims, dimc);
if logHalt==1, return; end
```

The Boolean `logHalt` is raised to 1 by the function `NASAdata` (see Section 5.6.2), when the user temperature range in `rTemp` does not agree with the table range.

4. *Solving equilibria at each temperature for each case/reaction.*

We first refer to `NH3_equilV3.m` (Ammonia synthesis, Section 5.8). The script performs a loop on the absolute temperature range defined by `dimTemp`. The centigrade temperature is saved by `Var(i,1)=T-Temp0;`. Given the interpolation points `TxH` and `TxS`, enthalpy, entropy, and free energy variations `DeltaH`, `DeltaS`, and `DeltaG` are computed and saved into `Var(i,2:4)`. The equilibrium constant `Kp` is computed, and the equilibrium algebraic equation is solved by the symbolic function `vpasolve` in terms of the reaction coordinate `S`. Pressure and equilibrium concentration are saved into `Var(i,5:6)`.

A similar procedure is implemented by `SO2_SO3V3.m` (sulfur trioxide, SO_3, concerned with the production of sulfuric acid from sulfur dioxide, SO_2, Section 5.11), but only the equilibrium concentration is saved in `Var(i,2)`.

In the script `CH4_combustionV3.m` (methane combustion, Section 5.9), six procedures are implemented. Two of them, water and carbon dioxide dissociation (reactions 2 and 3) follow the previous procedure and save their concentrations into `Var(i,4:5)`. In addition, the heat emitted by fumes and by reaction 1 is computed and saved into `Var(i,2:3)`. A further addition is the flame temperature computation to be explained in Section 5.9.

The same procedure is repeated twice in `CH4_H2O_CO_3H2_equilV3.m` (hydrogen production at high and low temperature, Section 5.10). The equilibrium concentrations of the low- and high-temperature reactions are collected into `Var(i,3)` and `Var(i,5)`. The relevant free energies are collected into `Var(i,2)` and `Var(i,4)`.

The same procedure is performed in `H2O_equilV3.m` (hydrogen combustion, Section 5.7). Thermodynamic variables are saved into `Var(i,2:4)`, and the heat emitted by reaction and fumes is saved into `Var(i,5:6)`.

In the script `C_CaO_SO2V4.m` ($CaSO_4$ production from lime and sulfur impurity, Section 5.12), only the reaction free energy is computed and saved into `Var(i,2)`.

In `CaCO3_dec_kineticV4.m` (calcium carbonate decomposition and kinetics, Section 5.13), concentration versus temperature and the relevant rate are computed and saved into `Var(i,2:3)`.

5. *Result display and plot*. They depend on the specific topic to be explained from Section 5.7 to Section 5.13. As a baseline, a table of the numerical results, previously saved in the matrix `Var`, is displayed in the `command window` and appropriately plotted.

5.6.2 The function `NASAdata`

Given the symbol of one or more substances, the function reads the thermochemical coefficients from the excel file `NineCoeff.xlsx` and builds appropriate matrices to be employed by the script

algorithm. The function call is the following:

```
%% Reading NASA table and building data base
[row,upperTemp,Tcoef,logHalt]=...
    NASAdata(sr,TempMax, TempMin,dims, dimc);
if logHalt==1, return; end
```

The function input data to be specified by users in the main script are the following:

1. `sr`, matrix of the symbols, one row for each reaction or case to be treated. Let us remark that some cases do not refer to a chemical reaction.
2. `TempMax` and `TempMin` define the range of temperature, the same for all the cases.
3. `dims`, vector containing the number of substances for each case/reaction; `dimc`, number of cases/reactions.

The function output data are as follows:

1. `row` of size (`dimc, max(dims), 2`). It consists of a pair of matrices. The first matrix contains the first-row number in the table of Fig. 5.2 for each substance collected in the matrix `sr` for the current `dimc` cases. The second matrix contains the number of the table rows to be employed in order to cover the temperature range `TempMin, TempMax`.
2. `upperTemp` of size (`dimc, max(dims), 3`). It consists of three matrices, where *three* is the max number of temperature ranges for each substance in the NASA table of Fig. 5.2. Each matrix contains the upper temperature of each range to be employed, according to the number of rows reported in the second matrix of `row`. For instance, if the number of necessary rows is *two*, the third matrix will be zero.
3. `Tcoef` is the matrix of the thermochemical coefficients, including the last nine columns of the table in Fig. 5.2.
4. `logHalt` is a logical variable (0/1) indicating (1) or not (0) the function failure. The script execution halts if the given temperature range `TempMin, TempMax` is not covered by the table ranges. In that case, the function writes on the Matlab `command window` a message as in the following statements.

```
if max(upperTemp(c,j,:))<TempMax
    logHalt=1;
    format='\nHALT:  Max temp too high, substance %s\n';
    fprintf(format,cell2mat(s));
end
if min(lowerTemp(c,j,:))>TempMin
    logHalt=1;
    format='\nHALT:  Max temp too low, substance %s\n';
    fprintf(format',cell2mat(s));
end
```

The function details follow.

```
%% Reading NASA data and building reaction data base
function [row,upperTemp,Tcoef,logHalt]=...
   NASAdata(sr,TempMax, TempMin,dims, dimc)
   % Reading the table: it contains thermochemical coefficients,
   % each row a substance
   GTab=readtable('NineCoeff.xlsx','ReadVariableNames',0);
   [nrow,ncol]=size(GTab);
   Tdim=GTab{:,2};Tmin=GTab{:,3};Tmax=GTab{:,4};
   % row No, min temp, max temp
   Tcoef=GTab{:,5:ncol};Tchem=GTab{:,1};
   % coefficients subtable
   logHalt=0;
   dimMax=max(dims);
   nrow=length(Tchem);
   upperTemp=zeros(dimc,dimMax,3); % upper temp of each range <=3
   lowerTemp=NaN(dimc,dimMax,3);
   row=zeros(dimc,dimMax, 2); % first row, # rows=temp ranges
   for c=1:dimc
       for j=1:dims(c)
           row(c,j,1)=0; row(c,j,2)=1; % single range
           s=sr(c,j);
           i=0; Tend=0;
           while i<nrow && Tend<TempMax
               i=i+1;
               if strcmp(s,Tchem{i})==1
                   r=i;
                   if Tmax(r)>=TempMax, Tend=Tmax(r); end
               end
           end
           row(c,j,2)=Tdim(r); % number of rows
           row(c,j,1)=r-Tdim(r)+1; % initial row
           for k=1:row(c,j,2)
               upperTemp(c,j,k)=Tmax(row(c,j,1)+k-1);
               lowerTemp(c,j,k)=Tmin(row(c,j,1)+k-1);
           end
           if max(upperTemp(c,j,:))<TempMax
               logHalt=1;
               format='\nHALT:  Max temp too high, substance %s\n';
               fprintf(format,cell2mat(s));
           end
           if min(lowerTemp(c,j,:))>TempMin
               logHalt=1;
               format='\nHALT:  Max temp too low, substance %s\n';
               fprintf(format',cell2mat(s));
           end
       end
   end
   disp('NASA data have been read');
end
```

5.6.3 The function `ThermoCoef`

The function computes, at the temperature T, the thermochemical coefficient $c_k(c,T)$, $k = 1,...,9$ of the reaction c (do not confuse it with the coefficient symbol), as a linear combination with signed stoichiometric coefficients. Positive sign applies to products, whereas negative sign applies to reactants. For instance, given the following reaction with symbol c:

$$a_1R_1 + a_2R_2 \Rightarrow b_1P_1 + b_2P_2, \tag{5.23}$$

the relevant generic coefficient holds

$$c_k(T,c) = -a_1c_k(T,R_1) - a_2c_k(T,R_2) + b_1c_k(T,P_1) + b_2c_k(T,P_2). \tag{5.24}$$

The nine coefficients $c_k(T, c)$ of each reaction/case c at temperature T are saved in the matrix `Rx` with dimensions `dimc,9`.

The function details are as follows.

```
function Rx = ThermoCoef(T,Tcoef,upperTemp,row,cr,dimc,dims, dimr)
   % Computes the substance thermochemical coefficients
   % as a linear combination with signed stoichiometric parameters
   % Positive sign=product, negative sign= reactant
   [nrow,dimCoef]=size(Tcoef);
   Rx=zeros(dimc,dimCoef);
   for c=1:dimc  % loop on the cases
        for j=1:dims(c) % loop on subtances
        % reactant (-) and product (+) sign
        ss=-1; if j>dimr(c), ss=1; end
        krow=1;
        while T> upperTemp(c,j,krow), krow=krow+1; end
             irow=row(c,j,1)+krow-1; % table row
             Rx(c,:)=Rx(c,:)+ss*Tcoef(irow,:)*cr(c,j);
        end
   end
end
```

5.7 Hydrogen combustion

5.7.1 Description

Hydrogen, often indicated as H_2, is a fuel whose combustion is free of CO_2 emissions (CO_2-neutral). When used by fuel-cell technologies, it can afford a high-efficient production of electrical energy.

Combustion with air or pure oxygen produces water vapor, which can be also cooled and condensed, to yield extra energy and pure liquid water. Topics like heat evolved by the reaction, flame temperature, and water molecule dissociation at high temperatures (greater than 1500K) need a thermochemical treatment, like in the following.

The flame temperature T_{flame} corresponds to the intersection of the heat emitted by the reaction, $-x\Delta H_{reaction}$ (`Var(i,5)`, scaled to kJ unit), and the heat absorbed by fumes, ΔH_{fumes} (`Var(i,6)`, scaled to kJ unit). In the hottest regions of the flame, the evolved heat is not yet transferred to the surroundings, being captured by the gaseous reaction products (fumes) which increase their temperature. In order to compute this value, we must account also for water dissociation, an endothermic reaction which decreases the flame temperature. It is modeled by the thermochemistry of the equilibrium $H_2O_{(gas)} \Leftrightarrow 2H_2 + O_2$ as it is done in the Matlab script of Section 5.7.2.

5.7.2 The Matlab script

The organization of the script is the same as illustrated in Section 5.6.1, except for the computation of the flame temperature T_{flame}, which derives, as already said, from the intersection of the heat emitted by the reaction $-x\Delta H_{reaction}$ and the heat absorbed by fumes ΔH_{fumes} (see Fig. 5.3). The relevant statements are as follows.

```
Var(i,5) = -S*DeltaH*KiloScale;
Var(i,6) =(R*sum(TxH.*Rx(2,:)) - H0)*KiloScale;
```

Two alternative methods have been adopted for computing the intersection.

1. *Linear interpolation*. The Matlab function `min` finds the minimum of the absolute heat difference, closed to zero. A linear interpolation is applied between the two different heat values across the zero value, allowing the zero-crossing temperature T_{flame} (`TempF`) to be found. The result is accurate as far as a small temperature step is used by simulation.
2. *Polynomial interpolation*. Both heat profiles versus temperature are fit with a second-order polynomial by means of the Matlab function `polyfit`. The polynomial difference is solved via the function `vpasolve` in order to find the zero-crossing temperature T_{flame} (`TempF`). Accuracy depends on the polynomial degree.

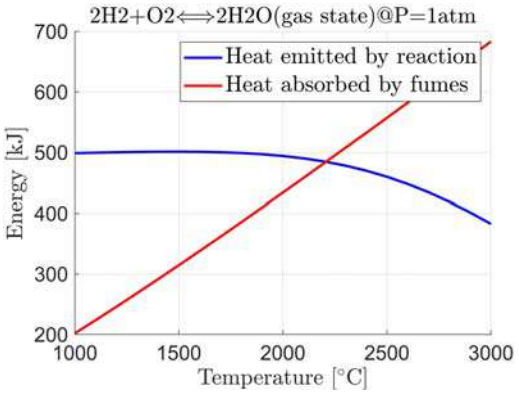


FIGURE 5.3

Heat emitted by reaction and absorbed by fumes.

Fig. 5.3 shows the profiles versus temperature of the heat emitted by reaction and absorbed by fumes. Their intersection defines the *flame temperature*.

The relevant script is as follows.

```
%% Hydrogen combustion (Chapter 5)
%       REACTION  :   2 H2 + O2  <==>  2 H2O  (gas state)
%% Initialization
InitChem;
% common parameters
Temp0=273.15;R = 8.314472;
P = 1; % relative to standard pressure 1 atm (dimless)
syms x; logDisp=0; % regulates printout
%%  Reaction parameters and temperature range
disp('Reaction:  2 H2 + O2  <==>  2 H2O  (gas state)');
disp('NASA CEA thermochemical coefficients');
% --- To be filled by user -------
% 1) Water gas synthesis
dimc=2; % case No.
dims=[3; 3]; % total substances
dimr=[2 ;0]; % reactant No
sr=cell(dimc,3); % substance name
cr=zeros(dimc,3); % stoichiometry
sr(1,:)={ 'H2', 'O2', 'H2O'};
cr(1,:)=[ 2,1, 2];
% 2) Heat content of gaseous reaction (fumes)
sr(2,:)={ 'H2O', 'N2', 'Ar'};
cr(2,:)=[ 2,3.714, 0.0476];
% Temperature range
rTemp=[1000; 3000; 100];  % Celsius degree, single interval
% --- End of user data
TempMin=rTemp(1)+Temp0;TempMax=rTemp(2)+Temp0;
dTemp=rTemp(3); % temp. step
Temp=TempMin:dTemp:TempMax; % temp. axis
dimTemp=length(Temp);
%% Reading NASA table and building data base
[row,upperTemp,Tcoef,logHalt]=...
    NASAdata(sr,TempMax, TempMin,dims, dimc);
if logHalt==1, return; end
%% Solving equilibrium at each temperature
T=TempMin-dTemp; % before initial
Var=zeros(dimTemp,6);
for i=1:dimTemp
  T=T+dTemp;  Var(i,1)=T-Temp0;
```

```
% interpolating points
TxH = [-1/T,log(T),T,T^2/2,T^3/3,T^4/4,T^5/5,1,0];
TxS = [-1/T^2/2,-1/T,log(T),T,T^2/2,T^3/3,T^4/4,0,1];
Rx = ThermoCoef(T,Tcoef,upperTemp,row,cr,dimc,dims,dimr);
% Calculating 1) DeltaH, Enthalpy variation
% 2) DeltaS, Entropy var. 3) DeltaG, free Energy variation
DeltaH = R*sum(TxH.*Rx(1,:)); DeltaS = R*sum(TxS.*Rx(1,:));
DeltaG = DeltaH - T*DeltaS;
Var(i,2)=DeltaH*KiloScale;
Var(i,3)=DeltaS*KiloScale;
Var(i,4)=DeltaG*KiloScale;
% Thermodynamics, Kp (equilibrium constant)
Kp = exp(-DeltaG*(R*T)^-1);
% We solve equilibrium for water synthesis reaction
%         H2          O2          H2O(gas)    Total
% Start   2 mol       1 mol       0 mol       3 mol
% Equil.  2-2x mol    1-x mol     2x mol      3-x mol
% Kp = (2x/(3-x))^2 / ((2-2x)/(3-x))^2 / (1-x)/(3-x) / P
% The 3th degree equation is solved by vpasolve()
y1=Kp*P*(1-x)^3-x^2*(3-x);
S = double(vpasolve(y1 == 0, x, [0 1]));
Var(i,5) = -S*DeltaH*KiloScale;
% Fumes
if i==1
Tf = 25+Temp0;
TxHf = [-1/Tf,log(Tf),Tf,Tf^2/2,Tf^3/3,Tf^4/4,Tf^5/5,1,0];
H0 = R*sum(TxHf.*Rx(2,:)); % heat content of fumes @ 25 °C
end
Var(i,6) =(R*sum(TxH.*Rx(2,:)) - H0)*KiloScale;
end
% Flame temperature: two methods
% 1) linear interpolation around zero difference
dVar=Var(:,5)-Var(:,6);  % difference between energies
[minF,indF]=min(abs(dVar)); % point close to zero difference
dE=abs(dVar(indF)-dVar(indF-1)); % Energy jump
TempF=Var(indF,1)-minF*dTemp/dE;
% 2) 2nd order polynomial fit
p1 = polyfit(
Var(:,1),Var(:,5),2);p2 = polyfit(Var(:,1),Var(:,6),2);
p = p1 - p2; y1 = p(1)*x^2 + p(2)*x + p(3);
S = double(vpasolve(y1 == 0, x, [TempMin TempMax]));
```

```
%% Result display and plot
disp('Results');
format=' Temp.[°C] DH[kJ/mol] DS[(kJ/mol)/K] DG [kJ/mol]';
format=strcat(format,' ReactHeat[kJ/mol] Fumes[kJ/mol]\n');
fprintf(format);
format=' %6.1f %12.3f %12.3f %12.3f %12.1f %12.3f\n';
fprintf(format,Var(1,:));
if logDisp==1
    for i=2:dimTemp-1,fprintf(format,Var(i,:));end
end
fprintf(format,Var(dimTemp,:));
% Flame
fprintf('Flame temperature (polynomial fit) = %6.1f [°C]\n',S);
format='Flame temperature (linear interpolation) = %6.1f [°C]\n';
fprintf(format,TempF);
% Plot
figure('name', 'Equilibrium conc'); hold on; grid on;
plot(Var(:,1),Var(:,5),'b') ;
plot(Var(:,1),Var(:,6),'r') ;
xlabel('Temperature [$^\circ$C]');
ylabel('Energy [kJ]');
title('2H2+O2$\Longleftrightarrow$2H2O(gas state)@P=1atm');
legend('Heat emitted by reaction','Heat absorbed by fumes',...
    'FontSize',18)
```

5.8 Ammonia synthesis (Haber process)

5.8.1 Description and graphical plot

Optimization of the yield of the ammonia synthesis is a topic of prominent interest in industry, due to ever increasing demand of this raw material. Ammonia is formed from nitrogen and hydrogen by the following reversible reaction:

$$N_2 + 3H_2 \Leftrightarrow 2NH_3. \tag{5.25}$$

Its production is favored by high pressures and low temperatures. In order to design an ammonia synthesis reactor and find the optimum operating conditions in terms of temperature, pressure, and catalyst, the basic thermochemistry has to be tackled.

As an example, the graphical plot in Fig. 5.4 shows the equilibrium concentration in molar fractions versus temperature. They are calculated by the Matlab script in Section 5.8.2.

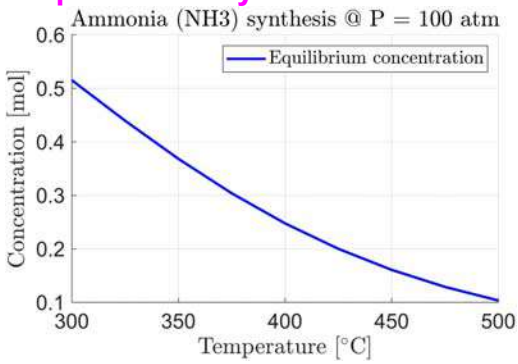


FIGURE 5.4

Ammonia concentration versus temperature.

5.8.2 The Matlab script

The script below follows the scheme of Section 5.6.1.

```
%% Ammonia synthesis, Haber process (Chapter 5)
% REACTION: 3 H2 + N2 <==> 2 NH3(gas state) (Haber synthesis)
%% Initialization
InitChem;
% common parameters
Temp0=273.15;R = 8.314472;
P = 100; % relative to standard pressure 1 atm (dimensionless)
syms x; logDisp=0; %regulates printout
%% Reaction parameters and temperature
disp(' Ammonia Synthesis (Haber process)');
disp(' REACTION:  3 H2 + N2  <==>  2 NH3  (gas state)');
% --- To be filled by user -------
dimc=1; % case No.
dims=3; % total substances
dimr=2; % reactant No
sr=cell(dimc,dims); % substance name
cr=zeros(dimc,dims); % stoichiometry
sr(1,:)={'H2', 'N2', 'NH3'};
cr(1,:)=[ 3,1, 2];
% Temp. range
rTemp=[300; 500; 25];  % Celsius degree
% --- End of user data
TempMin=rTemp(1)+Temp0;TempMax=rTemp(2)+Temp0;
dTemp=rTemp(3);
Temp=TempMin:dTemp:TempMax; % temp. axis
dimTemp=length(Temp);
%% Reading NASA table and building data base
[row,upperTemp,Tcoef,logHalt]=...
    NASAdata(sr,TempMax, TempMin,dims, dimc);
if logHalt==1, return; end
```

```
%% Solving equilibrium at each temperature
T=TempMin-dTemp; % before initial
Var=zeros(dimTemp,6);
for i=1:dimTemp
  T=T+dTemp;  Var(i,1)=T-Temp0;
  % interpolating points
  TxH = [-1/T,log(T),T,T^2/2,T^3/3,T^4/4,T^5/5,1,0];
  TxS = [-1/T^2/2,-1/T,log(T),T,T^2/2,T^3/3,T^4/4,0,1];
  Rx = ThermoCoef(T,Tcoef,upperTemp,row,cr,dimc,dims,dimr);
  % Calculating enthalpy, entropy, free energy variations
  DeltaH = R*sum(TxH.*Rx);  DeltaS = R*sum(TxS.*Rx);
  DeltaG = DeltaH - T*DeltaS;
  Var(i,2)=DeltaH*KiloScale;
  Var(i,3)=DeltaS*KiloScale;
  Var(i,4)=DeltaG*KiloScale;
  % Thermodynamics, Kp (equilibrium constant) derives from DeltaG
  Kp = exp(-DeltaG*(R*T)^-1);
  % Solving equilibrium from stoichiometric mixture of N2 and H2
  % Start   H2 = 3 mol  (volume does not matter)
  %         N2 = 1 mol  NH3 = 0 mol
  % Equilibrium H2 = 3-3x; N2 = 1-x; NH3 = 2x;Total moles=4-2x
  % Kp*P^2 = [16x^2 (2-x)^2] / [27(1-x)^4]
  % The 4th degree equation is solved by vpasolve()
  y1=Kp*P^2*27*(1-x)^4-16*(x*(2-x))^2;
  S = double(vpasolve(y1 == 0, x, [0 1]));
  Var(i,5)=P;Var(i,6) = S/(2 - S);
End
%% Result display and plot
format='Temp.[°C] DeltaH[kJ/mol] DeltaS[kJ/mol] DeltaG[kJ/mol]';
format=strcat(format,' P[atm]  Conc.[mol]\n');
fprintf(format);
format=' %6.1f %12.3f %12.3f %13.3f %12.1f %12.3f\n';
fprintf(format,Var(1,:));
if logDisp==1
    for i=2:dimTemp-1, fprintf(format,Var(i,:));end
end
fprintf(format,Var(dimTemp,:));
% plot
figure('name', 'Equilibrium conc'); hold on; grid on;
plot(Var(:,1),Var(:,6),'b') ;
xlabel('Temperature [$^\circ$C]');
ylabel('Concentration [mol]');
title('Ammonia (NH3) synthesis @ P = 100 atm');
legend('Equilibrium concentration','FontSize', 18);
```

5.9 Methane (CH_4) combustion

5.9.1 Description and graphical plot

Chemically, the combustion process of all fossil fuels follows a similar reaction between hydrocarbon and oxygen in the air:

$$C_nH_{2n+2} + \left(\frac{3}{2}n + \frac{1}{2}\right)O_2 \Rightarrow nCO_2 + (n+1)H_2O. \tag{5.26}$$

When the reaction takes place, the products are carbon dioxide, CO_2, water, H_2O, and a great deal of energy. The following reaction represents the combustion of methane, with $n = 1$, every substance being in the gaseous state:

$$CH_4 + 2O_2 \Rightarrow CO_2 + 2H_2O. \tag{5.27}$$

The reaction usually gives off either steam or water vapor together with a large amount of energy, which is calculated as the enthalpy variation ΔH of the reaction by the Matlab script in Section 5.9.2.

Natural gas is the cleanest burning fossil fuel. Coal and oil, the other fossil fuels, are more chemically complicated than natural gas, and when combusted, release a variety of potentially harmful air pollutants. Burning methane releases only carbon dioxide and water. Since the natural gas is mostly methane, the combustion of natural gas releases fewer byproducts than other fossil fuels.

At high temperatures, say $> 1800°C$, the emitted water and carbon dioxide dissociate via endothermic reactions. The Matlab script takes into account this dissociation, in order to properly balance the heat emitted by the reaction itself. In many instances the stoichiometric intake air is preheated, and this fact is also accounted for in the Matlab script.

As for hydrogen combustion, Fig. 5.5 shows the profiles of the heat emitted by reaction and absorbed by fumes versus temperature. The intersection temperature is the flame temperature to be computed by the script.

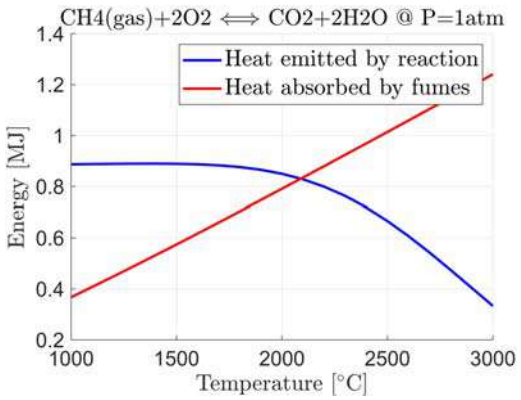


FIGURE 5.5

Methane combustion: heat emitted by reaction and absorbed by fumes.

5.9.2 Matlab script

The user section of the Matlab script is rather complex since the script treats five cases: the methane combustion reaction, two dissociation reactions (water and carbon dioxide) at high temperature as well as the heat computation for fumes and the intake air. Indeed, methane combustion may be carried out by preheating the intake air, usually by the same exhaust gases. In the script, the preheated air temperature is fixed to 600°C in the very beginning, but the value can be varied. The intersection temperature of the profiles in Fig. 5.1 is the flame temperature, which is computed by the pair of alternative algorithms of Section 5.7.2 (hydrogen combustion). Except for the number of different cases, which reflects in the equilibrium and flame temperature computations, the script organization is standard.

```
%% Methane combustion
%  REACTION:CH4(gas)+2O2-->CO2+2H2O(gas state)
%           - Methane combustion
%  Thermal dissociation at high temperatures is considered:
%  2H2O <==> 2H2 + O2    ;    2CO2 <==> 2CO + O2
%% Initialization
InitChem;
% common parameters
Temp0=273.15;R = 8.314472;
P = 1; % relative to standard pressure 1 atm (dimensionless)
Ta=25+Temp0; Ti=600; % ambient and intake
syms x; logDisp=0; % regulates printout
%%  Reaction parameters and temperature
disp('Reaction 1:CH4(g)+2O2-->CO2+2H2O(g),Methane combustion');
disp('Reaction 2:2H2O <==> 2H2 + O2, high temp. dissociation');
disp('Reaction 3:2CO2 <==> 2CO + O2, high temp. dissociation');
% --- To be filled by user -------
% 1) methane combustion
dimc=5; % case No.
dims=[4; 3;3; 4; 3]; % total substances
dimr=[2 ;1; 1; 0; 0]; % reactant No
sr=cell(dimc,4); % substance name
cr=zeros(dimc,4); % stoichiometry
sr(1,:)={ 'CH4', 'O2',  'CO2',  'H2O'} ;
cr(1,:)=[ 1,2, 1,2];
% 2) water dissociation
sr(2,:)={ 'H2O', 'H2','O2',''};
cr(2,:)=[ 2,2, 1,0];
% 3) CO2 dissociation
sr(3,:)={'CO2', 'CO', 'O2',''};
cr(3,:)=[ 2,2, 1,0];
% 4) Heat of fumes
sr(4,:)={'N2', 'Ar', 'H2O','CO2'};
cr(4,:)=[ 7.429,0.0952, 2,1];
```

```
% 5) Heat of intake air
sr(5,:)={'N2', 'Ar', 'O2',''};
cr(5,:)=[ 7.429,0.0952, 2,0];
% Temp. range
rTemp=[1000; 3000; 100];  % Celsius degree, same interval
% --- End of user data
TempMin=rTemp(1)+Temp0;TempMax=rTemp(2)+Temp0;
dTemp=rTemp(3); % temp. step
Temp=TempMin:dTemp:TempMax; % temp. axis
dimTemp=length(Temp);
%% Reading NASA table and building data base
[row,upperTemp,Tcoef,logHalt]=...
    NASAdata(sr,TempMax, TempMin,dims, dimc);
if logHalt==1, return; end
%% Solving equilibrium at each temperature
T=TempMin-dTemp; % before initial
Var=zeros(dimTemp,5);
heat=zeros(5,1);
for i=1:dimTemp
  T=T+dTemp;  Var(i,1)=T-Temp0;
  % interpolating points
  TxH = [-1/T,log(T),T,T^2/2,T^3/3,T^4/4,T^5/5,1,0];
  TxS = [-1/T^2/2,-1/T,log(T),T,T^2/2,T^3/3,T^4/4,0,1];
  Rx = ThermoCoef(T,Tcoef,upperTemp,row,cr,dimc,dims,dimr);
  %  Enthalpy, Entropy
  % 2) Reaction 2 - H2O dissociation
  DeltaH = R*sum(TxH.*Rx(2,:));  DeltaS = R*sum(TxS.*Rx(2,:));
  DeltaG = DeltaH - T*DeltaS;
  % From thermodynamics, Kp (equilibrium constant)
  Kp = exp(-DeltaG/(R*T));
  y1 = (Kp - P)*x^3 - 3*Kp*x + 2*Kp;
  S = double(vpasolve(y1 == 0, x, [0 1]));
  Var(i,4)=S;
  % heat absorbed by dissociation of 2 mol H2O
  heat(2) = S*2*DeltaH;
  % 3) Reaction 3 - CO2 dissociation
  DeltaH = R*sum(TxH.*Rx(3,:));  DeltaS = R*sum(TxS.*Rx(3,:));
  DeltaG = DeltaH - T*DeltaS;
  % Thermodynamics, Kp(equil. constant) derives from DeltaG
  Kp = exp(-DeltaG/(R*T));
  y1 = (Kp - P)*x^3 - 3*Kp*x + 2*Kp;
  S = double(vpasolve(y1 == 0, x, [0 1]));
  Var(i,5)=S;
  % heat absorbed by dissociation of 2 mol H2O
  heat(3) = S*DeltaH;
```

```
% 4) fumes
  if i==1
    TxHa = [-1/Ta,log(Ta),Ta,Ta^2/2,Ta^3/3,Ta^4/4,Ta^5/5,1,0];
    DeltaH4a = R*sum(TxHa.*Rx(4,:)); % heat of fumes @ 25degC
    DeltaH5a = R*sum(TxHa.*Rx(5,:)); % heat of air @ 25degC
  end
  DeltaH = R*sum(TxH.*Rx(4,:)) ;
  heat(4)=DeltaH-DeltaH4a; % heat of fumes
  Var(i,2)=heat(4)*MegaScale;
  % 5) intake air
  if i==1
    TxHi = [-1/Ti,log(Ti),Ti,Ti^2/2,Ti^3/3,Ti^4/4,Ti^5/5,1,0];
    DeltaH5i = R*sum(TxHi.*Rx(5,:)); % heat of air @ 600K
    heat(5) = DeltaH5i - DeltaH5a;
  end
  % 6) Reaction 1
  DeltaH = R*sum(TxH.*Rx(1,:));
  heat(1)= DeltaH+heat(2)+heat(3)-heat(5); % heat balance
  Var(i,3)=-heat(1)*MegaScale;
end
%% Flame temperature: two methods
% 1) linear interpolation around zero difference
dVar=Var(:,3)-Var(:,2);  % difference between energies
[minF,indF]=min(abs(dVar)); % point close to zero difference
dE=abs(dVar(indF)-dVar(indF-1)); % Energy jump
TempF=Var(indF,1)-minF*dTemp/dE;
% 2) 2nd order polynomial fit
p1 = polyfit(Var(:,1),Var(:,3),2);
p2 = polyfit(Var(:,1),Var(:,2),2);
p = p1 - p2;y1 = p(1)*x^2 + p(2)*x + p(3);
S = double(vpasolve(y1 == 0, x, [TempMin TempMax]));
%% Result display and plot
disp('Results');
fprintf('Reaction 1 CH4+H2O<>CO+3H2, Reaction2 CO+H2O<>CO2+H2\n');
format='Temp.[°C] Fumes[kJ/mol] Reaction[kJ/mol] H2O[mol/mol] ';
format=strcat(format,'CO2[mol/mol]\n');
fprintf(format);
format=' %6.1f %12.3f %12.3f %14.3f %12.1f \n';
fprintf(format,Var(1,:)); % first
if logDisp==1
    for i=2:dimTemp-1, fprintf(format,Var(i,:)); end
end
fprintf(format,Var(dimTemp,:)); % last
```

```
% Flame temp.
fprintf('Flame temp. (polynomial fit) = %6.1f [°C]\n',S);
fprintf('Flame temp. (linear interp.) = %6.1f [°C]\n',TempF);
% plot
figure('name', 'Equilibrium'); hold on; grid on;
plot(Var(:,1),Var(:,3),'b') ;
plot(Var(:,1),Var(:,2),'r') ;
xlabel('Temperature [$^\circ$C]');
ylabel('Energy [MJ]');
title('CH4(gas)+2O2 $\Longleftrightarrow$ CO2+2H2O @ P=1atm');
legend('Heat emitted by reaction','Heat absorbed by fumes',...
    'FontSize',18)
```

5.10 Hydrogen production at high and low temperature

5.10.1 Description and graphical plot

The so-called steam-methane reforming is a widely used method for hydrogen production, being currently the least expensive if compared with electrolysis of water or solar energy technologies. It accounts for nearly all the commercially produced hydrogen in the United States and about 95% worldwide. Commercial hydrogen producers and petroleum refineries use steam-methane reforming to separate hydrogen atoms from carbon atoms in methane (CH_4). In steam-methane reforming, high-temperature steam (700°C−900°C) under 0.3−2.5 MPa pressure reacts with methane in the presence of a nickel catalyst to produce hydrogen, carbon monoxide, and a relatively small amount of carbon dioxide. The stoichiometry of this reversible reaction is $CH_4 + H_2O \Leftrightarrow CO + 3H_2$.

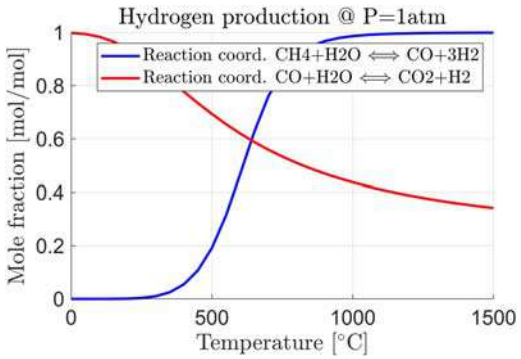


FIGURE 5.6

Reaction coordinates of hydrogen production. Blue line: high-temperature production. Red line: low-temperature production.

The reaction is strongly endothermic, implying $\Delta H < 0$, and produces the so-called *syngas*, an admixture of gaseous hydrogen and carbon monoxide. The latter is then cooled down to 360°C, when, in a second stage, additional hydrogen is generated through the lower-temperature, exothermic ($\Delta H > 0$) water gas shift reaction $CO + H_2O \Leftrightarrow CO_2 + H_2$. The latter is a reversible reaction. The calculation of the equilibrium concentration as a function of temperature and pressure for both reactions is the basis for plant design and conduction.

Fig. 5.6 shows the profiles versus temperature of the reaction coordinate x of the previous reactions as a result of the thermochemical algorithm implemented by the Matlab script in Section 5.10.2.

5.10.2 Matlab script

The next Matlab script follows the standard scheme described in Section 5.6.1.

```
%% Hydrogen production at high and low temperature
% REACTIONS: CH4 + H2O<==>CO+3H2 (industry,700 to 1100°C)
% CO + H2O <==> CO2 + H2 (at lower temperatures)
%% Initialization
InitChem;
% common parameters
Temp0=273.15;R = 8.314472;
P = 1; % pressure relative to standard pressure 1 atm (dimensionless)
syms x; logDisp=0; % regulates printout
%% Reaction parameters and temperature
disp('Reaction 1:  CH4 + H2O <==> CO + 3 H2 (high temp.)');
disp('Reaction 2:   CO + H2O <==> CO2 + H2 (low temp.)');
% --- To be filled by user -------
% 1) water vapour synthesis
dimc=2; % case No.
dims=[4; 4]; % total substances
dimr=[2 ;2]; % reactant No
sr=cell(dimc,4); % substance name
cr=zeros(dimc,4); % stoichiometry
sr(1,:)={ 'CH4', 'H2O','CO', 'H2'};
cr(1,:)=[ 1,1, 1,3];
```

```
% 2) Heat content of gaseous reaction (fumes)
sr(2,:)={'CO', 'H2O', 'CO2', 'H2'};
cr(2,:)=[ 1,1, 1,1];
% Temp. range
rTemp=[0; 1500; 50];  % Celsius degree, same interval
% --- End of user data
TempMin=rTemp(1)+Temp0;TempMax=rTemp(2)+Temp0;
dTemp=rTemp(3); % temp. step
Temp=TempMin:dTemp:TempMax; % temp. axis
dimTemp=length(Temp);
%% Reading NASA table and building data base
[row,upperTemp,Tcoef,logHalt]=...
    NASAdata(sr,TempMax, TempMin,dims, dimc);
if logHalt==1, return; end
%% Solving equilibrium at each temperature
T=TempMin-dTemp; % before initial
Var=zeros(dimTemp,5);
for i=1:dimTemp
  T=T+dTemp;  Var(i,1)=T-Temp0;
  % interpolating points
  TxH = [-1/T,log(T),T,T^2/2,T^3/3,T^4/4,T^5/5,1,0];
  TxS = [-1/T^2/2,-1/T,log(T),T,T^2/2,T^3/3,T^4/4,0,1];
  Rx = ThermoCoef(T,Tcoef,upperTemp,row,cr,dimc,dims,dimr);
  % DeltaH, Enthalpy, DeltaS, Entropy, DeltaG, free Energy
  % Reaction 1
  DeltaH = R*sum(TxH.*Rx(1,:));  DeltaS = R*sum(TxS.*Rx(1,:));
  DeltaG = DeltaH - T*DeltaS;
  Var(i,2)=DeltaG*KiloScale;
  Kp = exp(-DeltaG*(R*T)^-1); % dimensionless
  % The 4th degree equation is solved by vpasolve()
  y1 = (4*Kp - 27*P^2)*x^4 - 8*Kp*x^2 + 4*Kp;
  S = double(vpasolve(y1 == 0, x, [0 1]));
  Var(i,3) = S;
  % Reaction 2
  DeltaH = R*sum(TxH.*Rx(2,:));  DeltaS = R*sum(TxS.*Rx(2,:));
  DeltaG = DeltaH - T*DeltaS;
  Var(i,4)=DeltaG*KiloScale;
  Kp = exp(-DeltaG*(R*T)^-1);
  % The 2nd degree equation is solved by vpasolve()
  y1 = (Kp - 1)*x^2 - 2*Kp*x + Kp;
  S = double(vpasolve(y1 == 0, x, [0 1]));
  Var(i,5) = S;
end
```

```
%% Result display and plot
disp('Results');
fprintf('Reaction 1 CH4+H2O<>CO+3H2- Reaction2 CO+H2O<>CO2+H2\n');
fvar='Temp.[°C] DeltaG[kJ/mol] MoleFraction[mol/mol] DeltaG [kJ/mol]';
fvar=strcat(fvar,' MoleFraction[mol/mol]\n');fprintf(fvar);
format=' %6.1f %12.3f %12.3f %20.3f %12.1f \n';
fprintf(format,Var(1,:));
if logDisp==1
    for i=2:dimTemp-1, fprintf(format,Var(i,:));end
end
fprintf(format,Var(dimTemp,:));
% Figure
figure('name', 'Equilibrium conc'); hold on; grid on;
plot(Var(:,1),Var(:,3),'b') ;
plot(Var(:,1),Var(:,5),'r') ;
xlabel('Temperature [$^\circ$C]');
ylabel('Mole fraction [mol/mol]');
title('Hydrogen production @ P=1 atm');
legend('Reaction coord. CH4+H2O $\Longleftrightarrow$ CO+3H2',...
    'Reaction coord. CO+H2O $\Longleftrightarrow$ CO2+H2', 'FontSize',14)
```

5.11 Sulfur trioxide (SO_3) production from sulfur dioxide (SO_2)

5.11.1 Description and graphical profiles

In this section, we compute energy and equilibrium of the reaction $2SO_2 + O_2 \Rightarrow 2SO_3$ between sulfur oxides in gaseous state at various temperatures.

Fossil fuels such as coal and oil can contain a significant amount of sulfur. When fossil fuels are burned, sulfur is generally converted into sulfur dioxide, SO_2, a colorless gas. Such a conversion occurs under normal conditions of temperature and of the oxygen present in the fuel gas. SO_2 can further oxidize into sulfur trioxide, SO_3, in the presence of excess oxygen and when gas temperatures are sufficiently high, according to the reaction $2SO_2 + O_2 \Rightarrow 2SO_3$. At about 800°C, the formation of SO_3 is favored. SO_3 can also be formed from the catalysis of metals in the fuel. Such a reaction is typical of heavy fuel oil, due to the presence of a significant amount of vanadium. In the presence of water vapor, SO_3 is readily transformed into sulfuric acid mist with the formation of acid aerosols.

In order to remove SO_2 and SO_3 from combustion gases (flue gases), we should account for their acid nature. Therefore sorbent slurries (or other appropriate materials) must be alkaline. The process is called fluegas desulfurization (FDG). In industrial power plants, limestone, $CaCO_3$, is used to react with sulfur oxides yielding calcium sulfate (gypsum), $CaSO_4$, as detailed in Section 5.12.

Fig. 5.7 shows the mole fraction of the sulfur trioxide, SO_3, as a function of the temperature.

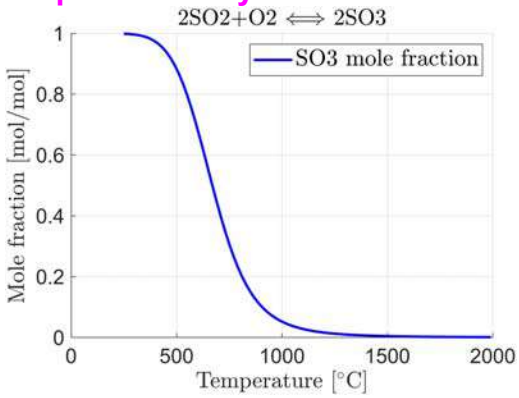


FIGURE 5.7

SO_3 mole fraction.

5.11.2 Matlab script

The next Matlab script follows the standard scheme of Section 5.6.1.

```
%% Sulphur trioxide (SO3) production
%% from sulphur dioxide (SO2) (Chapter 5)
%% Initialization
InitChem;
% common parameters
Temp0=273.15;R = 8.314472;
P = 1; % relative to standard pressure 1 atm (dimensionless)
syms x;logDisp=0; %regulates printout
%%  Reaction parameters and temperature
disp('Reaction: 2 SO2 + O2 <==> 2 SO3');
% --- To be filled by user -------
% reaction data from user
dimc=1; % case No.
dims=3; % substance No
dimr=2; % reactant No
sr=cell(dimc,dims); % substance name
sr(1,:)={ 'SO2','O2' ,'SO3'};
cr=zeros(dimc,dims); % stoichiometry
cr(1,:)=[ 2,1,2];
% Temp. range
rTemp=[250; 2000; 20];  % Celsius degree, same range
% --- End of user data
TempMin=rTemp(1)+Temp0;TempMax=rTemp(2)+Temp0;
dTemp=rTemp(3); % temp. step
dimTemp=floor((TempMax-TempMin)/dTemp)+1;
%% Reading NASA table and building data base
[row,upperTemp,Tcoef,logHalt]=...
    NASAdata(sr,TempMax, TempMin,dims, dimc);
if logHalt==1, return; end
```

```
%% Solving equilibrium at each temperature
T=TempMin-dTemp; % before initial
Var=zeros(dimTemp,2);
for i=1:dimTemp
  T=T+dTemp;  Var(i,1)=T-Temp0;
  % interpolating points and coeff table
  TxH = [-1/T,log(T),T,T^2/2,T^3/3,T^4/4,T^5/5,1,0];
  TxS = [-1/T^2/2,-1/T,log(T),T,T^2/2,T^3/3,T^4/4,0,1];
  Rx = ThermoCoef(T,Tcoef,upperTemp,row,cr,dimc,dims,dimr);
  % DeltaH, Enthalpy , DeltaS, Entropy,  DeltaG, free Energy
  DeltaH = R*sum(TxH.*Rx(1,:));  DeltaS = R*sum(TxS.*Rx(1,:));
  DeltaG = DeltaH - T*DeltaS;
  Kp = exp(-DeltaG/(R*T));
  % Solving Kp = p(SO3)^2 / (p(SO2)^2 * p(O2))
  y1 = (1 - Kp*P)*x^3 + (3*Kp*P - 3)*x^2 - 3*Kp*P*x + Kp*P;
  S = double(vpasolve(y1 == 0, x, [0 1]));
  Var(i,2) = 2*S/(3-S);
end
%% Result display and plot
disp('Results');
fprintf('Temp.[°C] MoleFraction(SO3)[mol/mol] \n');
format=' %6.1f %18.3f \n';
fprintf(format,Var(1,:)) % first
if logDisp==1
    for i=2:dimTemp-1,  fprintf(format,Var(i,:)); end
end
fprintf(format,Var(dimTemp,:)) %last
% plot
figure('name', 'Equilibrium'); hold on; grid on;
plot(Var(:,1),Var(:,2),'b') ;
xlabel('Temperature [$^\circ$C]');
ylabel('Mole fraction [mol/mol]');
title(' 2SO2+O2 $\Longleftrightarrow$ 2SO3');
legend('SO3 mole fraction', 'FontSize',18);
```

5.12 $CaSO_4$ production from lime and sulfur impurity in clean coal combustion

5.12.1 Description and graphical results

Different technologies are employed in the effort of mitigating health and environmental impact of coal combustion, in particular air pollution from coal-fired power stations or metallurgical processes. The sulfur content of coal, which may reach 5% by weight in some cases, causes emissions of sulfur dioxide, SO_2, during combustion. This is the most important gas at the origin of acid rains. SO_2 can be removed by fluegas desulfurization (see Section 5.11.1), but an emergent technology exploits the fluidized bed combustion.

In the most basic form, fuel particles are suspended in a hot bubbling fluid layer of ash (the *bed*) and other particulate materials (sand, limestone, ...), through which jets of air are blown for providing the necessary oxygen of combustion and gasification. The resultant fast and intimate mixing of gas and solids promotes rapid heat transfer and chemical reactions within the bed. Limestone is added in the form of a fine powder so as to promote the in situ formation, during combustion, of calcium sulfate, $CaSO_4$, thus avoiding the need of desulfurizing flue gases.

Besides the reduction of the sulfur amount emitted in the form of a SO_2/SO_3 mixture, the technique is favoring a more efficient heat transfer from the combustion area to the apparatus (usually water tubes) where heat energy is captured. Moreover, since the technology allows coal plants to burn at cooler temperatures, less NOx is also emitted.

The Matlab script simulates, according to thermodynamic data of the NASA CEA database, the above reaction environment, where CaO, from $CaCO_3$ decomposition, SO_2, and O_2 are present. The relevant equilibrium reaction is as follows:

$$2CaO(s) + 2SO_2(g) + O_2(g) \Leftrightarrow 2CaSO_4(s). \tag{5.28}$$

Simulation starts at 250°C and ends at 2900°C. In between, it accounts for the phase change of the solid $CaSO_4$ at 1473K and for its melting point at 1733K.

Fig. 5.8 shows the reaction free energy at lower and higher temperatures. At low temperatures, the reaction proceeds, whereas at higher temperatures, the reaction reverses. The relevant equilibrium point is denoted by a red circle. Melting point (the blue square) and the phase change point (the green diamond) are also indicated.

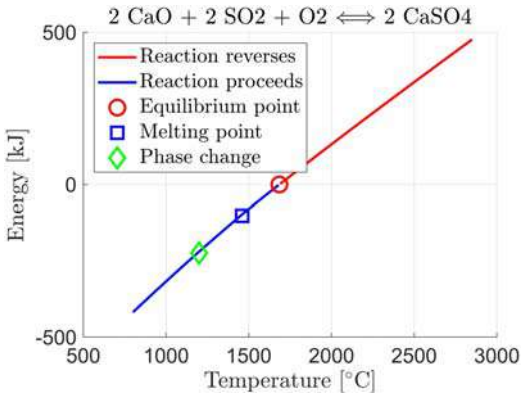


FIGURE 5.8

Free energy of the reaction showing direction of the reversible reaction.

5.12.2 Matlab script

The script follows the standard scheme of Section 5.6.1 except for the separation of the profile of the free energy $\Delta G^0(T)$ into negative and positive values through the Matlab function `find`, which yields the vector of the indices at which a variable satisfies a given condition (the condition is TRUE), like for instance to be positive or negative. The relevant statement is the following:

```
indP=find(Var(:,2)>0);indN=find(Var(:,2)<=0);
```

Positive values mean that the reaction is direct, whereas negative values mean that the reaction reverses.

```
%% Calcium sulphate production from lime and sulphur impurity
%% in clean coal combustion (Chapter 5)
% Reaction is 2 CaO + 2 SO2 + O2 <==> 2 CaSO4
%% Initialization
InitChem;
% common parameters
Temp0=273.15;R = 8.314472;
P = 1; % relative to standard pressure 1 atm  (dimensionless)
logDisp=0;  % regulates printout
%%  Reaction parameters and temperature
disp('Reaction:  2 CaO + 2 SO2 + O2 <==> 2 CaSO4');
% --- To be filled by user -------
dimc=1; % case No.
dims=4; % substance No
dimr=3; % reactant No
sr=cell(dimc,dims); % substance name
sr(1,:)={ 'CaO','SO2' ,'O2', 'CaSO4'};
cr=zeros(dimc,dims); % stoichiometry
cr(1,:)=[ 2,2,1,2];
% temp. range
rTemp=[800; 2850; 10];  % Celsius degree, single range
Tmelt=1733-Temp0; Tphase=1473-Temp0;
% --- End of user data
TempMin=rTemp(1)+Temp0;TempMax=rTemp(2)+Temp0;
dTemp=rTemp(3); % temp. step
dimTemp=floor((TempMax-TempMin)/dTemp)+1;
%% Reading NASA table and building data base
[row,upperTemp,Tcoef,logHalt]=...
    NASAdata(sr,TempMax, TempMin,dims, dimc);
if logHalt==1, return; end
%% Solving equilibrium at each temperature
T=TempMin-dTemp; % before initial
Var=zeros(dimTemp,2);
for i=1:dimTemp
  T=T+dTemp;  Var(i,1)=T-Temp0;
```

```
  % interpolating points and coeff table
  TxH = [-1/T,log(T),T,T^2/2,T^3/3,T^4/4,T^5/5,1,0];
  TxS = [-1/T^2/2,-1/T,log(T),T,T^2/2,T^3/3,T^4/4,0,1];
  Rx = ThermoCoef(T,Tcoef,upperTemp,row,cr,dimc,dims,dimr);
  % DeltaH, Enthalpy , DeltaS, Entropy , DeltaG, free Energy
  DeltaH = R*sum(TxH.*Rx(1,:));  DeltaS = R*sum(TxS.*Rx(1,:));
  DeltaG = DeltaH - T*DeltaS;
  Var(i,2)=DeltaG*KiloScale;
end
%% Result display and plot
disp('Results'); fprintf('  Temp.[°C] DeltaG[kJ/mol] \n');
format=' %6.1f %12.3f \n';
fprintf(format,Var(1,:)) % first
if logDisp==1
    for i=2:dimTemp-1, fprintf(format,Var(i,:)); end
end
fprintf(format,Var(dimTemp,:)) % last
% separating negative and positive values
indP=find(Var(:,2)>0);indN=find(Var(:,2)<=0);
sT=(Var(min(indP),1)+Var(max(indN),1))/2;
sE=(Var(min(indP),2)+Var(max(indN),2))/2;
% melting and phase change points
indM=find(Var(:,1)<=Tmelt); Emelt=Var(max(indM),2);
indPh=find(Var(:,1)<=Tphase); Ephase=Var(max(indPh),2);
% plot
figure('name', 'Equilibrium'); hold on; grid on;
plot(Var(indP,1),Var(indP,2),'r') ;
plot(Var(indN,1),Var(indN,2),'b') ;
plot(sT,sE,'ro','MarkerSize',12);
plot(Tmelt,Emelt,'bs','MarkerSize',12);
plot(Tphase,Ephase,'gd','MarkerSize',12);
xlabel('Temperature [$^\circ$C]');
ylabel('Energy [kJ]');
title(' 2 CaO + 2 SO2 + O2 $\Longleftrightarrow$ 2 CaSO4');
legend('Reaction proceeds','Reaction reverses',...
    'Equilibrium point','Melting point', 'Phase change',...
    'FontSize',16,'Location','northwest');
```

5.13 Calcium carbonate ($CaCO_3$) decomposition and kinetics

5.13.1 Description and graphical plot

Since ancient times, *lime*, being known to possess adhesive property with bricks and stones, is commonly used as a binding material in masonry works. It is also employed in whitewashing as a wall coat to make whitewash to adhere onto the wall. Limestone is extracted from either quarries or mines. The extracted stone is calcinated in different types of lime kilns/furnaces, which produce quicklime according to the following equilibrium reaction:

$$CaCO_3(s) \Leftrightarrow CaO(s) + CO_2(g). \tag{5.29}$$

Temperature strongly influences the equilibrium of Eq. (5.29), which is shifted to the right side during the kiln heating cycle from 800°C to 1000°C.

Before being employed, quicklime is hydrated, or combined with water, in a process called *slaking*. Hydrated lime is therefore known as slaked lime and is produced by the reaction:

$$CaO + H_2O \Rightarrow Ca(OH)_2. \tag{5.30}$$

Since reaction (5.30) completes at room temperature, it must be studied not only in terms of the thermodynamic pair $\{\Delta H, \Delta S\}$, and therefore of ΔG, but also from the point of view of the reaction yield.

In chemistry, *yield*, also referred to as *reaction yield*, is a measure of the quantity either in [mol] or [kg] units of the product formed, in agreement with the theoretical amount which is provided by *stoichiometry*. In chemical reaction engineering, *conversion* and *yield* are terms used to describe (1) the ratio of how much reactant has been consumed (*conversion*) and (2) how much desired product has been formed (*yield*) in agreement with theoretical values. Yield is usually expressed as a percentage ($0\% \sim 100\%$) or a fraction ($0 \sim 1$). In a chemical process, yield depends on several factors, like reaction time, mixing degree of reactants, diffusion, and others. In the following, the temporal simulation of the reaction (5.29), as well as thermodynamic variables and CO_2 equilibrium pressure, are calculated for a heating rate of 2 °C/min and a simulation time step of 1 min. In this time interval, the reaction yield (fraction) is adjusted to be 0.005, but the users can modify the value. The small value is justified by the slow diffusion of the gaseous product, CO_2, throughout the solid mass of lime and by the short simulation step. Simulation develops from 500 K up to 1300 K, but the CO_2 emission is assumed to occur as soon as the equilibrium $p(CO_2)$ reaches or overcomes the standard value in the atmosphere, that is 0.40 milliatm.

The script follows the standard scheme of thermodynamic calculations in Section 5.6.1, with additional paramaters which provide the initial material purity, in other terms 100% $CaCO_3$ (molar fraction = 1), the standard pressure in the atmosphere $p(CO_2) = 0.40$ milliatm, and the yield which is fixed to 0.005. As soon as the reaction proceeds, CaO is obtained from $CaCO_3$, so its molar fraction decreases from 1 to 0.

5.13.2 Matlab script

The script follows the evolution of the reaction in a stepwise procedure, by first calculating in Section 5.6.1, at each temperature, the equilibrium constant from the NASA CEA database [3], and then the equilibrium concentration of CO_2. As the temperature continuously grows, at each temperature step, only a part of the computed CO_2 will be produced, in agreement with to the concept of the reaction yield. Under a heating rate of 2°C/min, the reaction yield has been fixed to 0.5 %/min but the rate can be varied by users. Fig. 5.9 shows, versus temperature, the profile of the molar fraction of the unreacted $CaCO_3$ and the reaction rate as well.

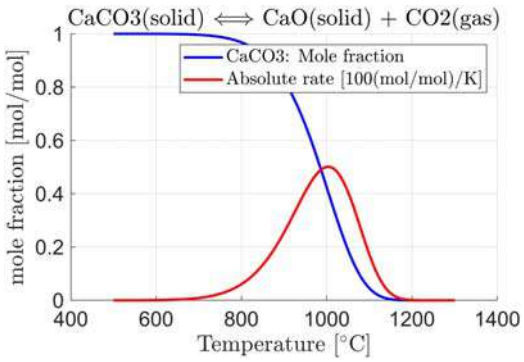


FIGURE 5.9

Reaction coordinate as a molar fraction and the absolute rate scaled by 100.

```
%% Calcium carbonate decomposition and kinetics (Chapter 5)
% Reaction is CaCO3(solid) <==> CaO(solid) + CO2(gas)
%% Initialization
InitChem;
Temp0=273.15;R = 8.314472;% common parameters
Kp0=0.00041; yield=0.005; Conc0=1; scale=100;
% kinetic parameters
logDisp=0; %regulates printout
%%  Reaction parameters and temperature
disp('Reaction: CaCO3(solid) <==> CaO(solid) + CO2(gas)');
% --- To be filled by user -------
% reaction data from user
dimc=1; % case No.
dims=3; % substance No
dimr=1; % reactant No
sr=cell(dimc,dims); % substance name
sr(1,:)={ 'CaCO3','CaO' ,'CO2'};
cr=zeros(dimc,dims); % stoichiometry
cr(1,:)=[ 1,1,1];
% temperature range
rTemp=[500; 1300; 2];  % Celsius degree, same range
% --- End of user data
TempMin=rTemp(1)+Temp0;TempMax=rTemp(2)+Temp0;
dTemp=rTemp(3); % temp. step
dimTemp=floor((TempMax-TempMin)/dTemp)+1;
```

```
%% Reading NASA table and building data base
[row,upperTemp,Tcoef,logHalt]=...
    NASAdata(sr,TempMax, TempMin,dims, dimc);
if logHalt==1, return; end
%% Solving equilibrium at each temperature
T=TempMin-dTemp; % before initial
ConcOld=Conc0; % kinetics initial
Var=zeros(dimTemp,2);
for i=1:dimTemp
  T=T+dTemp;  Var(i,1)=T-Temp0;
  % interpolating points and coeff table
  TxH = [-1/T,log(T),T,T^2/2,T^3/3,T^4/4,T^5/5,1,0];
  TxS = [-1/T^2/2,-1/T,log(T),T,T^2/2,T^3/3,T^4/4,0,1];
  Rx = ThermoCoef(T,Tcoef,upperTemp,row,cr,dimc,dims,dimr);
  %  DeltaH, Enthalpy, DeltaS, Entropy , DeltaG, free
  DeltaH = R*sum(TxH.*Rx(1,:));  DeltaS = R*sum(TxS.*Rx(1,:));
  DeltaG = DeltaH - T*DeltaS;
  Kp=exp(-DeltaG/(R*T));
  % Kinetics vs temperature
  Conc=ConcOld*(1-yield*max((Kp-Kp0),0));  Var(i,2)=Conc;
  AbsRate=scale*abs(Conc-ConcOld)/dTemp;Var(i,3)=AbsRate;
  ConcOld=Conc;
end
%% Result display and plot
disp('Results');
fprintf('Temp.[°C] MoleFract.[mol/mol] Abs.Rate[(mol/mol)/K]\n');
format=' %6.1f %14.3f %18.3f\n'; fprintf(format,Var(1,:))
if logDisp==1
    for i=2:dimTemp-1,  fprintf(format,Var(i,:)); end
end
fprintf(format,Var(dimTemp,:))
% plot
figure('name', 'Kinetics'); hold on; grid on;
plot(Var(:,1),Var(:,2),'b') ;
plot(Var(:,1),Var(:,3),'r') ;
xlabel('Temperature [$^\circ$C]');
ylabel('Mole fraction [mol/mol]');
legend('CaCO3:Mole fraction','Absolute rate[100(mol/mol)/K]',...
    'FontSize',14);
title('CaCO3(solid)$\Longleftrightarrow$CaO(solid)+CO2(gas)')
```

References

[1] M.M. Abbott, H.C. van Ness, Thermodynamics, McGraw-Hill, 1972.
[2] R.H. Petrucci, F.G. Herring, J.D. Madura, C. Bissonette, General Chemistry Principles and Modern Applications, Tenth Edition, Pearson, Canada, 2011.
[3] B.J. McBride, M.J. Zehe and S. Gordon, *NASA Glenn Coefficients for Calculating Thermodynamic Properties of Individual Species*, September 2002. Available from: https://ntrs.nasa.gov/api/citations/20020085330/downloads/20020085330.pdf.
[4] CEA, *Chemical Equilibrium with Applications*. Available from: https://cearun.grc.nasa.gov/ThermoBuild/ and https://ntrs.nasa.gov/citations/20020085330.
[5] Companion Website of the Book. Elsevier, 2021. Available from: https://www.elsevier.com/books-and-journals/book-companion/9780323913416.

CHAPTER

Physical properties of gases and vapors aided by Matlab

6

6.1 Introduction

In an ideal gas, particles, either atoms or molecules, freely move inside a constant volume container without each other interacting, except for very brief collisions in which they exchange energy and momentum. By colliding with the container's walls, they exchange energy with their thermal environment. Particle energy follows what is known as Maxwell−Boltzmann statistics (J.C. Maxwell 1831−79, L.E. Boltzmann 1844−1906), and the derived statistical distribution of speeds will be discussed in Section 6.2.

As a consequence of the above random motion, the three macroscopic parameters of ideal gases, namely pressure P, volume V, and absolute temperature T, are related by what is known as the *ideal gas law*. Further assumptions are dimensionless particles and interaction with elastic collisions, not subjected to other attractive/repulsive forces. Under such assumption, the ideal gas law takes the form

$$PV = nRT, \tag{6.1}$$

where n is the number of moles of the particles, in other terms, it is their real number in the container divided by the Avogadro's number. If the pressure P is measured in atm units, the volume V in dm^3(L), and T in Kelvin, the universal gas constant R holds:

$$R = 0.082 \text{ atm L } K^{-1}mol^{-1} \tag{6.2}$$

In other instances of this book, the SI value of R is employed, namely 8.314 JK^{-1} mol^{-1}.

As a consequence, the product PV is a constant value under constant temperature (*Boyle's law*), implying that, at a very high pressures, the gas volume would shrink to an extremely small size, approaching zero. This fact cannot occur, because the molecules themselves occupy a finite space (called *covolume*) as they are practically incompressible. Another consideration is that intermolecular attractive forces exist between gas molecules. Because of the intermolecular attractive forces, the momentum exchanged with the container walls during the collisions of the gas molecules must be smaller than that expected by an ideal gas. As a consequence, the measured gas pressure must drop to a smaller value. In summary, gases tend to behave ideally at high temperatures and low pressures, but they tend to depart from the ideal gas behavior (real gases) at low temperatures and high pressures.

A number of equations are available for describing the behavior of real gases in a wide range of temperatures and pressures. Such equations are not as generic as the ideal gas equation, since they must contain terms which are specific of each gas.

Usually, such equations are capable of correcting for the proper volume or *covolume* of the molecules and intermolecular forces of attraction. Of the equations that chemists employ for

Fundamental Chemistry with Matlab. DOI: https://doi.org/10.1016/B978-0-323-91341-6.00012-4

modeling the behavior of *real gases*, the van der Waals (1837−1923) equation

$$\left(P + \frac{an^2}{V^2}\right)(V - nb) = nRT \tag{6.3}$$

only requires two specific coefficients, a and b, whose values vary from molecule to molecule [1,2].

In the course of this chapter, Eq. (6.3) will be presented in graphical form, paying attention, in Section 6.4, to the case of gas *condensation* into liquid, and in Section 6.3 to the *compressibility factor* Z, which is defined by the dimensionless ratio $Z = PV(nRT)^{-1}$.

6.2 Distribution of molecular velocities in the case of oxygen

6.2.1 Description and graphical results

Gas particles randomly move, continuously exchanging their kinetic energy during mutual collisions. The particle absolute velocity v is related to kinetic energy by $E_c = 0.5mv^2$. The probability density function for the particle speed in a gas follows the Maxwell−Boltzmann probability density [3]:

$$f(v)dv = 4\pi v^2 \left(\frac{m}{2\pi RT}\right)^{3/2} \exp\left(\frac{-mv^2}{2RT}\right) dv, \tag{6.4}$$

where v is the absolute velocity in $[\mathrm{m/s}]$, m is the mass in [kg] of 1 mole of the substance, $R = 8.314\ \mathrm{JK^{-1}mol^{-1}}$ is usually given in SI units, and T[K] is the absolute temperature. A simple Matlab® script computes and graphically plots in Fig. 6.1, left, the above probability density for the oxygen, with molar mass = 0.032 kg, at three different temperatures. The relationship between temperature and mean kinetic energy is also displayed in Fig. 6.1, right. Different gases can be simulated by the script, by simply changing the oxygen molar mass.

6.2.2 Matlab script

The script refers to the oxygen molecule, but relevant data can be modified by users. The script sections are initialization, user data, computation, and graphical plot of the results.

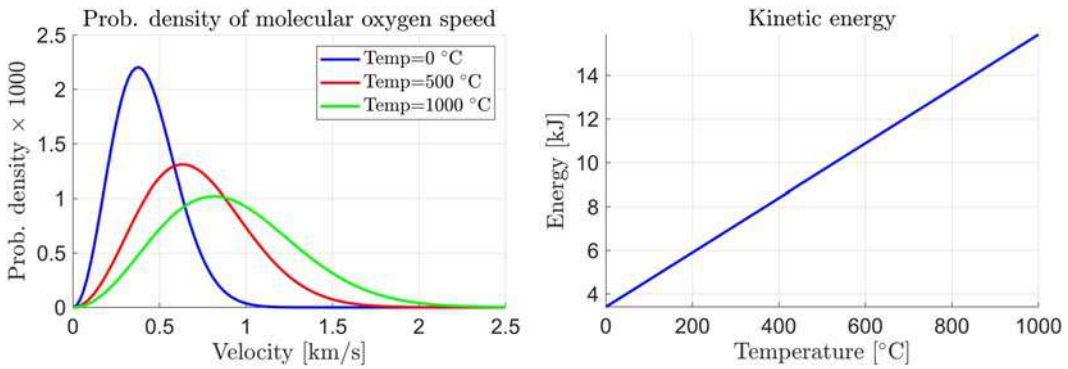


FIGURE 6.1

Oxygen molecule (O_2). (Left) Probability density of the molecular speed. (Right) Kinetic energy versus temperature.

```
%% Probability density of oxygen molecular speed (Chapter 6)
%% Initialization
InitChem;
R = 8.3145;     % gas constant J/(mol*K)
Temp0=273.15;
%% Parameters
% --- user can change them
M = 0.032;    % mass [kg] of 1.00 mol of oxygen (O2)
u=0:20:2500; % velocity range [m/s]
Temp=[0; 500; 1000]+Temp0;
%  --- end of user data
%% Computation: probability density and kinetic energy
dimu=length(u);dimTemp=length(Temp);
% probability density
P=zeros(dimu,dimTemp);
for i=1:dimTemp
T=Temp(i);
P0=4*pi*(M/(2*pi*R*T))^1.5;
P(:,i)=P0*u.^2.*exp(-u.^2*M/(2*R*T));
end
P=P/KiloScale;u=u*KiloScale;
% kinetic energy
Temp=0:200:1000;
E= 1.5*R*(Temp+Temp0);
%% plot
figure('name','Prob. distr.');hold on;grid on;
plot(u,P(:,1),'b',u,P(:,2),'r',u,P(:,3),'g')
xlabel('Velocity [km/s]');ylabel('Probability density $\times$ 1000')
title('Probability density of molecular oxygen speed')
legend('Temp=0 $^\circ$C','Temp=500 $^\circ$C','Temp=1000 $^\circ$C',…
    'FontSize',18);
figure('name','Kinetic energy');hold on;grid on;
plot(Temp,E*KiloScale,'k')
xlabel('Temperature [$^\circ$C]');ylabel('Energy [kJ]');
title('Kinetic energy')
```

6.3 Compressibility of a real gas

6.3.1 Description and graphical results

The ideal gas equation

$$PV = nRT \tag{6.5}$$

tells us that the ratio $Z = PV/(nRT)$ is equal to $Z = 1$ for ideal gases, whereas, for real gases, the dimensionless compressibility factor Z may have values which are significantly different from 1.

The compressibility factor is usually plotted for real gases versus the pressure P under isothermal conditions. Here, the pressure is given in the standard atmosphere unit [atm] and consequently $R = 0.082$ atm L $K^{-1}mol^{-1}$. At very high pressures, we have $Z > 1$. In fact, because of the finite size of molecules, the product PV at high pressures is larger than that predicted for ideal gases. Intermolecular forces of attraction account for $Z < 1$. These forces become increasingly important at low temperatures, where the molecular translational motion slows down [2]. The van der Waals Eq. (6.3) for real gases can be rearranged into the third-degree polynomial

$$nRTV^2 = PV^3 - PnbV^2 + an^2V - abn^3, \tag{6.6}$$

which, solved by the Matlab function `vpasolve`, provides the value of the volume V for a given P and finally the compressibility factor Z.

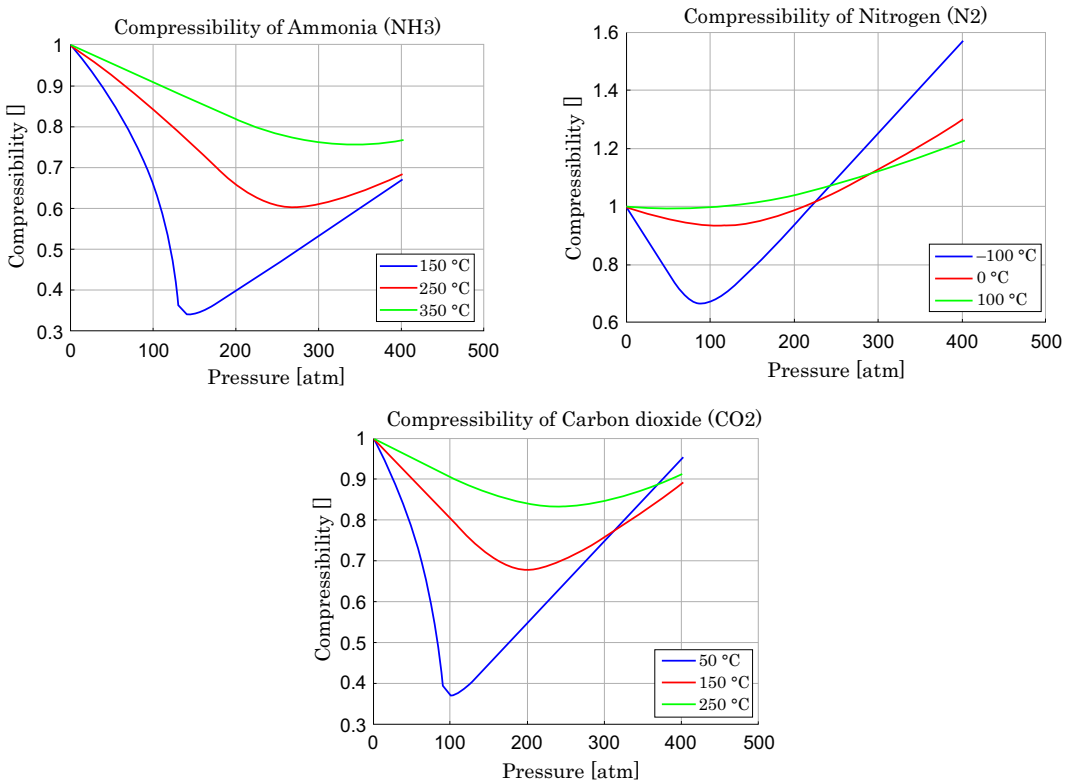


FIGURE 6.2

Compressibility profiles versus pressure. (Top left) Ammonia (NH_3), (Top right) nitrogen (N_2), (Bottom) carbon dioxide (CO_2).

Fig. 6.2 shows the compressibility profiles (dimensionless) versus pressure in [atm]. The three diagrams refer to ammonia (top left), nitrogen (top right), and carbon dioxide (bottom). The readers

may exploit the Matlab script in Section 6.3.2 to add other chemical species or to change the previous species.

6.3.2 Matlab script

The Matlab script is organized into three sections, initialization, user data, and computation. Computation and graphical results are inside the loop of the species with dimension `dims`, here set equal to three. User data can be changed or enlarged paying attention to maintain coherent dimensions between the different vectors and matrix, namely `a, b, s, Temp`. The first three vectors contain `dims` rows, whereas the size of the matrix `Temp` is `dimsx3`. Given pressure, temperature, and number of moles, the volume `V` is obtained by solving the third-order algebraic Eq. (6.6) using the symbolic function `vpasolve`.

```
%% Real gas compressibility (Chapter 6)
%% (nitrogen, ammonia, carbon dioxide)
%% Initialization
InitChem;
syms V; Temp0=273.15;
% WARNING pressure in atm , energy in atm*L
R=0.082; % gas constant atm*L/(K*mol)
%% Parameters
% Van der Waals coefficients for common gases
% ---- user data
% NH3 (ammonia), N2 (nitrogen), CO2 (carbon dioxide)
a=[4.225;1.37;3.658];
b=[0.0371;0.0387;0.0429];
s={' Ammonia (NH3)',' Nitrogen(N2)',' Carbon dioxide (CO2)'};
n=1; % unit mole of gas
% Temperature points
% min temp. (first column in the matrix below) must be greater
% than critical temp.: 130 °C (NH3), -147 °C (N2), 37 °C (CO2)
Temp=[150 250 350;  % NH3
-100 0 100;  %  N2
50 150 250];  % Co2
% Pressure range
P= 1:10:401;  %pressure range
% --- end of user data
```

```
%% computation and graphical results
dimP=length(P); dims=length(s);
[nrow,dimTemp]=size(Temp);
Var=zeros(dimP,dimTemp+1); Var(:,1)=P;
for k=1:dims% loop on species
   for i=1:dimTemp% loop on temp. points
        T=Temp(k,i)+Temp0;
        for j=1:dimP% loop on pressure range
           % P*V^3 - P*n*b*V^2 + a*n^2*V - a*b*n^3=n*R*T solved in V
           y1=P(j)*V^3-(P(j)*n*b(k)+n*R*T)*V^2+a(k)*n^2*V-a(k)*b(k)*n^3;
           S =double( vpasolve(y1 == 0, V,[0 Inf]));
           Var(j,i+1) = P(j)*double(S)/(n*R*T);
       end
   end
   % plot
   figure('name','Gas compress'); hold on; grid on;
   plot(Var(:,1), Var(:,2),'b');
   plot(Var(:,1), Var(:,3),'r');
   plot(Var(:,1), Var(:,4),'g');
   xlabel('Pressure [atm]');  ylabel('Compressibility []')
   stitle='Compressibility of '; stitle=strcat(stitle,cell2mat(s(k)));
   title(stitle);
   Tunit=' $^\circ$C';
   leg1=strcat(num2str(Temp(k,1)),Tunit);
   leg2=strcat(num2str(Temp(k,2)),Tunit);
   leg3=strcat(num2str(Temp(k,3)),Tunit);
   legend(leg1,leg2,leg3,'Location','southeast')
end
```

6.4 van der Waals isotherm of real gases

6.4.1 Description and graphical results

The van der Waals equation in Eq. (6.3) can be rearranged into an expression of the type $P = P(V)$, as follows:

$$P(V) = \frac{nRT}{V - nb} - \frac{an^2}{V^2}, V > nb = V_{min}. \tag{6.7}$$

The graph of $P = P(V)$ described by the above equation, is known as the *van der Waals isotherm*. The critical point of a gas is defined as the temperature limit above which the examined gas cannot be liquefied for whatever pressure is applied. If the gas temperature is lower than this critical point, the isotherm is nonmonotonic versus the volume V. In other terms, in a certain critical volume interval of the curve, the calculated pressure should decrease with increasing volume, a fact clearly inconsistent. Therefore van der Waals isotherm should be adjusted in this critical region where local minimum and maximum take place so as to become ideally constant. The physical

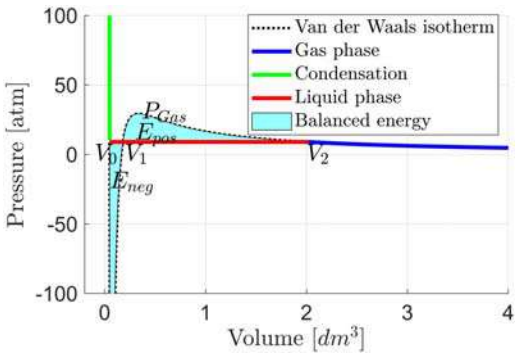


FIGURE 6.3

van der Waals isotherm with energy areas to be balanced and the volumes intersected by the condensation line.

meaning of this region (known as the *condensation region*), between the extreme volumes $\{V_0, V_2\}$ to be found, is that both gas and liquid phases coexist, since the gas phase progressively condenses to become liquid as far as the volume decreases at constant pressure. J.C. Maxwell (1831−79) suggested to better define this region by balancing the areas $\{E_{\text{neg}}, E_{pos}\}$ of negative and positive energy in [atm L] unit, below and above the rectified curve (*condensation line*), which corresponds to a constant pressure P_{cond}. The adjustment is known as the Maxwell construction [3,4] (see Fig. 6.3).

The relevant mathematical problem can be converted into the following minimum problem:

$$P_{cond} = \text{argmin}_{p>0} \left| E_{\text{neg}}(p) + E_{pos}(p) \right|. \tag{6.8}$$

During the minimum search, the extreme volumes $\{V_0(p), V_2(p)\}$, where the condensation line joins the van der Waals isotherm, are varying together with the current pressure p.

The problem (6.8) requires to express the above areas in term of the current pressure p. To this end, let us repeat the expression (6.7) of the van der Waals isotherm (pressure P, volume V):

$$P(V) = \frac{nRT}{V - nb} - \frac{an^2}{V^2}, V > nb = V_{min}. \tag{6.9}$$

Given a line at constant pressure which intersects the above curve in three points defined by

$$(V_0, p), (V_1, p), V(V_2, p), \tag{6.10}$$

the relevant volumes are found as the real roots of the algebraic equation

$$p = \frac{nRT}{V - nb} - \frac{an^2}{V^2}. \tag{6.11}$$

Given the intersection points, the energy areas $\{E_{\text{neg}}, E_{pos}\}$ can be proved to have the following expressions (the reader is asked to prove them):

$$\begin{aligned} E_{\text{neg}}(p) &= nRT \log \frac{V_1(p) - nb}{V_0(p) - nb} - \left(\frac{an^2}{V_0(p) V_1(p)} + p \right) (V_1(p) - V_0(p)) \\ E_{\text{pos}}(p) &= nRT \log \frac{V_2(p) - nb}{V_1(p) - nb} - \left(\frac{an^2}{V_2(p) V_1(p)} + p \right) (V_2(p) - V_2(p)) \end{aligned}. \tag{6.12}$$

In order to find the minimum of Eq. (6.9) in an iterated way, we need to start from a feasible pressure $p_0 > 0$ located below the maximum pressure P_{gas} of the gas phase. The corresponding point (V_{gas}, P_{gas}) can be found by searching the roots of the algebraic equation of the isotherm stationary points:

$$\frac{dP(V)}{dV} = \frac{-nRT}{(V-nb)^2} + \frac{2an^2}{V^3} = 0. \tag{6.13}$$

Only two real roots $\{V_{liq}, V_{gas}\}$, if they exist, belong to the volume range $V > nb$: they indicate the local minimum and maximum points of the original van der Waals isotherm. In the case that the roots become complex, van der Waals isotherm approaches a monotonic decreasing ideal gas isotherm, thus providing a criterion for discriminating between the existence or not of the condensation phase. In the Matlab script below, we set $p_0 = P_{gas}/2 > 0$.

Given the initial pressure, the search is mechanized by the Matlab function `fminsearch`, as in the following statements of the script.

```
VGas=max(Vroot); p=Pressure(VGas,par)/2;
fer =@(p)Maxwell(p,par); % fer= sum of energy (magnitude)
if logDisp==1
   options = optimset('Display','iter','PlotFcns',@optimplotfval);
   [p1,fer1,exitflag,output]=fminsearch(fer,p,options);
else
   [p1,fer1,exitflag,output]=fminsearch(fer,p);
end
```

`Vgas` is the max volume of the gas phase, which allows us to compute the initial pressure `p`. The function to be minimized is denoted by `fer`, and the final value is denoted by `fer1` together with the optimal condensation pressure `p1`. The Boolean variable `logDisp` allows the Matlab function to printout each minimization step together with a graphical view. Iterations are halted by a default tolerance on the absolute change of the functional.

The statement `fer =@(p)Maxwell(p,par);` illustrates the technique of the *anonymous functions* (a frozen-structure function) `fer =@(p)UserFunction(p, other parameters)` with a single argument `p` and a single output `fer`. The function is capable of executing the attached `UserFunction()` in charge of detailing the computational rule of the functional `fer`. Here, the user function is `Maxwell(p,par)`. Let us further remark that `@optimplotfval` is just the call to the function `optimplotfval` in the heading of another function, `optimset`.

Fig. 6.4 shows the van der Waals isotherm at 30°C for the gas species, ammonia (NH_3, top left), nitrogen (N_2, top right), and carbon dioxide (CO_2, bottom). At 30°C, only ammonia and carbon dioxide undergo condensation as shown by the adjusted profiles obtained by the Maxwell construction. The balanced positive and negative energy areas are colored in faint cyan. The adjusted isotherm is split into three sections: gas phase (blue), condensation (red), and liquid phase (green).

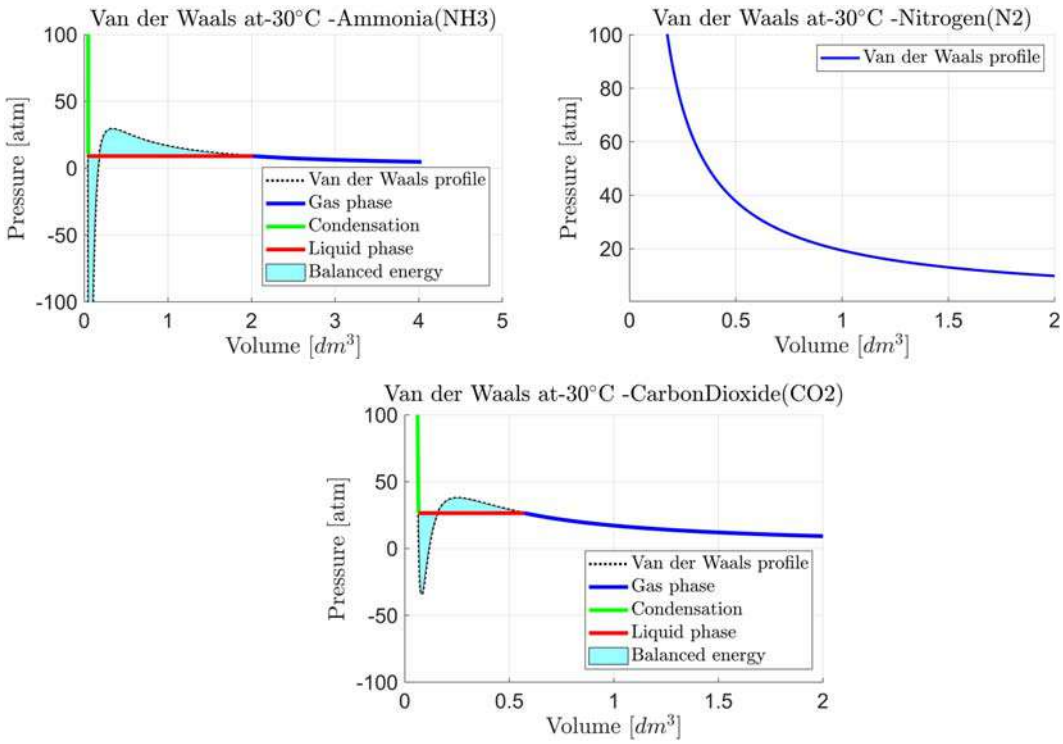


FIGURE 6.4

Real gas pressure versus volume profile for three gas species (the van der Waals isotherm is adjusted to become monotonically decreasing).

6.4.2 Matlab script

The main script splits into initialization, user data section which includes the real gas parameters and the definition of the volume and pressure ranges, and the search of the condensation line which includes graphical results. The search statements are preceded by a test on the existence of the condensation phase, which splits the following statements into two cases. The *search phase* employs the Matlab function `fminsearch`. The function in turn requires another function, `Maxwell`, in charge of computing the functional `fer`, which has to be minimized on the basis of the current variable `p` (pressure) and other parameters. The complete printout of `fminsearch` (including the iteration plot) is regulated by the Boolean variable `logDisp`.

```
%% Real gases, Van der Waals equation and Maxwell adjustment
%% Initialization
InitChem;
% WARNING Pressure in atm
Temp0=273.15; R = 0.082; % [ atm*L/(mol*K)]
logDisp=1; % regulates optimization printout
logLG=1; % condensation occurs
logText=0; % adds text to case 1 curve
%% Parameters, volume range and pressure bound
% ---- user data
disp('Van der Waals profile and condensation search');
s ={'Ammonia(NH3)','Nitrogen(N2)','CarbonDioxide(CO2)'};
a =[ 4.225;1.37; 3.658];
% ammonia(NH3) nitrogen(N2)carbon diox.(CO2)
b =[ 0.0371;  0.0387;0.0429];
k=1; % substance to be selected <<< user
n=1; % number of moles
Temp =[ -30; -30;-30]; % Temperature in °C of the isotherm
T = Temp(k)+Temp0; % Temperature in K
% 1) Vol. resolution, small to fit liquid phase
% 2) Vol. bound, it may  be enlarged by search
% 3) Init Volume, given by max pressure
Pmax=100; Vmax=2; dV=0.00005;
c=zeros(3,1);nIter=4; % init volume
for i=1:nIter, c =R*T./((Pmax+a*n./(b+c*dV).^2)*dV); end
V=n*(b(k)+c(k)*dV):dV:Vmax; % volume range
% --- end of user data
%% Search of condensation range
% condensation occurs?
par(1)=n*R*T;par(2)=n^2*a(k);par(3)=n*b(k);
P = Pressure(V,par);
[dP,Vroot]=dPressure(V,par);
if sum(abs(imag(Vroot)))>0, logLG=0; end
species=cell2mat(s(k));
% two cases
if logLG==1  % condensation
    fprintf('%s - Condensation at %6.1f°C\n',species,Temp(k));
    VGas=max(Vroot); p=Pressure(VGas,par)/2;
    fer =@(p)Maxwell(p,par); % fer= sum of energy (magnitude)
    if logDisp==1
        options = optimset('Display','iter','PlotFcns',
        @optimplotfval);
        [p1,fer1,exitflag,output]=fminsearch(fer,p,options);
```

```
    else
        [p1,fer1,exitflag,output]=fminsearch(fer,p);
    end
% enlargement if needed of volume range
    if V3>max(V)
        V=min(V):dV:V3*2;P=Pressure(V,par);
        [dP,Vroot]=dPressure(V,par);
    end
% gas, condensation,liquid regions
    logCond= V>=V1 & V<=V3;
    indCond=find(logCond); dimCond=length(indCond);
    indGas=find(V>=V3); indLiq=find(V<=V1);
    % plot
    PCond=ones(dimCond,1)*p1; %condensation line
    figure('name','PV');hold on; grid on;
    plot(V,P,'k:');  % Van der Waals
    plot(V(indGas),P(indGas),'b','LineWidth',3);
    plot(V(indLiq),P(indLiq),'g','LineWidth',3)
    % filling balanced energy
    plot(V(indCond),PCond,'r','LineWidth',3);
    x=[V(indCond),fliplr(V(indCond))];
    inBetween=[PCond.',fliplr(P(indCond))];
    fill(x,inBetween,[ 0.6 1 1]); % faint cyan
    % condensation line
    plot(V(indCond),PCond,'r','LineWidth',3);
    xlabel('Volume $[dm^3]$'); ylabel('Pressure [atm]');
    ylim([-Pmax Pmax])
    legend('Van der Waals profile','Gas phase',...
        'Condensation', 'Liquid phase','Balanced energy');
    % adding text
    if k==1 && logText==1
        text(0.3,15,'$E_{pos}$', 'FontSize',18,'Interpreter','Latex');
        text(0.05,-20,'$E_{neg}$', 'FontSize',18,'Interpreter','Latex');
        text(02.,0,'$V_{2}$', 'FontSize',18,'Interpreter','Latex');
        text(0.2,0,'$V_{1}$', 'FontSize',18,'Interpreter','Latex');
        text(-0.1,0,'$V_{0}$', 'FontSize',18,'Interpreter','Latex');
        text(0.365,30,'$P_{Gas}$', 'FontSize',18,'Interpreter','Latex');
    end
else % 2) no condensation
    fprintf('%s - No condensation at %6.1f\n',species,Temp(k));
    figure('name','PV');hold on; grid on;
    plot(V,P,'k');
    xlabel('Volume $[dm^3]$'); ylabel('Pressure [atm]');
    ylim([0 Pmax])
    legend('Van der Waals profile','FontSize',18);
end
t1=('Van der Waals at');t2=num2str(Temp(k));t3='$^\circ$C - ';
t=strcat(t1,t2,t3,species);title(t);
```

The main script invokes three functions: `Pressure()`, `dPressure()`, and `Maxwell()`, which are reported below. They are self explaining, except for the following statements of `Maxwell()`.

```
% Intersection volumes to workspace
assignin('base','V1',V(1));
assignin('base','V2',V(2));
assignin('base','V3',V(3));
```

The previous statements convert the entries of the local vector `V` into the `workspace` variables `V1`, `V2`, and `V3`. We must recall that the internal variables of a function are local and not visible by the `workspace`. As a second point, the *anonymous function* called by `fminsearch` can only provide as output the functional `fer`, which justifies the need of the above conversion.

```
%% Derivative and stationary volumes
function [dP,Vroot]=dPressure(V,par)
    nRT=par(1); an2=par(2); nb=par(3);
    c=2*an2/nRT;
    coeff=[1 ; -c; 2*nb*c; -nb^2*c];
    Vroot=roots(coeff);
    dP=-nRT./(V-nb).^2+2*an2./V.^3;
end
%% Pressure
function P = Pressure(V,par)
    nRT=par(1); na2=par(2); nb=par(3);
    % Van der Waals equation in the form P = f(V)
    P = nRT./(V-nb) - na2./V.^2;
end
%% Function Maxwell
function fer=Maxwell(p,par)
    % roots of line intersection
    nRT=par(1);an2=par(2); nb=par(3);
    r=nRT/p;d=an2/p;
    c=[1;-(nb+r);d; -d*nb]; % intersection eq. coeff.
    V=sort(roots(c));% ascending order
    % Intersection volumes to workspace
    assignin('base','V1',V(1));
    assignin('base','V2',V(2));
    assignin('base','V3',V(3));
    % Functional to be minimized
    V12=V(1)*V(2);    V23=V(2)*V(3);
    V1=V(1)-nb;V2=V(2)-nb; V3=V(3)-nb;
    dV=V2-V1; Eneg=nRT*log(V2/V1)-(an2/V12+p)*dV;
    dV=V3-V2; Epos=nRT*log(V3/V2)-(an2/V23+p)*dV;
    fer=abs(Eneg+Epos);
end
```

6.5 Water vapor pressure

6.5.1 Description and graphical results

Water left in an open beaker evaporates completely. A different condition establishes if water is placed in a closed container. If both liquid and vapor are present in the container, vaporization and condensation simultaneously occur. If a sufficient liquid volume is present, the condition eventually is reached in which the amount of vapor remains constant. This is an example of dynamic equilibrium: the two opposing processes, evaporation and condensation, take place simultaneously and at equal rates. Even in the absence of chemical reactions, we can apply Equations (5.7), (5.13), and (5.14) of Chapter 5, employ the NASA-CEA coefficients [5,6] to extrapolate the thermodynamic variables H (enthalpy) and S (entropy) of the solid, liquid, and gaseous water, and then compute ΔG^0 of the following transformations:

$$\begin{matrix} H_2O(\text{solid}) \Leftrightarrow H_2O(\text{gas}) \\ H_2O(\text{liquid}) \Leftrightarrow H_2O(\text{gas}) \end{matrix}. \tag{6.14}$$

From the values of ΔG^0, we can deduce the water vapor pressure as follows:

$$K_{eq} = \frac{p\left(H_2O_{\text{gas}}\right)}{H_2O_{\text{liq}}} \Rightarrow K_{eq} = K_P = p\left(H_2O_{\text{gas}}\right) = \exp\left(\frac{-\Delta G^0}{RT}\right), \tag{6.15}$$

where $\left\{H_2O_{\text{liq}}\right\} = 1$, being the activity of pure liquid water, and ΔG^0 refers to a constant pressure equal to $P = 1$ atm, but to a variable temperature. In Eq. (6.15), the equilibrium constant K_{eq} has been renamed K_P since we are treating the equilibrium of a gaseous substance. The Matlab script automatically switches at $0°C$ from solid water (ice) to liquid water. This is obtained by comparing the current temperature T with the temperature range (columns 3 and 4) of the table in Figure 5.2 of Chapter 5. Three rows of the table have been repeated below in Fig. 6.5. They are the rows of H_2O_{cl} where the subscript cl refers to solid and liquid phase (c = cryo-, cold, l = liquid). Accordingly, the correct coefficient row is selected.

Water coefficients (solid and liquid)

H2Ocl	1	200	273	-4.027e+05	2.748e+03	5.738e+01	-8.268e-01	4.413e-03	-1.054e-05	9.694e-09	-5.530e+04	-1.903e+02
H2Ocl	2	273	373	1.326e+09	-2.448e+07	1.879e+05	-7.679e+02	1.762e+00	-2.151e-03	1.093e-06	1.102e+08	-9.780e+05
H2Ocl	3	373	600	1.264e+09	-1.680e+07	9.278e+04	-2.722e+02	4.479e-01	-3.919e-04	1.426e-07	8.113e+07	-5.134e+05

FIGURE 6.5

Rows of the table in Figure 5.2 of Chapter 5 restricted to water phases.

Fig. 6.6 shows the water vapor equilibrium constant $K_P(T) = p\left(H_2O_{\text{gas}}\right)(T)$ versus temperature.

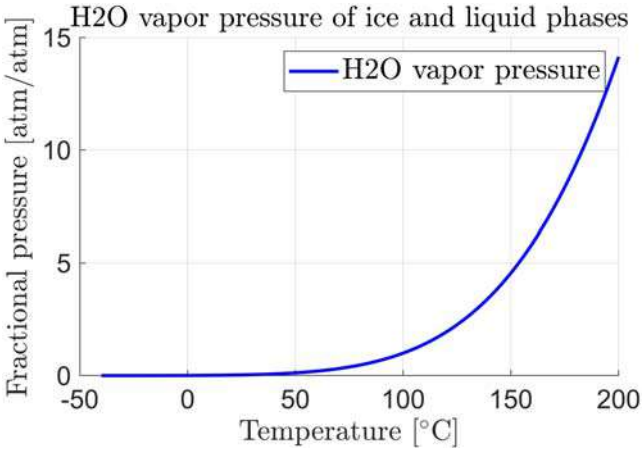


FIGURE 6.6

Water vapor pressure versus temperature.

6.5.2 Matlab script

The organization of the Matlab script follows the scheme of Section 5.6.1 in Chapter 5, where NASA-CEA thermochemical data [5,6] have been employed. The section where equilibrium is computed, ends with the equilibrium constant $K_P(T)$ as a function of temperature, which is the variable to be graphically plotted. As usual, numerical values can be partially or fully listed in the `command window` by setting the Boolean variable `logDisp` to (0/1).

```
%% Water pressure at ice and liquid phase (sea level) (Chapter 6)
%% Initialization
InitChem;
Temp0=273.15;R = 8.314472; % common parameters
logDisp=0; % regulates printout
%%  Reaction parameters and temperature
disp('H2O vapourequilibrium, H2O (gas) - H2O (solid-liquid)');
% --- data from user
dimc=1; % case No.
dims=2; % substance No
dimr=1; % reactant No
sr=cell(dimc,dims); % substance name
sr(1,:)={ 'H2Ocl','H2O'};
cr=zeros(dimc,dims); % stoichiometry
cr(1,:)=[ 1,1];
% temperature range
rTemp=[-40; 200; 1];  % Celsius degree,single range
% ----- end of user data
TempMin=rTemp(1)+Temp0;TempMax=rTemp(2)+Temp0;
dTemp=rTemp(3); % temp. step
dimTemp=floor((TempMax-TempMin)/dTemp)+1;
%% Reading NASA table and building data base
[row,upperTemp,Tcoef,logHalt]=NASAdata(sr,TempMax, TempMin,dims,dimc);
if logHalt==1, return; end
```

```
%% Solving equilibrium at each temperature
T=TempMin-dTemp; % before initial
Var=zeros(dimTemp,2);
for i=1:dimTemp
   T=T+dTemp;  Var(i,1)=T-Temp0;
   % interpolating points and coeff table
   TxH = [-1/T,log(T),T,T^2/2,T^3/3,T^4/4,T^5/5,1,0];
   TxS = [-1/T^2/2,-1/T,log(T),T,T^2/2,T^3/3,T^4/4,0,1];
   Rx = ThermoCoef(T,Tcoef,upperTemp,row,cr,dimc,dims,dimr);
   % DeltaH, Enthalpy, DeltaS, Entropy, DeltaG, free Energy
   DeltaH = R*sum(TxH.*Rx(1,:));  DeltaS = R*sum(TxS.*Rx(1,:));
   DeltaG = DeltaH - T*DeltaS;
   Kp = exp(-DeltaG/(R*T));  Var(i,2)=Kp;
end
%% Result display and plot
disp('Results'); fprintf('  Temp.[°C]   Kp \n');
format=' %6.1f %12.3f \n';
fprintf(format,Var(1,:)) % first
if logDisp==1
   for i=2:dimTemp-1,  fprintf(format,Var(i,:)); end
end
fprintf(format,Var(dimTemp,:)) % last
% plot
figure('name', 'Equilibrium'); hold on; grid on;
plot(Var(:,1),Var(:,2),'k') ;
xlabel('Temperature [$^\circ$C]');
ylabel('Relative pressure [atm/atm]');
title(' H2O vapor pressure of ice and liquid phases');
legend('H2O vapor pressure','FontSize',18);
```

6.6 Water vapor pressure at different altitude and humidity

6.6.1 Description and graphical results

As a further application of the NASA-CEA coefficients in the table of Figure 5.2 (Chapter 5), we compare the water vapor pressure with the CO_2 partial pressure at different altitudes in the Earth's atmosphere. Since both gases are infrared active as greenhouse gases, knowledge of the relative concentrations is of utmost interest. The greenhouse effect is particularly active in the upper troposphere, where the concentration of water vapor drops to very low values, due to decreasing temperature and consequent condensation.

The atmosphere's temperature profile, which assumes a constant temperature decrement, is written as $T = T_0 - Lh = 15 - 0.00649\,h$ °C, where h is the altitude [m] from sea level.

By applying the ideal gas law in Eq. (6.5), one obtains $P = \rho RT/M$, where P is in [atm] units, ρ, the atmosphere density, is given in [kg/m^3], $R = 8.314$ J K^{-1}mol^{-1} is expressed in SI units, T is the absolute temperature, and $M = 0.02896$ is the mean molecular mass of the air in [kg/mol].

We assume hydrostatic pressure, namely a pressure differential equal to $dP = -\rho g\, dh$, where $g = 9.806$ m/s^2 is the gravity acceleration at sea level. The pressure differential divided by P provides the differential equation:

$$\frac{dP}{P} = \frac{-Mg}{RT} dh, P_0 = P(h = 0). \tag{6.16}$$

Integration of Eq. (6.16) from sea level to altitude h and the previous assumption of a linear temperature variation $T = T_0 - Lh$ together with a constant air composition, provide the pressure profile

$$P(T) = P_0 \left(\frac{T}{T_0}\right)^{\frac{Mg}{RL}} = \left(\frac{T}{288}\right)^{5.256}, \; T = T[K] = T[^\circ C] + 273.15. \tag{6.17}$$

Fig. 6.7, left, plots the water vapor pressure versus the altitude in two colors: the blue color refers to ice and the red color to liquid water. The ice melting point at about 2.2 km altitude, indicated by a black circle, corresponds to the liquid–solid state in which vapor is in equilibrium (100% relative humidity, RH). In Fig. 6.7, right, the partial pressures of water and carbon dioxide (at 410 ppmv) are compared, by assuming 50% relative humidity. From about 2.2 km altitude, the contribution of CO_2 is prevalent, and from 8.0 km altitude, the water contribution becomes very low and nearly negligible.

6.6.2 Matlab script

The script follows the organization of Section 5.6.1 in Chapter 5, where NASA-CEA thermochemical coefficients are employed [5,6], with the following exceptions. The temperature range is fixed by the altitude range `rh` of the troposphere up to 11 km from the sea level. Parameters to be used for computing the species partial pressures are given in the initialization. The section for the computation of the equilibrium halts at the equilibrium constant `Kp`. After that, several variables are computed.

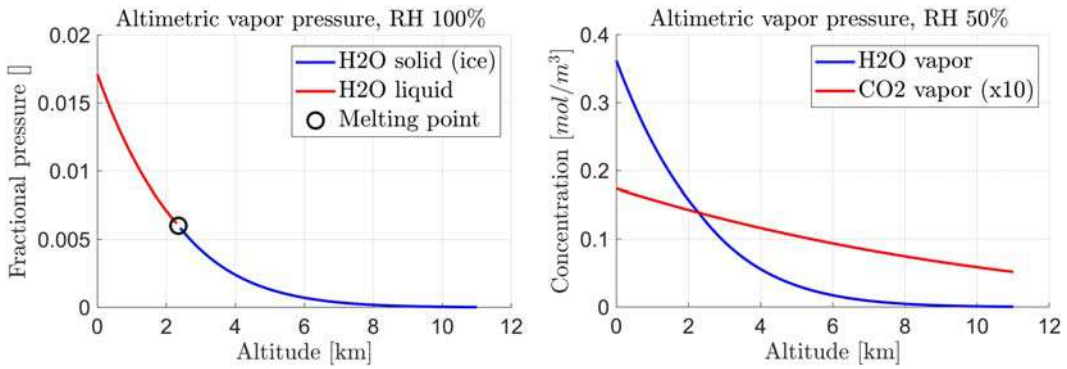


FIGURE 6.7

(Left) Water vapor pressure at different altitudes and RH = 100%. (Right) Comparison of water and carbon dioxide concentration versus altitude.

```
%% Water vapor pressure (Chapter 6)
%% with altitude and humidity (RH)
%% Initialization
InitChem;
Temp0=273.15;R = 8.314472; % common parameters
% linear thermal profile with altitude
Th0=15.04; dTh=0.00649;
% pressure
P0=101.29/101.325; Tp=288.08; Pexp=5.256;
% H2O and CO2 moles
cH2O=0.082; cCO2=0.00041;
logDisp=0; % regulates printout
%%  Reaction parameters and temperature
disp('H2O vapourequilibrium, H2O (gas) - H2O (solid-liquid)');
% ---- data from user
dimc=1; % case No.
dims=2; % substance No
dimr=1; % reactant No
sr=cell(dimc,dims); % substance name
sr(1,:)={ 'H2Ocl','H2O'};
cr=zeros(dimc,dims); % stoichiometry
cr(1,:)=[ 1,1];
% altitude range
rh=[0; 11000; 100];
% --- end of user data
% temperature range
TempMax=Th0+Temp0;TempMin=TempMax-rh(2)*dTh;
dTemp=rh(3)*dTh; % temp. step
dimTemp=ceil((TempMax-TempMin)/dTemp)+1;
%% Reading NASA table and building data base
[row,upperTemp,Tcoef,logHalt]=NASAdata(sr,TempMax, TempMin,dims,
dimc);
if logHalt==1, return; end
%% Solving equilibrium at each temperature
T=TempMin-dTemp; % before initial
h=rh(2)+rh(3); % before initial
Var=zeros(dimTemp,5);
for i=1:dimTemp
    T=T+dTemp;  Var(i,1)=T-Temp0;
    % interpolating points and coeff table
   TxH = [ 1/T,log(T),T,T^2/2,T^3/3,T^4/4,T^5/5,1,0];
   TxS = [-1/T^2/2,-1/T,log(T),T,T^2/2,T^3/3,T^4/4,0,1];
```

```
    Rx = ThermoCoef(T,Tcoef,upperTemp,row,cr,dimc,dims,dimr);
    % DeltaH, Enthalpy ,DeltaS, Entropy, DeltaG, free Energy
    DeltaH = R*sum(TxH.*Rx(1,:));  DeltaS = R*sum(TxS.*Rx(1,:));
    DeltaG = DeltaH - T*DeltaS;
    Kp = exp(-DeltaG/(R*T));  Var(i,3)=Kp;
    % variables
    P=P0*(T/Tp)^Pexp; % pressure
    h=h-rh(3); Var(i,2)=h*KiloScale;  % altitude
    H2Om=Kp/(2*cH2O*T*KiloScale) ;Var(i,4)=H2Om; % H2O  mole/m^3 RH 50%
    CO2m=P*cCO2/(cH2O*T*KiloScale);Var(i,5)=CO2m; % CO2 mole/m^3
end
%% Result display and plot
disp('Results');
fprintf('     Temp.[°C]  Altitude[km]  Pressure     H2O
MFraction[mole/m^3]    CO2    MFraction[mole/m^3]\n');
format=' %6.1f %12.3f %12.3f %12.3f %12.3f \n';
fprintf(format,Var(1,:)) % first
if logDisp==1
   for i=2:dimTemp-1,  fprintf(format,Var(i,:)); end
end
fprintf(format,Var(dimTemp,:)) % last
% separating ice from liquid
indL=find(Var(:,1)>=0);indI=find(Var(:,1)<0);
% switching point
xs=(Var(max(indI),2)+Var(min(indL),2))/2;
ys=(Var(max(indI),3)+Var(min(indL),3))/2;
% plot
figure('name', 'H2O vapor with altitude'); hold on; grid on;
plot(Var(indI,2),Var(indI,3),'b') ;
plot(Var(indL,2),Var(indL,3),'r') ;
plot(xs,ys,'ko','MarkerSize',12);
xlabel('Altitude [km]');
ylabel('Relative pressure [atm/atm]');
title('Altimetric vapor pressure, RH 100\% ','FontWeight','normal');
legend('H2O solid (ice)','H2O liquid','Melting point','FontSize',18);
% moles
figure('name', 'H2O vapor with altitude'); hold on; grid on;
plot(Var(:,2),Var(:,4),'b') ;
plot(Var(:,2),Var(:,5)*10,'r') ;
xlabel('Altitude [km]');
ylabel('Concentration $[mol/m^3]$');
title(' Altimetric vapor pressure, RH 50\% ','FontWeight','normal');
legend('H2O vapor','CO2 vapor (x10)','FontSize',18);
```

References

[1] Wikipedia, *Van der Waals equation*, available from https://en.wikipedia.org/wiki/Van_der_Waals_equation

[2] R.H. Petrucci, F.G. Herring, J.D. Madura, C. Bissonnette, General Chemistry Principles and Modern Applications, 10th (ed.), Pearson, Canada, 2011.

[3] J.C. Maxwell, On the dynamical evidence of the molecular constitution of bodies, Nature 11 (279) (1875) 357−359.

[4] Wikipedia, *Maxwell construction*, available from https://en.wikipedia.org/wiki/Maxwell_construction.

[5] NASA C.E.A., Chemical Equilibrium with Applications, available from https://cearun.grc.nasa.gov/ThermoBuild/

[6] B.J. McBride, M.J. Zehe and S. Gordon, NASA Glenn Coefficients for Calculating Thermodynamic Properties of Individual Species, September 2002, available from https://ntrs.nasa.gov/api/citations/20020085330/downloads/20020085330.pdf.

CHAPTER

Exploring with Matlab acid–base equilibria in water

7

7.1 The hydrogen ion in solution

Chemical reactions in aqueous solutions, including the chemistry of life processes, very often depend on the concentration of hydrogen ions, H^+, in the solution.

As we shall see later, we will deal with hydrogen ion concentrations which range from over 1 mol/L to less than 10^{-14} mol/L. Consequently, it is convenient to express concentrations on a logarithmic basis. For this purpose, the symbols pH and pOH have been introduced in the chemical literature.

Hydrogen ion concentration is represented by pH, whereas hydroxide ion concentration is represented by pOH, with the definitions $\mathrm{pH} = -\log_{10}\left[H^+\right]$ and $\mathrm{pOH} = -\log_{10}[OH^-]$.

In keeping with this usage, we also use $\mathrm{pK_w} = -\log_{10}(K_w)$, where K_w is the ionic product of water defined by $K_w = \left[H^+\right][OH^-]$.

Water is a very weak electrolyte, ionizing slightly and reversibly by transferring one H^+ ion from one molecule to another, which is expressed by the dissociation

$$H_2O + H_2O \Leftrightarrow H_3O^+ + OH^- \tag{7.1}$$

The H_3O^+ ions are usually indicated with a shorthand notation as H^+, H_3O^+ being considered the hydrated form. Just as we ignore the hydration of all the metal ions (for convenience in writing equations), we also often ignore the hydration of H^+. We must always remember, however, that a bare proton, one H^+ ion, can never exist in solution by itself.

The dissociation reaction (7.1) is always at equilibrium, under an extremely rapid formation and recombination of H^+ and OH^-. Because always at equilibrium, the principles of chemical equilibriums apply, and we can write the expression of the equilibrium constant.

This equilibrium is of such importance that K bears the special subscript w. In pure water at $T_{\text{standard}} = 25.0°\text{C}$, the concentration of H^+ is 1.0×10^{-7} mol/L, that is, $\left[H^+\right] = 1.0 \times 10^{-7}$ mol/L. Because the dissociation provides equal numbers of H^+ and OH^- ions, it follows that in pure water $[OH^-] = 1.0 \times 10^{-7}$ mol/L. By knowing the equilibrium concentrations, we can evaluate K_w at the standard temperature of 25°C as follows

$$K_w = \left(1.0 \times 10^{-7}\right)^2 = 1.0 \times 10^{-14}\left(\text{mol/L}\right)^2. \tag{7.2}$$

By recalling that $\log AB = \log A + \log B$, it follows the important constant:

$$\mathrm{pH} + \mathrm{pOH} = \mathrm{pK_w} = 14. \tag{7.3}$$

The above constant applies to *all water (aqueous) solutions*. Therefore, if acid is added to water, thereby increasing $\left[H^+\right]$, a corresponding decrease of $[OH^-]$ should occur, and vice versa. For instance, HCl is a strong acid that completely dissociates in water. This means that a solution with $[HCl] = 0.10$ mol/L implies $\left[H^+\right] = 0.10$ mol/L, and, since $\left[H^+\right][OH^-] = 1.0 \times 10^{-14}$, it follows that

Fundamental Chemistry with Matlab. DOI: https://doi.org/10.1016/B978-0-323-91341-6.00011-2

$[OH^-] = 1.0 \times 10^{-13}$ mol/L, an abrupt decrease to one millionth of the concentration in pure water. As a further instance, NaOH is a strong base, which also completely dissociates in water. A solution with [NaOH] = 0.10 mol/L in turn implies $[OH^-] = 0.1$ mol/L and $[H^+] = 1.0 \times 10^{-13}$ mol/L.

7.2 Monoprotic acid

7.2.1 Description and results

When a strong electrolyte, such as HCl, is put into solution, fairly all the molecules dissociate into ions, in this case H^+ and Cl^-. But when a weak electrolyte is put into solution, such as the acetic acid CH_3COOH, only a small fraction of the molecules dissociates. The relevant reaction is

$$CH_3COOH \Leftrightarrow CH_3COO^- + H^+. \tag{7.4}$$

Because this reaction is at equilibrium, the following expression applies:

$$K_a = [H^+][CH_3COO^-]/[CH_3COOH]. \tag{7.5}$$

The equilibrium constant of the ionization of a weak electrolyte is usually denoted by K_a and is referred to as the *ionization constant*. Ionization constants are derived by experimental measurements of equilibrium concentrations. For instance, in order to determine K_a of the acetic acid, a solution of known concentration is prepared, and by suitable instruments and methods, H^+ or, which is equivalent, pH, is measured. Today, the widely used instruments are *pH-meters*, as they provide a direct pH reading.

However, in practical calculations with weak and very weak acids, self-dissociation of water must be considered, which leads to a system of two equations and four unknowns, $[H^+], [OH^-], [CH_3COO^-], [CH_3COOH]$. The pair of equations consists of Eq. (7.5) and

$$K_w = [H^+][OH^-] = 1 \times 10^{-14}. \tag{7.6}$$

Due to four unknowns, we need two equations more, which are derived below, in the case of the hydrofluoric acid HF, a simple *monoprotic acid* found for instance in seawater. The Matlab® script in Section 7.2.2 refers to this acid.

Firstly, we choose an initial concentration of HF, say $A_{tot} = 0.08$ mol/L, which as soon as the rapid dissociation has reached equilibrium, equals to the sum of $[F^-]$ and [HF]. Such an equality provides the first equation

$$A_{tot} = [F^-] + [HF]. \tag{7.7}$$

The second and third equations derive from the water self-dissociation in Eq. (7.6) and from the hydrofluoric acid dissociation in Eq. (7.5), but rewritten for HF as follows

$$K_{diss} = [H^+][F^-]/[HF]. \tag{7.8}$$

The fourth equation which completes the system, expresses *electrical neutrality*, which should be always accounted for when dealing with electrolyte solutions:

$$\text{Neutrality} = [H^+] - [OH^-] - [F^-] = 0, \ \text{pH} = -\log_{10}(H^+) \tag{7.9}$$

The set of the four equations from (7.6) to (7.9), together with the four unknowns $[F^-], [HF], [H^+], [OH^-]$, could be directly solved in this simple case. To be generic, we search, in an *iterative way*, for a value of $0 \le \text{pH} \le 14$ which satisfies (7.9).

Different iterative methods may be employed, each one corresponding to a Matlab functions. For instance, we started with `fminbnd`, in order to minimize the magnitude of the *neutrality* $f(\text{pH}) = |\text{neutrality(pH)}|$ with respect to pH within the relevant finite domain. At the end, we preferred to employ `fzero`, since it searches for the zero crossing of *neutrality*, which looks more appropriate since neutrality separates between negative and positive charge concentration.

The result printout follows.

```
Monoprotic acid
1) Looking for neutrality
2) Results
Elements        H^+      HO^-       HF       F^-
log10(c)     -2.291    -11.709    -1.126     -2.291
conc      1.953e+02 1.953e-12 7.488e-02 5.119e-03
log(neutrality=  -17.284, neutrality=  5.204e-18
```

7.2.2 Matlab script

The Matlab script of this case is very similar to that of Sections from 7.3 to 7.6. The main script is organized in two parts.

1. *Heading* contains initialization and user data, symbols, electrical charges, and the bounds of pH. It differs from case to case. It ends by calling the main script `MainAcidBaseEqZero`. The vector `x` is the vector of the neutrality concentrations, which has been declared global since it cannot be the output of the function `neutrality` called by `fzero`.

```
%% Monoprotic acid base equilibrium
%% heading: Initialization
InitChem;
global x
%% Heading: User data
disp('Monoprotic acid')
symbol={'  H^+','  HO^-','  HF','  F^-'};
dims=length(symbol);
c=zeros(dims,2); %concentrations charge
c(1:dims,2)=[1;-1;0; -1]; % H+ OH-HF F^-+
x=c(:,1);  % final concentration vector
Atot=0.08;Ka=[3.5e-4];
dima=length(Ka)+1;
Kw=1e-14; p0=7; % initial pH
pHmin=0;pHmax=14;dpH=0.00001;
pHmin=pHmin+dpH;
%% Main and functions
MainAcidBaseEqZero
```

2. *Main script*. The main script searches for the species concentrations which satisfy neutrality, by varying pH within its bounds with the help of the Matlab function `fzero`. To this end, `fzero` calls the function `neutrality`, which computes (i) the current neutrality which is assigned to the output function `fer` and (ii) the current concentration `x`, which is a global vector. The main script ends with the printout of the results as shown in Section 7.2.1.

```
%% Main acid base equilibria in water
%% Looking for neutrality
disp('1) Looking for neutrality')
par(1:dima-1)=Ka;
par(dima)=Kw;
par(dima+1)=Atot;
% search for zero crossing concentrations
p=p0;plim=[pHmin;pHmax];
fer=@(p) neutrality(p,par,c,dima,dims);
options =optimset('PlotFcns',@optimplotfval);
[p1,fer1,exitflag,output] = fzero(fer,plim,options);
% final concentrations and their negative log10
pNeutr=log10(abs(fer1));
px=zeros(dima+1,1);
px(1)=-p1;
x(1)=10^(px(1));
for i=2:dims;    px(i)=log10(x(i));end
%% Results printout
disp('2) Results');
fprintf('Elements');
for i=1:dims,  fprintf('   %s',symbol{i});end
fprintf('\nlog10(c)');
for i=1:dims,    fprintf('%9.3f',px(i) );end
fprintf('\nconc    ');
for i=1:dims,fprintf('%9.2e',x(i)); end
format='\nlog(neutrality)= %7.3f, neutrality= %10.3e\n';
fprintf(format,pNeutr,abs(fer1));
```

3. *The function* `neutrality`. It works in the monoprotic, biprotic (Sections 7.3 and 7.4), and triprotic (Sections 7.5 and 7.6) cases, by building the generic linear system of equations driven by the total concentration `Atot` and the proton concentration `H`. The parameters are the dissociation constants collected in the vector `Ka`. In the case of a triprotic acid, the most complex case treated in the book with the function `neutrality`, the linear system holds:

$$\begin{bmatrix} A_{tot} \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ K_{a1} & -H & 0 & 0 \\ 0 & K_{a2} & -H & 0 \\ 0 & 0 & K_{a3} & -H \end{bmatrix} \begin{bmatrix} x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} \quad . \tag{7.10}$$

$$x_1 = [\mathrm{H^+}] = 10^{-\mathrm{pH}},\ x_2 = [\mathrm{OH^-}] = K_w/[\mathrm{H^+}],\ x_{k=3,\ldots,N} = [\text{other substances}]$$

The neutrality is computed as

$$\text{neutrality} = \sum_{k=1}^{N} x_k q_k \tag{7.11}$$

where the pair $\{x_k, q_k\}$ denotes the concentration and the signed electric charge of the species indicated by subscript k. The number N of substances corresponds, in the script, to `dims`.

```
%% Function neutrality called by fzero
function fer=neutrality(p,par,c,dima,dims)
    global x
    Ka=par(1:dima-1);
    Kw=par(dima);
    Atot=par(dima+1);
    H=10^(-p);    c(1,1)=H;
    OH=Kw/H;    c(2,1)=OH;
    % linear equation
    b=zeros(dima,1);
    A=zeros(dima,dima);
    % generic construction of vector and matrix
    b(1)= Atot;
    A(1,1:dima)=ones(1,dima);
    for i=2: dima
        A(i,i-1)=Ka(i-1);
        A(i,i)=-c(1,1);
    end
    c(3:dima+2,1)=A\b;
    % neutrality
    s=0;
    for i=1:dims,   s=s+c(i,1)*c(i,2);     end
    fer=s;     x=c(:,1); % concentration vector
end
```

7.3 Biprotic acid

7.3.1 Description and results

In some cases, we find two or even three hydrogen atoms in the same molecule that are able to dissociate in aqueous solutions. An important seawater example is the carbonic acid, H_2CO_3, which originates from CO_2(aq), the dissolved form of CO_2(gas). In the dissolved form, the water dipoles interact with the charges of the CO_2 molecule without forming new chemical bonds. The concentration of H_2CO_3 is, however, much smaller, about 0.3%, than that of CO_2(aq). The sum of the two electrically neutral forms, carbonic acid H_2CO_3 and aqueous carbon dioxide, which are chemically inseparable, is usually denoted by $H_2CO_3^*$ or simply by H_2CO_3.

Carbonatic equilibria will be treated in more detail when dealing, in Chapter 9, with seawater and the equilibria with atmospheric carbon dioxide. Here, we treat another common biprotic organic acid, the oxalic acid, possessing the ionization constants $K_{a1} = 5.4 \times 10^{-2}$ and $K_{a2} = 5.3 \times 10^{-5}$, for the first and second dissociation, respectively.

In the Matlab script below, three equilibria, including water self-dissociation, are simultaneously solved by means of the iterative procedure already described in Section 7.2.1. The dissociation constants K_{a1} and K_{a2} refer to pure water at 25°C, and given that the reaction vessel is deemed as closed, it is not necessary, at this stage, to account for the CO_2 dissolution in water. By progressively varying pH values, the following equilibria are solved:

$$\begin{aligned} HOOC^-COOH &\Leftrightarrow HOOC^-COO^- + H^+ \\ HOOC^-COO^- &\Leftrightarrow -OOC^-COO^- + H^+ \end{aligned} \quad . \tag{7.12}$$

The solution must account for the oxalic acid mass conservation and the electric neutrality, as follows

$$\begin{aligned} A_{tot} &= [HCOO^-COOH] + [HOOC^-COO^-] + [-OOC^-COO^-] \\ \text{Neutrality} &= [H^+] - [HOOC^-COO^-] - 2[-OOC^-COO^-] - [OH^-] = 0 \end{aligned} \quad . \tag{7.13}$$

The results obtained with the Matlab script of Section 7.3.2 are reported below.

```
Biprotic acid
1) Looking for neutrality
2) Results
Elements    H^+   HO^-   H_2C_2O_4   HC_2O_4^-   C_2O_4^2-
log10(c)    -1.807   -12.193    -2.350    -1.810    -4.279
conc      1.56e-02 6.42e-13 4.47e-03 1.55e-02 5.26e-05
log(neutrality)= -17.530, neutrality=  2.954e-18
```

7.3.2 Matlab script

Only the heading script is here reported.

```
%% Biprotic acid
%% Initialize
InitChem;
global x
%% User data
disp('Biprotic acid')
symbol={'H^+','HO^-','H_2C_2O_4','HC_2O_4^-','C_2O_4^2-'};
dims=length(symbol);
c=zeros(dims,2); %concentrations charge
c(1:dims,2)=[1;-1;0; -1;-2]; % H+ OH- H2C2O4 HC2O4- C2O4--
x=c(:,1);  % final concentration vector
Atot=0.02;
Ka=[5.4e-2;5.3e-5];
dima=length(Ka)+1;
Kw=1e-14; p0=7; % initial pH
pHmin=0;pHmax=14;dpH=0.00001;
pHmin=pHmin+dpH;
%% Main and functions
MainAcidBaseEqZero
```

7.4 Titration of a weak biprotic acid with a strong base

7.4.1 Description and graphical results

In an acid−base titration, one of the solutions to be neutralized, say the acid, is placed in a flask or beaker, whereas the other solution, the base to be used in the titration, is added from a buret and is known as the *titrant*. The titrant is added to the acid, first rapidly, and then drop by drop, up to the equivalence point, which is usually identified by noticing the color change of the acid−base indicator. The point in a titration at which the indicator changes color is called the *end point of the indicator*. The end point must match the equivalence point of the neutralization.

Further addition of titrant allows us to obtain a complete graph of pH versus the titrant volume, namely the solution volume in the buret. A graph as in Fig. 7.1 is known as *titration curve*. Titration curves are easily constructed by measuring pH during titration with a pH-meter and by plotting the relevant data with a recorder.

In a typical titration, the solution pH is plotted versus the volume of the titrant solution delivered by a buret. In the following example, one litre of a 0.2 mol/L aqueous solution of oxalic acid, a weak biprotic acid, is titrated with 1 mol/L solution of NaOH, a strong base, each step adding 5 mL of the base up to the 600 mL final volume. The only remarkable difference is that the following neutrality identity must account also for Na^+ ions:

$$\text{Neutrality} = [H^+] + [Na^+] - [HOOC^-COO^-] - 2[-OOC^-COO^-] - [OH^-] = 0. \tag{7.14}$$

The titration curve is shown in Fig. 7.1.

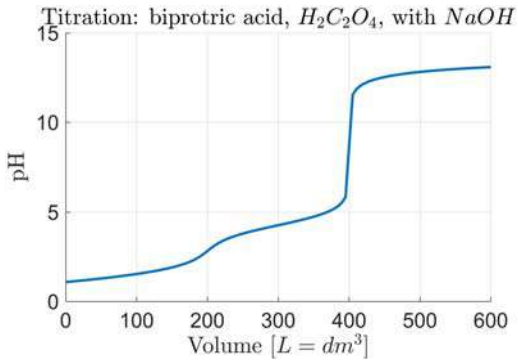


FIGURE 7.1

Titration of a biprotic acid with NaOH.

7.4.2 Matlab script

The Matlab script is an extension of the scripts in Sections 7.2.2 and 7.3.2. The function `neutrality` in Section 7.2.2 remains the same, but the main script `MainAcidBaseEqZero` is changed into `MainAcidTitrationZero`, because of the loop on the titrant volume added to the acid solution. The heading script is shown below. The Boolean variable `logDisp` regulates the printout of numerical results. `vTitr` defines the titrant volume range and step in liters. The scale `milliScale` comes from `InitChem`.

```
%% Biprotic acid titration
%% Initialize
InitChem;logDisp=0;
global x
%% User data
disp('Biprotic acid titration with NaOH')
symbol={'H^+','HO^-','H_2C_2O_4','HC_2O_4^-','C_2O_4^2-','NaOH'};
dims=length(symbol);
c=zeros(dims,2); %concentrations charge
c(1:dims,2)=[1;-1;0; -1;-2;1]; % H+ OH- H2C2O4 HC2O4- C2O4-- NaOH
x=c(:,1);  % final concentration vector
% titrant added NaOH
cTitr=1; % molar conc of NaOH
vTitr=(0:5:600)/milliScale; % Volume added [L]
dimv=length(vTitr);pH=zeros(dimv,1);
Atot=0.2;Ka=[5.4e-2;5.3e-5];
dima=length(Ka)+1; par=zeros(dima+1,1);
st='Titration: biprotric acid, $H_2C_2O_4$, with $NaOH$';
Kw=1e-14; p0=7; % initial pH
pHmin=0;pHmax=14;dpH=0.00001;
pHmin=pHmin+dpH;
%% Main and functions
MainAcidTitrationZero
```

The main script, which performs the loop on the titrant volume, is as follows. It applies also to the triprotic acid in Section 7.6.

```
%% Main: acid titration by base
%% Loop on titrant
disp('1) Loop on titrant ')
for i=1:dimv
   par(1:dima-1)=Ka;
   par(dima)=Kw;
   par(dima+1)=Atot/(1+vTitr(i));
   c(dims,1)=cTitr*vTitr(i)/(1+vTitr(i));
   % search for zero crossing concentrations
   p=p0;plim=[pHmin;pHmax];
   fer=@(p) neutrality(p,par,c,dima,dims);
   % options =optimset('PlotFcns',@optimplotfval);
   % [p1,fer1,exitflag,output] = fzero(fer,plim,options);
   [p1,fer1,exitflag,output] = fzero(fer,plim);
   pH(i)=p1;
end
```

```
% final concentrations and their negative log10
pNeutr=-log10(fer1);
px=zeros(dima+1,1);
px(1)=p1;
x(1)=10^(-px(1));
for i=2:dims; px(i)=-log10(x(i));end
%% Graphical result
disp('2) Graphical results');
figure('name','Titration');hold on; grid on;
plot(vTitr*milliScale,pH,'k');
xlabel('Volume [L]');
ylabel ('pH');title(st);
%% Numerical result printout
if logDisp==1
   disp('3) Numerical esults');
   fprintf('Elements  ');
   for i=1:dims, fprintf('   %s',symbol{i});end
   fprintf('\nlog10(c)');
   for i=1:dims, fprintf(' %9.3f',px(i) ); end
   fprintf('\nconc    ');
   for i=1:dims, fprintf(' %9.3e',x(i)); end
   fprintf('\nlog(neutrality= %7.3f,
   neutrality= %10.3e\n',pNeutr,fer1)
end
```

7.5 Triprotic acid and sodium salt: H_3PO_4 + Na_3PO_4

7.5.1 Description and results

In this section and the following Section 7.6, a triprotic acid is considered, namely the phosphoric acid H_3PO_4 which is widely present in aqueous systems like seawater (see Section 9.6.1). The three dissociation constants at 25°C have the following decreasing values:

$$\begin{aligned} K_{a1} &= [H^+][H_2PO_4^-]/[H_3PO_4] = 7.5 \times 10^{-3} \\ K_{a2} &= [H^+][HPO_4^{2-}]/[H_2PO_4^-] = 6.2 \times 10^{-8} \\ K_{a3} &= [H^+][PO_4^{3-}]/[HPO_4^{2-}] = 4.4 \times 10^{-13} \end{aligned} . \tag{7.15}$$

Since we assume that the neutral sodium phosphate Na_3PO_4 is added to the solution, the neutrality is expressed by the equality:

$$\text{Neutrality} = [H^+] + [Na^+] - [H_2PO_4^-] - 2[HPO_4^{2-}] - 3[PO_4^{3-}] - [OH^-]. \tag{7.16}$$

The starting concentration of the phosphoric acid is assumed to be 0.4 mol/L. 0.4 mol/L of Na_3PO_4 are added, in order to obtain a total concentration of phosphates equal to 0.8 mol/L. These values can be modified by users.

The results of the Matlab script in Section 7.5.2 are reported below.

```
Triprotic acid
1) Looking for neutrality
2) Results
Elements    H^+    OH^-    H_3PO_4    H_2PO_4^-    HPO_4^2-    PO_4^3-    Na^+
log10(c)    -7.208    -6.792    -5.481    -0.398    -0.398    -5.547    0.079
conc      6.20e-08 1.61e-07 3.31e-06 4.00e-01 4.00e-01 2.84e-06 1.20e+00
log(neutrality)=    -Inf, neutrality=  0.000e+00
```

7.5.2 Matlab script

Only the heading part of the main script is reported.

```
%% Triprotric acid
%% Initialize
InitChem;
global x
%% User data
disp('Triprotic acid')
symbol={'H^+','OH^-','H_3PO_4','H_2PO_4^-','HPO_4^2-',...
    'PO_4^3-','Na^+'};
dims=length(symbol);
c=zeros(dims,2); %concentrations charge
c(1:dims,2)=[1;-1;0; -1;-2;-3; 1];
% H+ OH- H3PO4 H2PO4- HPO4-- PO4--- Na+
% Sodium
Na=0.4; c(dims,1)=Na*3; % Na+
x=c(:,1);  % final concentration vector
Atot=0.4;Atot=Atot+Na;
Ka=[7.5e-3;6.2e-8;4.4e-13];
dima=length(Ka)+1;
Kw=1e-14; p0=7; % initial pH
pHmin=0;pHmax=14;dpH=0.00001;
pHmin=pHmin+dpH;
%% Main and functions
MainAcidBaseEqZero
```

7.6 Titration of a triprotic acid with addition of NaOH

7.6.1 Description and graphical results

As in Section 7.5, H_3PO_4 is the triprotic acid, whereas 1.00 mol/L of NaOH solution is added as a titrant. The resulting titration curve in Fig. 7.2 has two marked jumps. Between them, the curve reaches a less steep trend, where the solution pH remains fairly insensible (actually it slowly increases) to the addition of the strong base NaOH. In chemistry, this behavior is known as a *buffer system* [1].

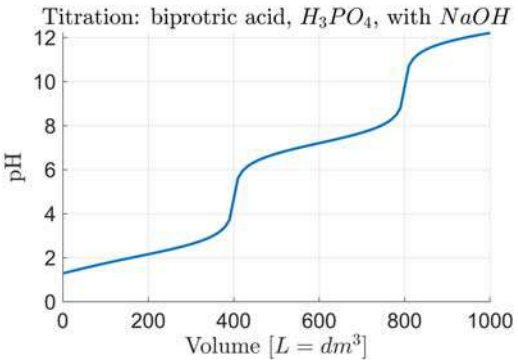


FIGURE 7.2

Titration of a triprotic acid with addition of NAOH.

7.6.2 Matlab script

Only the heading script is reported, since main script and function are the same as for the biprotic acid titration in Section 7.4.

```
%% Triprotric acid titration
%% Initialize
InitChem;logDisp=1;
global x
%% User data
disp('Triprotic acid')
symbol={'H^+','OH^-','H_3PO_4','H_2PO_4^-','HPO_4^2-',...
    'PO_4^3-','NaOH'};
dims=length(symbol);
c=zeros(dims,2); %concentrations charge
c(1:dims,2)=[1;-1;0; -1;-2;-3; 1];
% H+ OH- H3PO4 H2PO4- HPO4-- PO4--- Na+++
x=c(:,1);  % final concentration vector
% added NaOH
cTitr=1; % molar conc of NaOH
vTitr=(0:10:1000)/milliScale; % Volume added [L]
dimv=length(vTitr);pH=zeros(dimv,1);
% Dissociation constants and pH range
Atot=0.4;Ka=[7.5e-3;6.2e-8;4.4e-13];
dima=length(Ka)+1; par=zeros(dima+1,1);
st='Titration: biprotric acid, $H_3PO_4$, with $NaOH$';
Kw=1e-14; p0=7; % initial pH
pHmin=0;pHmax=14;dpH=0.00001;
pHmin=pHmin+dpH;
%% Main and functions
MainAcidTitrationZero
```

7.7 Carbonatic acid–base equilibria involving the precipitation of $CaCO_3$ and $Mg(OH)_2$

7.7.1 Description and numerical results

The following is an example of multiple acid–base equilibria coupled with the possible precipitation of two sparingly soluble compounds, calcium carbonate, a salt, and magnesium hydroxide. Their solubility products, denoted with $K_{spx}, x = 1, 2$, are expressed by the relationships:

$$\begin{aligned} K_{sp1} &= \left[Ca^{2+}\right]\left[CO_3^{2-}\right] = 3.8 \times 10^{-9} \\ K_{sp2} &= \left[Mg^{2+}\right][OH^-]^2 = 8.9 \times 10^{-12}\,. \end{aligned} \tag{7.17}$$

If the concentration products in Eq. (7.17) are higher than $K_{spx}, x = 1, 2$, a portion of insoluble salt/hydroxide will be formed, leaving behind the ions in solution so as to satisfy Eq. (7.17). In parallel, carbonatic equilibria occur, due to the presence of sodium carbonate, Na_2CO_3, sodium bicarbonate, $NaHCO_3$, solid calcium carbonate and magnesium hydroxide, and some amount of carbonic acid, H_2CO_3, chloridric acid, and the sodium hydroxide added to the solution. All the relevant concentrations may be varied by users.

The script implements the acid–base iterative procedure described in Sections 7.2 to 7.6 together with a step-by-step check of the solubility product, in order to simulate the salt/hydroxide precipitation when the solubility products overcome the upper bounds in Eq. (7.17). The reaction vessel is deemed as closed, so as CO_2 cannot be desorbed to or absorbed from the air.

Numerical results are reported below.

```
Carbonatic acid base
1) Looking for neutrality
2) Results
Elements      H^+    HO^-    H_2CO_3    HCO_3^-    CO_3^2-    Ca^2+    CaCO_3
log10(c)     -8.22    -5.78     -7.66     -5.79     -7.90     -0.52     -0.52
conc      5.99e-09 1.67e-06 2.18e-08 1.62e-06 1.27e-08 3.00e-01 3.00e-01
Elements      Ca^2+    CaCO_3    Mg^2+    Mg(OH)_2    Na^+    Cl^-
log10(c)     -0.52     -0.52     -0.70      -Inf     -0.22     0.20
conc      3.00e-01 3.00e-01 2.00e-01 0.00e+00 6.00e-01 1.60e+00
log(neutrality)=    -Inf, neutrality=  0.000e+00
```

7.7.2 Matlab script

The Matlab script is similar to that of Section 7.2.2, but due to the addition of soluble compounds which can precipitate, a specific function `CarboNeutr` is provided, in charge of computing neutrality. In turn, the function calls the function `cLinEq` for building and solving the linear equation system in Eq. (7.10), which in this case reduces to be second order. The main script below consists of the heading which includes the user data part and of the main body which includes the search for neutrality and the result printout.

```
%% Carbonatic acid base equilibrium
%% Initialize
InitChem; global x
%% User data
disp('Carbonatic acid base ')
symbol={'H^+','HO^-','H_2CO_3','HCO_3^-','CO_3^2-',...
    'Ca^2+','CaCO_3','Mg^2+','Mg(OH)_2','Na^+','Cl^-'};
dims=length(symbol);
c=zeros(dims,2); %concentrations charge
c(1:dims,2)=[1;-1;0; -1;-2;2; 0; 2;0; 1;-1]; % H+ OH- ...
x=c(:,1);  % final concentrations
Ka=[4.45e-7;4.69e-11];
dima=length(Ka)+1;
Na2CO3=0.2; NaHCO3=0.1; NaOH=0.1; % Na+
c(10,1)=Na2CO3*2+NaHCO3+ NaOH;
MgCl2=0.2; Mgtot=MgCl2; % Mg++
HCl=0.4;CaCl2=0.6; % Cl-
c(11,1)=HCl+CaCl2*2;
CaCO3=0;Catot=CaCO3+CaCl2;  % Ca++
H2CO3=0; Ctot=Na2CO3+NaHCO3+CaCO3+H2CO3;  % CO3--
% Solubility Ca++CO3--, Mg++(OH-)^2
Ksp=[3.8e-9; 8.9e-12];
Kw=1e-14;
pHmin=0;pHmax=14;dpH=0.00001;pHmin=pHmin+dpH;
pmin=pHmin;pmax=pHmax;p0=7; %initial pH
%% Looking for neutrality
disp('1) Looking for neutrality')
par(1)=Ka(1);par(2)=Ka(2);
par(3)=Kw;
par(4)=Ksp(1);par(5)=Ksp(2);
par(6)=Ctot;par(7)=Catot;par(8)=Mgtot;
plim=[pHmin;pHmax];
fer=@(p) CarboNeutr(p,par,c,dims,dima);
options =optimset('PlotFcns',@optimplotfval);
[p1,fer1,exitflag,output] = fzero(fer,plim,options);
pNeutr=log10(abs(fer1));
x(10)=c(10,1); x(11)=c(11,1);
px=zeros(dims,1);
px(1)=-p1;x(1)=10^(px(1));
for i=2:dims, px(i)=log10(x(i));end
```

```
%% Numerical results
disp('2) Results');
End=[7; dims];Init=[1;6];
for j=1:2
    fprintf('Elements  ');
    for i=Init(j):End(j), fprintf('   %s',symbol{i});end
    fprintf('\nlog10(c)');
    for i=Init(j):End(j), fprintf('%9.2f',px(i) );end
    fprintf('\nconc    ');
    for i=Init(j):End(j), fprintf('%9.2e',x(i));end
    fprintf('\n');
end
format='log(neutrality)= %7.3f, neutrality= %10.3e\n';
fprintf(format,pNeutr,abs(fer1));
```

The functions `CarboNeutr` and `cLinEq` are as follows.

```
%% Function CarboNeutr
function fer=CarboNeutr(p,par,c,dims,dima)
    global x
    Kw=par(3); Ka(1)=par(1);   Ka(2)=par(2);
    Ksp(1)=par(4);   Ksp(2)=par(5);
    Ctot=par(6);   Catot=par(7);   Mgtot=par(8);
    H=10^(-p); c(1,1)=H;
    OH=Kw/H;  c(2,1)=OH;
    % Solving for Atot
    c0=-Ksp(1)*(1+H/Ka(2)+H*H/(Ka(1)*Ka(2)));
    c1=Catot-Ctot;
    y=0.5*c1*(-1+sqrt(1-c0*4/c1^2));
    Atot=min(y,Ctot);
    % Linear equation
    parc=zeros(dima+1,1);
    parc(1:dima)=par(1:dima);
    parc(dima+1)=Atot;
    z=cLinEq(p,parc,dima);
    c(1:dima+2,1)=z;  % 'H_2CO_3','HCO_3^-','CO_3^2-',
    % End linear equation
    % Ca++ (6) CaCO3 (7)  Mg Mg(OH)2
    c(7,1)=Ctot-Atot;
    c(6,1)=Catot-c(7,1);
    c(8,1)=min(Ksp(2)/(OH)^2,Mgtot);
    c(9,1)=max(c(8,1)-Mgtot,0);
    % neutrality
    s=0;
    for i=1:dims,   s=s+c(i,1)*c(i,2);    end
    fer=s;  x=c(:,1); % concentration vector
end
```

```
%% Function cLinWq: after Atot has been found
function c=cLinEq(p,par,dima)
    c=zeros(dima+2,1);
    Ka=par(1:dima-1);
    Kw=par(dima);
    Atot=par(dima+1);
    H=10^(-p);   c(1)=H;
    OH=Kw/H;     c(2)=OH;
    b=zeros(dima,1);
    A=zeros(dima,dima);
    b(1)= Atot;
    A(1,1:dima)=ones(1,dima);
    for i=2: dima
        A(i,i-1)=Ka(i-1);
        A(i,i)=-c(1,1);
    end
    c(3:dima+2)=A\b;
end
```

7.8 Pure water electric conductivity at different temperatures

7.8.1 Description and graphical results

An electrically conducting solution is obtained when an electrolyte is dissolved either in water or other polar solvents. The solution is electrically neutral, as explained in the previous sections, although the electrolyte separates, in water, into cations (positively charged) and anions (negatively charged), which uniformly disperse. If an electric potential is applied to the solution, cations are drawn to the negative electrode and anions to the positive one. In such a way, an electric current propagates in the solution, whose intensity, at least for diluted solutions, is proportional to ion concentration at a given temperature and to the potential difference ΔV applied between electrodes.

The *specific conductance* (today *conductivity*) is defined as the conductance of the unit volume of solution divided by the unit surface of electrodes. It is expressed in $[(\mathrm{S/m^3})\,\mathrm{m^2}] = [\mathrm{S/m}]$ and often in [S/cm], where [S] is the symbol of the Siemens unit. It is convenient to refer the specific conductance to the concentration of the electrolyte. The relevant ratio, known as the *molar conductivity*, is denoted by Λ_m with unit $[\mathrm{S\ cm^2/mol}] = [(\mathrm{S/cm})/(\mathrm{mol/cm^3})]$. The molar conductivity can be retained constant at sufficiently low concentrations, and as soon as the concentration increases, conductivity no longer rises in proportion. The *limiting conductivity* Λ^0 in [S/cm] is defined at the so called *infinite dilution*. F. Kohlrausch (1840–1910) found that the limiting conductivity Λ^0 of an electrolyte is equal to the sum of the limiting molar conductivities $\lambda_i^0, i = 1, \ldots, n$ of the individual ions multiplied by their concentrations $c_i\left[\mathrm{mol/L}\right]$:

$$\Lambda^0 = 0.001 \sum_{i=1}^{n} c_i \lambda_i^0 \ [\mathrm{S/cm}]. \tag{7.18}$$

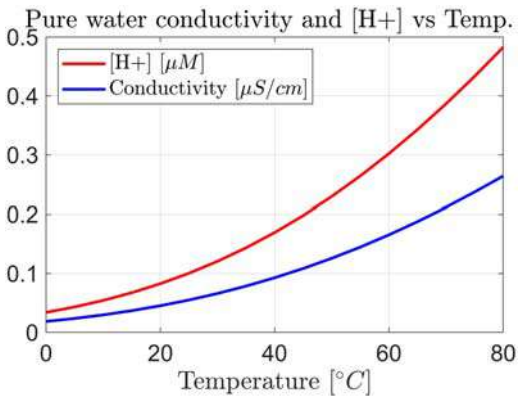


FIGURE 7.3

Pure water conductivity and $[H]^+$ versus temperature.

For pure water in the temperature range 0–80°C, since the concentrations $[H^+]$ and $[OH^-]$ lie in the range $0.4 \sim 5 \times 10^{-7}$ mol/L (see Fig. 7.3), the limiting law of molar conductivities is deemed valid, and by denoting ionic concentrations with $c_H = c_{OH}$, we can write

$$\Lambda^0 = 0.001\left(c_H\lambda_H^0 + c_{OH}\lambda_{OH}^0\right) = c_H 0.3497 + c_{OH}0.1976 \quad [S/cm]. \tag{7.19}$$

In order to establish the concentrations of the two conducting ions in pure water, namely $[H^+]$ and $[OH^-]$, we have to solve the self-dissociation equilibrium of water, expressed by the reaction

$$H_2O = H^+ + OH^-, K_w = [H^+][OH^-], \tag{7.20}$$

in an appropriate temperature range, say from 0°C to 80°C.

We use equation (5.19) of Chapter 5 under the assumption of a constant molar heat capacity C_P^0 in such a temperature interval, implying $C_P^0 = a_3 R$. In fact, the assumption implies that all the coefficients in equations (5.20) and (5.21) are equal to zero except a_3. As a consequence, we can write:

$$H_T^0 = H^0 + (T - T_{lab})C_P^0, \quad S_T^0 = S^0 + \log\left(T/T_{lab}\right)C_P^0, \quad T_{lab} = (273.15 + 25)\text{K}. \tag{7.21}$$

From the Gibbs equation

$$\Delta G = \Delta H - T\Delta S = H_{T,H}^0 - H_{T,OH}^0 - T\left(S_{T,H}^0 - S_{T,OH}^0\right), \tag{7.22}$$

and $K_{eq} = \exp\left(-\Delta G/(RT)\right)$ with $K_{eq} = K_w$, the equal concentrations $[H^+]$ and $[OH^-]$ can be computed from

$$c_H = [H^+] = \sqrt{K_w}, \; c_{OH} = [OH^-] = \sqrt{K_w}. \tag{7.23}$$

7.8.2 Matlab script

The Matlab script of this section employs the parameters in Table 7.1 [1].

Table 7.1 Coefficients employed by the Matlab script of this section.

Element	H^0 [J/mol]	S^0 [J/(mol K)]	C_P^0 [J/(mol K)]	λ_i^0 [S cm^2/mol^{-1}]
H^+	0	0	0	349.7
OH^-	−229990	−10.75	−148.75	197.6
H_2O	−285830	69.91	75.29	0

```
%% Pure water conductivity and [H+] (Chapter 7)
%% Initialization
InitChem;
R = 8.314; T0=273.15; Tc=25; TLab=T0+Tc;
%% Parameters
lambdaH=349.7; lambdaOH= 197.6;  % S cm^2 /mol
H_H=-229990; H_OH=-285830; % enthalpy J/mol
CP_H =-148.75; CP_OH=75.29; % specific heat J/(mol*K)
S_H=-10.75; S_OH =69.91; % entropy  J/(mol *K)
%% Loop on temperature
fprintf('    Temp.[°C]      Kw        [H+]        [OH-]');
fprintf('         pH        Cond. \n');
Temp=0:5:80;
dimTemp=length(Temp);
T=zeros(dimTemp,1); cond=zeros(dimTemp,1); H=zeros(dimTemp,1);
OH=zeros(dimTemp,1);
for j=1:dimTemp
   T (j)= T0 + Temp(j);
   DT=T(j)-TLab;
   DeltaH = H_H + DT*CP_H - (H_OH + DT*CP_OH);
   DeltaS = (S_H + log(T(j)/TLab)*CP_H) - (S_OH + log(T(j)/TLab)*CP_OH);
   DeltaG = DeltaH - T(j)*DeltaS;
   Kw = exp(-DeltaG/(R*T(j)));
   H(j) = sqrt(Kw);OH(j) = H(j);
   pH = -log10(H(j));
   cond(j) = (lambdaH*H(j) + lambdaOH*OH(j))/milliScale;
   fprintf(' %10.3f %10.3e %10.3e  %10.3e %8.3f  %10.3e\n',...
   Temp(j),Kw,H(j),OH(j),pH,cond(j));
end
%% Graphical plot
plot(Temp,H*microScale,'r','LineWidth',2);grid on;hold on;
plot(Temp,cond*microScale,'b','LineWidth',2);
xlabel('Temperature ${^\circ} C]$')
title('Pure water conductivity and [H+] vs Temp. ')
legend('[H+] $[\mu mol/L]$','Conductivity $[\mu S/cm]$',
'Location','northwest')
```

7.9 pH and water ionic product from 0°C to 80°C

In this section, the water ionic product is calculated in the range from 0°C to 80°C using the procedure of Section 7.8, together with the concentrations $[H^+] = [OH^-] = \sqrt{K_w}$.

Matlab script and the result table follow. The resulting data appearing in the `command window` are tabulated versus temperature in the range from 0°C to 80°C with a step of 5°C.

```
InitChem;
R = 8.314;
tC = (0:5:80);
T = 273.15 + tC;
deltaH = (-229990 - (T-298).*148.5) - (-285830 + (T-298).*75.29);
deltaS = (-10.75 - (log(T)-log(298)).*148.5) - (69.91 + (log(T)-
log(298)).*75.29);
deltaG = deltaH - T.*deltaS;
Kw = exp(-deltaG./(R*T));
H = sqrt(Kw);
pH = -log10(H);
% transpose the 4 vectors, in order to be nicely printed
tC = tC.';Kw = Kw.';H = H.';pH = pH.';
table (tC,Kw,H,pH)
```

tC	Kw	H	pH
0	1.1542e-15	3.3973e-08	7.4689
5	1.8683e-15	4.3223e-08	7.3643
10	2.9478e-15	5.4294e-08	7.2652
15	4.5406e-15	6.7384e-08	7.1714
20	6.837e-15	8.2686e-08	7.0826
25	1.0077e-14	1.0038e-07	6.9983
30	1.4553e-14	1.2064e-07	6.9185
35	2.0621e-14	1.436e-07	6.8428
40	2.8694e-14	1.6939e-07	6.7711
45	3.925e-14	1.9812e-07	6.7031
50	5.2823e-14	2.2983e-07	6.6386
55	7.0005e-14	2.6458e-07	6.5774
60	9.1429e-14	3.0237e-07	6.5195
65	1.1777e-13	3.4317e-07	6.4645
70	1.497e-13	3.8691e-07	6.4124
75	1.8793e-13	4.3351e-07	6.363
80	2.3313e-13	4.8284e-07	6.3162

Reference

[1] R.H. Petrucci, F.G. Herring, J.D. Madura, C. Bissonnette, General Chemistry Principles and Modern Applications, 10th (ed.), Pearson, Canada, 2011.

CHAPTER

Colligative properties of solutions aided by Matlab

8

8.1 Aqueous solutions of NaCl

8.1.1 Description

In chemistry, *colligative properties* are those properties of solutions which only depend on the number of solute particles, measured as either *molarity*, in [mol/L − soln], briefly [mol/L], or *molality*, in [mol/kg − solvent], briefly [mol/kg], and on the temperature, but not on the chemical constitution of the solute. The assumption that solution properties are independent of the nature of the solute particles only applies to ideal solutions and becomes approximate for dilute real solutions.

The term was introduced in 1891 by W. Ostwald (1853−1932), who classified solution properties in three categories.

1. *Colligative* properties only depend on the solute molar concentration and temperature, being independent of the nature of the solute particles.
2. *Additive* properties, in addition, depend on the molecular mass of the solute particles and therefore on the molecular formula of the solute.
3. *Constitutional* properties such as density, further depend on the molecular structure of the given solute and on its interaction with solvent [1].

The first category essentially collects those properties of the solvent which are changed by the presence of the solute. The solute particles displace some solvent molecules in the liquid phase and thereby reduce the concentration of solvent, so that the colligative properties are independent of the nature of the solute. Colligative properties include:

1. relative lowering of vapor pressure (*Raoult's law*, F.-M. Raoult, 1830−1901),
2. elevation of boiling point,
3. depression of freezing point, and
4. osmotic pressure.

These properties are influenced by the dissociation into ions of the solute molecules, like solutions of salts, because dissociation affects the total number of particles in solution. For example, the colligative properties of a diluted solution of sodium chloride (NaCl) are exactly twice compared to those of a solution of a nonionized solute such as urea or glucose in water.

Fundamental Chemistry with Matlab. DOI: https://doi.org/10.1016/B978-0-323-91341-6.00001-X

One unit of NaCl dissociates into two ions, Na^+ and Cl^-, which fact is accounted for by the van't Hoff coefficient $i_{vH} = 2$ (J.H. van't Hoff, 1852–1911). The relevant equations, valid for ideal diluted solutions, are as follows:

$$1.\ \text{vapor pressure lowering} = \left(1 + \frac{H_2O\, i_{vH}\, cw}{NaCl(100 - cw)}\right)^{-1} \qquad (8.1)$$

$$H_2O = 18.0\ \text{g/mol},\ NaCl = 58.4\ \text{g/mol}$$

$$2.\ \text{elevation of boiling point} = 0.513\ i_{vH} \times \text{molality}\ [\text{K}] \qquad (8.2)$$

$$3.\ \text{depression of freezing point} = 1.860\ i_{vH} \times \text{molality}\ [\text{K}] \qquad (8.3)$$

$$4.\ \text{osmotic pressure} = 0.0821\ Ti_{vH} \times \text{molarity}\ [\text{atm}] \qquad (8.4)$$

The constant in Eq. (8.2) is the solvent (water) ebullioscopic constant K_b defined by

$$K_b = RT_b^2 \times H_2O \times \Delta H_{vap}^{-1} = 0.513\ \text{K(kg/mol)}$$
$$T_b = 373.15\ \text{K},\ H_2O = 0.018\ \text{kg/mol},\quad \Delta H_{vap} = 40660\ \text{J/mol} \qquad (8.5)$$

where T_b is the water boiling temperature, and ΔH_{vap} is the molar enthalpy of vaporization. The constant in Eq. (8.3) is the solvent (water) cryoscopic constant K_f defined by

$$K_f = RT_f^2 \times \Delta H_{fus}^{-1} = 1.860\ \text{K(kg/mol)}$$
$$T_f = 273.15\ \text{K},\ \Delta H_{fus} = 333550\ \text{J/kg} \qquad (8.6)$$

where T_f is the water fusion temperature, and ΔH_{fus} is the water enthalpy of fusion per unit of mass. The constant in Eq. (8.4) is the universal gas constant $R = 0.0821$ atm L $K^{-1}mol^{-1}$ in appropriate units.

As already mentioned, *molarity* is defined as the number of moles in one liter of solution in units $[\text{mol/L}]$, *molality* is the number of moles dissolved in 1kg of solvent, in units [mol/kg − solvent], and *cw* is the weight percentage of NaCl in the solution.

The previous colligative properties will be tabulated at different solute concentrations in the Matlab® script of Section 8.1.2 for the case of the NaCl aqueous solution. A constitutional property, the *density*, will be studied in more details in the script of Section 8.2.5, for the same NaCl solution, at different concentration and temperature.

Colligative properties are mostly studied for dilute solutions, since their behavior can be approximated by that of an ideal solution. In fact, all the properties listed above are colligative only in the dilute limit: at higher concentrations, they became dependent on the chemical nature of solvent and solute.

8.1.2 Matlab script

```
%% Let us examine colligative properties of NaCl solutions in water,
% with increasing concentration
% Pressure in MPa April 2021
%% Initialization
InitChem;
disp('NaCl aqueous solution: colligative properties');
R = 8.314; % J/(mol x K)
NaCl = 22.9898 + 35.453; %atomic weight of sodium chloride [g/mol]
Tc = 4;T = Tc + 273.15; % tC temperature in °C, T in K
i_vH = 2; % van't Hoff coefficient
% expansion coefficients of density
D=[919.0202567;8.661163416;0.854264859;0.027175948;-0.00199299;
-0.00585389];
cw=1:1:20;dimcw=length(cw); %NaCl weight percentage in solution
%% Loop on cw, weight fraction of NaCl in solution
density = D(1) + D(2)*cw + D(3)*T + D(4)*cw.^2 + D(5)*T^2 +
D(6)*T*cw; % g/dm^3
molality =(milliScale/NaCl).* cw./(100-cw); % molality=mol/(kg-solvent)
molarity =(density/NaCl).*(cw/100); % molarity = mol/(L-solution)
freeze = 1.86*i_vH*molality;
ebull = 0.513*i_vH*molality;
osmotic = i_vH*molarity*R*T/milliScale; % MPa
vap_press = 1./(1+(18*i_vH/NaCl).*cw./(100-cw));
%% numerical results
fprintf('NACl Density Molarity Molality Freezing Boiling OsmoticPr
VaporPr \n');
fprintf(' [%%] [g/l]   [mol/l] [mol/kg]   [K]      [K]     [MPa]\n');
for i=1:dimcw
   fprintf( '%3.0f %6.0f %8.3f %8.3f %8.3f %8.3f %8.3f %8.3f \n', ...
   cw(i),density(i),molarity(i), molality(i), freeze(i),...
   ebull(i),osmotic(i),vap_press(i));
end
```

8.1.3 Output in command window

The printout on the `command window` is reported in the following table.

NACl [%]	Density [g/l]	Molarity [mol/l]	Molality [mol/kg]	Freezing [K]	Boiling [K]	OsmoticPr [atm]	VaporPr
1	1010	0.173	0.173	0.643	0.177	7.9	0.994
2	1017	0.348	0.349	1.299	0.358	15.8	0.988
3	1024	0.526	0.529	1.969	0.543	23.9	0.981
4	1031	0.706	0.713	2.652	0.731	32.1	0.975
5	1039	0.889	0.901	3.350	0.924	40.4	0.969
6	1046	1.074	1.092	4.063	1.121	48.9	0.962
7	1053	1.262	1.288	4.791	1.321	57.4	0.956
8	1061	1.452	1.488	5.535	1.527	66.1	0.949
9	1068	1.645	1.692	6.295	1.736	74.9	0.943
10	1076	1.841	1.901	7.072	1.951	83.8	0.936
11	1083	2.039	2.115	7.867	2.170	92.8	0.929
12	1091	2.240	2.333	8.680	2.394	102.0	0.923
13	1099	2.444	2.557	9.511	2.623	111.2	0.916
14	1107	2.651	2.785	10.362	2.858	120.6	0.909
15	1114	2.860	3.020	11.233	3.098	130.2	0.902
16	1122	3.072	3.259	12.124	3.344	139.8	0.895
17	1130	3.288	3.505	13.037	3.596	149.6	0.888
18	1138	3.506	3.756	13.972	3.854	159.5	0.881
19	1146	3.726	4.014	14.931	4.118	169.6	0.874
20	1154	3.950	4.278	15.913	4.389	179.8	0.867

8.2 Density of aqueous solutions: linear regression

In this section, a 2D polynomial model of the NaCl aqueous solution *density*, a constitutional property of the solution, will be derived from experimental data in terms of concentration and temperature [2]. Physics and chemistry only suggest the kind of variables (the *predictor variables*), that is temperature and concentration, that must enter the model in order to fit experimental data. Since a polynomial model is linear in the unknown coefficients to be estimated, the fitting procedure is known as *linear regression*. No information on the polynomial degree can be derived from physical and chemical laws. Therefore we are obliged to test different degrees by means of the statistical tools of the *linear regression*, including *analysis of variance* and *statistical tests of significance*, like *Student's t-test* and *F-test*. A brief introduction to linear regression, relevant tools, variables, notations, and terms is available in Appendix C.

8.2.1 The regression of density data in terms of temperature and concentration

The density $y = \rho \left[\mathrm{g/m^3}\right]$ of aqueous solutions must be expressed as a function of the temperature samples $u_1(i) = T_i$ [K] (the first row in Fig. 8.1) and concentration percentage $u_2(j) = c_j$ [100 g/g] (the first column in Fig. 8.1), which are indexed by i and j, respectively. The relevant raw data are reported in the table of Fig. 8.1. The $k-$th measurement of $y = \rho$ will be denoted according to Appendix C by $\check{y}(k)$. We are looking for a polynomial function $\check{y}(T_i, c_j)$.

NaN	273.15	283.15	293.15	298.15	303.15	313.15	323.15	333.15	353.15	373.15
1	1007.47	1007.07	1005.34	1004.09	1002.61	999.08	994.82	990.00	978.50	965.10
2	1015.09	1014.42	1012.46	1011.12	1009.57	1005.93	1001.61	996.70	985.20	971.90
4	1030.38	1029.20	1026.80	1025.30	1023.61	1019.77	1015.31	1010.30	998.80	985.50
6	1045.75	1044.08	1041.27	1039.63	1037.81	1033.78	1029.19	1024.10	1012.50	999.40
8	1061.21	1059.07	1055.89	1054.12	1052.19	1047.98	1043.26	1038.10	1026.40	1013.40
10	1076.77	1074.19	1070.68	1068.79	1066.76	1062.38	1057.53	1052.30	1040.50	1027.60
12	1092.44	1089.46	1085.66	1083.65	1081.53	1076.99	1072.02	1066.70	1054.90	1042.00
14	1108.24	1104.91	1100.85	1098.72	1096.51	1091.82	1086.74	1081.30	1069.40	1056.50
16	1124.19	1120.56	1116.21	1114.01	1111.71	1106.88	1101.70	1096.20	1084.20	1071.30
18	1140.31	1136.43	1131.90	1129.54	1127.15	1122.18	1116.91	1111.30	1099.30	1086.40
20	1156.63	1152.54	1147.79	1145.33	1142.85	1137.74	1132.38	1126.80	1114.60	1101.70
22	1173.18	1168.91	1163.95	1161.40	1158.83	1153.58	1148.12	1142.50	1130.30	1117.20
24	1189.99	1185.57	1180.40	1177.76	1175.11	1169.71	1164.14	1158.40	1146.30	1133.10
26	1207.09	1202.54	1197.17	1194.43	1191.70	1186.14	1180.45	1174.70	1162.60	1149.20

FIGURE 8.1

Raw data of the NaCl solution density.

As already said, the expression of $\check{y}(T_i, c_j)$ must be experimentally obtained and cannot be derived by chemistry laws. A simple model is a 2D polynomial with a sufficient degree to be found from data regression. Here, we assume as a starting point to be checked, a third-degree polynomial as follows:

$$\begin{gathered}\check{y}(T_i, c_j) = a_0 + \alpha_1 \Delta T_i + a_2 \Delta c_j + a_{11} \Delta T_i^2 + a_{22} \Delta c_j^2 + a_{12} \Delta T_i \Delta c_j \\ + a_{112} \Delta T_i^2 \Delta c_j + a_{122} \Delta T_i \Delta c_j^2 + a_{111} \Delta T_i^3 + a_{222} \Delta c_j^3 + \tilde{y}(T_i, c_j) \\ \Delta T_i = T_i - \overline{T}_i, \Delta c_j = c_j - \overline{c}_j\end{gathered} \tag{8.7}$$

The symbol $\tilde{y}(T_i, c_j)$ denotes the cumulative error due to polynomial truncation, the effect of neglected variables and measurement errors. The range of the predictor variables is denoted by $i = 1, \ldots, n_1, j = 1, \ldots, n_2$, and the pair $\{\overline{T}_i, \overline{c}_j\}$ indicates data average. ΔT_i will be referred to as the temperature anomaly. The sequence of variables and coefficients in Eq. (8.7) is the same as in the Matlab script of Section 8.2.5. It is convenient to arrange the density samples into a vector $\check{\boldsymbol{y}}$ with components $\check{y}(k), k = j + n_1(i-1), k = 1, \ldots, N$. The same is done by building the vectors $\{\boldsymbol{f}_0, \ldots, \boldsymbol{f}_\mu, \ldots, \boldsymbol{f}_m\}, m = 9$ of the predictor variables, with the following components and acronyms:

$$\begin{gathered}f_0(k) = 1, \quad \text{constant} \\ f_1(k) = \Delta T_i \text{ (T1)}, \; f_2(k) = \Delta c_j \text{ (C1)} \\ f_3(k) = \Delta T_i^2 \text{ (T2)}, \; f_4(k) = \Delta c_j^2 \text{ (C2)}, \; f_5(k) = \Delta T_i \Delta c_j \text{ (T1C1)} \\ f_6(k) = \Delta T_i^2 \Delta c_j \text{ (T2C1)}, \; f_7(k) = \Delta T_i \Delta c_j^2 \text{ (T1C2)}, \; f_8(k) = \Delta T_i^3 \text{ (T3)}, \; f_9(k) = \Delta c_j^3 \text{ (C3)}\end{gathered} \tag{8.8}$$

The polynomial regression coefficients are collected into the vector (using the inline notation)

$$\boldsymbol{a} = [a_0, a_1 a_2, a_{11}, a_{22}, a_{12}, a_{112}, a_{122}, a_{111}, a_{222}] = \left[a_0, a_1, \ldots, a_\mu, \ldots, a_m\right], m = 9. \tag{8.9}$$

The matrix version of Eq. (8.7) becomes

$$\check{\boldsymbol{y}} = \left[\boldsymbol{f}_0 .. \boldsymbol{f}_\mu .. \boldsymbol{f}_9\right]\boldsymbol{a} + \tilde{\boldsymbol{y}} = F\boldsymbol{a} + \tilde{\boldsymbol{y}},\ \check{y}(k) = \boldsymbol{f}^{\mathrm{T}}(k)\boldsymbol{a} + \tilde{y}(k) \tag{8.10}$$

The unknown vector $\boldsymbol{a}$ of the model (8.10) has been fit to data (the linear regression) with the aid of the Matlab function `fitlm`. The function requires the following table:

$$R = [F, \check{\boldsymbol{y}}], \tag{8.11}$$

which is constructed by the function `table` in the script of Section 8.2.5 (in the last section of the script, headed by the comment 'Analysis of variance'). In the following, the mean $\bar{y}$ of the raw data defined by

$$\bar{y} = \sum_{k=1}^{N} \check{y}_k(T_i, c_j) \tag{8.12}$$

will be employed, The reader should pay attention that in Eq. (8.7) the subscript k was dropped.

8.2.2 Numerical results of the regression

The function `fitlm` provides the tabular results of Fig. 8.2.

1. The first numerical column (Estimate) is the vector $\hat{\boldsymbol{a}} = \left[\hat{a}_0, \hat{a}_1, \ldots, \hat{a}_\mu, \ldots \hat{a}_m\right]$ of the estimated regression coefficients.
2. The second column (SE, standard error) shows the estimated (*a posteriori*) standard deviation $\tilde{p}_\mu$ of the previous estimate (see Appendix C, Eq. C.12).
3. The third column (tStat) is the *t-statistics* t_μ under the assumption that the hypothetical value of the relevant coefficient is zero (*null hypothesis*), hence not contributing to data explanation (see Appendix C, Section C.3.2). The statistics is defined by the following ratio and exemplified with the third row (predictor C1) of Fig. 8.2:

$$t_\mu = \frac{\hat{a}_\mu}{\tilde{p}_\mu} \Rightarrow t_{C1} = \frac{7.40}{0.00603} \simeq 1230. \tag{8.13}$$

   Intuitively, as soon as $t_\mu \to 0$, or practically when $t_\mu < t_{max}$, the null hypothesis becomes TRUE, which occurs either because $\left|\hat{a}_\mu\right| \to 0$ (hence not contributing to the model) or because the standard error $\tilde{p}_\mu$ becomes large, thus removing any significance to the estimate.
4. The fourth column (pValue) shows the probability that the *null hypothesis* occurs according to the *t-Student distribution* (see Appendix C, Section C.3.2). We observe that after the coefficient $a_6 = a_{112}$ with $t_6 = 42.3$ (the red row in Fig. 8.2), the *t*-statistics drops to $t_n < 8$, for $n = 8, 9, 10$. However, the reader should be warned that the range $6 < t_\mu < 8, \mu = 7, 8, 9$ does not suggest that *the null hypothesis is FALSE*, which is confirmed by the last column which reports a *very small probability* that the null hypothesis is TRUE. Notwithstanding, we should take this drop as a suggestion that the last three variables are *poorly significant* in explaining raw data. The same conclusion is confirmed by the analysis of variance to be done below.

```
Linear regression model:
    Dens ~ 1 + T1 + C1 + T2 + C2 + T1C1 + T2C1 + T1C2 + T3 + C3
Estimated Coefficients:
                    Estimate        SE          tStat        pValue
                   __________    __________    __________    __________
    (Intercept)     1.0843e+00    3.1251e-05     3.4698e+04    0.0000e+00
    T1             -4.9026e-04    1.5067e-06    -3.2540e+02    3.9985e-191
    C1              7.4037e-03    6.0277e-06     1.2283e+03    4.3546e-266
    T2             -2.1205e-06    2.5002e-08    -8.4813e+01    1.1177e-115
    C2              2.6870e-05    3.1316e-07     8.5805e+01    2.5324e-116
    T1C1           -7.5259e-06    8.1413e-08    -9.2442e+01    1.8388e-120
    T2C1            1.0091e-07    2.3844e-09     4.2321e+01    6.8400e-78
    T1C2            7.1100e-08    1.0390e-08     6.8428e+00    2.7554e-10
    T3              5.5543e-09    7.0992e-10     7.8239e+00    1.5570e-12
    C3              3.2776e-07    4.7380e-08     6.9176e+00    1.8747e-10
Number of observations: 140, Error degrees of freedom: 130
Root Mean Squared Error: 0.0002
R-squared: 1, Adjusted R-Squared: 1
F-statistic vs. constant model: 1.48e+06, p-value = 3.76e-321
```

FIGURE 8.2

Results of the linear regression.

The last rows of Fig. 8.2 detail the following parameters.

1. The *number of observations* N, namely the dimension of the measurement vector $\check{\boldsymbol{y}}$.
2. The *residual degrees of freedom* (DF) $N-m-1$ equal to the data size N less the total number of variables $m-1$, with the exclusion of the data mean value (in other terms of the unknown a_0). To better understand $N-m-1$, we need to recall that also the samples $\tilde{y}(T_i, c_j)$ of the cumulative error in Eq. (8.7) (known as the *regression residuals*) are estimated by the regression algorithm, and of course, the cardinality of the estimated components cannot be larger than raw data dimension N.
3. The *root mean squared (RMS) residual* holds

$$\hat{\sigma}_{res} = (N-m-1)^{1/2} \quad \sqrt{\sum_{k=1}^{N} \left(\check{y}(k) - \boldsymbol{f}^T(k)\hat{\boldsymbol{a}}\right)^2} = \\ = (N-m-1)^{1/2}\sqrt{\hat{s}^2_{res}} = \sqrt{\frac{5.22\times10^{-6}}{140-10}} = 0.2 \text{ mg/m}^3 \tag{8.14}$$

and has been checked via the residual sum of square (RSS) $\hat{s}^2_{res}$ in the last row of the table in Fig. 8.4.

4. The *R-squared* R^2 is just the ratio between the explained variance $\hat{s}^2$ (equal to the data variance s_y^2 minus the error sum of square $\hat{s}_{res}^2$) and the data variance (equal to the square of the data deviations from the mean value $\overline{y}$). The data variance (the total sum of squares) can be found in the first row of Fig. 8.3. R^2 is computed and checked as follows:

$$R^2 = \frac{\hat{s}^2}{s_y^2} = \frac{s_y^2 - \hat{s}_{res}^2}{s_y^2} = 1 - \frac{\hat{s}_{res}^2}{\sum_{k=1}^{N}(\check{y}(k) - \overline{y})^2} = 1 - \frac{5.22 \times 10^{-6}}{0.536} \simeq 1.00$$
$$\overline{y} = \frac{1}{N}\sum_{k=1}^{N} \check{y}(k) \qquad (8.15)$$

When close to 1 as in this case, it means that the third-order model in Eq. (8.7) fully explains raw data.

5. The *adjusted R-squared* accounts for the variance DF, namely $N - 1 = 139$ for the data variance (which is free of the data mean value $\overline{y}$) and $N - m - 1 = 130$ for the residual variance. It holds

$$\overline{R}^2(\mu) = 1 - \frac{N-1}{N-m-1}\frac{\hat{s}_{res}^2}{\sum_{k=1}^{N}(\check{y}(k) - \overline{y}\quad)^2} \simeq 1.00. \qquad (8.16)$$

6. *F-statistics versus the constant model* is a further statistical test for checking the significance of the model (or a part of the model, see Section 8.2.3) in explaining the variance of the data with respect to the data mean value (the constant model, the relevant null hypothesis is that all the polynomial coefficients except a_0 are equal to zero). A more generic way with respect to the *t-statistics* (testing a single estimate) is to compare the *explained variance* $\hat{s}^2$ to the *residual sum of squares* $\hat{s}_{res}^2$ (or to an *alternative explained variance*). The *null hypothesis* corresponds to the case when the explained variance is ideally zero with respect to the alternative, in other terms, the whole set $A = \{a_1, \ldots, a_m\}$ is equal to zero. Here, the test is implemented by the ratio explained/residuals, account taken of the respective DF (the numerical values come from the table in Fig. 8.3):

$$F_A = \frac{(N-m-1)\hat{s}^2}{m\hat{s}_{res}^2} = \frac{130 \times 0.536}{9 \times 5.22 \times 10^{-6}} \simeq 1.48 \times 10^6. \qquad (8.17)$$

The extremely high value of the statistics imply that the model fully explains experimental data, which is further testified by a negligible probability (pValue) that the null hypothesis is TRUE. In other terms, a third-degree polynomial is sufficient and the effect of neglected predictor variables cannot be separated from residuals.

	SumSq	DF	MeanSq	F	pValue
Total	5.3627e-01	1.3900e+02	3.8580e-03		
Model	5.3626e-01	9.0000e+00	5.9584e-02	1.4836e+06	3.7648e-321
Residual	5.2211e-06	1.3000e+02	4.0162e-08		

FIGURE 8.3

Summary of the linear regression results from Matlab function anova.

8.2.3 Analysis of variance

Using the Matlab function `anova`, we have performed the *analysis of variance* of the regression. The relevant results are shown in the tables of Figs. 8.3 and 8.4.

The rows of Fig. 8.3 show the data (Total), explained (Model), and residual sum of squares (SumSq) $s^2, \hat{s}^2, \hat{s}^2_{res}$ already encountered. They are accompanied by their respective DF $\{N-1, m, N-m-1\}$ and their mean square, equal to the sum of squares normalized by DF. The reported *F-statistics* and pValue refer to the model variance (free of the mean value $\overline{y}$) and have been already explained.

The model sum of squares in the table of Fig. 8.3 is decomposed in the table of Fig. 8.4 among the $m = 9$ predictor variables of the density model in Eqs. (8.7) and (8.8).

1. The first column reports the *sum of squares* $\hat{s}(\mu)^2$ explained by each variable. It is repeated in the third column but divided by the DF in the second column.
2. The fourth column reports the *F-statistics* $F(\mu)$ of each predictor variable, which, as already said, is a more generic test to decide whether a predictor variable can explain or not the model variance. The test is the ratio between the sum of squares explained by the variable of index μ and the residual sum of squares, both normalized by their DF $\{1, N-m-1\}$:

$$F_\mu = \frac{(N-m-1)\hat{s}^2_\mu}{\hat{s}^2_{res}} = \frac{(N-m-1)\boldsymbol{f}^T_\mu \boldsymbol{f}_\mu \hat{a}^2_\mu}{\hat{s}^2_{res}}, \mu = 1, \ldots, m = 9. \tag{8.18}$$

	SumSq	DF	MeanSq	F	pValue
T1	0.0042526	1	0.0042526	1.0588e+05	3.9985e-191
C1	0.060592	1	0.060592	1.5087e+06	4.3546e-266
T2	0.0002889	1	0.0002889	7193.3	1.1177e-115
C2	0.00029569	1	0.00029569	7362.5	2.5324e-116
T1C1	0.00034321	1	0.00034321	8545.5	1.8388e-120
T2C1	7.1933e-05	1	7.1933e-05	1791.1	6.84e-78
T1C2	1.8805e-06	1	1.8805e-06	46.824	2.7554e-10
T3	2.4585e-06	1	2.4585e-06	61.214	1.557e-12
C3	1.9219e-06	1	1.9219e-06	47.854	1.8747e-10
Error	5.2211e-06	130	4.0162e-08		

FIGURE 8.4

Results of the analysis of variance for each predictor variable.

The statistics of Eq. (8.18) as well (fourth column of the table in Fig. 8.4), undergoes a significant drop, from 1791.1 to 46.824, while passing from the sixth variable (in red color) to the seventh one. As a consequence, it seems reasonable to halt the polynomial model (8.7) at the sixth predictor variable $T2C1$, as follows:

$$\check{y}(T_i, c_j) = \alpha_0 + \alpha_1 \Delta T_i + \alpha_2 \Delta c_j + \alpha_{11}\ \Delta T_i^2 + \alpha_{22}\Delta c_j + \alpha_{12}\Delta T_i \Delta c_j + \alpha_{112}\Delta T_i^2 \Delta c_j + \tilde{y}^* \tag{8.19}$$

where the cumulative error $\tilde{y}^*$ includes the neglected polynomial components and therefore is different from the error in Eq. (8.7) (the asterisk denotes the difference).

The above model, which includes all the second-order components plus the third-order $T2C1$, will be referred to as the *accepted model*. This decision will be confirmed by the graphical analysis in Section 8.2.4.

8.2.4 Graphical analysis

Fig. 8.5 shows the density surface (ordinate in [g/m^3]) provided by raw data. Comparison between the explained surfaces of different order models can only be made through regression residuals, because of their small values with respect to the density surface.

Residual comparison is done in Fig. 8.6 (ordinate in [mg/m^3]). The left figure shows the residual surface of a *second-order model*, based on the *first five* predictor variables plus the constant. The right figure shows the residuals of the *accepted model* in Eq. (8.19). We observe that the residual surface of the second-order model looks rather smooth, implying that it may be better explained by a higher-order polynomial, as it has been done by adding an appropriate third-order predictor variable in Eq. (8.19).

To be exhaustive, Fig. 8.7 shows the residual surface of the complete third-order model in Eq. (8.7), which includes all the nine predictor variables plus the constant. The surface looks very similar to the accepted model residuals in Fig. 8.6, right: a further confirmation of the previous decision.

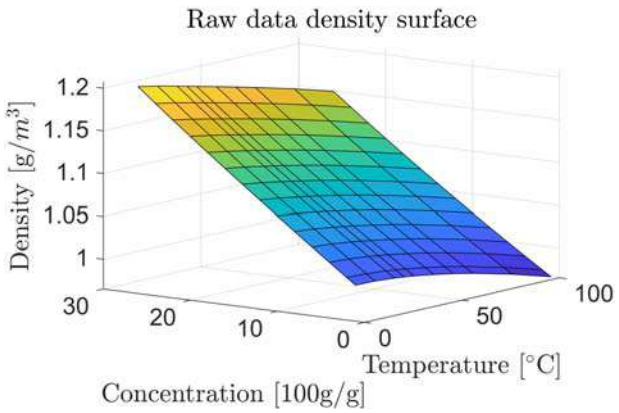


FIGURE 8.5

Raw data density surface vs concentration and temperature anomaly.

Fig. 8.8 shows that both t and F-*statistics* (in logarithmic scale) decrease in the same way by increasing model complexity, from left to right. The residual RMS $\hat{\sigma}_{res}$ decays in a similar way. The jump that was previously discussed occurs between the sixth and seventh parameter (the constant is. The RMS errors $\hat{\sigma}(n)$, $n = 5, 6, 9$ of the models with five, six, and nine variables (plus constant) are as follows:

$$\begin{aligned} \hat{\sigma}(5) &= 0.77 \text{ mg/m}^3 \\ \hat{\sigma}(6) &= 0.29 \text{ mg/m}^3 \\ \hat{\sigma}(9) &= 0.19 \text{ mg/m}^3 \end{aligned} \tag{8.20}$$

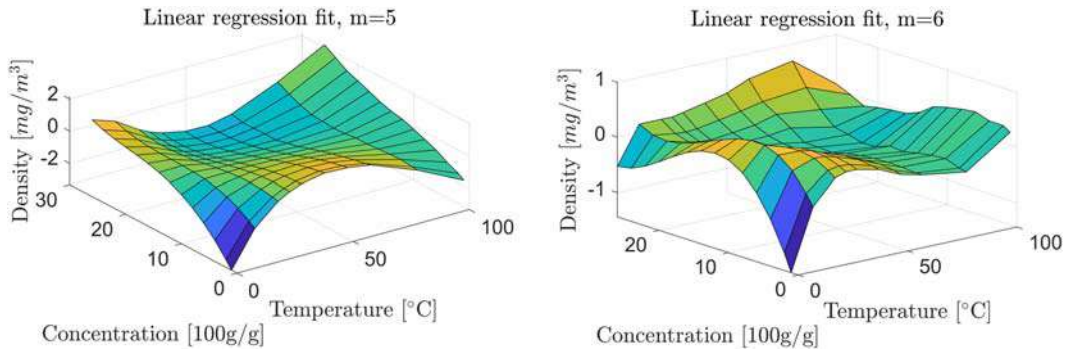


FIGURE 8.6

Residual surface. (Left) The second-order model residuals (five variables plus constant), (Right) The accepted model residuals (six variables plus constant).

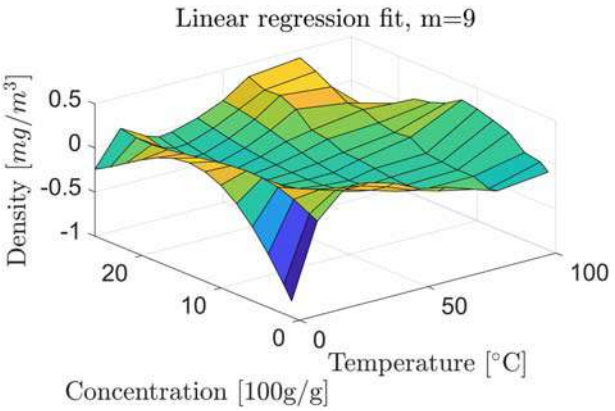


FIGURE 8.7

Residual surface of the complete third-order model.

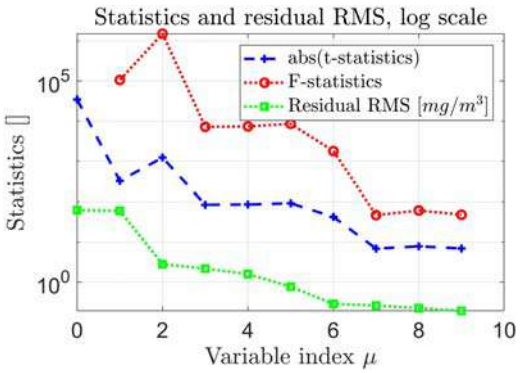


FIGURE 8.8

F- and *t*-statistics together with the residual RMS (log scale) versus the variable index μ.

The previous RMS errors show that residuals significantly drop beyond the five-variable model (seven times in terms of the residual square), and about two times from the six- to nine-variable model. Fig. 8.8 shows that the last reduction is not significant, as the residual profile becomes close to be flat. This is a further confirmation for accepting as significant the six-variable model.

8.2.5 Matlab script

The Matlab script is organized into five parts:

1. *Initialization:* users may select the regression order dimPred between 0 (only the constant model) and 9 (full model); it allows to fit data with different order models in order to plot the relevant residuals.
2. *Raw data reading and organization:* The raw data of the NaCl aqueous solution density are read from the file `NACLSolTable.txt` which is available from the companion website of the book [3] and is reported in Fig. 8.1. Temperature T_i and concentration c_j data are converted to anomalies by subtracting their mean values, in order to obtain a set of quasilinearly independent predictor variables. Density is converted into [g/m^3].
3. *Construction of the predictor variables:* the predictor variables correspond to a third-order 2D polynomial (constant plus nine variables).
4. *Least squares regression with dimPred + 1 variables*. This part allows to collect the residual RMS `ResRMS` of the regression of an arbitrary order not greater than nine and plot the residual surface of the higher order regression as in Figs. 8.6 and 8.7.
5. *Analysis of variance*. The analysis is performed by the Matlab functions `fitlm` (it replicates the regression with the complete set of variables) and `anova`. They provide the printouts in Figs. 8.2–8.4. At the end, *t* and *F-statistics* together with the residual RMS are plotted versus the variable index μ as in Fig. 8.8.

```
%% NACL aqueous solution density fit (Chapter 8)
%% Initialization
InitChem;
Temp0=273.15;
dimPred=6; % user selected, regression order without constant
%% Raw data extraction and 2D plot
disp('Polynomial fit of raw density data vs temp & conc')
% Temp & conc anomaly make variables quasi-independent
RawTab=load('NACLSolTable.txt');
[dimRow,dimCol]=size(RawTab);
% Temp. extraction [degree Celsius]
Temp=RawTab(1,2: dimCol);
Temp=Temp-Temp0; % [degree Celsius]
TempAn=Temp-mean(Temp);  % [K]
%Conc. extraction [100g/g]
Conc=RawTab(2: dimRow,1);
ConcAn=Conc-mean(Conc);
% Density extraction
dimRaw=(dimCol-1)*(dimRow-1);
Dens=zeros(dimRaw,1);
Dens2D=RawTab(2:dimRow,2:dimCol)/milliScale;  %[g/m^3]
% Raw surface density plot
[th,c]=meshgrid(Temp,Conc);
figure('name','Density surface');grid on; hold on;
surf(th,c,Dens2D);
xlabel('Temperature [$^\circ$C]');ylabel('Concentration [100g/g]');
zlabel('Density [g/${m^3}$]');title('Raw data density surface')
view(-40,20);hold off;
%% Regression cubic predictors
dimVar=10;U=ones(dimRaw,dimVar);k=1;
for i=2:dimRow
    for j=2:dimCol
        Dens(k)=RawTab(i,j)/milliScale;  % g/m^3
        U(k,2)=TempAn(j-1); % T1
        U(k,3)=ConcAn(i-1); %C1
        U(k,4)=TempAn(j-1)^2; %T2
        U(k,5)=ConcAn(i-1)^2; %C2
        U(k,6)=U(k,2)*U(k,3); %T1C1
        U(k,7)=U(k,2)^2*U(k,3); %T2C1
        U(k,8)=U(k,2)*U(k,3)^2; %T1C2
        U(k,9)=TempAn(j-1)^3; %T3
        U(k,10)=ConcAn(i-1)^3; %C3
        k=k+1;
    end
end
```

```
%% Any order regression by least squares & residual plot
ResRMS=zeros(dimVar,1);
x=zeros(dimVar,dimVar); % Estimated coefficients per column
if dimPred <0, dimPred=0; end
if dimPred>dimVar-1, dimPred=dimVar-1; end
for kp=1:dimPred+1  % selected above
    Up=U(:,1:kp); x(1:kp,kp)=Up\Dens; % least squares fit
    Est=Up*x(1:kp,kp); % estimated density
    Res=Dens-Est; ResRMS(kp)=rms(Res); % residual
    % from vector to matrix
    Res2D=zeros(dimRow-1,dimCol-1); k=1;
    for i=1:dimRow-1
        for j=1:dimCol-1
            Res2D(i,j)=Res(k);k=k+1;
        end
    end
end
% residual surface plot
sPred=num2str(dimPred);
stitle=strcat('Linear regression fit, m=',sPred);
figure('name','Residual surface');grid on; hold on;
surf(th,c,Res2D*milliScale);
xlabel('Temperature [$^\circ$C]');
ylabel('Concentration [100g/g]');
zlabel('Density [$mg/{m^3}$]');
title (stitle);hold off;
view(-40,30);
%% Analysis of variance
disp('Analysis of variance')
% preparation to fitlm
T1=U(:,2);C1=U(:,3);
T2=U(:,4);C2=U(:,5);T1C1=U(:,6);
T2C1=U(:,7);T1C2=U(:,8);T3=U(:,9);C3=U(:,10);
% table of fitlm
UTab=table(T1,C1,T2,C2,T1C1,T2C1,T1C2,T3,C3,Dens,'VariableNames',...
{'T1','C1','T2','C2','T1C1','T2C1','T1C2','T3','C3','Density'});
model=fitlm(UTab)
tabVar=anova(model)
anova(model,'summary')
```

```
% statistical indices extraction
FTest=tabVar.F;
tStat=model.Coefficients.tStat;
xTest=0:dimVar-1;
% plot
figure('name','Test'); grid on;
semilogy(xTest,abs(tStat),'b--+');hold on;
semilogy(xTest(2:10),FTest(1:9),'r:o');hold on;
semilogy(xTest,ResRMS*milliScale,'g:s');hold on;
xlabel('Variable index $\mu$');
ylabel('Statistics []');
legend('abs(t-statistics)', 'F-statistics','Residual RMS [$mg/m^3$]');
title('Statistics and residual RMS, log scale');hold off;
```

References

[1] D. McQuarrie, P. Rock, E.B. Gallogly, General Chemistry, University Science Books, *Mill Valley*, CA, 2011.

[2] A.I. Simion, C.-G. Grigoraş, A.-M. Roşu, L. Gavrilă, Mathematical modelling of density and viscosity of NaCl aqueous solutions, J. Agroalimentary Process. Technol. 21 (1) (2015) 41−52.

[3] Companion Website of the Book. Elsevier, 2021. Available from: https://www.elsevier.com/books-and-journals/book-companion/9780323913416.

CHAPTER

Exploring seawater chemical equilibria with Matlab

9

9.1 Introduction

Seawater is not a simple reservoir of different dissolved salts, like sodium chloride, magnesium sulfate, and others, but is capable of reacting with different substances like the atmosphere's carbon dioxide, by modifying its concentration and buffering the anthropogenic increase. Since oceans cover about 71% of the Earth's surface, even a slight variation of the chemical parameters, like pH, salinity, and so on, have a huge effect on global climate and therefore on our way of life.

The involved reactions, however, cannot be mathematically treated as the usual aqueous solutions of chemistry textbooks. Seawater has a high salt content (measured by the ionic strength), and the potency of the ionic charge density significantly alters equilibrium constants based on pure water ionic equilibrium, so as to make them of little use. Furthermore, chemical equilibria and the corresponding parameters are greatly affected by the high hydrostatic pressures to be found in the dark abysses. The hydrostatic pressure, which, in marine chemistry, is usually measured in *bar* (1 bar = 100 kPa), may reach values above 1000 bar = 100 MPa. Since as remarked in the book Introduction (Section about Measurements units and universal constants), the unit *bar* is not accepted by SI, we will ordinarily use the SI unit Pascal and its multiples.

In this chapter, appropriate algorithms for solving chemical equilibria in seawater are proposed via simple scripts which are ready for use and easily adjusted by readers. The amount of numerical data to be employed by the scripts, has been collected in excel spreadsheets, to be downloaded from the companion website of the book [1].

Before explaining scripts, we will address some questions about seawater chemistry. To this end, we should remark that the concentration of dissolved substances is expressed with a slightly different *molality* (let us call it s*olution molality*), equal to the amount of moles of substances in 1 kg of solution (not of solvent as in the standard definition of Chapter 8). This is done in order to avoid the effect of volume reduction due to increasing pressure with the ocean depth. The corresponding unit will be denoted by [mol/kg−soln]. By denoting the standard (solvent) molality with m, the solution molality with m_{soln}, and the mass fraction solute/solution with μ, the following relation holds

$$m_{soln} = m(1 - \mu). \tag{9.1}$$

9.2 Why seawater reacts with atmospheric CO_2?

Breathe in, breathe out. Like a giant lung, oceans absorb vast amounts of carbon dioxide, CO_2, from the atmosphere and release it once again as soon as cold-water currents reach warmer areas of

Fundamental Chemistry with Matlab. DOI: https://doi.org/10.1016/B978-0-323-91341-6.00009-4

the globe. In fact, CO_2 solubility varies with temperature as well as with other factors such as salinity and pressure.

Chemically speaking, why does seawater so readily absorb carbon dioxide, thereby buffering anthropogenic emissions of the gas? Gaseous exchange occurs through the ocean's surface, but the answer to the question lies deeper, in what is a widely underestimated fact: the seawater pH (the acidity level) is substantially alkaline, ranging from 8.0 to 8.7. This means that the balance of positive and negative ions is reached through a concentration of hydroxide ions, OH^-, higher than the concentration of hydrogen ions, H^+.

Having a pH value greater than seven, enables seawater to react with and dissolve huge amounts of CO_2, thus absorbing part of atmospheric excess and affecting its concentration. The reason behind seawater alkalinity lies in the current chemical composition. Different salts are present in seawater, the primary one being sodium chloride. As soon as they dissolve in water, positive charges (cations) and negative charges (anions) are generated (see References [2] and [3] and Table 9.1).

In order to calculate the dissolution equilibria of the carbon dioxide, we need to explore the mean seawater composition in greater detail. If we refer to the standard seawater composition (see References [4] and [5]), summing up all the positive charges, namely Na^+, K^+, Mg^{2+}, Ca^{2+}, and Sr^{2+}, one obtains 0.60585 mol/kg−soln. Carrying out the same operation for the negative charges, namely for Cl^-, Br^-, F^-, and SO_4^{2-}, the sum is slightly smaller, namely 0.60325 mol/kg−soln. As a result, 2.60 mmol/kg−soln (millimoles per kilogram of solution) *of anions are missing!*

Since like all ionic solutions seawater must obey the law of electro-neutrality, it becomes evident that some negative charges (anions) have been forgotten! They are indeed HCO_3^-, to a minor extent OH^- and, to a far lesser extent, CO_3^{2-}. All the last three ions react with the atmospheric CO_2 and are therefore designated as *reactive*. On the contrary, the former cations and anions are classified as *spectator ions* (see Table 9.1). Reactive ions have an active role in chemical equilibria, as mentioned in the same table.

Table 9.1 Spectator and reactive ions in standard seawater.

No.	Cation	mol/kg-soln	g/kg-soln	Anion	mol/kg-soln	g/kg-soln
Spectator ions						
1	Na^+	0.46906	10.7836	Cl^-	0.54586	19.3524
2	Mg^{2+}	0.05282	1.2837	SO_4^{2-}	0.02824	2.7123
3	Ca^{2+}	0.01028	0.4121	Br^-	0.00084	0.0673
4	K^+	0.01021	0.3991	F^-	0.00007	0.0013
5	Sr^{2+}	0.00009	0.0079			
6	Sum	0.60565	12.8864	Sum	0.60325	22.1333
Reactive ions and molecules						
7	H_2CO_3	$\Leftrightarrow$	HCO_3^-	$\Leftrightarrow$	CO_3^{2-}	
8	$B(OH)_3$	$\Leftrightarrow$	$B(OH)_4^-$			
9	$2H_2O$	$\Leftrightarrow$	H_3O^+	$\Leftrightarrow$	OH^-	

The presence of the hydroxide anion OH^- is the reason for $pH > 7$. Its concentration at $pH = 8.0$ (due to the logarithmic nature of the pH scale) is equal to 0.001 mmol/L in pure water. Under the same conditions, the H^+ ion concentration is 100 times smaller. But OH^- ions alone are insufficient to fill the gap: other negative ions are required, mainly HCO_3^- ions and a smaller amount of CO_3^{2-}.

This fact has a significant impact on the CO_2 equilibrium between atmosphere and oceans. Compared to the atmosphere, which contains around 850 Gt (gigatons) of carbon in the form of CO_2, the oceans hold about 38,000 Gt of carbon. That is nearly 45 times more.

Thus, when we talk about CO_2 [ppmv] in the atmosphere, this is only the *top of the iceberg*! CO_2 dissolves in seawater like the atmospheric O_2 and N_2. However, being a reactive gas, an almost immediate reaction takes place with the water itself (no reaction takes place with N_2 and O_2), yielding HCO_3^- and CO_3^{2-}. Carbon dioxide itself also dissolves in meteoric waters, although to a far less extent, because those waters are not basic, but neutral. This phenomenon decreases the pH value of rain droplets to about 5.5. This fact is not to be confused with acid rains due to SO_2 pollution and formation of sulfuric acid in the rain droplets, with a much larger decrease of pH.

After the completion of these reactions, a further slow reaction takes place, which very often is disregarded: the formation of solid calcium carbonate $CaCO_3$ [4]. In chemistry, this reaction is known as *precipitation*. The precipitated $CaCO_3$ usually has the calcite structure; aragonite, the other polymorphic structure, is slightly more soluble. Seawater is oversaturated, both in terms of calcite and aragonite, due to its relatively high Ca^{2+} ion concentration equal to 10.28 mmol/kg − soln. However, this reaction requires nucleation and growth of crystal nuclei, and is usually sluggish (it may speed up in the cells of calcifying organisms like invertebrates). In other words, it is a heterogeneous reaction between a liquid phase and a solid one. The destiny of this salt is to eventually settle on the ocean floor. When the ocean is very deep, calcium carbonate may fail to reach the bottom, by dissociating again into ions due to the extremely high pressure, and recycle. In any case, CO_2, removed from the atmosphere, will eventually form limestone.

9.3 Methods and techniques for dealing with seawater chemistry

Every year hundreds of publications and articles appear about this topic: some of them worry about ocean *acidification* (lowering of ocean pH values, although remaining in the alkaline range) and the possible effects on calcifying organisms, some of them stress a possible increase of the ocean's ability to uptake anthropogenic CO_2 [6].

To this purpose, scientists are employing complex computer-aided models to simulate the chemical/physical behavior of the ocean water and predict the effects of man-made activities such as fossil burning. A comparison of the 10 most used computer packages to this purpose can be found in Reference [7]. One of the most addressed packages, SEACARB, is referenced in Reference [8]. Uncertainty and error propagation in numerical calculation are discussed in Reference [9].

These packages have been designed as professional tools to be employed by users through their input and output interfaces. As such, their scope is not to provide an insight into logic and procedures of the code—they are described in manuals and relevant scientific papers—, but just to provide reliable results. The goal of this textbook is somewhat different, as chemistry problems, in this Chapter about seawater equilibria, are solved with Matlab® scripts which have been designed to be as simple as possible. The underlying theory and the code itself are explained in some detail, and as such, the code may

be adjusted by users. Also, key Matlab functions are pointed out with a brief explanation, to be completed by users with a reference to Matlab help. Large input data have been collected into excel files. Anybody with a little chemical knowledge will be able to follow them. Fairly all the scripts are reported in the book. Scripts and excel files can be downloaded from the companion website of the book [1].

Despite frequent discussions and analysis, in both scientific papers and public media, about the relationship between ocean chemistry and environmental issues, such as CO_2 uptake, ocean acidification, and carbonate sedimentation, the basic underlying chemistry looks poorly understood.

On the other hand, by exploiting Matlab functions and capabilities, scripts not longer than a few hundred lines can be developed, to aid chemistry in producing a variety of simple and interesting results for anybody curious about seawater reactions. Well, let us not oversimplify! Seawater solution possesses a high ionic strength (a high density of oppositely charged ions), a fact that hinders the direct usage of equilibrium constants taken from standard thermodynamic databases. For the same reason, temperature, pressure, and salinity dependence of the above constants is not at all straightforward and should be carefully accounted for. Consequently, simple equilibrium constants are of limited use in the numerical solution of equilibria. On the contrary, by encoding the expansions of the equilibrium constants which are available in the literature, such as in References [5,10–13], the results of the simultaneous seawater reactions are just the matter of few seconds. The computational speed should not lead us to neglect the kinetics of reactions, that in some case, like heterogeneous precipitation reactions, are far from immediate and may have a time scale of centuries!

9.4 Surface chemistry

9.4.1 Description and graphical results

The Matlab script of this section only applies to ocean surface, where *hydrostatic pressure is zero*, namely 0 Pa.

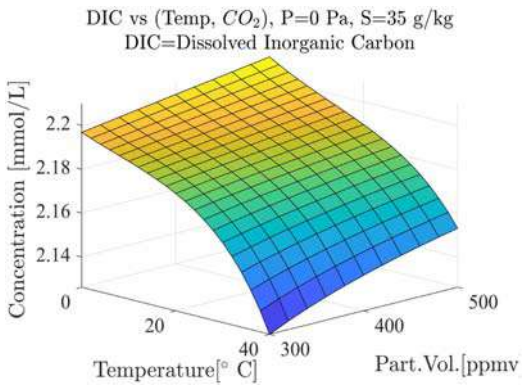


FIGURE 9.1

Concentration of dissolved inorganic carbon (DIC) versus seawater temperature and CO_2 volume fraction in dry air.

This section being introductory to Section 9.5 and deliberately simple, the actual seawater chemical composition is not explicitly accounted for and thus cannot be varied. To the purpose, the script in Section 9.4.2 only accounts for the *electrical imbalance* of the spectator ions, which has been computed for the standard salinity $S = 35\text{g/kg}$, to be meant as a mass fraction solute/solution. As a consequence, salinity must be kept constant.

The script only aims to compute the concentration of the Dissolved Inorganic Carbon (DIC) at the ocean surface and to provide the relevant graphical plot in Fig. 9.1. The latter shows the 3D plot of the DIC concentration in $[\text{mmol/L}]$, versus (1) the seawater temperature in [°C] and (2) the volume fraction in $[\text{ppmv} = \text{cm}^3/\text{m}^3]$ of the carbon dioxide, CO_2, in dry air. The DIC concentration decreases with increasing seawater temperature and with a decrease of the CO_2 fractional volume.

9.4.2 Matlab script

As explained in the previous section, the DIC concentration is searched at each seawater absolute temperature T and fractional volume V of CO_2 in dry air, by guaranteeing the dissolved species neutrality as in Chapter 7 about acid–base equilibria in water. To this end, each dissociation constant K_x has to be computed as a function of T and salinity S, the latter being expressed as a mass fraction solute/solution in $[\text{g/ kg}]$.

Let $K_x(T,S)$, with $x = w, 0\text{H2CO3}, \text{H2CO3}, \text{HCO3}$, denote a generic dissociation constant as a function of the variable pair $\{T,S\}$, which accounts for air and seawater conditions of interest. The function is expanded into a suitable series as follows

$$\begin{aligned}\log(K_x) \simeq \sigma(c_{x0} + c_{x1}T^{-1} + c_{x2}\log(T) + \sqrt{S}\left(c_{x3} + c_{x4}T^{-1} + c_{x5}\log(T)\right) + \\ + \quad S\left(c_{x6} + c_{x7}T^{-1} + c_{x8}T + c_{x9}T^2\right) + c_{x10}S^2)\end{aligned}, \tag{9.2}$$

where σ is a scale factor. Coefficient and scale factor values are reported in the Appendix E. The list of the dissociation constants can be found in Table 9.2. The constant in the first row is set to zero (no symbol).

The previous dissociation constants are employed to compute the concentration of the dissolved ions as a function of the pH value which ensures the solution neutrality. The neutrality equation includes H^+, OH^-, the three components of the dissolved carbon H_2CO_3, HCO_3^-, and CO_3^{2-}, and the so-called *imbalance* concentration with positive charge. Neutrality zero crossing is achieved by the Matlab function `fzero`, as done in Chapter 7. The function in turn calls `SeaWNeutr` in charge of computing the current neutrality. The main script, after initialization and the section about user data, reads the excel file `SeaWaterTable.xlsx`, and loops on temperature and CO_2 partial volume to compute the DIC concentration, defined by

$$\text{DIC}(T,V,S) = [H_2CO_3] + \left[HCO_3^-\right] + \left[CO_3^+\right]. \tag{9.3}$$

Table 9.2 Dissociation constants.

No.	Symbol	Name
1	None	void
2	K_w	Dissociation constant, $H_2O \Longleftrightarrow H^+ + OH^-$
3	K_{0H2CO3}	Dissociation constant, $CO_2 + H_2O \Longleftrightarrow H_2CO_3$
4	K_{H2CO3}	Dissociation constant, $H_2CO_3 \Longleftrightarrow H^+ + HCO_3^-$
5	K_{HCO3}	Dissociation constant, $HCO_3^- \Longleftrightarrow H^+ + CO_3^{--}$

The main script ends by plotting the surface of DIC versus temperature and the CO_2 partial volume.

The script is briefly explained, since it is a simplified version of the script in Section 9.5. It consists of the following parts.

1. *Initialization*. The logical variable `logDisp` regulates the iteration figure of the zero-crossing function`fzero` (see the item 4 below).

```
%% Sea water equilibria: surface chemistry (Chapter 9)
%% 1) Initialization
disp('SeaWater: surface chemistry (0 Pa) ');
disp('1) Initialization');
InitChem;logDisp=0;
global x
T0=273.15;
```

2. *User data*. Users may modify the temperature and partial volume ranges as well as the iteration step. Salinity must be kept constant because of the fixed imbalance.

```
%% 2) User data
disp('2) User data');
sh1Name='Temperature'; sh2Name='SalTemp'; %excel sheets
symbol={'H^+','HO^-','H_2CO_3','HCO_3^-','CO_3^2-','Imbalance'};
dims=length(symbol);
dimK=5; % dissociation constants:  H2O, H2CO3 (2) HCO3
c=zeros(dims,2); %concentration and charge
c(1:dims,2)=[1;-1;0; -1;-2;1]; % H+ OH- H2CO3 HCO3- CO3-- imbalance
x=c(:,1); % final concentration vector
Atot=1/microScale; %scale factor
Imbalance=0.00218;
% Ranges
Smin=35; Smax=35; dS=1; %salinity, gram/kg
S=Smin:dS:Smax; dimS=length(S);
Tmin=0; Tmax=40; dT=2; % Temperature, degree Celsius  <<< user
T=Tmin:dT:Tmax; dimT=length(T);
Vmin=300; Vmax=500; dV=20;
% CO_2 partial volume in ppmv in dry air <<< user
V=Vmin:dV:Vmax; dimV=length(V);
pHmin=0;pHmax=14;p0=7;
% matrices for plotting surface
Temp=zeros(dimT,dimV);
CO2Vol=zeros(dimT,dimV);
DIC=zeros(dimT,dimV);
```

3. *Reading and printing the excel table of expansion coefficients*. See Appendix E. Expansion coefficients versus temperature and salinity, according to Eq. (9.2), are printed in the `command window`.

```
%% 3) Reading excel table of dissociation coefficients
disp('3) Reading coefficient tables, Temperature and Salinity');
% temperature
fprintf('SeaWater:  Temperature coefficients');
TTab=readtable('SeaWaterTable.xlsx','ReadVariableNames',true,...
      'ReadRowNames',true,'PreserveVariableNames',true,'Sheet',
       sh1Name);
[nrowT,ncolT]=size(TTab);
sprint=TTab.Species; % species names
fprintf('\nVariable      ');
for i=1:dimK, fprintf('%12s', cell2mat(sprint(i))); end
      for k=1:ncolT
      sprint=TTab.Properties.VariableNames(k);
      fprintf('\n%14s ',cell2mat(sprint));
      for i=1:dimK
            if TTab{i,k}==0 || abs(TTab{i,k})==1, fprintf('%12.1f',
            TTab{i,k}); else fprintf('%12.4e', TTab{i,k}); end
      end
end
% salinity
fprintf('\nSeaWater: Salinity coefficients');
STab=readtable('SeaWaterTable.xlsx','ReadVariableNames',true,...
      'ReadRowNames',true,'PreserveVariableNames',true,'Sheet',
      sh2Name);
[nrowS,ncolS]=size(STab);
for k=1:ncolS
      sprint=STab.Properties.VariableNames(k);
      fprintf('\n%14s ',cell2mat(sprint));
      for i=1:dimK
           if STab{i,k}==0 || abs(STab{i,k})==1, fprintf('%12.1f',
           STab{i,k}); else fprintf('%12.4e', STab{i,k}); end
      end
end
fprintf('\n');
```

4. *Search for electro-neutrality*: loop on independent variables, temperature and CO_2 partial volume.

   As already said, neutrality is looked for by the zero-crossing function `fzero`, which in turn calls `SeaWNeutr(p,K,c,dims)` in charge of computing concentrations and electro-neutrality, on the basis of the input pH variable `p`. The iteration sequence of the function `fzero` can be shown and graphically plotted by setting `logDisp = 1` in the introductory script.

```
%% 4) Neutrality: loop on temperature and CO2 partial volume
disp('4) Neutrality: Loop on temperature and CO2 partial volume');
K=zeros(dimK,1);
c(dims,1)=Imbalance*S(1)/Smin; %imbalance
for i=1:dimT
      Ta=T(i)+T0; Sl=S(1);
      rS=sqrt(Sl); S2=Sl^2;
      Ta2=Ta^2; lnTa=log(Ta);
      fT=[1 1/Ta lnTa Ta ]; % temperature
      fS=[rS rS/Ta rS*lnTa rS*Ta rS*Sl rS*Sl/Ta Sl Sl/Ta Sl*lnTa
          ... Sl*Ta Sl*Ta2 S2 S2/Ta ]; % salinity
      for k=2:dimK % loop on species
           coT=TTab{k,2:ncolT};
           coS=STab{k,1:ncolS};
           scale=TTab{k,1};
           K(k)=exp(scale*(coT*fT.'+coS*fS.'));
      end
      for j=1:dimV
           c(3,1)=Atot*V(j)*K(3); % H2CO3
           p=p0;plim=[pHmin;pHmax];
           fer=@(p) SeaWNeutr(p,K,c);
           if i==1 && j==1&& logDisp==1
                options =optimset('PlotFcns',@optimplotfval);
                [p1,fer1,exitflag,output] =
                fzero(fer,plim,options);
           else
                [p1,fer1,exitflag,output] = fzero(fer,plim);
           end
           DIC(i,j) = (x(3)+x(4)+x(5))*milliScale;
           Temp(i,j) = T(i);CO2Vol(i,j) = V(j);
      end
end
```

5. *Graphical tool*. The DIC surface versus temperature and carbon dioxide partial volume is graphically plotted as in Fig. 9.1.

```
%% 5) Graphical result: plotting DIC surface
disp('5) Plotting DIC (dissolved inorganic carbon)');
figure('name', 'Surface');
surf(Temp,CO2Vol,DIC);
title({'DIC vs (Temp, $CO_2$ppmv), P=0 Pa, S=35 g/kg'; ...
      'DIC=Dissolved Inorganic Carbon'},'FontSize',16);
view([45 20]); %set viewpoint direction
xlabel('Temperature[$^\circ$ C]'); ylabel('Part.Vol.[ppmv]');
zlabel('Concentration [mmol/L] ');
```

6. *The electro-neutrality function* `SeaWNeutr`. The function produces the electro-neutrality balance `fer` to be brought to zero in search of equilibrium. Input data are the scalar `p` which stands for pH, the vector `K` of the dissociation constants and the matrix `c` of concentration and charge.

```
%% 6) Function SeaWNeutr
function fer=SeaWNeutr(p,K,c)
      global x
      H=10^(-p); c(1,1)=H;
      OH=K(2)/H; c(2,1)=OH;
      c(4,1)=K(4)*c(3,1)/H; %HCO3
      c(5,1)=K(5)*c(3,1)/H; %CO3
      % neutrality
      x=c(:,1); % concentration vector
      fer=x.'*c(:,2);
end
```

9.5 Ocean chemistry under pressure up to 100 MPa

9.5.1 Different pH scales in ocean chemistry

In chemical oceanography, three different pH scales are currently used: *free*, *total*, and *seawater*. This point is not to be neglected. When dealing with the acidity constants of hydrogen ion transfer reactions as is the case of H_2CO_3, the use of a consistent pH scale is mandatory. The values of different pH scales in seawater differ by up to 0.12 pH units [5].

We recall that the pH value is defined as the negative logarithm of the concentration of hydrogen ions, namely $\mathrm{pH} = -\log_{10}\left[\mathrm{H}^+\right]$. Unfortunately, individual ion activities cannot be experimentally determined. Indeed, the concentration of a single ion cannot be varied independently, because electro-neutrality is required. Therefore the *free scale* for seawater has been proposed, denoted by $\mathrm{pH_F}$ and defined by

$$\mathrm{pH_F} = -\log_{10}\left[\mathrm{H}^+\right]_\mathrm{F}, \tag{9.4}$$

where $\left[\mathrm{H}^+\right]_\mathrm{F}$ stands for the concentration of free hydrogen ions, including hydrated forms, like H_3O^+ [14]. We recall, as already done, that, as usual in ocean chemistry, concentrations in square brackets are expressed in $\left[\mathrm{mol/kg-soln}\right]$ (aqueous solution) and not in $\left[\mathrm{mol/L}\right]$ as usual in general chemistry.

In 1973 Hanson [15] introduced the *total scale*, denoted by $\mathrm{pH_T}$, which includes the contribution of sulfate ions as follows

$$\mathrm{pH_T} = -\log_{10}\left[\mathrm{H}^+\right]_\mathrm{T},\ \left[\mathrm{H}^+\right]_\mathrm{T} = \left[\mathrm{H}^+\right]_\mathrm{F} + \left[\mathrm{HSO}_4^-\right]. \tag{9.5}$$

The bisulfate ion HSO_4^- is a rather weak acid. Since the dissociation constant holds $K_{a1} \simeq 0.021$, the bisulfate does not completely dissociate into H^+ and SO_4^{2-} ions. By knowing the value of the dissociation constant, a relationship between the two pH scales, free and total, can be inferred. This in turn requires an accurate determination of the K_{a1} value in seawater, which would be difficult to obtain. By using the Hanson's total scale, the determination of the bisulfate ion constant K_{a1} may be avoided.

The third scale is the so-called *seawater scale*, denoted by pH_{SW}, which slightly differs from the preceding one. The reason of this scale is the presence, in seawater, of fluoride ions F^- [16]. Consequently, we have to account for the protonation of the F^- ions according to the equilibrium:

$$HF \Leftrightarrow H^+ + F^-, \text{ dissociation constant } \quad K_{a2} = 0.35 \times 10^{-3}. \tag{9.6}$$

Hydrofluoric acid also is a weak acid. In standard seawater, the concentration of the fluoride ions amounts to 0.07 mmol/kg − soln of water, about 400 times smaller than the concentration of the sulfate ions, which amounts to 0.028 mol/kg − soln. Therefore the seawater scale differs by no more than 0.01 pH units from the total scale.

The following identities provide the transformation between the different scales, in terms of concentration and pH units:

$$\begin{aligned} pH_T &= pH_F - \log_{10}\left(1 + \frac{[SO_4^{2-}]}{K_{a1}}\right) \\ pH_{SW} &= pH_F - \log_{10}\left(1 + \frac{[SO_4^{2-}]}{K_{a1}} + \frac{[F^-]}{K_{a2}}\right) \\ pH_{SW} &= pH_T - \log_{10}\left(1 + \frac{[F^-]}{K_{a2}}\right) \end{aligned} \tag{9.7}$$

9.5.2 Heterogeneous reactions in ocean chemistry

Generally speaking, reactants of heterogeneous reactions are in different phases, like solid, liquid solution, or gaseous mixture. One of the most relevant heterogeneous reaction is the formation or dissolution of calcium carbonate (solid/solution), according to the oversaturation ratio, which is denoted by Ω. Oversaturation, whose value is greater than unit, is defined as the ratio between the concentration product and the solubility product K_{sp} as follows

$$\Omega = \frac{[Ca^{2+}][CO_3^{2-}]}{K_{sp}}. \tag{9.8}$$

Oversaturation differs between calcite and aragonite, the former being the greater [4].

9.5.3 Hydrostatic pressure acting on homogeneous and heterogeneous equilibria

The effect of pressure on equilibrium constants is of paramount importance. Sinking down into ocean depths, pressure increases by 100 kPa every 10 m. Since the intermolecular distance between water molecules slightly decreases, density of liquid water accordingly increases. Therefore interionic interaction and equilibrium constants become progressively altered in function of the pressure itself. The effect becomes noticeable when pressure reaches tens of MPa. pH and solubility of calcium carbonate alter to such an extent that aragonite oversaturation, as well as the calcite at greater depths, vanish and, whether oversaturation occurs, salts readily re-dissolve.

Calcium carbonate, once has been formed either biologically by calcifying organisms or by an inorganic route, and has reached a density greater than unit, sinks down into the dark abyss. However, due to increasingly high pressure, the solid $CaCO_3$ begins to dissolve below a certain depth, which is referred to as the *saturation horizon*, where oversaturation equals one, namely

$\Omega = 1$. Since dissolution of the solid $CaCO_3$ is not instantaneous, the downward flux continues until the solid particles of calcium carbonate are completely dissolved. The relevant depth is known as the *carbonate compensation depth*. When the sea bottom does not reach this depth, $CaCO_3$ becomes undissolved carbonate sediment. The two crystallographic forms of $CaCO_3$, calcite and aragonite, have a different solubility product k_{sp}, the former being less soluble. Therefore the saturation horizon and the compensation depth of aragonite happen to have greater values than calcite. Most calcifying organisms, like for instance the submillimetric Coccolithophores, produce calcite, whereas coral reefs are made of aragonite [15].

By solving seawater equilibria, which include $CaCO_3$ formation as in the Matlab scripts of Sections 9.5.5 to 9.5.8, the oversaturation profile at different depths can be calculated and graphically reported. Some of the results can be seen in the graphical plots of Fig. 9.2.

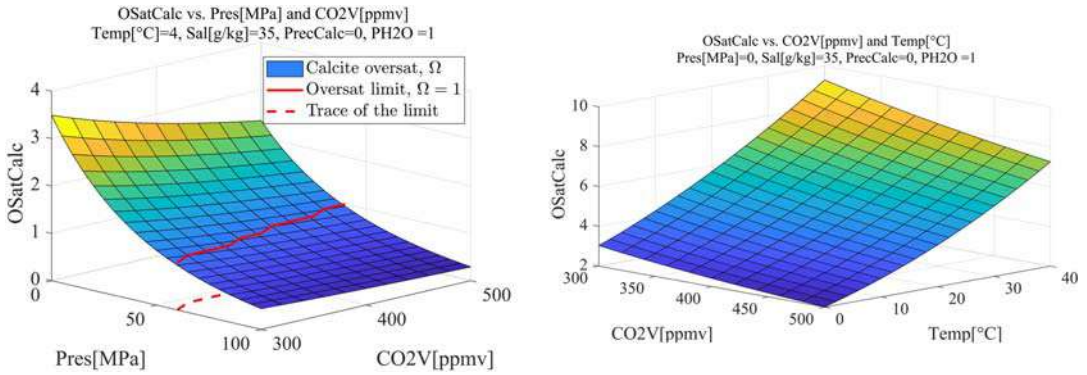


FIGURE 9.2

(Left) Calcite oversaturation versus pressure and CO_2 partial volume. (Right) Calcite oversaturation at surface pressure versus temperature and CO_2 partial volume.

In Fig. 9.2, left, calcite oversaturation Ω (indicated by `OSatCalc`) is plotted against pressure and CO_2 partial volume, the temperature being fixed at $4°C$. The latter value is the average ocean temperature at depths below the thermocline, which is situated at about 300m. The CO_2 partial volume covers approximately the range from the preindustrial value of 280 ppmv, to the mid-20th century value of 345 ppmv, to the present-day value of 410 ppmv up to the 2050 forecast of 460 ppmv. Below $\Omega = 1$, indicated by a red line in Fig. 9.2, left, carbonates begin their dissolution process.

A frequently debated topic is the potential hazard for coral reefs of the rising concentrations of CO_2, because of the reduction of the ocean pH and of carbonate ion concentration (the ordinate variable which decreases under CO_2 partial volume increase). The latter effect, which is evident from Fig. 9.2, right, should be however mitigated by a concomitant increase of the temperature, due to global warming of oceans, where calcifying organisms and coral reefs prosper. Global warming, estimated to be about $1.2(\pm 0.1)°C$ from the beginning of the twentieth century to present days, favors oversaturation and thereby should counteract the effect of increasing CO_2 due to anthropogenic emissions. Therefore Fig. 9.2, left and right, should be considered together to have a complete picture.

To conclude the section, let us again remark that since the sea surface pressure is set to zero, the water pressure in the script means hydrostatic pressure.

The effect of the pressure on the generic equilibrium constant K_i, where i denotes a species as in Table 9.6, can be calculated [8] by expanding into a second-order polynomial, the natural logarithm of the ratio between K_{iP}, the value at a pressure P, and K_{i0}, the value at zero pressure, as follows:

$$\log\left(\frac{K_{iP}}{K_{i,0}}\right) = -\frac{\Delta V_i}{RT}P + \frac{\Delta Z_i}{2RT}P^2, \tag{9.9}$$

where $R = 8.314\ \mathrm{cm^3 MPa K^{-1} mol^{-1}} = 8.314\ \mathrm{J K^{-1} mol^{-1}}$ is the universal gas constant, ΔV_i $\left[\mathrm{cm^3/mol}\right]$ is the molar volume change, and ΔZ_i $\left[\mathrm{MPa^{-1}}\right]$ denotes the compressibility change. ΔV_i and ΔZ_i are in turn expanded in terms of the following second-order polynomials:

$$\Delta V_i = a_{V0} + a_{V1}T_c + a_{V2}T_c^2, \quad \Delta Z_i = a_{Z0} + a_{Z1}T_c. \tag{9.10}$$

The coefficient values, from References [5,11,12] and [17], are collected in the excel spreadsheet `SeaWaterTable.xlsx`. The coefficient values are reported in Appendix E, and the excel spreadsheet can be downloaded from the companion website of the book [1]. T_c stands for the temperature in [°C], P denotes the pressure in [MPa], and $T = T_c + 273.15$ is the absolute temperature in [K]. As already said, pressure alters the values of the equilibrium constants and must be properly accounted for in the script flow.

Care must be taken not to extrapolate the script algorithm to pure water (zero salinity). The valid range of the salinity is different according to various authors, usually being $5 \sim 45$ g/kg. The actual value for ocean waters lies in the range $32 \sim 38$ g/kg [5]. The sum in Table 9.1, row 6, of the spectator ions amounts to 35.02 g/kg, in the middle of the last range. As seen from Eq. (9.9), the effect of the salinity change is not considered at this stage of calculations.

9.5.4 Ocean chemistry in a broad perspective

Transferring the results of seawater equilibria to the real oceans is far from straightforward.

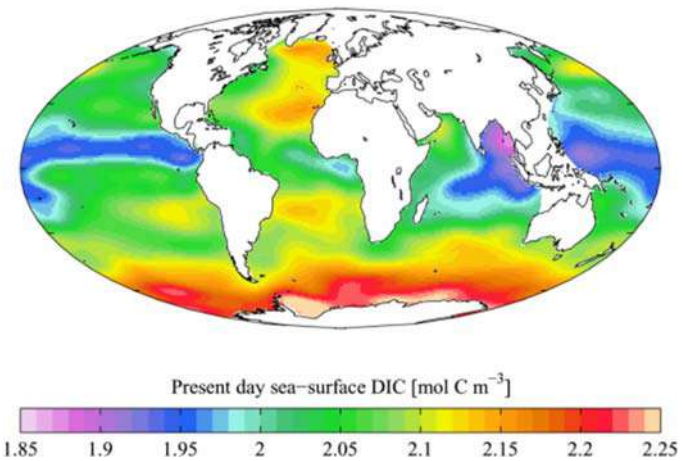


FIGURE 9.3

Current DIC distribution on the ocean surface. *DIC*, dissolved inorganic carbon.

A complex pattern emerges since temperature, salinity, and DIC vary from arctic waters to tropical ones. By way of example, DIC distribution on the ocean's surface is depicted in Fig. 9.3 as taken from Reference [18] (DIC concentration is expressed in moles of carbon per cubic meter [mol C m^{-3}], equivalent to mmol/L).

As in the atmosphere, ocean streams, like the Gulf Stream, move huge masses of waters, which eventually sink, a fact known as vertical mixing. Vertical mixing in the oceans is driven by buoyancy. The factors that determine buoyancy in the oceans are temperature and salinity. Sinking of waters from surface to the ocean depths takes place in polar regions where surface water is cold and salty, hence heavier. Polar sinking in turn forces the movement of tropical waters toward polar regions, which causes evaporation and increase in salinity, thereby closing the cycle. In other regions, the ocean surface is warmer than the water underneath, so that any vertical mixing is suppressed. Some vertical mixing still takes place near the surface due to wind stress, resulting into an oceanic mixed layer which extends down to about 100 m in depth and slowly exchanges with the deeper ocean. Time variations of the gaseous exchange with the air in the mixed layer (the layer volume equals 36×10^{15} m^3) develops over a time scale of 1 year. Equilibration of the whole ocean, for example in response to a change in the atmospheric CO_2, has the much longer time scale of about 120 years or more.

Therefore, in applying the results of the following Matlab scripts to the ocean as a whole, many precautions should be taken, bearing in mind that while CO_2 concentration is practically uniform in air, there are marked temperature, salinity, and DIC variations across the globe.

As we know, there is an ocean uptake of CO_2 in colder areas (at the high latitudes) and an ocean outgassing in warmer ones (tropics). The blue curve in Fig. 9.4, left, corresponds to the DIC at equilibrium, as calculated by the Matlab script in Section 9.5.5 under a standard seawater composition, in equilibrium with 410 ppmv of CO_2 (no calcite precipitation occurs) and as a function of temperature from 0°C to 30°C. The abbreviated concentration unit notation [mM = mmol/kg-soln] is employed.

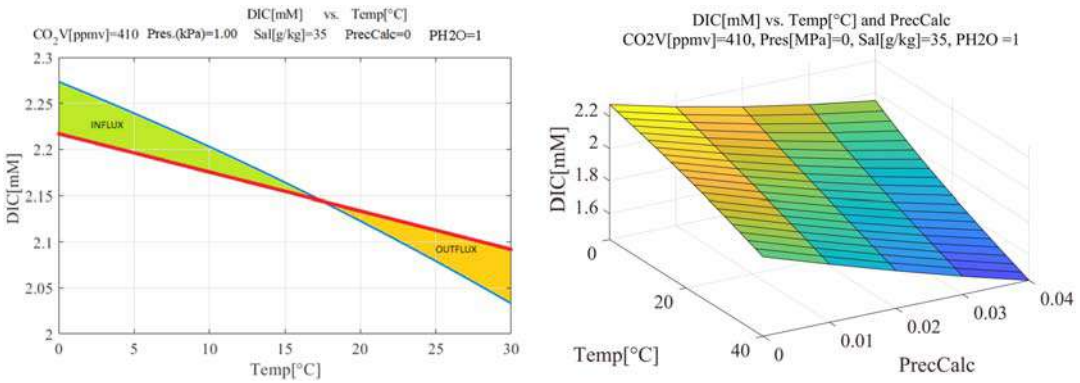


FIGURE 9.4

(Left) Equilibrium DIC at ocean surface compared to in situ measured. DIC and to the relevant CO_2 influx and outflux zones. (Right) Surface DIC versus temperature and 0% to 4% of calcite precipitation. *DIC*, dissolved inorganic carbon.

Since ocean waters are not in a complete equilibrium, the real DIC values DIC_{meas} are reported as an estimated red line on the same figure. They are taken from Reference [4] (figure 6.15(b), p. 147). In this way, it is possible to assess, in the same Fig. 9.4, the DIC disequilibrium $\Delta DIC = DIC - DIC_{meas}$, where DIC has been computed by the script of Section 9.5.5 and DIC_{meas} has been measured in situ. The difference ranges from $+0.06$ to -0.06 mmol/kg − soln across the whole ocean surface of the globe [4], indicating a CO_2 intake (influx) at temperatures lower than 17°C and an outflux at higher.

The time response of the oceans has to be accounted for, as well as the temporal evolution of calcite/aragonite formation. For example, by imposing in our simulation, an increasing calcite precipitation from 0.0 to 0.04 (4%), the DIC surface in Fig. 9.4, right, shifts progressively to smaller values, thus indicating that a significant part of DIC has disappeared, being now in the sediments. The real behavior needs of course a deeper insight into the time response of the oceans and appropriate dynamic models and time simulation, which are not the goal of this introductory book.

9.5.5 Matlab script

The Matlab script is organized in two parts: computation and graphical results.

1. *Computation*. The first part, `SeaWaterDeepChemistry.m`, computes the seawater equilibria of the different dissolved substances which ensure electro-neutrality, by varying five independent variables, namely, the water temperature T [K], the partial volume V of the carbon dioxide in the dry air [ppmv], the water pressure P [MPa], the salinity S [g/kg], and the fraction of precipitated calcite with respect to the amount derived from the solubility product of calcium carbonate (`pptF`,φ). This can be considered as the reaction yield (see Section 5.13.1 of Chapter 5) of the reaction $Ca^{2+} + CO_3^{2-} \Rightarrow CaCO_3 \quad (\text{solid, calcite})$. Being the seawater usually oversaturated, a *unitary* reaction yield would correspond to a complete calcite precipitation from a seawater solution where the identity $\left[Ca^{2+}\right]\left[CO_3^{2-}\right] = K_{sp}$ (calcite) holds. On the contrary, *zero* yield would mean no formation of $CaCO_3$. All the values between *zero* and *one* are allowed by the script computation. The matrix `Var` of the independent variables is saved in the Matlab file `SWIndVariables.mat`. The numerical values of different concentrations are provided and saved in the file `SWResults.mat`. Numerical results are printed in the `Command Window` versus a single-independent variable, whose index, `ipr`, must be selected by users, as shown by the statements of the user section to be listed below. The statements define the table `Rvar`, which, provided by users, includes the range and the incremental step of each independent variable. The suggestion is to simulate the excursion of a single pair of variables, leaving the other three with the same minimum or maximum value (or any other intermediate value chosen by users) like the last three variables in the statements below, namely pressure, salinity, and `pptF`. The three values of each row are the minimum value, the maximum value, and the incremental step. They are checked by the script and, when wrong, corrected in order to guarantee the maximum value not to be smaller than the minimum, and the incremental step not be smaller than the default values in the vector `minStep`. Of course, the amended indices may not provide the expected graphical results. The entries of the vector `minStep` must be positive.

Units have been already mentioned, except for the temperature to be provided in [°C]. Absolute temperatures are computed by the script.

```
% --- User selection: printout variable and graphical range
ipr=1; % <<<< user  Index of independent variable for printout
if ipr<1  || ipr>dimVar, ipr=1; end  % default ind var =temp
fprintf('   Ind var to be printed  %8.1f  \n',ipr);
dimv=zeros(dimVar,1); % variable size
% Computing and graphical range of independent variables
% column 1 = min value, 2 = max value, 3 = step
minStep=[2; 20; 5; 1; 0.01]; % WARNING: minStep > 0
for i=1:dimVar
      if minStep(i)<=0, minStep(i)=1; end
end
% SUGGESTION:  only two independent variables with max > min value
% The other three variables with min=max
RVar=[ 4 4 1; % Temp degree centigrade <<< user
      300 500 20; % Vol CO2 partial volume ppmv
      0 100 5; % Pressure MPa
      35 35 1; % Sal
      0. 0.0 0.01]; % pptF CaCO3 fraction that precipitates
for i=1: dimVar
      if RVar(i,2)<RVar(i,1), RVar(i,2)=RVar(i,1); end % max >= min
      if RVar(i,3)<minStep(i), RVar(i,3)=minStep(i); end % step>0
end
fprintf('   Ind. var    Min value  Max value      Step   \n');
for i=1:dimVar
      fprintf('  %8.1f    %8.1f   %8.1f   %8.1f  \n',...
           i, RVar(1,1),RVar(i,2),RVar(i,3));
end
% --- End of user part
```

As an aid to readers, Table 9.3 provides the range of the independent variables in agreement with the expansion coefficients of the equilibrium constants. The same table also provides the actual range of the same variables across the oceans.

Table 9.3 Suggested ranges of independent variables and actual range across oceans.

No.	Variable	Script name	Unit	Symbol	Suggested range	Actual range
1	Temperature	`Temp[°C]`	°C	T_c	$-2 \sim 40$	$-2 \sim 30$
2	CO_2 partial volume	`CO2[ppmv]`	ppmv	V	$100 \sim 5000$	$280 \sim 500$
3	Pressure	`Pres[MPa]`	MPa	P	$0 \sim 100$	$0 \sim 110$
4	Salinity	`Sal[g/kg]`	g/kg	S	$0 \sim 45$	$32 \sim 38$
5	Fraction of precipitated calcite	`PrecCalc`		φ	$0 \sim 0.1$	$0 \sim 0.01$

2. *Graphical results*. The second part, `SeaWaterDeepGraphTool.m`, reads the previous files and provides two kinds of graphical results, surfaces and 2D plots. The indices of the variables to be plotted are provided by users in the tables `Sgr` and `Pgr`, like in the following statements. Each row corresponds to a figure, either a surface (`Sgr`) or a 2D plot (`Pgr`). The last index (the

third one in `Sgr`, the second in `Pgr`) selects one of the 14 dependent variables (see Table 9.4) produced by the first script. The first indices (a pair in `Sgr`, a single one in `Pgr`) indicate one of the five independent variables. In the case that one of the independent variables has only a single value, equal to the minimum value of the range table `RVar` in the main script, the figure is not provided, and the message `Insufficient dimensions` is printed out. In the case of a single-value variable, the maximum value must be equal to the minimum.

Both tables, `Sgr` and `Pgr`, are checked as in the script below, and the wrong indices are corrected to their default values: to the unit when out of the range (less than 1 or greater than 5 for independent variables, greater than 14 for dependent variables), the index of the second-independent variable in `Sgr` is increased by 1 (set to 1 when equal to 5) when equal to the first index, or out of range. WARNING messages are printed out.

```
% --- user selection: graphical variable
% two matrices: Sgr for surface plot, Pgr for 2D plot
% Sgr: ind. variables (Var) column 1: surface x, column 2 surface y
% Sgr: dependent variable (Res), column 3 : surface z
% WARNING: if an independent variable has been computed
%           with min=max values in the main script,
%           'Insufficient dimensions' will be printed out
% More than one surface can be plotted
% WARNING: wrong variable indices are set to 1 or added by 1
% WARNING: 2nd equal ind. var. index is added by 1
Sgr=[ 3 2 8;
      5 4 5];
% 1 2 10;
% 1 2 11]; % <<<< user surface [IndVar1 IndVar2 Res]
[rowS,colS]=size(Sgr);
% check
fprintf('Sgr Check\n');
for i=1: rowS
      if Sgr(i,1)<0 || Sgr(i,1) > dimVar
           fprintf('WARNING: row=%2.0f, col=1, index=%4.0f
            set =1\n',...i,Sgr(i,1));
           Sgr(i,1)=1;
      end
      if Sgr(i,2)<0 || Sgr(i,2) > dimVar
           fprintf('WARNING: row=%2.0f, col=2, index=%4.0f
                set=Sgr(i,1)+1\n',...i,Sgr(i,2));
           Sgr(i,2)=mod(Sgr(i,1),dimVar)+1;
      end
      if Sgr(i,3)<0 || Sgr(i,3) > dimRes
           fprintf('WARNING: row=%2.0f, col=3, index=%4.0f
                set =1\n',...i,Sgr(i,3));
           Sgr(i,3)=1;
      end
      if Sgr(i,1)==Sgr(i,2)
           fprintf('WARNING: row=%2.0f, col=1, index=%4.0f
                set=Sgr(i,1)+1\n',...i,Sgr(i,1));
           Sgr(i,2)=mod(Sgr(i,1),dimVar)+1;
      end
end
```

```
% Pgr: ind. variable (Var)  column 1: graph x,
% Pgr: dependent variable (Res), column 3 graph y
% WARNING: if the independent variable has been computed
%           with min=max values in the main script,
%           'Insufficient dimensions' will be printed out
% More than one 2D graph can be plotted
% WARNING: wrong variable indices are set to 1
Pgr=  [3  1;
      -2  8]; % <<< user 2D plot  [IndVar1 Res]
[rowP,colP]=size(Pgr);
% check
fprintf('Pgr Check\n');
for i=1: rowP
      if Pgr(i,1)<0 || Pgr(i,1) > dimVar
           fprintf('WARNING: row=%2.0f, col=1, index=%4.0f
               set =1\n',...i,Pgr(i,1));
           Pgr(i,1)=1;
      end
      if Pgr(i,2)<0 || Pgr(i,2) > dimRes
           fprintf('WARNING: row=%2.0f, col=2, index=%4.0f
               set =1\n',...i,Pgr(i,2));
           Pgr(i,3)=1;
      end
end
```

The ordered list of the dependent variables (the script results), of their numerical index, symbol, unit, and name, is provided in Table 9.4.

Table 9.4 Symbol, unit, and name of dependent variables.

No.	Script name	Unit	Name
1	DIC[mM]	mmol/kg-soln	Dissolved inorganic carbon
2	H2CO3[mM]	mmol/kg-soln	Carbonic acid
3	HCO3-[mM]	mmol/kg-soln	Bicarbonate ion
4	CO3—[mM]	mmol/kg-soln	Carbonate ion
5	InOutC[mM]	mmol/kg-soln	Intake or outtake carbon dioxide
6	Ca++[mM]	mmol/kg-soln	Calcium ion
7	CaCO_3[mM]	mmol/kg-soln	Calcium bicarbonate
8	OSatCalc	fraction	Oversaturated calcite
9	OSatArag	fraction	Oversaturated aragonite
10	Alkali[mM]	mmol/kg-soln	Alkalinity
11	pHFree	pH unit	pH free scale, $-\log_{10}[H^+]$
12	pOH	pH unit	Hydroxide scale, $-\log_{10}[OH^-]$
13	pHTot	pH unit	Total pH scale, $-\log_{10}([H^+] + [SO_4{}^-])$
14	pHSW	pH unit	Seawater pH scale, $-\log_{10}([H^+] + [SO_4{}^-] + [F^-])$

The variable names in the script have been arranged to be short, which asks for some explanation. `PrecCalc` in Table 9.3 corresponds to the precipitated calcite fraction (pptF, φ). In Table 9.4, `DIC[mM]`is the total DIC, `InOutC[mM]` is the concentration of carbon dioxide which is absorbed (positive) or outgassed (negative) by the sea water, and `OSatCalc` and `OsatArag` provide the fraction Ω of oversaturated calcite and aragonite. As already said, oversaturation corresponds to a value greater than one and dissolution into seawater to values smaller than one. WARNING: the unit $[\text{mmol/kg} - \text{soln}]$ has been abbreviated as [mM] in the variable names of Table 9.4, second column.

9.5.6 The sea water equilibria computation

The script parts are as follows.

1. *Initialization*: the Boolean variables `logDisp = 0;logInput = 0` can be raised to 1 by users. The first one triggers the `fzero` iteration figure within the loops along the steps where electroneutrality is searched for. The second one triggers the printout of input data in the script `SWInputDataPrintout`, called by the main script (see Appendix E). The relative humidity (`PH2O` in the script) is expressed as a fraction between 0 and 1 with respect to the saturation value. Usually, this value is set to 1, since, over the oceans, air is likely near to saturation in H_2O(vap), but it may be varied by users.

```
%% Sea water equilibria: deep sea water (Chapter 9)
% Pressure in MPa, last row of pressure coefficient dropped T^2P^2
%% 1) Initialization
disp('Sea Water equilibria: deep sea water');
disp('Initialization'); InitChem;
global x
T0=273.15; PH2O=1; % =100%
R = 8.314;
logDisp=0; % iteration graphic of the search for neutrality << user
logInput=0; %to printout excel data <<< user
```

2. *User data and independent variable range*. The names of independent and dependent variables are defined (see Tables 9.3 and 9.4). As already said, users are asked to specify the printout index `ipr` and the range matrix `RVar`. The section ends by constructing the variable matrix `Var` and the dimension vector `dimv`. The script of the user part is repeated for completeness.

```
%% 2) User data and independent variable range
disp('2) User data and independent variable range');
global Var
VarNames={'Temp[°C]  ','CO2V[ppmv] ','Pres[MPa] ','Sal[g/kg]      ','PrecCalc'};
ResNames={'DIC[mM] ','H2CO3[mM] ','HCO3-[mM] ','CO3--[mM] ','InOutCO2[mM] ',...
      'Ca++[mM] ', 'CaCO3[mM] ','OSatCalc ','OSatArag ','Alkali[mM] ',...
      'pHFree  ','pOH    ','pHTot ','pHSW'};
% --- User selection: printout variable and graphical range
ipr=1; % <<<< user Index of independent variable for printout
if ipr<1 || ipr>dimVar, ipr=1; end % default ind var =temp
fprintf('   Ind var to be printed  %8.1f  \n',ipr);
dimv=zeros(dimVar,1); % variable size
% Computing and graphical range of independent variables
% column 1 = min value, 2 = max value, 3 = step
% RVar=[ 0 40 2; % Temp degree centigrade <<< user
minStep=[2; 20; 5; 1 ; 0.01]; % WARNING: minStep > 0
for i=1:dimVar
      if minStep(i)<=0, minStep(i)=1; end
end
% SUGGESTION:  only two independent variables with max > min value
% The other three variables with min=max
RVar=[ 4   4    1; % Temp degree centigrade <<< user
      300 500 20; % Vol CO2 partial volume ppmv
      0   100  5; % Pressure MPa
      35  35   1; % Sal
      0. 0.0 0.01]; % pptF CaCO3 fraction that precipitates
for i=1: dimVar
      if RVar(i,2)<RVar(i,1), RVar(i,2)=RVar(i,1); end % max >= min
      if RVar(i,3)<minStep(i), RVar(i,3)=minStep(i); end % step>0
end
fprintf('   Ind. var    Min value  Max value      Step   \n');
for i=1:dimVar
      fprintf('  %8.1f    %8.1f   %8.1f   %8.1f  \n',...
           i, RVar(1,1),RVar(i,2),RVar(i,3));
end
% --- End of user part
for i=1:dimVar
      dimv(i)=floor((RVar(i,2)-RVar(i,1))/RVar(i,3))+1;
end
dimvMax=max(dimv);Var=zeros(dimVar,dimvMax);
for i=1:dimVar
      Var(i,1)=RVar(i,1);
      for k=2:dimv(i),  Var(i,k)=Var(i,k-1)+RVar(i,3);  end
end
```

3. *Reading the excel spreadsheet*. The excel spreadsheet `SeaWaterTable.xlsx`, organized into six sheets, is read and saved into appropriate Matlab tables. The first sheet (`sh5Name = 'StandardSal'`) to be read, includes concentration and charge of the *standard salinity*. The other tables include the coefficients of the dissociation constant expansions. Some expansions require a scale factor which is reported in the first column of the tables `sh1Name = 'Temperature'` and `sh6Name = 'Fugacity'`. The

pressure table includes as the last column the charge of the species listed in the species column. They are: `H^{+}`, `(OH)^{-}`, `H_2CO_3`, `HCO_3^{-}`, `CO_3^{2-}`, `HSO_4^{-}`, `CaCO_3(Cal)`, `CaCO_3(Arg)`, `B(OH)_3`, `HF`. Ten of the twelve dissociation constants to be computed refer to these species. `Cal` and `Arg` stand for calcite and aragonite. The last two constants are the fugacity and the water vapor pressure.

```
%% 3) Reading excel table
disp('3) Reading excel table');
sh1Name='Temperature';sh2Name='SalTemp';sh3Name='IonicTemp';
sh4Name='Pressure';
sh5Name='StandardSal'; sh6Name='Fugacity'; % sheet name
% Standard salinity table
disp('SeaWater: reading standard salinity');
SSTab=readtable('SeaWaterTable.xlsx','ReadVariableNames',true,...
     'ReadRowNames',true,'PreserveVariableNames',true,'Sheet',sh5Name);
[nrow,ncol]=size(SSTab);
symbolS=SSTab.Species; % species names
dimsS=length(symbolS);
% Temperature coeff. table
disp('SeaWater: reading temperature coefficients');
TTab=readtable('SeaWaterTable.xlsx','ReadVariableNames',true,...
     'ReadRowNames',true,'PreserveVariableNames',true,'Sheet',sh1Name);
[nrowT,ncolT]=size(TTab);
symbolT=TTab.Species;  % species names
dimsT=length(symbolT);
% Salinity coeff. table
STab=readtable('SeaWaterTable.xlsx','ReadVariableNames',true,...
     'ReadRowNames',true,'PreserveVariableNames',true,'Sheet',sh2Name);
[nrowS,ncolS]=size(STab);
% Pressure coefficient table
PTab=readtable('SeaWaterTable.xlsx','ReadVariableNames',true,...
      'ReadRowNames',true,'PreserveVariableNames',true,'Sheet',sh4Name);
[nrowP,ncolP]=size(PTab);
% Ionic strength coefficient table
ITab=readtable('SeaWaterTable.xlsx','ReadVariableNames',true,...
      'ReadRowNames',true,'PreserveVariableNames',true,'Sheet',sh3Name);
[nrowI,ncolI]=size(ITab);
% Fugacity and water pressure table
FTab=readtable('SeaWaterTable.xlsx','ReadVariableNames',true,...
      'ReadRowNames',true,'PreserveVariableNames',true,'Sheet',sh6Name);
[nrowF,ncolF]=size(FTab);
symbolF=FTab.Species; % species names
%% 4) Possible printout of input data
SWInputDataPrintout;
```

4. *Printout of input data*. See Appendix E.

5. *Initial concentration and charge vectors.*

The section builds up the initial concentration and charge of the species involved in the electro-neutrality equation, which, when brought to zero, decides the final concentration equilibrium.

```
%% 5) Initial concentration and  and charge vectors
disp('5) Initial concentration and charge vectors');
dims=dimsS+dimsT;
c=zeros(dims,2); %concentration and charge
c(1:dimsS,2)=SSTab{1:dimsS,2}; % charge
c(1:dimsS,1)=SSTab{1:dimsS,1}; %conc
c(dimsS+1:dims,2)=PTab{1:dimsT,ncolP}; %conc
x=c(:,1); % concentration vector
x0=x; % initial concentrations
```

Table 9.5 reports all the species involved. The left list, which consists of the first three columns, includes the species of the standard salinity with their initial concentration. The right list, the last two columns, reports the species whose initial concentration varies with the independent variables. As such, initial concentration values are not provided. The electric charge is provided for all the species.

Table 9.5 The species involved in seawater equilibria.

Species	Concentration [mol/kg-soln]	Charge	Species	Charge
Cl^{-}	0.5458600	−1.00	H^{ + }	1.00
Na^{ + }	0.4690600	1.00	(OH)^{-}	−1.00
K^{ + }	0.0102100	1.00	H_2CO_3	0.00
Sr^{2 + }	0.0000900	2.00	HCO_3^{-}	−1.00
Mg^{2 + }	0.0528200	2.00	CO_3^{2-}	−2.00
SO_4^{2-}	0.0282400	−2.00	HSO_4^{-}	−1.00
Ca^{2 + }	0.0102800	2.00	CaCO_3(Cal)	0.00
Br^{-}	0.0008400	−1.00	CaCO_3(Arg)	0.00
B(OH)_4^{-}	0.0004200	−1.00	B(OH)_3	0.00
F^{-}	0.0000700	−1.00	HF	0.00

6. *Electro-neutrality: loop on independent variables*

At the core of the five loops, 12 constants, collected in the vector `K`, are required for computing concentrations and results. They are listed in Table 9.6 (the table can be found after the script pages and the numeric item 7) with reference to their position in the vector `K`.

Before calling the zero-crossing function `fzero`, the concentration of H_2CO_3 is computed. The zero crossing function calls an anonymous function which in turn calls the function `SeaWNeutr2(p, K,c,dimsS,x0,i5)` in charge of computing electro-neutrality. The current neutrality is coded in the variable `fer`, to be brought to zero on the basis of concentrations and charges. The function is listed in Section 9.5.7. At the end, the six-dimensional ipermatrix of the results, `Res`, is computed.

```
%% 6) Neutrality: loop on independent variables
disp('6) Neutrality: loop on ind. variables');
% Result matrices
Res=zeros(dimv(1),dimv(2),dimv(3),dimv(4),dimv(5),14);
% Standrad salinity and pH range
S0=Var(4,1);pHmin=0;pHmax=14;p0=7;
dimK=dimsT+2; K=zeros(dimK,1); % number of constants
% loops
for i1=1:dimv(1)
      for i2=1:dimv(2)
      for i3=1:dimv(3)
      for i4=1:dimv(4)
      for i5=1:dimv(5)
           S=Var(4,i4);
           for i=1:dimsS
                c(i,1)=c(i,1)*S/S0;
           end
           % ind. variable preparation
           Tc=Var(1,i1); % centigrade
           Ta=Tc+T0; Tc2=Tc^2;
           V=Var(2,i2); P=Var(3,i3);
           rS=sqrt(S); S2=S^2; % salinity
           Ta2=Ta^2; lnTa=log(Ta); % absolute
           PT=P/(R1*Ta); P2T=P*PT; % pressure
           Sn=1-0.001005*S; lnSn=log(Sn);
           IS=0.019924*S/Sn; rIS=sqrt(IS);IS2=IS^2;
           fT=[1 1/Ta lnTa Ta ]; % temperature
           fS=[rS rS/Ta rS*lnTa rS*Ta rS*S rS*S/Ta ...
               S S/Ta S*lnTa S*Ta S*Ta2 S2 S2/Ta ]; % salinity
           fI=[rIS rIS*lnTa rIS/Ta IS IS*lnTa IS/Ta ...
               rIS*IS/Ta IS2/Ta lnSn]; %ionic strength
           fP=[PT Tc*PT Tc2*PT P2T Tc*P2T ]; %pressure
           fF=[1 1/Ta lnTa Ta Ta2 S];
           % diss. const. Kw KH2CO3 KHCO3 KCO3 KHSO4 KCal Karg KBOH3 KHF
           for k=2:dimsT  % loop on species
               coT=TTab{k,2:ncolT}; aT=coT*fT.';
               coS=STab{k,1:ncolS}; aS=coS*fS.';
               coI=ITab{k,1:ncolI}; aI=coI*fI.';
               coP=PTab{k,1:ncolP-1}; aP=coP*fP.'; %last col=charge
               scale=TTab{k,1};
               K(k)=exp(scale*(aT+aS)+aI+aP);
            end
            for k=1:2 % fugacity & vapor pressure
                 kk=dimsT+k;
                 coF=FTab{k,2:ncolF};
                 scale=FTab{k,1};
                 if k==1, scale=scale/R; end
                 K(kk)=exp(scale*coF*fF.');
```

```
            end
            scale=K(dimsT+1)*K(3)*(1-K(dimsT+2)*PH2O)/microScale;
            c(dimsS+3,1)=scale*V; %H2CO3 conc
            % search for neutrality
            p=p0;plim=[pHmin;pHmax];
            fer=@(p) SeaWNeutr2(p,K,c,dimsS,x0,i5);
            if i1==1 && logDisp==1   % <<< user
                  options =optimset('PlotFcns',@optimplotfval);
                  [p1,fer1,exitflag,output] = fzero(fer,plim,options);
            else
                  [p1,fer1,exitflag,output] = fzero(fer,plim);
            end
            % end of search
            % Results: calcium and  alkalinity
            d=dimsS;
            Res(i1,i2,i3,i4,i5,6)=x(7)*milliScale; %Ca
            Res(i1,i2,i3,i4,i5,7)=(x0(7)-x(7))*milliScale; %CaCO3
            Res(i1,i2,i3,i4,i5,8)=x(7)*x(d+5)/K(7); %cal
            Res(i1,i2,i3,i4,i5,9)=x(7)*x(d+5)/K(8); %arg
            Res(i1,i2,i3,i4,i5,10)=(2*x(d+5)+x(d+4)-...
                 x(d+1)+x(d+2)+x(9))*milliScale;
            % carbon
            Res(i1,i2,i3,i4,i5,1) = (x(d+3)+x(d+4)+x(d+5))*milliScale;
            Res(i1,i2,i3,i4,i5,2)=x(d+3)*milliScale; % H2CO3
            Res(i1,i2,i3,i4,i5,3)=x(d+4)*milliScale; % HCO3
            Res(i1,i2,i3,i4,i5,4)=x(d+5)*milliScale; % CO3
            dc=Res(i1,i2,i3,i4,i5,1)+Res(i1,i2,i3,i4,i5,7);
            if i2==1, Cin=dc; end
             ìRes(i1,i2,i3,i4,i5,5)=Cin-dc;
            % pH
            Res(i1,i2,i3,i4,i5,11)=p1; %pH
            Res(i1,i2,i3,i4,i5,12)=-log10(x(d+2)); %pOH
            Res(i1,i2,i3,i4,i5,13)=-log10(x(d+1)+x(d+6)); % p(H+HSO4)
            Res(i1,i2,i3,i4,i5,14)=-log10(x(d+1)+x(d+6)+x(d+10)); % p(H+HSO4+HF)
      end
      end
      end
      end
end
```

7. *Saving numerical results*

Results are saved by the following statements. They are reloaded by the graphical script in Section 9.5.8.

```
%% 7) Saving numerical results
disp('7) Saving numerical results')
save('SWResults.mat','Res');
save('SWIndVariables.mat','Var');
```

Table 9.6 The list of dissociation constants and other parameters to be used in seawater equilibria.

No.	Symbol	Name
1	None	void
2	K_w	Dissociation constant, $H_2O \Leftrightarrow H^+ + OH^-$
3	K_{0H2CO3}	Dissociation constant, $CO_2 + H_2O \Leftrightarrow H_2CO_3$
4	K_{H2CO3}	Dissociation constant, $H_2CO_3 \Leftrightarrow H^+ + HCO_3^-$
5	K_{HCO3}	Dissociation constant, $HCO_3^- \Leftrightarrow H^+ + CO_3^{--}$
6	K_{HSO4}	Dissociation constant, $HSO_4^- \Leftrightarrow H^+ + SO_4^{--}$
7	K_{Cal}	Dissociation constant, $CaCO_3$ calcite $\Leftrightarrow Ca^{++} + CO_3^{--}$
8	K_{arg}	Dissociation constant, $CaCO_3$ aragonite $\Leftrightarrow Ca^{++} + CO_3^{--}$
9	K_{BOH3}	Dissociation constant, $B(OH)_3 + H_2O \Leftrightarrow H^+ + B(OH)_4^-$
10	K_{HF}	Dissociation constant, $HF \Leftrightarrow H^+ + F^-$
11	Fugacity	
12	Vapor pressure	

8. *Printing numerical results*

Extraction of the dimension `ipr` from the ipermatrix `Res` is done with the help of the Matlab function `permute(A,dimorder)`, which rearranges the matrix `A` dimensions according to the dimension order in the vector `dimorder`. The key statements are the following.

```
indPer=[ipr indVar 6];
% matrix permutation to set selected variable as first dimension
Perm=permute(Res,indPer);
Mpr=squeeze(Perm(:,1,1,1,1,:));
sMpr=size(Mpr); if sMpr(2)==1; Mpr=Mpr.'; end % squeeze= column vector
```

The vector `indper`, which corresponds to `dimorder`, contains `ipr` as the first dimension, the remaining four dimensions of the independent variables are in `indVar`, the dimension of the single variable containing the result remains the last one. From the permuted matrix `Perm`, only the entries of the first and last dimension are extracted by `Perm(:,1,1,1,1,:)`, and the resulting matrix is squeezed to become a two-dimensional matrix by the function `squeeze`. The reader must be warned that `squeeze` tends to produce a column vector when one of the dimensions has only one entry. Since the dimension of the result variable must be the second one, the last statement does the task.

```
%% 8) Printing numerical results
disp('8) Printing numerical results');
% preparation
s=''; indVar=[]; sizeVar=[];
for i=1:dimVar
      if ne(i,ipr)==1
           s=strcat(s,VarNames(i)); indVar=[indVar i];
      end
end
indPer=[ipr indVar 6];
% matrix permutation to set selected variable as first dimension
Perm=permute(Res,indPer);
Mpr=squeeze(Perm(:,1,1,1,1,:));
sMpr=size(Mpr); if sMpr(2)==1; Mpr=Mpr.'; end  % squeeze= column vector
fprintf('%s PH2O ',cell2mat(s));
fprintf('\n %8.1f   %8.1f   %8.1f   %8.1f   %8.1f  ',
Var(indVar,1),PH2O);
% Carbonatic
fprintf('\n%s %s %s %s %s %s ',...
cell2mat(VarNames(ipr)), cell2mat(ResNames(1:5)));
for i=1:dimv(ipr)
      fprintf('\n  %8.1f %11.3e  %11.3e  %11.3e  %11.3e  %11.3e',
       ... Var(ipr,i),Mpr(i,1:5));
end
% Calcium
fprintf('\n%s  %s %s %s %s %s',...
      cell2mat(VarNames(ipr)), cell2mat(ResNames(6:10)));
for i=1:dimv(ipr)
      fprintf('\n  %8.1f %11.3e  %11.3e  %11.3e  %11.3e  %11.3e',
       ... Var(ipr,i),Mpr(i,6:10));
end
% pH
fprintf('\n%s %s %s %s %s',...
      cell2mat(VarNames(ipr)), cell2mat(ResNames(11:14)));
for i=1:dimv(ipr)
      fprintf('\n  %8.1f %11.3e  %11.3e  %11.3e  %11.3e', ...
           Var(ipr,i),Mpr(i,11:14));
end
fprintf('\n');
```

The example of printed results, which is provided in Section 9.5.9 for `ipr = 2`, corresponds to the CO_2 partial volume.

9.5.7 The electro-neutrality function

The electro-neutrality function `SeaWNeutr2` computes the current neutrality `fer` from the current pH coded in the variable `p`, the dissociation constants `K`, the charge vector `c(:,2)`, the dimension `dims = 10` of the standard salinity vector, the initial concentration `x0`, and the index `iv` of the `ppfT` variable. The current concentration `x`, which drives the result computation, is a global variable.

```
%% Function SeaWNeutr2
function fer = SeaWNeutr2(p,K,c,dims,x0,iv)
      global x Var
      pptF=Var(5,iv); % fraction of precipitated calcite
      x=c(:,1);
      % concentrations
      x(dims+1) = 10^(-p); H=x(dims+1); % H
      % 1) carbonatic
      x(dims+4) =K(4)*x(dims+3)/H; %HCO3= KH2CO3*H2CO3/H
      x(dims+5)=K(5)*x(dims+4)/H; %CO3 = KHCO3*HCO3/H
      % 2) HSO4
      x(dims+6)=x(6)*H/(H+K(6)); %HSO4 = H*SO4/(KHSO4 + H)
      x(6)=x0(6)-x(dims+6); %SO4- corrected
      % 3) Calcium: solubility check for CaCO3
      x(7)=x0(7); dSat=x0(7)*x(dims+5)-K(7); % K(7)=KCal
      if dSat>0, x(7)=x0(7)-dSat*pptF/x(dims+5); end
      % 4) BOH4-, F-
      x(9)= x0(9)*K(9)/(H+x(dims+6)+K(9)); %BOH4 = KBOH3*B/(H+HSO4+KBOH3)
      x(dims+10)=x(10)*H/(K(10)+H); % HF = H*F/(KHF + H;
      x(10)=x0(10)+x(dims+10); % F- corrected
      % 5) (OH)-
      x(dims+2)=K(2)/(H+x(dims+10)+x(dims+6)); % Kw/( H + HF + HSO4)
      % neutrality
      fer=x.'*c(:,2);
end
```

9.5.8 The graphical script

The graphical script is separated by the main script, but it may be added to the end of the latter. The binary Matlab files `SWResults.mat` and `SWIndVariables.mat` are loaded, and their content converted into the ipermatrices `Res` and `Var`, by the following statements.

```
Res=load('SWResults.mat');
Res=cell2mat(struct2cell(Res)); sRes=size(Res);
Var=load('SWIndVariables.mat');
Var=cell2mat(struct2cell(Var)); dimv=size(Var);
```

The two-step conversion is due to the `structured array` where data are loaded. Firstly, the array is converted into a `cell array` by `struct2cell` and then into a `double valued matrix` by `cell2mat`. The second issue is to extract, from the multidimensional matrix `Res`, the desired

independent variables, a pair for surface plots and a single variable for 2D plots. This is done with the help of the Matlab functions `permute` and `squeeze`, as already explained at the end of Section 9.5.6. The pair of matrices, where users select the graphical variables, have been explained in Section 9.5.5.

The script is organized into the following parts.

1. *Initialization*.

```
%% Sea water equilibria: deep sea water, graph (Chapter 9)
%% 1) Initialization
disp('Sea Water equilibria, deep sea water: graphical part');
disp('1) Initialization');
InitChem; PH2O=1;
```

2. *Variable loading and selection*. The reader should refer to Section 9.5.5. The user section has been repeated for completeness.

```
%% 2) Variable loading and selection
disp('2) Loading variables and selection');
VarNames={'1-Temp[°C]   ','2-CO_2V[ppmv] ','3-Pres[kPa] ',
      '4-Sal[g/kg] ',...
      '5-LostCalc'};
dimVar=length(VarNames);
ResNames={'1-DIC[mM]','2-H_2CO_3[mM]','3-HCO_3-[mM] ',
      '4-CO_3--[mM]',...
      '5-LostCO_2[mM]',...
      '6-Ca++[mM]', '7-CaCO_3[mM]','8-SatCalc','9-SatArag',
      '10-Alkali[mM]',...
      '11-pH    ','12-pOH   ','13-pH(+SO_4)','14-pH(+SO_4+F)'};
dimRes=length(ResNames);
Res=load('SWResults.mat');
Res=cell2mat(struct2cell(Res)); sRes=size(Res);
Var=load('SWIndVariables.mat');
Var=cell2mat(struct2cell(Var)); dimv=size(Var);
% --- user selection: graphical variable
% tow matrices: Sgr for surface plot , Pgr for 2D plot
% Sgr: ind. variables (Var) column 1: surface x, column 2 surface y
% Sgr: dependent variable (Res), column 3 : surface z
% WARNING: if an independent variable has been computed
%           with min=max values in the main script,
%           'Insufficient dimensions' will be printed out
% More than one surface can be plotted
% WARNING: wrong variable indices are set to 1 or added by 1
% WARNING: 2nd equal ind. var. index is added by 1
Sgr=[ 3 2 8;
      5 4 5];
%     1 2 10;
%     1 2 11]; % <<<< user surface [IndVar1 IndVar2 Res]
[rowS,colS]=size(Sgr);
% check
fprintf('Sgr Check\n');
```

```
for i=1: rowS
      if Sgr(i,1)<0 || Sgr(i,1) > dimVar
           fprintf('WARNING: row=%2.0f, col=1, index=%4.0f set =1\n',...
                i,Sgr(i,1));
           Sgr(i,1)=1;
      end
      if Sgr(i,2)<0 || Sgr(i,2) > dimVar
           fprintf('WARNING: row=%2.0f, col=2, index=%4.0f set=Sgr(i,1)+1\n',...
                i,Sgr(i,2));
           Sgr(i,2)=mod(Sgr(i,1),dimVar)+1;
      end
      if Sgr(i,3)<0 || Sgr(i,3) > dimRes
            fprintf('WARNING: row=%2.0f, col=3, index=%4.0f set =1\n',...
                  i,Sgr(i,3));
            Sgr(i,3)=1;
      end
      if Sgr(i,1)==Sgr(i,2)
            fprintf('WARNING:        row=%2.0f,        col=1,        index=%4.0f
set=Sgr(i,1)+1\n',...
                  i,Sgr(i,1));
            Sgr(i,2)=mod(Sgr(i,1),dimVar)+1;
      end
end
% Pgr: ind. variable (Var)  column 1: graph x,
% Pgr: dependent variable (Res), column 3 : graph y
% WARNING: if the independent variable has been computed
%           with min=max values in the main script,
%           'Insufficient dimensions' will be printed out
% More than one 2D graph can be plotted
% WARNING: wrong variable indices are set to 1
Pgr=  [3  1;
      -2  8]; % <<< user 2D plot  [IndVar1 Res]
[rowP,colP]=size(Pgr);
% check
fprintf('Pgr Check\n');
for i=1: rowP
      if Pgr(i,1)<0 || Pgr(i,1) > dimVar
           fprintf('WARNING: row=%2.0f, col=1, index=%4.0f set =1\n',...
                i,Pgr(i,1));
           Pgr(i,1)=1;
      end
      if Pgr(i,2)<0 || Pgr(i,2) > dimRes
            fprintf('WARNING: row=%2.0f, col=2, index=%4.0f set =1\n',...
                 i,Pgr(i,2));
            Pgr(i,3)=1;
      end
end
```

3. *Surface plot.*

The statements for the graphical plot of surfaces are as follows.

```
%% 3) Surface plot
disp('3) Plotting Surfaces')
for i=1:rowS
      vx=Sgr(i,1);  vy=Sgr(i,2); vz=Sgr(i,3);
      dimx=sRes(vx);    dimy=sRes(vy);
      fprintf('\n Surface No. %3.0f x=%3.0f y=%3.0f z=%3.0f',
      i,vx,vy,vz);
      Var2D=zeros(dimx,dimy,2);
      for ix=1:dimx
      for iy=1:dimy
           Var2D(ix,iy,1) = Var(vx,ix); Var2D(ix,iy,2) = Var(vy,iy);
      end
      end
      % result permutation
      indVar=zeros(1,dimVar+1); indVar(1,1:2)=[ vx vy ]; k=0;
      for iv=1:dimVar
           if ne(iv,vx)==1 && ne(iv,vy)==1
                k=k+1; indVar(1,1:2+k)=[indVar(1,1:2+k-1) iv];
           end
      end
      indVar(1,1:dimVar+1)=[indVar(1,1:dimVar) 6];
      Perm=permute(Res,indVar);
      % matrix permut. to set first selected variable
      Mpr=squeeze(Perm(:,:,1,1,1,:));
      sMpr=size(Mpr); if sMpr(2)==1; Mpr=Mpr.'; end
      % squeeze= column vector
      if dimx>1 && dimy>1
          figure('name', 'Surface');
          surf(Var2D(:,:,1),Var2D(:,:,2),Mpr(:,:,vz));
          st1=strcat(ResNames{vz},' vs. ',VarNames{vx},' and
          ',VarNames{vy});
          st2=strcat(VarNames{indVar(3)},'=',
          num2str(Var(indVar(3),1)));
          st2=strcat(st2,',',VarNames{indVar(4)},'=',
          num2str(Var(indVar(4),1)));
          st2=strcat(st2,',',VarNames{indVar(5)},'=',
          num2str(Var(indVar(5),1)));
          st2=strcat(st2,', PH2O =',num2str(PH2O));
          title({st1;st2},'Interpreter','tex','FontWeight',
          'normal',...'FontSize',14);
           view([45 20]); %set viewpoint direction
           xlabel(VarNames{vx},'Interpreter','tex');
           ylabel(VarNames{vy},'Interpreter','tex');
           zlabel(ResNames{vz},'Interpreter','tex');
      else
            fprintf(' Insufficient dimensions\n');
      end
end
fprintf('\n');
```

4. *2D graphical representation.*

The statements for the 2D graphical plot are as follows.

```
%% 4) 2D graphical representation
disp('4) 2D graphical representation')
for i=1:rowP
      vx=Pgr(i,1); vy=Pgr(i,2); dimx=sRes(vx); dimy=dimx;
      fprintf('\n 2D plot No. %3.0f x=%3.0f y=%3.0f ',i,vx,vy);
      % Result permutation
      indVar=zeros(1,dimVar+1); indVar(1,1)= vx ; k=0;
      for iv=1:dimVar
           if ne(iv,vx)==1
                k=k+1; indVar(1,1:1+k)=[indVar(1,1:1+k-1) iv];
           end
      end
      indVar(1,1:dimVar+1)=[indVar(1,1:dimVar) 6];
      Perm=permute(Res,indVar);
      % matrix perm. to set first selected variable
      Mpr=squeeze(Perm(:,1,1,1,1,:));
      sMpr=size(Mpr); if sMpr(2)==1; Mpr=Mpr.'; end
      % squeeze= column vector
      if dimx>1
           figure('name', '2D plot');
           plot(Var(vx,1:dimx),Mpr(1:dimx,vy));
           st1=strcat(ResNames{vy},' vs. ',VarNames{vx});
           st2=strcat(VarNames{indVar(2)},'=',num2str(Var(indVar(2),1)));
           st2=strcat(st2,VarNames{indVar(3)},
           '=',num2str(Var(indVar(3),1)));
          st2=strcat(st2,',',VarNames{indVar(4)},
          '=',num2str(Var(indVar(4),1)));
           st3=strcat(VarNames{indVar(5)},'=',num2str(Var(indVar(5),1)));
           st3=strcat(st3,', P(H_2O) =',num2str(PH2O));
           title({st1;st2;st3},'Interpreter','tex','FontWeight',
           'normal',...'FontSize',14);
           xlabel(VarNames{vx},'Interpreter','tex');
           ylabel(ResNames{vy},'Interpreter','tex');
      else
           fprintf(' Insufficient dimensions\n');
      end
end
fprintf('\n');
```

9.5.9 Typical numerical and graphical results

Typical numerical results of the first script in Section 9.5.6, without the printout of input data (see Appendix E, `logInput = 0`) are as follows. Let us recall the unit equality 1 mM = 1 mmol/kg − soln.

```
8) Printing numerical results
Temp[°C]      Pres[MPa]     Sal[g/kg]    PrecCalc    PH2O
0.0             0.0          35.0         0.0         1.0
CO2V[ppmv] DIC[mM]    H2CO3[mM]     HCO3-[mM]     CO3--[mM]     InOutCO2[mM]
300.0   2.225e+00    1.867e-02    2.079e+00    1.265e-01    0.000e+00
320.0   2.235e+00    1.991e-02    2.095e+00    1.204e-01   -1.057e-02
340.0   2.245e+00    2.116e-02    2.109e+00    1.148e-01   -2.027e-02
360.0   2.254e+00    2.240e-02    2.122e+00    1.098e-01   -2.922e-02
380.0   2.262e+00    2.364e-02    2.133e+00    1.051e-01   -3.751e-02
400.0   2.270e+00    2.489e-02    2.144e+00    1.009e-01   -4.524e-02
420.0   2.277e+00    2.613e-02    2.154e+00    9.697e-02   -5.245e-02
440.0   2.284e+00    2.738e-02    2.163e+00    9.335e-02   -5.922e-02
460.0   2.290e+00    2.862e-02    2.172e+00    9.000e-02   -6.559e-02
480.0   2.296e+00    2.987e-02    2.180e+00    8.687e-02   -7.161e-02
500.0   2.302e+00    3.111e-02    2.187e+00    8.396e-02   -7.730e-02
CO2V[ppmv] Ca++[mM]  CaCO3[mM]    OSatCalc      OSatArag    Alkali[mM]
300.0   1.028e+01    0.000e+00    3.032e+00    1.905e+00    2.400e+00
320.0   1.028e+01    0.000e+00    2.885e+00    1.812e+00    2.400e+00
340.0   1.028e+01    0.000e+00    2.751e+00    1.729e+00    2.400e+00
360.0   1.028e+01    0.000e+00    2.630e+00    1.652e+00    2.400e+00
380.0   1.028e+01    0.000e+00    2.519e+00    1.583e+00    2.400e+00
400.0   1.028e+01    0.000e+00    2.417e+00    1.519e+00    2.400e+00
420.0   1.028e+01    0.000e+00    2.323e+00    1.460e+00    2.400e+00
440.0   1.028e+01    0.000e+00    2.237e+00    1.405e+00    2.400e+00
460.0   1.028e+01    0.000e+00    2.156e+00    1.355e+00    2.400e+00
480.0   1.028e+01    0.000e+00    2.081e+00    1.308e+00    2.400e+00
500.0   1.028e+01    0.000e+00    2.012e+00    1.264e+00    2.400e+00
CO2V[ppmv] pHFree        pOH          pHTot        pHSW
300.0   8.220e+00    6.118e+00    8.189e+00    8.182e+00
320.0   8.196e+00    6.143e+00    8.164e+00    8.157e+00
340.0   8.172e+00    6.166e+00    8.141e+00    8.134e+00
360.0   8.150e+00    6.188e+00    8.119e+00    8.112e+00
380.0   8.129e+00    6.209e+00    8.098e+00    8.090e+00
400.0   8.109e+00    6.229e+00    8.078e+00    8.070e+00
420.0   8.089e+00    6.249e+00    8.058e+00    8.051e+00
440.0   8.071e+00    6.267e+00    8.040e+00    8.033e+00
460.0   8.054e+00    6.285e+00    8.022e+00    8.015e+00
480.0   8.037e+00    6.301e+00    8.006e+00    7.998e+00
500.0   8.020e+00    6.318e+00    7.989e+00    7.982e+00
```

Fig. 9.5, right, shows a typical surface plot, in this case the DIC surface versus temperature and CO_2 partial volume. The surface section at the minimum CO_2 partial volume is shown on the left. Let us recall that 1 mM = 1 mmol/kg − soln.

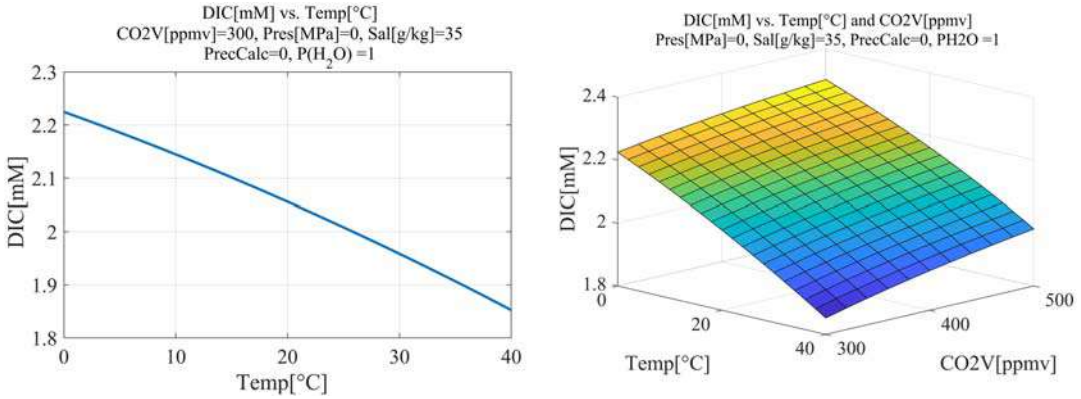


FIGURE 9.5

(Left) Section of the right surface at minimum CO_2 volume. (Right) DIC surface versus temperature and CO_2 partial volume. *DIC*, dissolved inorganic carbon.

9.6 Phosphate chemistry in seawater

9.6.1 Description and graphical results

Phosphate concentration in seawater locally varies to a great extent, which suggests to separately treat the relevant equilibrium. In this case, equilibrium concentrations are just obtained by solving the linear system of Equations (7.10) in Section 7.3.1 about pure water solutions, in the case of typical triprotic acids like H_3PO_4. The equations are repeated here:

$$\begin{bmatrix} A_{tot} \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ K_3 & -H & 0 & 0 \\ 0 & K_4 & -H & 0 \\ 0 & 0 & K_5 & -H \end{bmatrix} \begin{bmatrix} x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix}$$

$$x_1 = [H^+] = 10^{-pH},\ x_2 = [OH^-] = K_2/[H^+]$$

$$x_3 = [H_3PO_4],\ x_4 = [H_2PO_4^-],\ x_5 = [HPO_4^{2-}],\ x_6 = [PO_4^{3-}]$$

Because we are now dealing with seawater, the dissociation constants depend, as in Sections 9.4 and 9.5, on the salinity S and on the absolute temperature T. The total phosphate concentration A_{tot} drives the equilibrium equations. The relevant coefficients are taken from the literature [11] and have been collected in the excel spreadsheet already used in Sections 9.4 and 9.5 and specifically in the sheet `Phosphate`. The relevant coefficients are reported in the next table, where each column refers to a specific dissociation constant $K_i, i = 2, 3, 4, 5$, where $K_2 = K_w$. K_1 is unused.

```
3) Reading and printing coefficients, temperature and salinity
Variable          (OH)^{-}    H_3PO_4     H_2PO_4^{-} HPO_4^{2-}
Constant       1.4898e+02  1.1553e+02  1.7209e+02 -1.8141e+01
1/T           -1.3847e+04 -4.5768e+03 -8.8147e+03 -3.0708e+03
log(T)        -2.3652e+01 -1.8453e+01 -2.7927e+01          0.0
sqrt(S)       -5.9770e+00  6.9171e-01  1.3566e+00  2.8120e+00
sqrt(S)/T      1.1867e+02 -1.0674e+02 -1.6034e+02  1.7270e+01
sqrt(S)log(T)  1.0495e+00          0.0          0.0          0.0
S             -1.6150e-02 -1.8440e-02 -5.7780e-02 -9.9840e-02
S/T                    0.0 -6.5643e-01  3.7335e-01 -4.4995e+01
```

The concentration of phosphoric acid and the dissociation products are computed in a range of the seawater pH total scale. The pH values are determined in turn by the seawater composition, which affects the phosphate dissociation equilibria through pH, temperature, and salinity. Being the total seawater composition unknown, the calculation of electric neutrality is not pertinent to the script.

The script in Section 9.6.2 ends by printing numerical results (not reported here) and the graphical concentration in $[\text{mol/kg} - \text{soln}]$ of the three phosphate ions, $H_2PO_4^-$ (scaled 10 times), HPO_4^{2-}, and PO_4^{3-} (see Fig. 9.6).

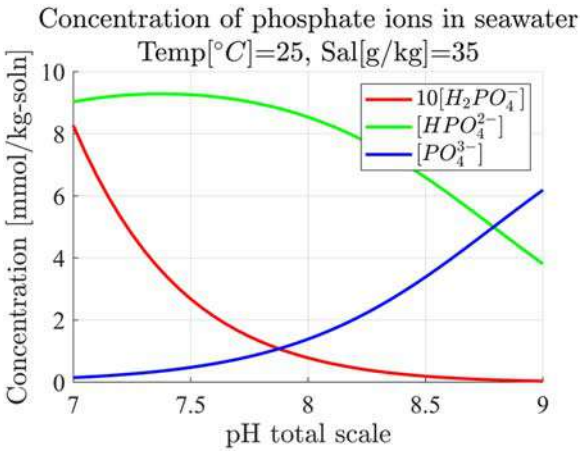


FIGURE 9.6

Concentration of phosphate ions in seawater.

9.6.2 Matlab script

The script follows the organization of the scripts in Sections 9.4 and 9.5.

```
%% Phosphate in seawater (Chapter 9)
%% 1) Initialization
InitChem; Temp0=273.15;
%% 2) User data
disp('Phosphate concentration in seawater');
%  H3PO4 <==> H+  +  H2PO4-,  Ka1 = [H+]*[H2PO4-]/[H3PO4]
%  H2PO4- <==> H+  +  HPO4--, Ka2 = [H+]*[HPO4--]/[H2PO4-]
%  HPO4-- <==> H+  +  PO4---,  Ka3 = [H+]*[PO4---]/[HPO4--]
% ---- user data
Atot = 0.01; % phosphate total conc. (H3PO4 + H2PO4- +  HPO4-- + PO4---)
Tc = 25; Ta = Tc + Temp0; S = 35; % Temp. and Salinity
pHmin=7 ; pHmax=9; dpH=0.05; % range pH = pH(total)
pH=pHmin:dpH:pHmax; dimpH=length(pH);
dimK=5; % dissociation constant vector
% --- end of user data
%% 3) Reading coefficients
disp('3) Reading and printing coefficients, temperature and salinity');
TTab=readtable('SeaWaterTable.xlsx','ReadVariableNames',true,...
      'ReadRowNames',true,'PreserveVariableNames',true,'Sheet',
      'Phosphate');
[nrowT,ncolT]=size(TTab);
sprint=TTab.Species;  % species names
fprintf('Variable');
for i=2:dimK, fprintf('%12s', cell2mat(sprint(i))); end
for k=1:ncolT
      vprint=TTab.Properties.VariableNames(k);
      fprintf('\n%14s ',cell2mat(vprint));
      for i=2:dimK
           if  TTab{i,k}==0 || abs(TTab{i,k})==1, fprintf('%12.1f',
           TTab{i,k});
           else fprintf('%12.4e', TTab{i,k}); end
      end
end
fprintf('\n');
%% 4) Sea water ionic concentration: loop on pH total scale
disp('4) Ionic concentration: loop on pH total scale')
c=zeros(dimpH,nrowT); K=zeros(dimK,1);
% concentration and diss. constants
lnTa=log(Ta);rS=sqrt(S);
fTS=[1 1/Ta lnTa rS rS/Ta rS*lnTa S S/Ta ];
% temperature and  salinity
for i=2:dimK, coTS=TTab{i,1:ncolT};  K(i)=exp(coTS*fTS.'); end
for i =1: dimpH
      H = 10.^(-1*pH(i));  c(i,1)=H; c(i,2) = K(2)/H; % H+ and OH-
      A = [1  1  1  1;-K(3) H 0 0 ;0 -K(4) H 0; 0 0 -K(5) H];
      b =[Atot; 0; 0;0];
      c(i,3:nrowT)= A\b; % H3PO4, H2PO4, HPO4--, PO4---
end
```

```
%% 5) Numerical and graphical results
disp('5) Numerical and graphical results');
fprintf('\npH total %11s  %11s %11s %11s %11s %11s',...
      sprint{1:6});
for i=1:dimpH
      fprintf('\n  %8.1f %10.3e  %10.3e  %10.3e  %10.3e
      %10.3e %10.3e', ...
            pH(i),c(i,1:6));
end
fprintf('\n');
figure('name','SW phosphate'); hold on; grid on;
scale=ones(nrowT,1);
scale(4)=scale(4)*10;
for i=4:nrowT
      plot(pH,c(:,i)*milliScale*scale(i),color(i-3)); grid on;
end
xlabel('pH total scale'); ylabel('Concentration [mmol/kg-soln]')
lgd={nrowT,1}; %legend
for i=4:nrowT, lgd(i)=strcat('$[',sprint(i),']$'); end
lgd{4}=strcat('10',lgd{4});legend(lgd{4},lgd{5},lgd{6});
% title
st=strcat('Temp$[^\circ C]$=',num2str(Tc),',','
Sal[g/kg]=',num2str(S) );
title({'Concentration of phosphate ions in seawater'; st},...
      'FontWeight','normal','FontSize',18);
```

9.7 Density of seawater versus salinity, temperature, and pressure

9.7.1 Description

In this section, the seawater density $\rho_{sw}\left[\mathrm{kg/m^3}\right]$ is computed as a function of temperature T_c [°C], pressure P[MPa], and salinity $S\left[\mathrm{g/kg}\right]$, according to the following expression [3,16]

$$\rho_{sw}(T_c,S,P)=\frac{\rho_{sw}(T_c,S,0)}{\left(1-P/K(T_c,S,P)\right)}, \tag{9.12}$$

where $\rho_{sw}(T_c,S,0)$ is the seawater density at zero hydrostatic pressure (sea surface), which is a function of temperature and salinity. $P/K(T_c,S,P)$ is the correction for nonzero hydrostatic pressure P. The surface seawater density is the pure water density $\rho_w(T_c)$ as a function of the temperature alone, but corrected by the salinity contribution as follows

$$\rho_{sw}(T_c,S,0)=\rho_{pw}(T_c,5)+\left(A(T_c,4)+B(T_c,2)\sqrt{S}+CS\right)S. \tag{9.13}$$

All the temperature functions in Eq. (9.13) are polynomials of the degree included in the round brackets after T_c. The values of the relevant coefficients are reported in a table below. The pressure correction $K(T_c, S, P)$ has the expression

$$K(T_c,S,P) = K_1(T_c,4) + \left(K_2(T_c,3) + K_3(T_c,2)\sqrt{S}\right)S + \\ + P\left(K_4(T_c,3) + K_4(T_c,2)S + K_6 S\sqrt{S}\right) + P^2(K_7(T_c,2) + K_8(T_c,2)S)\,, \tag{9.14}$$

where again the functions of temperature are polynomials and their degree is included in the round brackets.

The Matlab script reads the expansion coefficients of the pure water (functions of temperature [°C]) and of the sea water correction (function of temperature, salinity $[g/kg]$ and hydrostatic pressure [MPa]) from the sheet `SWDensity` of the excel spreadsheet excel spreadsheet mentioned in Sections 9.4 and 9.5. The coefficients values are reported in the next table (`SWP` means seawater pressure).

3) Reading and printing coefficients, temperature and salinity						
Multiplier	Constant	Tc	Tc^2	Tc^3	Tc^4	Tc^5
PureWater	9.998e+02	6.794e-02	-9.095e-03	1.002e-04	1.120e-06	6.536e-09
SeaWater,P=0,S	8.245e-01	-4.090e-03	7.644e-05	-8.247e-07	5.388e-09	0.0
SWP=0,sqrt(S)S	-5.722e-03	1.023e-04	-1.655e-06	0.0	0.0	0.0
SWP=0,S^2	4.831e-04	0.0	0.0	0.0	0.0	0.0
SeaWater,P>0,	11.965e+03	1.484e+01	-2.327e-01	1.360e-03	-5.155e-06	0.0
SWP>0,S	5.467e+00	-6.035e-02	1.100e-03	-6.167e-06	0.0	0.0
SWP>0,sqrt(S)S	7.944e-03	1.648e-03	-5.301e-05	0.0	0.0	0.0
SWP>0,P	3.240e+00	1.437e-03	1.161e-04	-7.779e-07	0.0	0.0
SWP>0,PS	2.284e-03	-1.098e-05	-1.608e-06	0.0	0.0	0.0
SWP>0,Psqrt(S)S	1.911e-04	0.0	0.0	0.0	0.0	0.0
SWP>0,P^2	8.509e-04	-6.123e-05	5.279e-07	0.0	0.0	0.0
SWP>0,P^2S	-9.935e-06	2.082e-07	9.170e-09	0.0	0.0	0.0

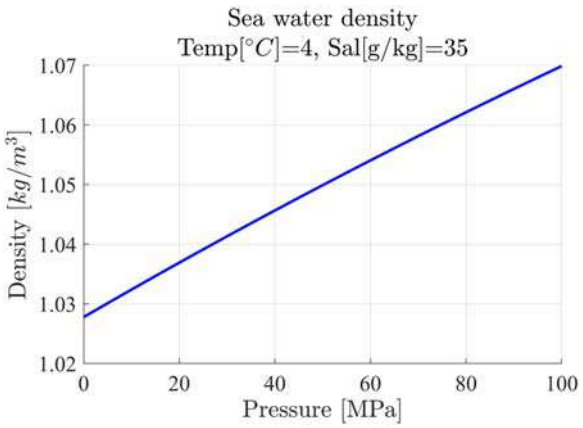


FIGURE 9.7

Sea water density versus hydrostatic pressure.

Numerical results are not reported here. Only the graphical plot of the sea water density in kg/m^3 units is shown in Fig. 9.7.

9.7.2 Matlab script

The Matlab script is organized like the scripts of Sections 9.4 to 9.6.

```
%% Density of seawater, according to temperature and
   salinity and pressure
disp('Density of seawater');
InitChem;
%% 2) User data
Tc=4; % temperature, degree Celsius <it can be varied>
S = 35; % salinity, g/(kg-soln) <it can be varied>
Pmin=0; Pmax=100; dP=5; % pressure range MPa
P=Pmin:dP:Pmax; dimP=length(P);
%% 3) Reading coefficients
disp('3) Reading and printing coefficients, temperature
and salinity');
TTab=readtable('SeaWaterTable.xlsx','ReadVariableNames',true,...
      'ReadRowNames',true,'PreserveVariableNames',true,'Sheet',
      'SWDensity');
[nrowT,ncolT]=size(TTab);
sprint=TTab.Species; % species names
vprint=TTab.Properties.VariableNames;
fprintf('Multiplier  ');
for i=1:ncolT, fprintf('%10s', cell2mat(vprint(i))); end
for k=1:nrowT
      fprintf(' \n%13s ',cell2mat(sprint(k)));
      for i=1:ncolT
            if TTab{k,i}==0 || abs(TTab{k,i})==1,
                  fprintf('%10.1f', TTab{k,i});
            else fprintf('%10.3e', TTab{k,i}); end
      end
```

```
end
fprintf('\n');
%% 4) Density computation
disp(' 4)Density computation');
fT=[1 Tc Tc^2 Tc^3 Tc^4 Tc^5 ]; % temperature
PWd=0; rS=sqrt(S); dimPW=4;
PWmult=[1; S; rS*S; S^2];  % Pure Water
for i=1:dimPW, co=PWmult(i)*TTab{i,1:ncolT}; PWd=PWd+co*fT.'; end
% Sea Water
for i=1: dimP
     p=P(i);d=0; SWmult=[1 ; S; rS*S; p; p*S; p*rS; p^2; p^2*S];
     for k=dimPW+1:nrowT
          co=SWmult(k-dimPW)*TTab{k,1:ncolT}; d=d+co*fT.';
     end
     SWd(i)=PWd/(1-p/d);
end
%% 5) Numerical and graphical results
disp('5) Numerical and graphical results');
fprintf('\n Sea water density  Temp. %8.0f °C,
Sal = %8.0f g/kg ',Tc,S);
fprintf('\nPressure [MPa]  SW density kg/dm^3]');
for i=1:dimP, fprintf('\n  %10.1f %14.3e ', P(i),SWd(i)); end
fprintf('\n');
figure('name','SW density'); hold on; grid on;
plot(P,SWd/milliScale,'k');
xlabel('Pressure [MPa]'); ylabel('Density $[kg/m^3]$')
st=strcat('Temp$[^\circ C]$=',num2str(Tc),',',
Sal[g/kg]=',num2str(S) );
title({'Sea water density',st},'FontWeight','normal','FontSize',18);
```

References

[1] Companion Website of the Book. Elsevier, 2021. Available from: https://www.elsevier.com/books-and-journals/book-companion/9780323913416.

[2] D. Mazza, *Algorithms in ocean chemistry*, Youcanprint, Lecce (Italy), 2019. https://polito.academia.edu/DanieleMazza/Books.

[3] A.E. Gill, Atmosphere-Ocean Dynamics, Academic Press, San Diego, 1982.

[4] R.G. Williams, M.J. Follows, Ocean Dynamics and the Carbon Cycle: Principles and Mechanisms, Cambridge University Press, 2011.

[5] R.E. Zeebe, D. Wolf-Gladrow, *CO_2 in seawater: equilibrium, kinetics, isotopes*. Elsevier Oceanography Series, 2005, Amsterdam.

[6] D. Easterbrook (Ed.), Evidence-Based Climate Science. Data opposing CO_2 Emissions as the Primary Source of Global Warming, Elsevier, 2016.

[7] J.C. Orr, J.-M. Epitalon, J.-P. Gattuso, Comparison of ten packages that compute ocean carbonate chemistry, Biogeosciences 12 (2015) 1483–1510.
[8] H. Lavigne, J.-P. Gattuso, Seacarb: seawater carbonate chemistry with R. Version 2.3.3, 2010, from http://cran-project.org/package = seacarb.
[9] J.C. Orr, J.-M. Epitalon, A.G. Dickson, J.-P. Gattuso, Routine uncertainty propagation for the marine carbon dioxide system, Mar. Chem. 207 (2018) 84–107. Available from. Available from: https://www.sciencedirect.com/science/article/pii/S030442031830149X.
[10] R.N. Roy, L.N. Roy, K.M. Vogel, C. Porter-Moore, T. Pearson, C.E. Good, et al., The dissociation constants of carbonic acid in seawater at salinities 5 to 45 and temperatures 0 to 45 °C, Mar. Chem. 44 (2–4) (1993) 249–267.
[11] J. Lueker, A.G. Dickson, C.D. Keeling, Ocean pCO2 calculated from dissolved inorganic carbon, alkalinity, and equations for K1 and K2: validation based on laboratory measurements of CO2 in gas and seawater at equilibrium, Mar. Chem. 70 (2000) 105–119.
[12] F.J. Millero, Thermodynamics of the carbon dioxide system in the oceans, Geochim. Cosmochim. Acta 59 (1995) 661–677.
[13] J. Waters, F.J. Millero, R.J. Woosley, Corrigendum to "The free proton concentration scale for seawater pH, Mar. Chem., 149 (2013) 8–22, Mar. Chem. 165 (2014) 66–67.
[14] A.G. Dickson, Standard potential of the reaction: AgCI(s) + 1/2H2(g) = Ag(s) + HCI (aq), and the standard acidity constant of the ion HSO4 in synthetic sea water from 273.15 to 318.15 K, J. Chem. Thermodyn. 22 (1990) 113–127.
[15] I. Hansson, A new set of pH-scales and standard buffers for sea water, Deep. Sea Res. Oceanographic Abstr. 20 (5) (1973) 489–491.
[16] F.F. Perez, F. Fraga, Association constant of fluoride and hydrogen ions in seawater, Mar. Chem. 21 (1987) 161–168.
[17] F.J. Millero, T.B. Graham, F. Huang, H. Bustos-Serrano, D. Pierrot, Dissociation constants of carbonic acid in sea waters a function of salinity and temperature, Mar. Chem. 100 (2006) 80–84.
[18] Anonymous, Global Ocean Data Analysis Project,Drawing from wikipedia/commons/d/df/WOA05_GLODAP_pd_DIC_Ayool.png, Author: Plumbago(CCBY-SA 3.0).

CHAPTER

Prevalence diagrams for common elements aided by Matlab

10

10.1 Introduction and scope

In order to graphically describe stability and distribution of various soluble and insoluble forms of chemical elements in aqueous solution, we need to consider their oxidation state according to *redox* and pH conditions of the solution.

The concept of pH is easier to understand, being associated with a concentration of H^+ ions in aqueous solution. The symbol H^+ is a shorthand notation of the H_3O^+ ion, which originates from proton transfer between two water molecules, according to the reaction

$$\begin{gathered} H_2O + H_2O \Leftrightarrow H_3O^+ + OH^- \\ \left[H_3O^+\right][OH^-] = K_w = 1.0 \times 10^{-14} \ @\, 25^\circ C \end{gathered} \tag{10.1}$$

where K_w is known as the *ionic product of water*, already defined in Chapter 7. We recall that square brackets indicate molar concentration with unit [mol/L]. Since the equilibrium constant K_w holds 1.00×10^{-14} at the standard laboratory temperature of 25°C, we find the following identities in pure water:

$$\begin{gathered} \left[H_3O^+\right] = \left[H^+\right] = 1.00 \times 10^{-7},\ [OH^-] = 1.00 \times 10^{-7} \\ pH = -\log_{10}\left(\left[H^+\right]\right) = 7.00 \end{gathered} . \tag{10.2}$$

In acid solution, $\left[H^+\right]$ increases, reaching, in the presence of strong acids, the value of 1 mol/L or even higher, and consequently, pH lowers to zero or even to slightly negative values. On the contrary, in the presence of strong bases, $[OH^-]$ increases up to 1 mol/L, whereas $\left[H^+\right]$ decreases down to 1.00×10^{-14} in order to obey the ionic product K_w. Hence, pH may reach values up to 14 or even slightly higher.

The concept of *redox potential* for aqueous solutions is in some way harder to grasp, but there is a conceptual analogy between acid–base and reduction–oxidation reactions. In a similar way that acids and bases have been interpreted as proton donors and proton acceptors, reductants and oxidants are defined as electron donors and electron acceptors [1].

Because there are no free electrons, every oxidation is accompanied by a reduction and vice versa. In other terms, an oxidant is a substance that causes oxidation to occur while itself being reduced. For instance,

$$O_2 + 4H^+ + 4e^- \Rightarrow 2H_2O \tag{10.3}$$

is a half-reaction of the oxygen reduction—free electrons appear on the left side—, whereas

$$4Fe^{2+} \Rightarrow 4Fe^{3+} + 4e^- \tag{10.4}$$

Fundamental Chemistry with Matlab. DOI: https://doi.org/10.1016/B978-0-323-91341-6.00006-9

is a half-reaction of Fe^{2+} oxidation—free electrons appears on the right side—, and the overall reaction

$$O_2 + 4Fe^{2+} + 4H^+ \Rightarrow 4Fe^{3+} + 2H_2O \tag{10.5}$$

is a redox reaction.

An experimental way exists for measuring the *redox potential*: a galvanic cell (also known as electrochemical cell) in which oxidation and reduction occur inside separate vessels and electrons are transferred through an external wire. Current intensity and difference of potential can be directly measured.

10.2 The electrode potential E^0 and the galvanic cell

The standard electrode potential E^0 (also known as *redox potential*) of a half-reaction, which is usually written in the reduction direction, like

$$Zn^{2+}(aq) + 2e^- = Zn(s), \tag{10.6}$$

corresponds to the standard electrode potential, in voltage unit [V], of an electrochemical cell where the half-reaction couples with a standard hydrogen electrode (SHE). We recall that a *standard state* corresponds to a pure substance in the case of solids and unit concentration in the case of solutions, as well to the standard laboratory temperature of 25°C and pressure equal to 100 kPa(1 bar), according to the IUPAC (International Union of Pure and Applied Chemistry) suggestion [2]. In Chapters 5 and 6, the past pressure standard, still widely in use, of 101.235 kPa(1 atm) was used to be coherent with experimental data.

On a SHE, according to the specific half-reaction which occurs at the other electrode, both oxidation of molecular H_2 to H^+ ions and reduction of H^+ ions to molecular hydrogen (in gaseous phase) may occur. A reference potential $E^0 = 0.000V$ is assigned to SHE, irrespectively of the reaction [3].

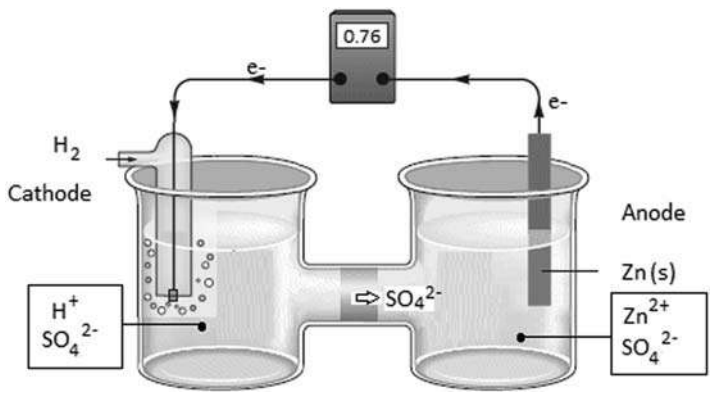


FIGURE 10.1

Sketch of a zinc/hydrogen galvanic cell.

If the half-reaction electrode has a negative standard potential, for instance −0.763 V in the case of zinc electrodes (see Fig. 10.1), the hydrogen electrode becomes positively charged and a reduction reaction occurs—the free electrons appear on the left side since electrons flow from negative to positive electrode—like

$$2H^+(aq) + 2e^- = H_2(g). \tag{10.7}$$

Instead, the following oxidation reaction—the free electrons appear on the right side—occurs on the zinc electrode

$$Zn(s) = Zn^{2+}(aq) + 2e^-, \tag{10.8}$$

even if, under standard conventions, the zinc half-reaction is written as a reduction.

On the contrary, if the half-reaction electrode has a positive standard potential, like in the reaction

$$Cu^{2+}(aq) + 2e^- = Cu(s) \Leftrightarrow E^0 = +0.337\ V, \tag{10.9}$$

where the copper electrode is coupled with a hydrogen electrode, the latter becomes negatively charged and an oxidation reaction takes place like

$$H_2(g) = 2H^+(aq) + 2e^-. \tag{10.10}$$

When all the species in a cell reaction are in their standard states, namely either pure solid or in unit concentration at 25°C and 100 kPa, the measured electromotive force (EMF, in voltage units, [V]) of the cell equals the standard potential E^0 of the cell. The identity would be TRUE, if we could measure the potential under zero current intensity, which is nearly the case when modern voltmeters are employed.

The electrode potential is sometimes known as *redox potential*. It is usually referred to the SHE. A list of standard redox potentials at standard laboratory temperature of 25°C, pressure of 100 kPa for gaseous substances, and unit concentration, for the most common half-reactions (always written as reduction reactions with electrons on the left side), is known as the *Electrochemical Series*. A subset of the whole list of standard electrode potentials is reported in Table 10.1. A full reference can be found elsewhere [3–5]. Very often, SHE is referred to the past standard pressure of 1 atm = 101.325 kPa instead of the current standard of 100.0 kPa (see Reference [2]). However, the correction $\Delta E^0 = 0.17$mV of the reference SHE potential $E^0(H_2\ @\ 1\ atm) = 0.000$ V, in passing to the current standard, only affects the fourth decimal figure as proven in Section 10.4 with the help of the Nernst equation. Therefore the potential values in Table 10.1 remain valid whichever be the underlying standard pressure.

10.3 Energy analysis of a galvanic cell

10.3.1 Introduction

A galvanic cell can be easily constructed by joining two half-elements with an external electric wiring and a salt bridge. Electrons flow in the conductor, whereas ionic neutrality is assured by a proper ionic movement in the salt bridge. The salt bridge may be substituted by other means which allow ions to flow, like a porous glass membrane in Fig. 10.2.

The pair of half-reactions in Fig. 10.2—both reactions are indicated as reduction half-reactions—and the corresponding electrode potentials are as follows:

$$\begin{aligned} &Cu^{2+}(aq) + 2e^- = Cu(s) \Leftrightarrow E^0 = +0.337\ V \\ &Zn^{2+}(aq) + 2e^- = Zn(s) \Leftrightarrow E^0 = -0.763\ V \end{aligned}. \tag{10.11}$$

Table 10.1 Subset of the electrochemical series.

No.	Reduction reaction	Standard electrode potential E^0[V], @25°C and 100.000 kPa
1	$Na^+(aq) + e^- = Na(s)$	−2.71
2	$Mg^{2+} + 2e^- = Mg(s)$	−2.35
3	$Zn^{2+} + 2e^- = Zn(s)$	−0.76
4	$Fe^{2+} + 2e^- = Fe(s)$	−0.44
5	$Co^{2+} + 2e^- = Co(s)$	−0.28
6	$V^{3+} + e^- = V^{2+}(s)$	−0.26
7	$2H^+ + 2e^- = H_2(g)$	0.00
8	$S(s) + 2H^+ + 2e^- = H_2S$	+0.14
9	$Cu^{2+} + e^- = Cu^+$	+0.16
10	$AgCl(s) + e^- = Ag(s) + Cl^-$	+0.22
11	$Cu^{2+} + 2e^- = Cu(s)$	+0.34
12	$Cu^+ + e^- = Cu(s)$	+0.52
13	$Fe^{3+} + e^- = Fe^{2+}$	+0.77
14	$Ag^+ + e^- = Ag(s)$	+0.80
15	$Fe(OH)_3(s) + 3H^+ + e^- = Fe^{2+} + 3H_2O$	+1.01
16	$IO_3^- + 6H^+ + 5e^- = \frac{1}{2}I_2 + 3H_2O$	+1.23
17	$MnO_2(s) + 4H^+ + 2e^- = Mn^{2+} + 2H_2O$	+1.29
18	$Cl_2(g) + 2e^- = 2Cl^-$	+1.36
19	$Co^{3+} + e^- = Co^{2+}$	+1.82
20	$F_2(g) + 2e^- = 2F^-(aq)$	+2.87

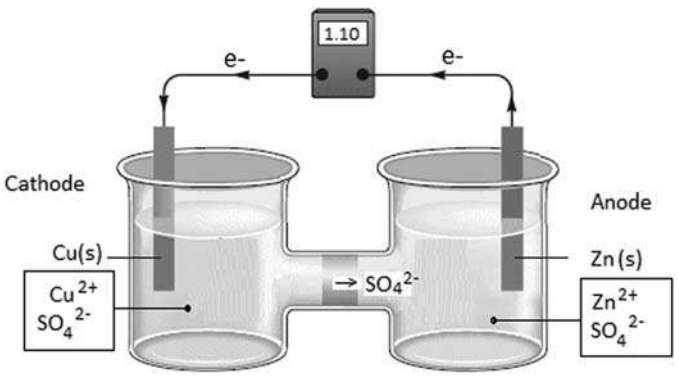


FIGURE 10.2

Sketch of a copper/zinc galvanic cell.

Due to the fact that electrons are only transferred—they are neither created nor destroyed—, electrons are emitted from the anode and collected through the wire by the cathode. It follows that reactions occur in opposite ways on the pair of electrodes, so that they can be added together, like the respective E^0 as follows:

$$\begin{array}{l}\text{(anode) } Cu^{2+}(aq) + 2e^- = Cu(s) \Leftrightarrow E^0 = +0.337\ \text{V} \\ \text{(cathode) } Zn(s) = Zn^{2+}(aq) + 2e^- \Leftrightarrow E^0 = +0.763\ \text{V} \\ \hline \text{(overall) } Cu^{2+}(aq) + Zn(s) = Cu(s) + Zn^{2+}(aq) \Leftrightarrow E^0 = +1.100\ \text{V}\end{array} \tag{10.12}$$

The reactions in Eq. (10.12) of the galvanic cell in Fig. 10.2 spontaneously produce an EMF or potential difference equal to $+1.100$V under standard conditions. The apparatus is very efficient, so that nearly all the chemical energy is transformed into electric energy.

By assuming that the efficiency reaches the ideal 100% fraction, thermodynamics tells us that the electrical work produced by the cell equals the negative Gibbs free energy variation $-\Delta G^0$ (J.W. Gibbs, 1839−1903), namely that

$$E^0 nF = -\Delta G^0. \tag{10.13}$$

In Eq. (10.13), the sign has been changed with respect to the standard convention of negative work toward the environment, n denotes the number of transferred electrons in the reaction, and F is the Faraday constant, namely the electric charge per mole of electrons, equal to the electron charge e multiplied by the Avogadro number:

$$F = eN_A = 1.6022 \times 10^{-19} \times 6.0221 \times 10^{23} = 96.485 \times 10^3\ \text{C/mol} \tag{10.14}$$

We recall that the Gibbs free energy G is a thermodynamic potential which corresponds to the maximum reversible work, if performed by a thermodynamic system at constant temperature and pressure. The superscript 0 denotes standard conditions. The Gibbs free energy SI unit is Joule [J], and its variation ΔG is related to the enthalpy variation ΔH and entropy variation ΔS by $\Delta G = \Delta H - T\Delta S$, where T is the absolute temperature. We can now write

$$E^0 nF = -\Delta G^0 \Rightarrow E^0 = \frac{-\Delta G^0}{nF} \tag{10.15}$$

Since we know from thermodynamics that $\Delta G^0 = -RT\log(K_{eq})$, we find the final identity:

$$E^0 = \frac{-\Delta G^0}{nF} = \frac{RT}{nF}\log(K_{eq}) = \frac{2.303RT}{nF}\log_{10}(K_{eq}), \tag{10.16}$$

where:

1. E^0 denotes the measured potential relative to the SHE, which is set to zero.
2. K_{eq} is the product at the equilibrium of all the oxidized concentrations divided by the product of the reduced ones.

In the copper/zinc galvanic cell above, we can write:

$$\begin{array}{ll}(1) & K_{eq} = \dfrac{\{Cu\}\{Zn^{2+}\}}{\{Zn\}\{Cu^{2+}\}} = \dfrac{\{Zn^{2+}\}}{\{Cu^{2+}\}} \\ (2) & K_{eq} = \dfrac{[Zn^{2+}]}{[Cu^{2+}]}\end{array}. \tag{10.17}$$

Identity (1) in Eq. (10.17) uses *thermodynamic activities*, denoted by a symbol within braces like $\{A\}$. Activities of solid substances and gasses at 100 kPa are exactly 1, so that they disappear on the right side of the top identity. Actually, in a solution, the thermodynamic activity of a compound A does not correspond to its molar concentration $[A]$. A correction should be introduced, so that $\gamma[A] = \{A\}$. In the following, we will assume $\gamma = 1$, and therefore $[A] = \{A\}$, although the assumption should only apply to diluted solutions. Under this assumption, identity (1) can be rewritten as identity (2).

If we assume the standard condition of 25°C, all the constants before $\log_{10}$ in Eq. (10.16) can be merged to yield:

$$E^0 = \frac{2.303RT}{nF}\log_{10}K_{eq} = \frac{2.303 \times 8.314 \times 298.15}{96485n}\log_{10}K_{eq} = \frac{0.0592}{n}\log_{10}K_{eq} \tag{10.18}$$

Eq. (10.18) applies to half-reactions and a complete redox system as well, but it will be mainly used for half-reactions, those with free e^- on the left side.

10.3.2 Example 1

Let us compute the constant K_{eq} for the copper/zinc galvanic cell in Fig. 10.2, with $n = 2$ and $E^0 = 1.1\text{V}$ from Eq. (10.12). With the help of Eq. (10.16), we find

$$K_{eq} = \exp\left(E^0\frac{nF}{RT}\right) = 10^{\frac{nE^0}{0.0592}} = 10^{\frac{2\times 1.100}{0.0592}} = 1.45 \times 10^{37}. \tag{10.19}$$

The extremely large ratio $\left[Zn^{2+}\right]/\left[Cu^{2+}\right] = 1.45 \times 10^{37}$ implies that nearly all copper has disappeared from the solution, that the thermodynamic equilibrium has been reached and the cell EMF equals 0.00 V.

10.4 The Nernst equation

10.4.1 Introduction

Under thermodynamic equilibrium, a galvanic cell is no more capable of providing nonzero potential difference, which implies $\Delta G = 0$ and $E = 0$. But what happens if the cell deviates from equilibrium conditions? Basically, a nonzero EMF establishes between electrodes. Let us start again from the thermodynamic relationship $\Delta G^0 = -RT\log\left(K_{eq}\right)$, which, under nonequilibrium conditions, is rewritten as

$$\Delta G = \Delta G^0 + RT\log(Q), \tag{10.20}$$

where Q is the product of all oxidized concentrations divided by the product of the reduced ones, under nonequilibrium condition. We can write the identities

$$E_H = -\frac{\Delta G}{nF} = -\frac{\Delta G^0 + RT\log(Q)}{nF}, \tag{10.21}$$

and the following equation, which should be compared with Eq. (10.18):

$$E_H = -\frac{\Delta G^0}{nF} - \frac{2.303RT}{nF}\log_{10}(Q) = E^0 - \frac{0.0592}{n}\log_{10}(Q), \tag{10.22}$$

where E_H denotes the voltage potential with respect to the SHE as calculated by Eq. (10.22), which is the well-known *Nernst* equation (W.H. Nernst, 1839−1903). The quantity Q is defined by

$$Q = \frac{\Pi_i\{\text{right side}\}}{\Pi_i\{\text{left side}\}} \simeq \frac{\Pi_i[\text{right side}]}{\Pi_i[\text{left side}]}. \tag{10.23}$$

The symbol Π_i denotes the product of the activities (or of the concentrations under our previous assumption of confusing them) of all the substances, where each substance concentration is raised by its own stoichiometric coefficient. The Nernst equation applies to complete redox systems as well as to half-reactions (those with free e^- on the left side). In the latter case, we rewrite Eq. (10.23) as follows:

$$Q = \frac{\Pi_i\{\text{reduced side}\}}{\Pi_i\{\text{oxidized side}\}} \simeq \frac{\Pi_i[\text{reduced side}]}{\Pi_i[\text{oxidized side}]}. \tag{10.24}$$

Using Eqs. (10.22) and (10.23), we can prove the correction ΔH^0 of the SHE potential E^0 in row 7 of Table 10.1. By setting $[H^+] = 1$, $P(H_2) = 100.000/101.235$, the Nernst equation provides

$$\Delta H^0 = E^0 - \frac{0.0592}{n}\log_{10}\frac{P(H_2)}{[H^+]^2 = 1} = 0.00 - \frac{0.0592}{2}\log_{10}\left(\frac{100000}{101325}\right) = 0.17 \text{ mV}. \tag{10.25}$$

10.4.2 Example 2

We refer again to the copper/zinc cell, sketched in Fig. 10.2, whose reaction is repeated here:

$$Cu^{2+}(aq) + Zn(s) = Cu(s) + Zn^{2+}(aq) \Rightarrow E^0 = +1.100\text{V}. \tag{10.26}$$

The following series of identities can be written based on the previous discussion:

$$\begin{aligned} E_{cell} &= 1.100 - \frac{0.0592}{2}\log_{10}\frac{\{Zn^2+\}}{\{Cu^2+\}}, \text{ Nernst equation of the whole cell} \\ E_{H,cathode} &= +0.337 - \frac{0.0592}{2}\log_{10}\{Cu^{2+}\}, \text{ Nernst equation of the cathode} \\ E_{H,anode} &= -0.763 - \frac{0.0592}{2}\log_{10}\{Zn^{2+}\}, \text{ Nernst equation of the anode} \end{aligned} \tag{10.27}$$

By subtracting the last identity from the second one, the identity $E_{cell} = E_{H,cathode} - E_{H,anode}$ is obtained, that is, the first equation in Eq. (10.27). We remark that the deponents *cathode* and *anode* distinguish the different voltage potentials E_H defined by Eq. (10.22).

10.4.3 Example 3

Let us calculate E_{cell} of the following half-reaction:

$$NO_3^-(aq) + 4H^+(aq) + 3e^- \Rightarrow NO(g) + 2H_2O. \tag{10.28}$$

From the Nernst Eq. (10.22), we write:

$$E_{cell} = E^0 - \frac{0.0592}{4}\log_{10}\frac{p_{[NO]}}{\{NO_3^-\}\{H^+\}^4}, E^0 = +0.92 \text{ V}, \tag{10.29}$$

where $p_{[NO]}$ is the pressure of the nitrogen oxide. If we assume that the pressure of the nitrogen oxide is unitary and we replace activities with concentrations, the final identity holds:

$$E_{cell} = E^0 + 0.0148\log_{10}\left(\left[NO_3^-\right]\left[H^+\right]^4\right) = 0.92 + 0.0148\log_{10}\left(\left[NO_3^-\right]\left[H^+\right]^4\right). \tag{10.30}$$

10.5 The electron activity

Aqueous solutions do not contain free protons and free electrons, but it is nevertheless possible to define a relative electron activity, in some way similar to pH. The relevant parameter, which has been defined as the negative base-10 logarithm of the concentration of H^+ ions, measures the relative tendency of a solution to accept or transfer protons. In acid solutions, this tendency is low, but in alkaline solutions, this tendency is high. We recall that the symbol p (from Latin *pondus*), which precedes H, conventionally indicates the negative base-10 logarithm, as follows:

$$\begin{gathered} \mathrm{pH} = -\log_{10}\{H^+\}, \\ \text{or by setting } \gamma = 1,\ \mathrm{pH} = -\log_{10}\left[H^+\right] \end{gathered} \tag{10.31}$$

Similarly, we can define an equally convenient parameter for the redox intensity:

$$\begin{gathered} p\varepsilon = -\log\{e^-\} \\ \text{or by setting } \gamma = 1,\ p\varepsilon = -\log[e^-]\text{'} \end{gathered} \tag{10.32}$$

where $p\varepsilon$ provides the (hypothetical) electron activity at equilibrium and measures the relative tendency of a solution to accept or transfer electrons. In a highly reducing solution, the tendency to donate electrons, that is, the hypothetical *electron pressure* or *electron activity*, is relatively large [1]. There is a similarity with the potential of electrons in the conduction band of metals or of electrical conductors. Two different metals in close contact generate a potential difference, due to electronic transfer.

The use of electron activity in describing redox equilibria, although intuitive, is not numerically compliant with the standard electrochemical potential series of the elements. Therefore, even if it is employed in some textbook, like Reference [1], it will not be used in the following.

10.6 The prevalence (or Pourbaix) diagrams

10.6.1 Introduction

The aim of the following Matlab® scripts is to solve equilibria between chemical species under the different oxidation states of the same element, at a particular pH and as a function of E_H, the redox potential of the solution relative to the SHE, which is defined by the Nernst Eq. (10.22). In this way, we obtain a $\{\mathrm{pH}, E_H\}$ diagram, known as prevalence (or Pourbaix) diagram (M. Pourbaix, 1904–98) [6,7].

A $\{\mathrm{pH}, E_H\}$ stability diagram shows in a comprehensive way how protons and electrons simultaneously shift the different equilibria under various conditions and may indicate which species predominates within a given range of E_H and pH.

The primary merit of a $\{\mathrm{pH}, E_H\}$ diagram consists of providing an aid for representating equilibrium species in a certain solution, on the basis of the redox potentials which are available in the scientific literature. The diagrams graphically plot the electrochemical stability for the different redox states of an element as a function of pH and E_H. The diagram is a phase diagram, which maps the prevalence/stability regions of the pair $\{\mathrm{pH}, E_H\}$ (in the case of aqueous solutions), where a redox species is stable.

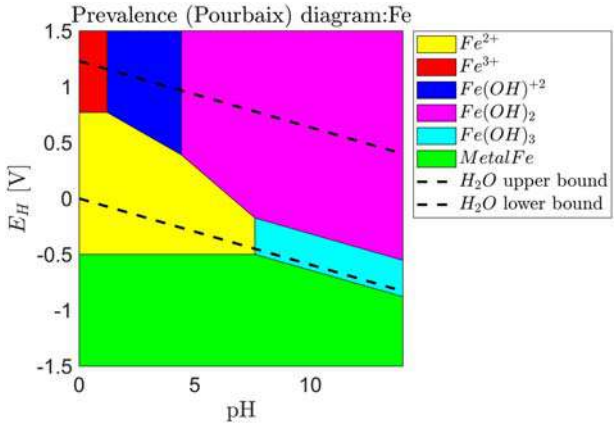


FIGURE 10.3

Prevalence/stability diagram $\{\text{pH}, E_H\}$ of Fe.

Table 10.2 Fe electrochemical potentials.

No.	Half-reaction (reduction)	Standard electrode potential E^0 [V]
1	$Fe^{2+} + 2e^- \Rightarrow Fe(s)$	−0.4402
2	$Fe^{3+} + e^- \Rightarrow Fe^{2+}$	+0.771
3	$Fe(OH)_3 + 3H^+ + 3e^- \Rightarrow Fe(s) + 3H_2O$	+0.059
4	$Fe(OH)_2 + 2e^- \Rightarrow Fe(s) + OH^-$	−0.877
5	$(FeOH)^{2+} + H^+ + e^- \Rightarrow Fe^{2+} + H_2O$	+0.914

The boundary lines in a Pourbaix diagram represent redox and acid−base reactions and are the parts of the diagram where two species can exist in equilibrium. For example, in the Pourbaix diagram of Fe in Fig. 10.3, the red area on the top left represents the stability region of Fe^{3+}, which is only stable in a strong oxidizing and acid environment, the yellow area represents the stability region of Fe^{2+}, and the horizontal line between Fe^{3+} and Fe^{2+} regions is described by the reaction $Fe^{3+}(aq) + e^- \Rightarrow Fe^{2+}(aq)$, which has a standard potential of +0.77V.

While we could use standard electrode potentials for all these lines (when all the concentrations of a soluble species are unitary), in practice, Pourbaix diagrams are plotted for lower ion concentrations (often 1 mmol/L or even smaller), since they are more relevant for corrosion and electrochemical experiments. In the diagrams of this book, concentration has been chosen to be 0.01 mol/L, but this value can be adjusted in the Matlab scripts by users.

The first step in the diagram construction, for instance the Fe diagram, is to collect the relevant electrochemical potentials, which have been reported in Table 10.2.

Usually, the above potentials do not refer to a common oxidation state of the element, so it is necessary to transform them with the reference to an oxidation state, which is equal to zero in the case of metallic elements.

For instance, the reference potential for Fe^{3+} is that of Fe^{2+} and not that of metallic iron, but adding the reaction in row 1 of Table 10.2 to the reaction in row 2, we obtain the desired relation as follows

$$\begin{aligned} Fe^{2+} + 2e^- &\Rightarrow Fe(s),\ E_1^0 = -0.4402\ V \\ Fe^{3+} + e^- &\Rightarrow Fe^{2+},\ E_2^0 = +0.771\ V \\ Fe^{3+} + 3e^- &\Rightarrow Fe(s),\ E_{tot}^0 = (-0.4402 \times 2 + 0.771)/3 = -0.0365\ V \end{aligned} \tag{10.33}$$

The electrochemical potentials are multiplied or divided by the number of transferred electrons. For the last half-reaction, the Nernst equation writes as:

$$E_H(Fe^{3+}/Fe) = -0.0365 - \frac{0.0592}{3}\log_{10}\frac{\{Fe\}}{\{Fe^{3+}\}}. \tag{10.34}$$

Since, for solid Fe, the activity is unitary, namely $\{Fe\} = 1$, like for pure solid substances, we can replace activities with concentrations and rearrange Eq. (10.34) as follows:

$$\log_{10}E_H(Fe^{3+}/Fe) = (E^0 + 0.0365)\frac{3}{0.0592} - \log_{10}[Fe^{3+}] = (E^0 + 0.0365)\frac{3}{0.0592} - \log_{10}c_1 \tag{10.35}$$

where $c_1 = 0.01\ mol/dm^3$ denotes the chosen concentration of the soluble species.

In order to carry out similar computations, we write the following statements:

```
logFe2 = (Eh + 0.4402)*2/R1 - logConcFe    % logConcFe = -2
logFe3 = (Eh + 0.0365)*3/R1 - logConcFe    % R1 = 0.0592
logFeOH = (Eh - 0.0112)*3/R1 + 2*pH - logConcFe
logFeOH3 = (Eh - 0.059)*3/R1 + 3*pH
logFeOH2 = (Eh + 0.0496)*2/R1 + 2*pH
logFe = 0                                  % unit activity
```

The coefficients appearing in the above rows are collected in the sheet Fe of the excel file `PrevalenceDiagram.xlsx`, which is available in the companion website of the book [8]. When a particular boundary line in Fig. 10.3 has to be computed, for instance, the line separating the blue and yellow regions assigned to $Fe(OH)^{2+}$ and Fe^{2+}, respectively, the corresponding reduction potentials are equated as follows:

```
logFe2 = log FeOH
(Eh + 0.4402)*2/R1 - logConcFe = (Eh - 0.0112)*3/R1 + 2*pH - logConcFe
Eh = 0.9138 - 0.118·pH
```

Repeating the procedure for every boundary line allows us to graphically plot the boundaries of every prevalence/stability domain, having a polygonal form, in a 2D diagram. The Matlab `fill` function, which requires to know the ordered set of the polygon vertices, is employed for coloring the different regions.

10.6.2 Two alternative algorithms for building prevalence/stability diagrams

Two alternative algorithms and scripts, both aiming to build the prevalence diagram of the oxidation states of a chemical species, are explained. To be generic, let us denote the different oxidation states of a chemical species with $\{S_1,\ldots,S_\mu,\ldots,S_m\}$ and the corresponding reduction potentials as $\{E_1,\ldots,E_\mu,\ldots,E_m\}$, where each potential is a function $E_\mu(\mathrm{pH},E_H)$ of the pair $\{\mathrm{pH},E_H\}$. For simplicity's sake, we assume that the function is linear as

$$E_\mu(\mathrm{pH},E_H)=a_\mu\mathrm{pH}+b_\mu E_H+c_\mu. \tag{10.36}$$

A state S_ν is said to prevail on the other states if

$$E_\nu(\mathrm{pH},E_H)>\max_{\mu\neq\nu}E_\mu(\mathrm{pH},E_H). \tag{10.37}$$

The lines $B_{\nu,\mu\neq\nu}(\mathrm{pH},E_H)$ defined by $E_\nu(\mathrm{pH},E_H)=E_{\mu\neq\nu}(\mathrm{pH},E_H),\mu=1,\ldots,m$ constitute the candidate boundaries of the prevalence region of the state S_ν, which possesses a polygonal form constituted by up to $m-1$ edges and vertices, unless it crosses the boundaries $\{\mathrm{pH}_{min}=0,\mathrm{pH}_{max}=14\}$ and $\{E_{H,min},E_{H,max}\}$ of the pair $\{\mathrm{pH},E_H\}$. The polygon is completely defined by the set V_ν of its vertices $\boldsymbol{v}_\kappa,\kappa=1,\ldots,k_\nu$.

1. The first algorithm and script scans the pixels $i,j,i=1,\ldots,N,j=1,\ldots,M$ of the $\{\mathrm{pH},E_H\}$ quadrant defined by the boundaries previously defined, where $N\Delta\mathrm{pH}=\mathrm{pH}_{max}-\mathrm{pH}_{min}$ and $M\Delta E_H=E_{H,max}-E_{H,min}$ define the resolution pair $\{\Delta\mathrm{pH},\Delta E_h\}$ of the search. A pixel is assigned to S_ν, if the inequality (10.37) holds, and consequently is colored in the graphical representation like in Fig. 10.3.
2. The second algorithm and script explicitly search, via the function `intersection`, the vertex set V_ν, which is a subset of the set C_ν of all the intersections between the $m-1$ lines $B_{\nu,\mu\neq\nu}(\mathrm{pH},E_H)$ and the four axes of the $\{\mathrm{pH},E_H\}$ quadrant, for a total of $m-1+4$ candidate boundary lines. The subset V_ν is found by searching around any intersection, whether a cone exists with a finite aperture $\Delta\alpha\geq\Delta\alpha_{min}>0$, where inequality (10.37) holds, via the function `prev`. The resolution $\Delta\alpha$ must be selected to be sufficiently larger than the least vertex angle of the expected polygonal forms. The reader must be warned that very narrow angles have been found in the Mn and S diagrams (see Fig. 10.7 and Fig. 10.8). Coloring of each polygonal form is performed by the Matlab function `fill`, which requires that the vertices of V_ν are ordered either clockwise or counterclockwise around their center. Ordering is performed by the function `Vorder`. Having selected a vertex as the origin and a second vertex to define a reference axis, the angle between any vector from the origin and the reference axis is the parameter to be ordered. To avoid alignment, the measured angles are appropriately perturbed.

10.7 The first algorithm

Once the Fe concentration is fixed to the default value 0.01 mol/L corresponding to `logConc = -2`, the script loads a blank diagram, `prevalence.bmp`, in the matrix `M`. The matrix is then filled pixel by pixel by the color pertaining to the highest potential among the different forms of Fe, which is detected by the Matlab function `max(x)`.

```
InitChem;
x = zeros (1,6);
logConc = -2;logFe = 0;
M = imread('prevalence.bmp');
for row = 50:437
  for column = 57:404
    pH = (column - 57)/24.786;
    Eh = (243.5 - row)/129;
    x(2) = (Eh + 0.4402)*2/0.0591 - logConc; % log Fe2+
    x(3) = (Eh + 0.0365)*3/0.0591 - logConc; % log Fe3+
    x(4) = (Eh - 0.0112)*3/0.0591 + 2*pH - logConc; % log Fe(OH)2+
    x(5) = (Eh - 0.059)*3/0.0591 + 3*pH;      % log Fe(OH)3
    x(6) = (Eh + 0.0496)*2/0.0591 + 2*pH;     % log Fe(OH)2
    [w,iw] = max(x);
    switch iw
      case 1;M(row,column,1:3) = [128,128,128]; % metallic Fe
      case 2;M(row,column,1:3) = [255,255,0]; % Fe2+ ions
      case 3;M(row,column,1:3) = [255,0,0]; % Fe3+ ions
      case 4;M(row,column,1:3) = [0,0,255]; % Fe(OH)2+ ions
      case 5;M(row,column,1:3) = [255,0,255]; % Fe(OH)3 solid
      case 6;M(row,column,1:3) = [0,255,255]; % Fe(OH)2 solid
    end
  end
end
image(M);
```

10.8 The second algorithm

10.8.1 The main script

The main script is organized in two parts. The heading is specific of the species, say Fe, whose prevalence diagram is aimed to, and contains the name of the species `SName` and other parameters, whose value should remain the same whichever be the species. Two parameters deserve attention: (1) the size `dimv = 10` of the vertex matrix `vTab`, whose value has been found sufficient for the book case studies, (2) the neighbor radius `rho = 0.0005` where to check the prevalence, which has been calibrated on the sulfur diagram (see Fig. 10.8). Increasing the radius value may generate spurious copies of the same vertex, which however should not impair the diagram construction. The second neighbor parameter is `Npsi = 180`, which defines the angular resolution `dpsi = 2*pi/Npsi` for the neighbor scan. The heading code in the case of `Sname = 'Fe'` is the following.

```
%% Prevalence diagram: S (Chapter 10)
%% Initialization
InitChem;
%% Common parameters
SName='S';
Rl=0.0592; Conc=0.01; logC=log10(Conc);
% Quadrant axes [0,y] [pHmax,y]  [x,EhMin] [x,EhMax]
pHMin=0; pHMax=14; dPh=0.05; % pH axis
EhMin=-1.5; EhMax=1.5; dEh=0.01; %Eh axis
lim=[pHMin pHMax;
    EhMin EhMax];
dimv=10; % WARNING max number of vertices per region
rho=0.0005;Npsi=180;  % cone for checking prevalence
dAngle=0.0001; % angle perturbation to avoid alignment
%% Main script
PrevalenceMain;
```

The heading starts with the initialization script `InitChem` and closes with the main script `PrevalenceMain`.

The main script consists of four parts.

1. The coefficients of the different states are read from a sheet of the excel file `Prevalence Diagram.xlsx`, which is available in the companion website of the book [8]. The sheet name is the same as `Sname`.
2. The coefficients of the H_2O boundary lines, to be added in the prevalence diagram, are read from the sheet H2Ob. The latter sheet should not to be confused with the sheet H2O listing the data for the water prevalence diagram.
3. The following dimensions and matrices are prepared: (i) `cTab` contains the coordinates of the intersections points together with the relevant prevalence logic (0/1, column 3, `logp`) and the boundary line indices (quadrant axes 1 to 4, state axes starting from 5). It may be checked in the case of anomalies. (ii) `vTab` contains the ordered list of vertices as printed out in the `command window`.
4. The vertices of the prevalence regions are searched out, the colored diagram of each polygonal region is plotted, and numerical results are printed out in the `command window`.

The numerical printout includes the electrochemical potentials and other data for building the prevalence diagram as in the following table, which refers to the Fe diagram in Fig. 10.3.

```
--- Prevalence diagram: Fe
State table
State          phScale EhScale Potential[V] log(c)Scale Color
Fe^{2+}        0.00     2.00     0.4402          -1      y
Fe^{3+}        0.00     3.00     0.0365          -1      r
Fe(OH)^{+2}    2.00     3.00    -0.0112          -1      b
Fe(OH)_2       3.00     3.00    -0.0590           0      m
Fe(OH)_3       2.00     2.00     0.0496           0      c
Metal Fe       0.00     0.00     0.0000           0      g
Prevalence region vertices
# dim ( pH, Eh[V]) ...
1  6 ( 0.00, 0.77)( 0.00,-0.50)( 7.60,-0.50)( 7.60,-0.17)
( 4.42, 0.39)( 1.21, 0.77)
2  4 ( 0.00, 1.50)( 0.00, 0.77)( 1.21, 0.77)( 1.21, 1.50)
3  4 ( 1.21, 1.50)( 4.42, 1.50)( 4.42, 0.39)( 1.21, 0.77)
4  5 (14.00, 1.50)(14.00,-0.55)( 7.60,-0.17)( 4.42, 0.39)
( 4.42, 1.50)
5  4 (14.00,-0.55)(14.00,-0.88)( 7.60,-0.50)( 7.60,-0.17)
6  5 ( 0.00,-1.50)( 0.00,-0.50)( 7.60,-0.50)(14.00,-0.88)
(14.00,-1.50)
```

The upper part reports the coefficients and voltage potential of each oxidation state, as they are read from the relevant excel sheet for building the boundary lines of the stability regions. The lower part provides the list of the region vertices. When an oxidation case is not prevalent on the other ones, its stability region is void and consequently no vertex is found. The relevant row will show NOT PREVALENT, and the acronym NP will appear in the prevalence diagram legend beside the oxidation state symbol.

The main script statements are as follows.

```
%% Main prevalence
%% Reading data
fprintf(' --- Prevalence diagram: %s',SName);
PTab=readtable('PrevalenceDiagram.xlsx','ReadVariableNames',true,...
   'ReadRowNames',true,'PreserveVariableNames',true,'Sheet',SName);
[nrow,ncol]=size(PTab);
sprint=PTab.State; % species names
% preparing symbols and colors for plot & legend
symbol={nrow+2,1};
for i=1:nrow, symbol(i)=strcat('$',PTab.State(i),'$'); end
colors=PTab{:,5};  % plot colors, WARNING: braces
fprintf('\nState table ');
fprintf('\nState          phScale EhScale Potential[V]
log(c)Scale Color ');
for i=1:nrow
   fprintf('\n%12s %7.2f  %7.2f %10.4f %10.0f %6s',...
        sprint{i}, PTab{i,1:4},colors{i});
end
%% Reading H2O boundaries
wName='H2Ob'; % not to be confused with the sheet H2O
Pw=readtable('PrevalenceDiagram.xlsx','ReadVariableNames',true,...
   'ReadRowNames',true,'PreserveVariableNames',true,'Sheet',wName);
aw=Pw{1:2,1:3}; linew=zeros(2,2,2); % extremes
for i=1:2
   linew(1,1,i)=pHMin; linew(2,1,i)=-aw(i,3)-aw(i,1)*pHMin/aw(i,2);
   linew(1,2,i)=pHMax; linew(2,2,i)=-aw(i,3)-aw(i,1)*pHMax/aw(i,2);
end
symbol(nrow+1)={'$H_2O$ upper bound'};
symbol(nrow+2)={'$H_2O$ lower bound'};
%% Preparation
%dimensions
dims=nrow; diml=dims+4; % #axes + substances
dimc=sum(1:diml-1); % #intersections
% potential line coefficients
a=zeros(diml,3); % a(1)=pH, a(2)=Eh, a(3)=b
a(1:4,:)=[1 0 -pHMin; 0 1 -EhMin; 1 0 -pHMax; 0 1 -EhMax]; %axes
a(5:diml,1)=PTab{:,1}; %pH scale
a(5:diml,2)=PTab{:,2}/R1; % Eh scale
a(5:diml,3)=PTab{:,3}.*PTab{:,2}/R1+PTab{:,4}*logC; % b
% intersections table
dimVar=7; % x y logp prevalence line 1 line 2
```

```
cTab=inf(dimc,dimVar);
% vertex table   1) x 2) y 3) link to intersection table 4) angle 5) 6)
vTab=inf(dimv,4,dims); % vertices WARNING: size dimv, see above
dimVert=zeros(dims,1); % each species
bn=zeros(diml,3);% boundary lines
bn(1:4,:)=a(1:4,:); % quadrant axes
fprintf('\nPrevalence region vertices ');
fprintf('  \n  # dim ( pH, Eh[V]) ...');
% common figure
figure('name', 'Prev. diagram');grid on;hold on;
%% Finding prevalence regions
% 1) loooking for intersection which are vertices
for s=1: dims  % loop on states
    indp=4+s; k=4;  %  prevalent states
    for i=5:diml  % loop on boundary lines excluding axes
        k=k+1;
        if ne(i,indp)
            bn(k,1)=a(indp,1)-a(i,1);
            bn(k,2)=a(indp,2)-a(i,2);
            bn(k,3)=a(indp,3)-a(i,3);
        end
    end
    % Finding intersections between boundary lines(including axes)
    k=0;   dimVert(s)=0;
    for i=1:diml
        if ne(i,indp)==1 %first boundary
            for j=i+1:diml % second boundary
                if ne(j,indp)==1
                    k=k+1;
                   [v,logv]=intersection(bn,i,j,rho,Npsi,lim,diml,indp);
                    cTab(k,:)=v;  % intersections
                    if logv==1
    dimVert(s)=dimVert(s)+1; vTab(dimVert(s),1:2,s)=v(1:2);
                        % x y
```

```
                vTab(dimVert(s),3,s)=k; % link to cTab
                vTab(dimVert(s),5,s)=i; % 1st line
                vTab(dimVert(s),6,s)=j; % 2nd line
                end % vertices
                end
            end
        end
end
% 2) Reorder
indVert=1:dimVert(s);
vert=Vorder(vTab(:,:,s),dimVert(s),dAngle);
vTab(indVert,:,s)=vert;
% 3) Display and  plot
fprintf('\n%3.1ld%3.1d ',s,dimVert(s));
if dimVert(s)>0
     for p=1:min(dimVert(s),5)
         fprintf('(%5.2f,%5.2f)',abs(vTab(p,1,s)),vTab(p,2,s));
        end
        if dimVert(s)>5
            fprintf('\n       ',s,dimVert(s));
            for p=6:dimVert(s)
            fprintf('(%5.2f,%5.2f)',abs(vTab(p,1,s)),vTab(p,2,s));
            end
        end
        fill(vTab(indVert,1,s),vTab(indVert,2,s),colors{s});hold on;
   else
        fprintf('NOT PREVALENT');
        line(0.0,0.0,'color','w');
        symbol(s)=strcat(symbol(s),'  NP');
   end
end
fprintf('\n');
% H2O boundary lines
line(linew(1,:,1),linew(2,:,1),'color','k','LineStyle','--');
line(linew(1,:,2),linew(2,:,2),'color','k','LineStyle','--');
% legends and title
xlabel('pH');ylabel('Eh [V]');
ylim([EhMin EhMax]);xlim( [pHMin pHMax]);
legend(symbol,'Interpreter','Latex','Location','bestoutside');
st=strcat('Prevalence (Pourbaix) diagram: ',SName);title(st);
```

10.8.2 The functions

The function `intersection` searches the intersection points between two boundary lines of the prevalence species indicated by the index `indp`. Intersection data including the Boolean `logp`, provided by the function `prev` and indicating that the intersection is a vertex of a prevalent region, are

collected in the row vector `v`. At the same time, the Boolean `logv` is created as an output of the function. It differs from `logp` since it excludes the vertices outside the diagram quadrant. Prevalence is checked by the function `prev`.

```
%% Function intersection
function [v,logv]=intersection(bn,k1,k2,rho,Npsi,lim,diml,indp)
    % input 1) lim=[pHMin pHMax; EhMin EhMax]
    % 2) bn coefficients of boundary lines pH, Eh, constant
    % 3) k1, k2 index of boundary lines
 % 4) rho,Npsi small radius and circle divisions to build the small cone
    % where to check prevalence
    % 5) diml, indp  # boundary lines, prevalence species
    % output: 1) v intersection data,
 % 2)logv=1 YES, intersection is a vertex in the diagram quadrant, =0 NO
   v=zeros(1,7); % intersection vector, x y logp psi p k1 k2
    v(1)=inf; v(2)=inf;
    v(5)=-inf;
    v(6)=k1; v(7)=k2;
    vdef=v; logv=0;
    % intersection
    det=bn(k1,1)*bn(k2,2)-bn(k1,2)*bn(k2,1);  % a1*b2-a2*b1
    numx=bn(k1,2)*bn(k2,3)-bn(k2,2)*bn(k1,3);  % b1*c2-b2*c1
    numy=bn(k1,3)*bn(k2,1)-bn(k2,3)*bn(k1,1);  % a2*c1-a1*c2
    if abs(det)>0  % not parallel, check prevalence
        v(1)=numx/det; v(2)=numy/det;
        [logp,pr,psi]=prev(v(1),v(2),bn,rho,Npsi, diml,indp);
        v(3)=logp;
        v(5)=pr;
        v(4)=psi;
        if logp>0, logv=1; end
    end
    % out of range
    if v(1)<lim(1,1) || v(1)>lim(1,2) || v(2)<lim(2,1) || v(2)>lim(2,2)
        v=vdef;logv=0;
    end
end
```

The function `prev` checks the prevalence around an intersection point. A circular neighbor of fixed radius `rho` is scanned with an angle resolution of `2*pi/Npsi`. As soon as the prevalence is found, *the intersection point is declared to be a vertex,* by raising the Boolean`logp` to unit.

```
%% Function prev
function [logp,pr,psi]=prev(x,y,bn,rho,Npsi, diml,indp)
    % input 1)x y intersection coordinates
    % 2) bn coefficients of boundary lines pH, Eh, constant
 % 3) rho,Npsi small radius and circle divisions to build the small cone
    % where to check prevalence
    % 5) diml, indp  # boundary lines, prevalence species
    % output: 1)logp=1, YES, intersection is a vertex, =0 NO
    % 2) pr, signed prevalence product
    % 3) angle where prev has been found , if not 2*pi
    dpsi=2*pi/Npsi;  psi=-dpsi;
    logp=0; pr=0;  j=0;
    while logp<1 && j<Npsi
        j=j+1;
        psi=psi+dpsi;
        xx=x+rho*cos(psi);
        yy=y+rho*sin(psi);
        np=0; p=1;
        for i=5:diml % warning bn sized diml-1
            if ne(i,indp)==1
                pc=bn(i,1)*xx+bn(i,2)*yy+bn(i,3);
                if pc>0, np=np+1; end
                p=p*abs(pc);
            end
        end
        pr=-p; % negative functional
        if np==diml-4-1, logp=1;pr=p;  end % positive functional
    end
end
```

The last function `Vorder` orders the vertex coordinates in a circle around the polygonal center, so as to exploit the coloring Matlab function `fill`.

```
%% Function Vorder
function vert = Vorder(vTab,dimV,dA)
    % Vertices must be ordered either clockwise or anticlockwise
    % with respect to any vertex pair axis
    % input: vTab= vertex table, dimV dimension, dA angle perturbation
    % output: ordered vertices
    % Reference axis
    indVert=1:dimV;
    rx=vTab(2,1)-vTab(1,1); ry=vTab(2,2)-vTab(1,2);
    rUnit=[rx;ry]/sqrt(rx^2+ry^2);
    vTab(1,4)=0;% pivot angle
    for i=2:dimV % looking for angles
         ux=vTab(i,1)-vTab(1,1); uy=vTab(i,2)-vTab(1,2);
         uUnit=[ux;uy]/sqrt(ux^2+uy^2);
         vTab(i,4)=acos(uUnit.'*rUnit)+dA*(i-1);
    end
    [u,iu]=sort(vTab(indVert,4),'ascend'); % reordering
    vert=vTab(iu,:);
end
```

10.9 The fundamental prevalence diagram: water

10.9.1 Description

The construction of the water $\{pH, E_H\}$ diagram in Fig. 10.4 can be introduced by considering the redox stability of the water molecule itself. It can be oxidized to O_2 or reduced to H_2 according to redox and pH values of the following half-reactions [7]:

$$\begin{array}{ll} \text{(a)} & 2H_2O + 2e^- \Rightarrow H_2(g) + 2OH^-,\ E^0 = -0.828\ \text{V, reduction of water} \\ \text{(b)} & 2H_2O \Rightarrow O_2(g) + 4H^+ + 4e^-,\ E^0 = -1.229\ \text{V, oxidation of water} \end{array} . \tag{10.38}$$

The standard electric potential E^0 of the above equations has been computed with the help of the following equations:

$$\begin{array}{ll} \text{(c)} & 2H^+ + 2e^- \Rightarrow H_2(g),\ E^0 = 0.000\ \text{V, reference electrode} \\ \text{(d)} & O_2(g) + 2H_2O + 4e^- \Rightarrow 4OH^-,\ E^0 = +0.401\text{V} \end{array} . \tag{10.39}$$

By adding $2OH^-$ ion to both sides of the reaction (c) and by transforming $2OH^- + 2H^+$ into $2H_2O$, the reaction (a) is obtained. In a similar manner, $4H^+$ is added to both sides of the reaction (d), $4OH^- + 4H^+$ is transformed into $4H_2O$, the reaction is inverted, and the left side is simplified into $2H_2O$, thus obtaining the reaction (b). Since reactions (c) and (d) refer to standard conditions, in other terms to the standard gas pressure of 100 kPa, to $[H^+] = 1\ \text{mol/L}$ and to $[OH^-] = 1 \times 10^{-14}[H^+]^{-1}$, we obtain the potentials in Eq. (10.38):

$$\begin{aligned} E^0(a) &= E^0(c) + 0.0592 \log_{10} \frac{1 \times 10^{-14}}{[H^+]} = -0.828\ \text{V} \\ E^0(b) &= E^0(d) + 0.0592 \log_{10} \frac{1 \times 10^{-14}}{[H^+]} = -0.401 - 0.828\ \text{V} = -1.229\ \text{V} \end{aligned} . \tag{10.40}$$

Were standard conditions not met and were pH varying between 0 and 14, the first-row equation in Eq. (10.40), expressing the reduction of water into hydrogen, would provide the following electric potential:

$$E_{H1} = E^0 - 0.0592 \log_{10} \frac{1 \times 10^{-14}}{[H^+]} = 0.00 + 0.0592 \log_{10} [H^+] = -0.0592 \text{ pH}. \tag{10.41}$$

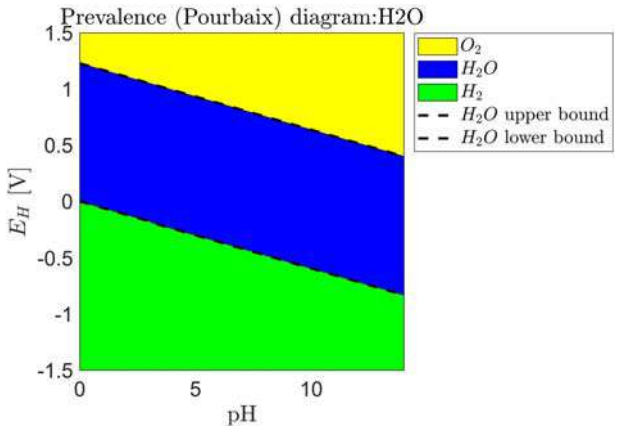


FIGURE 10.4

The $\{\text{pH}, E_H\}$ diagram of water.

The equation describes the boundary line between the green and blue regions in Fig. 10.4. For the oxidation of water to oxygen, the boundary line between the yellow and blue areas is given by:

$$E_{H2} = E^0 - \frac{0.0592}{4} \frac{1 \times 10^{-14}}{[H^+]^4} = 1.229 + 0.0592 \log_{10} [H^+] = 1.229 - 0.0592 \text{ pH}. \tag{10.42}$$

We can now answer questions like: when O_2 at $P = 100$ kPa is capable of oxidizing water and when not? When water can be converted into gaseous hydrogen?

1. The points in the diagram of Fig. 10.4 lying below the lower boundary defined by Eq. (10.41) correspond to an equilibrium of H_2/H_2O with $P(H_2) > 100$ kPa, implying that hydrogen is the predominant phase (the green area).
2. The points in the diagram lying above the upper boundary defined by Eq. (10.42) correspond to an equilibrium of O_2/H_2O with $P(O_2) > 100$ kPa, implying that oxygen is the predominant phase (the yellow area).
3. Within the H_2O domain (in blue color), O_2 acts as an oxidant and H_2 as a reductant.

Both boundaries have a slope equal to -0.0592, and they intersect (1) the ordinate axis pH = 0 at the electric potentials $E_H = 1.229$ V and $E_H = 0.000$ V, respectively, and (2) the ordinate axis pH = 14 at the electric potentials $E_H = 0.401$ V and $E_H = -0.828$ V, respectively.

Care must be taken as natural waters are often found in a highly dynamic state with regards to oxidation/reduction, rather than in or close to equilibrium. Most oxidation/reduction reactions have a tendency to be much slower than acid−base reactions, especially in the absence of appropriate bio-chemical catalysis. Nonetheless, equilibrium diagrams may greatly aid in understanding the possible redox pattern in natural waters and water technological systems.

10.9.2 Graphical and numerical results

The diagram in Fig. 10.4 has been obtained with the second algorithm of the prevalence diagrams. The next table shows the electrochemical potentials and other data which are read from the excel sheet H2O. The table also shows the list of the vertices of the three stability regions as previously explained.

```
--- Prevalence diagram: H2O
State table
State          phScale EhScale Potential[V] log(c)Scale Color
O_2             4.00     4.00    -1.2290          0      y
H_2O            0.00     0.00     0.0000          0      b
H_2            -2.00    -2.00     0.0000          0      g
Prevalence region vertices
# dim ( pH, Eh[V]) ...
1  4 ( 0.00, 1.50)( 0.00, 1.23)(14.00, 0.40)(14.00, 1.50)
2  4 ( 0.00, 1.23)( 0.00,-0.00)(14.00,-0.83)(14.00, 0.40)
3  4 ( 0.00,-1.50)( 0.00,-0.00)(14.00,-0.83)(14.00,-1.50)
```

The data in the previous table correspond to the prevalence diagram of water in Fig. 10.4.

10.10 Manganese oxides and hydroxides

10.10.1 Description, graphical, and numerical results

Manganese oxides and hydroxides cover a wide range of oxidation states, ranging from +2 to +7. The reactions to be considered are as follows.

```
Mn2+ + 2e- ---> Mn(s)                          E° = - 1.180 volt
Mn(OH)2 + 2e- ---> Mn(s) + 2 OH-               E° = - 1.555 volt
MnOOH + 3 H+ + 3e- ---> Mn(s) + 2 H2O          E° = - 0.335 volt
%MnO2 + 4 H+ + 2e- ---> Mn(s) + 2 H2O          E° = + 0.024 volt
MnO4- + 8 H+ + 5e- ---> Mn2+ + 4 H2O           E° = + 1.507 volt
MnO4-- + 2 H2O + 2e- ---> MnO2(s) + 4 OH-      E° = + 0.600 volt
```

Fig. 10.5 has been obtained with the script in Section 10.8 which explicitly finds the vertices of the prevalence regions.

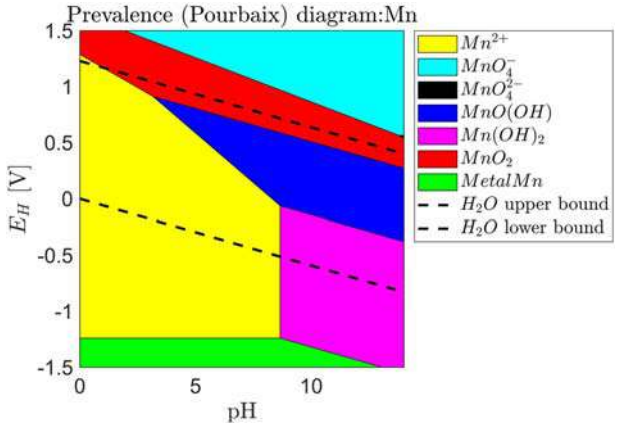


FIGURE 10.5

Prevalence diagram of Mn.

Electrical potentials and coefficients of the Nernst equation which define boundary lines and vertices in Fig. 10.5 are reported below together with the vertices, region by region.

```
--- Prevalence diagram: Mn
State table
State           phScale EhScale Potential[V] log(c)Scale Color
Mn^{2+}          0.00     2.00      1.1800          -1      y
MnO_4^{-}        8.00     7.00     -0.7390          -1      c
MnO_4^{2-}       8.00     6.00     -0.7680          -1      k
MnO(OH)          3.00     3.00      0.3350           0      b
Mn(OH)_2         2.00     2.00      0.7270           0      m
MnO_2            4.00     4.00     -0.0240           0      r
Metal Mn         0.00     0.00      0.0000           0      g
Prevalence region vertices
# dim ( pH, Eh[V]) ...
1  5 ( 0.00, 1.29)( 0.00,-1.24)( 8.65,-1.24)( 8.65,-0.06)
( 3.15, 0.91)
2  4 (14.00, 1.50)(14.00, 0.56)(13.78, 0.56)( 1.94, 1.50)
3  3 (14.00, 0.56)(14.00, 0.54)(13.78, 0.56)
4  4 (14.00,-0.38)(14.00, 0.27)( 3.15, 0.91)( 8.65,-0.06)
5  5 (14.00,-1.50)(13.06,-1.50)( 8.65,-1.24)( 8.65,-0.06)
(14.00,-0.38)
6  7 ( 0.00, 1.50)( 0.00, 1.29)( 3.15, 0.91)(14.00, 0.27)
(14.00, 0.54)(13.78, 0.56)( 1.94, 1.50)
7  4 ( 0.00,-1.50)( 0.00,-1.24)( 8.65,-1.24)(13.06,-1.50)
```

10.11 Lead and lead sulfate

10.11.1 Description

Lead oxides, sulfates, and chlorides are usual found among corrosion products of lead manufacts, like pipes, ingots, vessels, and so on, when exposed to natural waters. The stability/prevalence diagram is calculated in absence of atmospheric CO_2 so as $PbCO_3$ (lead carbonate) formation is not considered.

The reactions to be considered are as follows.

```
Pb2+ + 2e- ---> Pb(s)                    E° = - 0.1266 volt
PbO + 2 H+ + 2e- ---> Pb(s) * H2O        E° = + 0.248 volt
PbO2 + 4 H+ + 2e- ---> Pb2+ * 2 H2O      E° = + 1.455 volt
Pb3O4 + 2 H+  2e- ---> 3 PbO + H2O       E° = + 0.972 volt
PbSO4 + 2e- ---> Pb + SO4--              E° = - 0.3588 volt
PbCl2 + 2e- ---> Pb + 2 Cl-              E° = - 0.2675 volt
```

The following table shows the electrochemical potentials and other data for building stability/prevalence regions. The second part shows the list of the prevalence region vertices. Three oxidation states are not prevalent, and their regions are void.

```
--- Prevalence diagram: Pb
Sta e table
State          phScale EhScale Potential[V] log(c)Scale Color
Pb^{2+}         0.00     2.00      0.1266         -1      k
PbO             2.00     2.00     -0.2480          0      b
PbO_2           4.00     4.00     -0.6642          0      r
Pb_3O_4         2.67     2.67     -0.4290          0      c
PbSO_4          0.00     2.00      0.3588          1      y
Pb(OH)_2        2.00     2.00     -0.2770          0      m
PbCl_2          0.00     2.00      0.2675          2      k
Metal Pb        0.00     0.00      0.0000          0      g
Prevalence region vertices
# dim ( pH, Eh[V])
1  0 NOT PREVALENT
2  4 (14.00, 0.14)(14.00,-0.58)( 9.25,-0.30)( 9.25, 0.42)
3  4 (14.00, 1.50)(14.00, 0.31)( 8.33, 0.64)( 1.08, 1.50)
4  4 (14.00, 0.14)(14.00, 0.31)( 8.33, 0.64)( 9.25, 0.42)
5  6 ( 0.00, 1.50)( 0.00,-0.30)( 9.25,-0.30)( 9.25, 0.42)
( 8.33, 0.64) ( 1.08, 1.50)
6  0 NOT PREVALENT
7  0 NOT PREVALENT
8  5 ( 0.00,-1.50)( 0.00,-0.30)( 9.25,-0.30)(14.00,-0.58)
(14.00,-1.50)
```

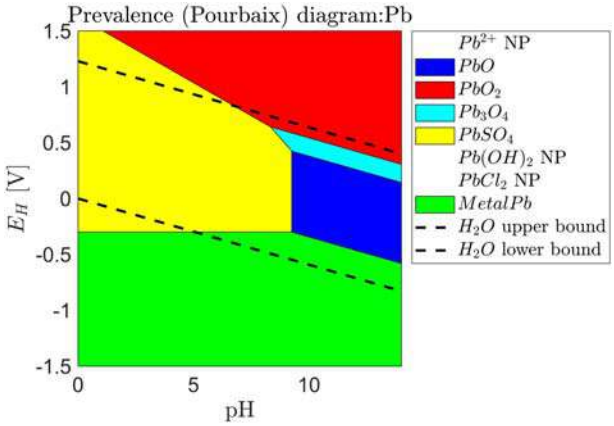


FIGURE 10.6

The prevalence diagram of the lead.

Fig. 10.6 shows the prevalence diagram of the lead oxidation states. Three states are not prevalent (NP), and their regions are void, since no vertex has been found.

10.12 Sulfur

10.12.1 Description, numerical, and graphical results

The stability diagram includes the stable oxidation states of sulfur, namely 0, −2, and +6, which occur as (1) elemental sulfur with nox = 0, (2) suphidric acid with nox = − 2, a weak biprotic acid, and (3) sulfuric acid with nox = + 6, a strong acid for which, however, a protonated form, $H_2SO_4^-$, exists which is stable at very low pH.

The reactions to be considered are as follows.

```
S + H+ + 2e- ---> HS-                    E° = - 0.065  volt
S + 2 H+ + 2e- ---> H2S(aq)              E° = + 0.142 volt
S + 2e-  ---> S(s)                       E° = - 0.4763 volt
SO4-- + 8 H+ + 6e- ---> S(s) + 4 H2O     E° = + 0.3572 volt
HSO4- + 7 H+ + 6e- ---> S(s) + 4 H2O     E° = + 0.339 volt
```

The electrochemical potentials and other data are reported below, together with list of the vertices for each oxidation state.

```
--- Prevalence diagram: S
State table
State          phScale EhScale Potential[V] log(c)Scale Color
HSO_4^{-}       7.00     6.00    -0.3390         -1      m
SO_4^{2-}       8.00     6.00    -0.3572         -1      g
H_2S           -2.00    -2.00    -0.1420         -1      b
HS^{-}         -1.00    -2.00     0.0650         -1      c
S^{2-}          0.00    -2.00     0.4763         -1      r
Solid S         0.00     0.00     0.0000          0      y
Prevalence region vertices
# dim ( pH, Eh[V]) ...
1  4 ( 0.00, 1.50)( 0.00, 0.32)( 1.84, 0.19)( 1.84, 1.50)
2  8 (14.00, 1.50)(14.00,-0.68)(13.90,-0.67)( 6.99,-0.21)
( 6.96,-0.21)( 6.91,-0.21)( 1.84, 0.19)( 1.84, 1.50)
3  5 ( 0.00,-1.50)( 0.00, 0.20)( 6.91,-0.21)( 6.99,-0.21)
( 6.99,-1.50)
4  4 ( 6.99,-1.50)(13.90,-1.50)(13.90,-0.67)( 6.99,-0.21)
5  4 (14.00,-1.50)(13.90,-1.50)(13.90,-0.67)(14.00,-0.68)
6  4 ( 0.00, 0.32)( 0.00, 0.20)( 6.91,-0.21)( 1.84, 0.19)
```

Fig. 10.7 shows the sulfur prevalence diagram. The reader should be warned that the central vertex, defined by the intersections of four boundary lines, actually consists of two very close, but not coincident, vertices as shown by the above results and by Fig. 10.8. The vertex has been used to calibrate the circular neighbor of intersection points, where to check the prevalence.

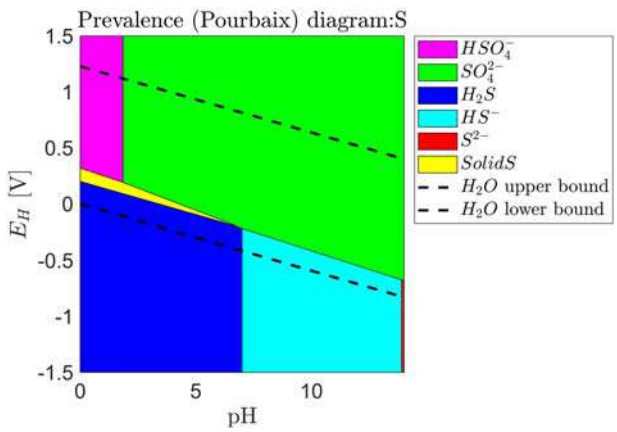


FIGURE 10.7

The sulfur prevalence diagram.

Fig. 10.8 is an enlargement of Fig. 10.7 which shows the pair of close intersections, denoted by red circles, at the boundary of the green and blue regions.

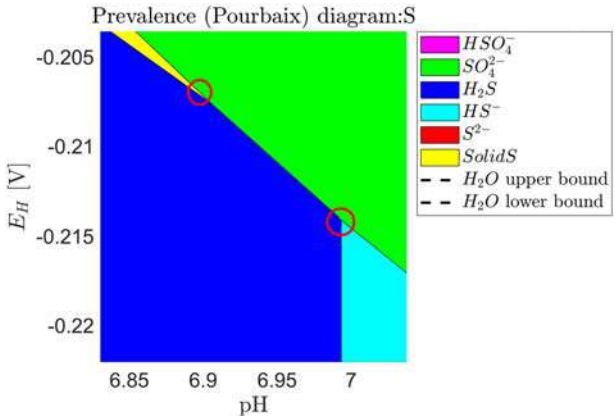


FIGURE 10.8

Enlargement of the sulfur prevalence diagram showing the pair of close central vertices.

10.13 Au/Cl in seawater

10.13.1 Description

The Earth's oceans contain tiny concentrations of gold, about 10^{-30} g/km^3. At 10^{-30} g/km^3, the Earth's oceans would hold 15,000 tons of gold [9].

In the past, a few people claimed to be able of economically recovering gold from sea water, but they were either mistaken or acted in an intentional deception. F. Haber (1868–1934) did research on the gold extraction from sea water in an effort to help Germany paying the World War I reparations [10]. Actually, his belief into a possible commercial extraction was founded on a largely overestimated analysis of seawater, and his efforts became unsuccessful.

In seawater, gold may be present as finely dispersed colloidal particles of metal and auric trihydroxide, $Au(OH)_3$. Gold may be also present in ionic form like chloroauric, $AuCl_4^-$, and auric ions, Au^{3+}. The prevalence diagram in Fig. 10.9 provides a detailed picture of the abundance of these species relative to the approximate concentration $[Cl^-] = 0.6$ mol/L of the chlorine ion in seawater and the total gold concentration of 0.01 mmol/L.

Reactions and electric potentials to be considered in order to build the stability diagram are the following ones.

```
Au+ + e- ---> Au(s)                      E° = + 1.692 volt
AuCl4- + 3 e- ---> Au(s) + 4 Cl-         E° = + 1.002 volt
Au(OH)3 + 3H+ + 3e- ---> Au + 3 H2O      E° = + 1.45 volt
Au3+ + 3e- --→ Au(s)                     E° = + 1.49 volt
```

10.13.2 Graphical and numerical results

The following table shows the data, from the sheet Au, for building the prevalence diagram and the list of the prevalence vertices. A pair of species has been found not prevalent (NP).

```
--- Prevalence diagram: Au
State table
State           phScale EhScale Potential[V] log(c)Scale Color
Au^{+}           0.00     1.00    -1.6920         -1      b
Au^{3+}          0.00     3.00    -1.4980         -1      c
Au(Cl)_4^{-}     0.00     3.00    -1.0020          2      g
Au(OH)_3         3.00     3.00    -1.4500          0      r
Metal Au         0.00     0.00     0.0000          0      y
Prevalence region vertices
# dim ( pH, Eh[V]) ...
1  0 NOT PREVALENT
2  0 NOT PREVALENT
3  4 ( 0.00, 1.50)( 0.00, 1.15)( 5.07, 1.15)( 5.07, 1.50)
4  4 (14.00, 1.50)(14.00, 0.62)( 5.07, 1.15)( 5.07, 1.50)
5  5 ( 0.00,-1.50)( 0.00, 1.15)( 5.07, 1.15)(14.00, 0.62)
(14.00,-1.50)
```

The data in the previous table correspond to the gold prevalence diagram in Fig. 10.9.

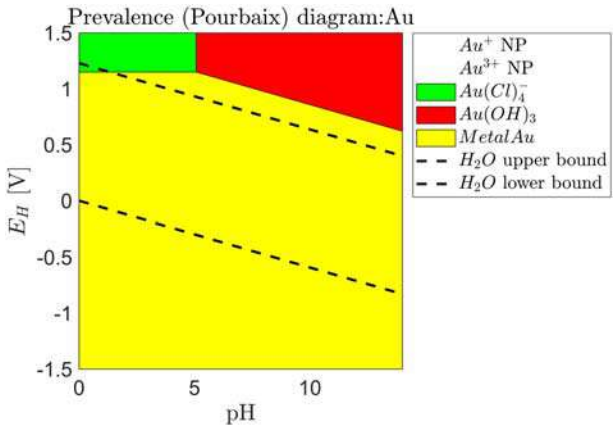


FIGURE 10.9

Prevalence diagram of Au/Cl in seawater.

References

[1] W. Stumm, J.J. Morgan, Aquatic Chemistry, Chemical Equilibria and Rates in Natural Waters, Third Edition, Wiley Interscience, 1996.
[2] Wikipedia, Standard state, Available from: https://en.wikipedia.org/wiki/Standard_state.
[3] R.H. Petrucci, F.G. Herring, J.D. Madura, C. Bissonnette, General Chemistry Principles and Modern Applications, Pearson Prentice Hall, 2010.
[4] Wikipedia, Standard electrode potential (data page), Available from: https://en.wikipedia.org/wiki/Standard_electrode_potential_(data_page).
[5] Chemistry LibreTexts, Electrochemistry. 2020, August 14. Retrieved April 3, 2021, Available from: https://chem.libretexts.org/@go/page/125386.
[6] Wikipedia, Pourbaix diagrams, Available from: https://en.wikipedia.org/wiki/Pourbaix_diagram, 2021.
[7] M. Pourbaix, Atlas of Electrochemical Equilibria in Aqueous Solution, National Association of Corrosion Engeneers, Huston, Texas, 1974, p. 644.
[8] Companion Website of the Book. Elsevier, 2021. Available from: https://www.elsevier.com/books-and-journals/book-companion/9780323913416.
[9] K. Kenison Falkner, J. Edmond, Gold in seawater, Earth Planet. Sci. Lett. 98 (2) (1990) 208−221. Available from: https://doi.org/10.1016/0012-821X(90)90060-B.
[10] F. Haber, Das Gold im Meerwasser, Z. für Angew. Chem. 40 (11) (1927) 303−314. Available from: https://doi.org/10.1002/ange.19270401103.

APPENDIX

Linear algebra

A

A.1 Bases of a vector space

A.1.1 Vectors, matrices, and bases

Linear algebra extends 3D *vector* properties and operations to a generic *vector space* $\mathscr{V}$, which is here assumed to be real and finite dimensional. In this case, $\mathscr{V} = \mathscr{R}^n$, where n is the space dimension, and a generic vector $\boldsymbol{v}$ (in bold font) is an n-ple of real numbers, which can be written either in column form or by using the inline convention [1], as follows

$$\boldsymbol{v} = [v_1, \ldots, v_k, \ldots v_n], \quad v = \begin{bmatrix} v_1 \\ \ldots \\ v_k \\ \ldots. \\ v_n \end{bmatrix}. \tag{A.1}$$

A *vector space* has the property that the sum of two vectors $\boldsymbol{v}_1 + \boldsymbol{v}_2 \in \mathscr{V}$ belongs to the space itself and the product $\alpha\boldsymbol{v} \in \mathscr{V}$ of a vector by a real scalar α, too.

When $n = 3$, the vector $\boldsymbol{v}$ can be interpreted as the coordinate vector $\boldsymbol{v} = [v_1, v_2, v_3]$ of the oriented segment from the origin O of a Cartesian frame to a generic point P, denoted by $\vec{v} = \overrightarrow{OP}$.

A set of m vectors of $\mathscr{R}^n$ can be collected into the columns of a *matrix* $V, n \times m$, as follows:

$$V = \begin{bmatrix} \boldsymbol{v}_1 & \ldots & \boldsymbol{v}_j & \ldots & \boldsymbol{v}_m \end{bmatrix} = \begin{bmatrix} v_{11} & \ldots & v_{1j} & \ldots & v_{1m} \\ & \ldots & & \ldots & \\ v_{i1} & \ldots & v_{ij} & \ldots & v_{im} \\ & \ldots & & \ldots & \\ v_{n1} & \ldots & v_{nj} & \ldots & v_{nm} \end{bmatrix}. \tag{A.2}$$

One may ask if it is possible to represent a generic vector as a linear combination of the matrix vectors, like

$$\boldsymbol{v} = x_1\boldsymbol{v}_1 + \ldots + x_k\boldsymbol{v}_k + \ldots + x_m\boldsymbol{v}_m = V\boldsymbol{x}. \tag{A.3}$$

This is always possible; but in general, the vector $\boldsymbol{x} \in R^m$ is not unique. The coefficients x_k are the coordinates of $\boldsymbol{v}$ with respect to the matrix V. The geometric meaning is that the m vectors of the matrix V generate a subspace $\mathscr{S}(V)$ in $\mathscr{R}^n$, which is the generalization of a plane passing through the origin in the 3D space. An important question is the dimension μ of this subspace, since in the case of $\mu = m$, the representation in Eq. (A.3) becomes unique. In other terms, a unique

vector $\boldsymbol{x} \in \mathscr{R}^m$ exists which premultiplied by V generates $\boldsymbol{v}$. One may say that the subspace $\mathscr{S}(V)$ is *spanned by the columns* of V and the columns become a *vector basis* of the subspace.

Which is the condition for the m vectors of the matrix V to span a subspace of dimension m? They must be *linearly independent*, in other terms, *no vector* $\boldsymbol{v}_k$ of the matrix V can be represented as a linear combination of the other vectors. Formally, the following inequality

$$\boldsymbol{v}_1 \neq x_2\boldsymbol{v}_2 + \ldots + x_k\boldsymbol{v}_k + \ldots + x_m\boldsymbol{v}_m \tag{A.4}$$

must hold, whichever be the $m-1$ coefficients. The geometric meaning is that each vector does not lie in the subspace spanned by the remaining vectors. In the 3D case, any of the three Cartesian axes $\{\vec{i}_1, \vec{i}_2, \vec{i}_3\}$ cannot be written in terms of the other two. Fig. A.1 shows three vectors $\vec{v}_1, \vec{v}_2, \vec{v}_3$ lying in the xy plane: they cannot be linear independent. In fact, only a pair of them is LI, and a pair of them plus $\vec{v}_4$, which does not lie in the xy plane, are LI in the 3D space, which is spanned by them.

Up to now, m is generic. We consider three specific cases:

1. $m = n$: if the n vectors $\boldsymbol{v}_k$, $k = 1, \ldots, n$ are linearly independent, they become a basis for the whole vector space $\mathscr{R}^n$, like the Cartesian axes for $\mathscr{R}^3$.
2. $m > n$: the m vectors cannot be linearly independent, like the three Cartesian axes of $\mathscr{R}^3$ and a fourth vector, which can be written as a linear combination of the three axes. Let us remark, that also in the case $m > n$, the number of linear independent vectors may be less than n, in which case they cannot span the whole vector space $\mathscr{R}^n$.
3. $m < n$: if the m vectors are linearly independent, they generate and span a subspace of dimension m, like x and y axes span the xy plane of dimension two, which is orthogonal to z axis in Fig. A.1.

A.1.2 Rank of a matrix

Does an algebraic index of the matrix V exist, which tells us whether the column vectors are or not linearly independent? Yes, it is the numerical rank of the matrix, $\rho = \text{rank}(V)$.

Rank test: the m column vectors of a matrix $V, n \times m$ are LI if only if $\text{rank}(V) = m$. When $\text{rank}(V) = min(m, n)$, the matrix is said to be *full rank*.

How to compute the rank of a matrix?

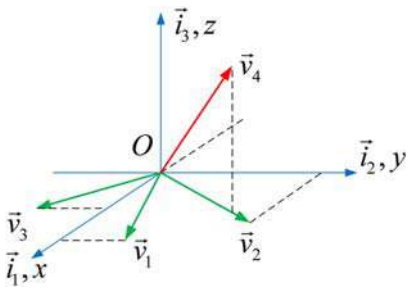


FIGURE A.1

Vectors in a 3D space.

1. In the case of a *square matrix* $A, n \times n$, *zero determinant*, $detA = 0$, implies that the matrix is not invertible and therefore $\text{rank}(A) < n$. However, zero determinant does not tell us the rank of the matrix, in other terms how many columns ρ are LI.
2. In terms of *determinant*, the rank of a generic matrix V is the size ρ of the largest minor of the matrix with nonzero determinant. A minor of a matrix of size ρ is any square matrix $M, \rho \times \rho$ which can be extracted by canceling rows and columns of the matrix V.
3. In general, the rank is not found by computing the minor determinants. A sound numerical way is the *QR factorization*, which, provided by the Matlab® function `[Q,R]=qr(V)`, converts a generic matrix $V, n \times m$, into the product $V = QR$ of an orthogonal matrix $Q, Q^TQ = I, n \times n$ and an upper triangular matrix $R, n \times m$, whose main diagonal values are nonnegative, namely $(R)_{ii} \geq 0, i = 1, \ldots, min(n, m)$ and are ordered in a decreasing way, $(R)_{11} \geq (R)_{22} \geq \ldots$. The number $\rho \geq 0$ such that $(R)_{\rho\rho} > 0, (R)_{\rho+1,\rho+1} = 0$ is the numerical rank of V. Of course, due to numerical errors, all the diagonal values may be nonzero, which asks for replacing the rank test by

$$(R)_{\rho\rho} > \epsilon, (R)_{\rho+1,\rho+1} \leq \epsilon, \tag{A.5}$$

where $\epsilon > 0$ is the tolerance. Actually, Matlab provides the specific function `k=rank(V)`, which employs a more sophisticated method than QR factorization.

Let us consider again the previous cases:

1. $m = n$: $\text{rank}(V) = n$ implies that the matrix is *full rank* and invertible, and $\boldsymbol{x} = V^{-1}\boldsymbol{v}$.
2. $m > n$: generically, the inequality $\text{rank}(V) < m$ holds, but if $\text{rank}(V) = n$, the equation $\boldsymbol{v} = V\boldsymbol{x}$ is satisfied by an infinite number of $\boldsymbol{x}$.
3. $m < n$: $\text{rank}(V) = m$ implies that the matrix is *full rank* but not invertible.

One may recognize that, given $\boldsymbol{v}$ and V, the following equation,

$$\boldsymbol{v} = V\boldsymbol{x} \tag{A.6}$$

is a set of n linear equations with m unknowns, the components of $\boldsymbol{x}$.

A.2 Orthogonal bases

The 3D Cartesian axes constitute an orthogonal basis. To this end, we recall the *scalar* or *inner product* between two vectors of the same dimensions $\boldsymbol{v}, \boldsymbol{u} \in \mathscr{R}^n$:

$$\boldsymbol{v}^T\boldsymbol{u} = |\boldsymbol{v}||\boldsymbol{u}|\cos\theta, \quad |\boldsymbol{v}| = \sqrt{\boldsymbol{v}^T\boldsymbol{v}}, \quad |\boldsymbol{u}| = \sqrt{\boldsymbol{u}^T\boldsymbol{u}}, \tag{A.7}$$

where $-\pi \leq \theta < \pi$ extends the concept of angle between two vectors to the space $\mathscr{R}^n$.

Orthogonality: two vectors $\boldsymbol{v}, \boldsymbol{u} \in \mathscr{R}^n$ are orthogonal if $\boldsymbol{v}^T\boldsymbol{u} = 0 \Leftrightarrow \theta = \pi/2 + k\pi$. An orthogonal basis for $\mathscr{R}^n$ is made of n vectors each other orthogonal. Let us collect the vectors in the matrix $V = [\boldsymbol{v}_1 \ldots \boldsymbol{v}_k \ldots \boldsymbol{v}_n]$. Orthogonality is tested by checking whether the following matrix products are diagonal

$$V^TV = VV^T = \text{diag}\left(|\boldsymbol{v}_1|^2, \ldots, |\boldsymbol{v}_k|^2, \ldots, |\boldsymbol{v}_n|^2\right), \boldsymbol{v}_k^T\boldsymbol{v}_j = 0, k \neq j. \tag{A.8}$$

If the previous matrix products are equal to the identity matrix I, the basis is said to be *orthonormal*, since the vector magnitudes become unitary. Cartesian axes are orthonormal. When the basis is orthogonal, the solution of the equation $\boldsymbol{v} = V\boldsymbol{x}, n \geq m$, is very simple

$$\begin{gathered} V^T\boldsymbol{v} = V^TV\boldsymbol{x} = \text{diag}\left(|\boldsymbol{v}_1|^2, \ldots, |\boldsymbol{v}_k|^2, \ldots, |\boldsymbol{v}_n|^2\right)\boldsymbol{x} \Rightarrow \\ \Rightarrow x_k = \left(\boldsymbol{v}_k^T\boldsymbol{v}\right)/|\boldsymbol{v}_k|^2 \end{gathered} . \tag{A.9}$$

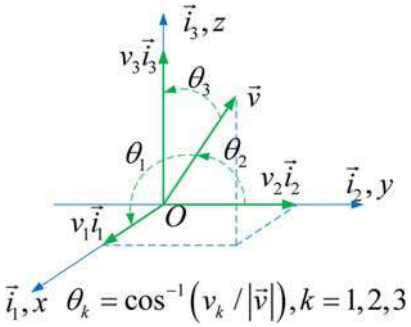


FIGURE A.2

Cartesian coordinates.

The scalar x_k is the coordinate of $\boldsymbol{v}$ in the orthogonal basis defined by the matrix V, like the Cartesian coordinates in $\mathscr{R}^3$ (see Fig. A.2). The inverse of the diagonal matrix of the vector magnitudes makes the matrix V and its basis orthonormal. As a warning, an orthonormal real matrix V, thus satisfying $V^T V = VV^T = I$, is often known as *orthogonal matrix*.

The concept of orthogonal vectors can be extended to a subspace $\mathscr{S}$ of $\mathscr{R}^n$. Here, we consider only a pair $\{\mathscr{S}_1, \mathscr{S}_2\} \in \mathscr{R}^n$ of orthogonal subspaces. Let us assume that their dimensions, which correspond to the number of the basis vectors, is the pair $\{m_1, m_2\}$ with the constraint that $m_1 + m_2 \leq n$. Two subspaces of $\mathscr{R}^n$ are each other orthogonal if two arbitrary vectors, one for each subspace, are always orthogonal:

$$\text{given the arbitrary pair } \boldsymbol{v}_1 \in \mathscr{S}_1, \boldsymbol{v}_2 \in \mathscr{S}_2 \Leftrightarrow \boldsymbol{v}_1^T \boldsymbol{v}_2 = 0. \tag{A.10}$$

Moreover, the sum of two arbitrary vectors, one for each subspace, defines a new subspace $\mathscr{S}$ of dimension $m = m_1 + m_2 \leq n$, which can be written as

$$\mathscr{S} = \mathscr{S}_1 + \mathscr{S}_2. \tag{A.11}$$

If $m = n$, the sum of the two subspaces completely spans the vector space $\mathscr{R}^n$, and each subspace can be indicated as the *orthogonal complement* of the other one with respect to $\mathscr{R}^n$. In $\mathscr{R}^3$, the axis z (a subspace of dimension one) is the orthogonal complement of the plane xy (a subspace of dimension two). The same property applies to the other axes.

Consider now a subspace $\mathscr{S} \in \mathscr{R}^n$ of dimension $m < n$ and a basis matrix $V, n \times m$, not necessarily orthogonal, but with $\text{rank}(V) = m$ (full rank). There should exist a subspace $\mathscr{S}^{oc}$ of dimension $m_{oc} = n - m$ which is the orthogonal complement of $\mathscr{S}$, such that $\mathscr{S} + \mathscr{S}^{oc} = \mathscr{R}^n$. It is straightforward to prove that any vector $\boldsymbol{x} \in \mathscr{S}^{oc}$ satisfies the system of m homogeneous linear equations

$$V^T \boldsymbol{x} = 0, \quad V^T = \left[\boldsymbol{v}_1^T, \ldots, \boldsymbol{v}_k^T \ldots, \boldsymbol{v}_m^T\right], \tag{A.12}$$

where $V^T, m \times n$ is the transpose of the basis matrix V. The orthogonal complement $\mathscr{S}^{oc}$ is referred to as the *nullspace* of the matrix V and possesses the dimension $\nu = n - m$.

It is well known that homogeneous linear equations possess a nontrivial solution (nonzero) if and only rank $V < n$. The rank complement $\nu = n - m$ is the dimension of the nullspace, which is

unique, and corresponds to the orthogonal complement of the subspace spanned by the columns of V. Thus the infinite solutions of Eq. (A.12) must be searched in the nullspace of V, indicated by $\mathcal{N}(V)$.

The simplest case considered in Chapter 2 is when the nullspace has minimal dimension, that is $\nu = n - m = 1$. In this case, all the solutions of Eq. (A.12) lie on a line passing through the origin and can be written as $\boldsymbol{x} = \lambda \boldsymbol{x}_0$, where λ, a scale factor, is a real scalar and $\boldsymbol{x}_0$ is any vector of the nullspace. To fix a value of λ, we have to add some constraint, as in Chapter 2, where V is integer and we look for a positive and integer vector of minimum norm.

We remark that for an integer matrix V, the equation solution can be found in the *rational vector space* $\mathcal{Q}^n$, which is *a vector space* since the sum of rational numbers is still rational, the scalar multiplication by a rational number remains rational and also the inverse of a rational number is rational. The latter property does not apply to integer numbers since their inverse is rational but not integer, except for the unit.

A subspace and its nullspace of a 3D space have been sketched in Fig. 2.1 of Chapter 2. In the same Chapter, Eq. (A.12) has been written as

$$AS\boldsymbol{x} = 0, \quad AS, \quad n \times m, \tag{A.13}$$

where A is a nonnegative integer matrix and the sign is given by the diagonal matrix S. Thus we write $V = SA^T$, and the rows of AS are the columns of V.

Actually, we have considered three kinds of equations depending on m.

1. *Standard redox equations* with $m = n - 1$ and $\text{rank}(AS) = m$, which implies a nullspace of dimension one;
2. *nonredox equations and redox exceptions* with $m = n$, but $\text{rank}(AS) = n - 1$, which again implies a nullspace of dimension one, and
3. *nonredox equations* with $m = n + 1$, but $\text{rank}(AS) = n - 1$, which again implies a nullspace of dimension one.

In summary, the rows of AS are always assumed to span a subspace of dimension $n - 1$, whichever be m. As a result, the nullspace turns out to always possess dimension one.

A.3 Eigenvalues and eigenvectors

When studying the state equations of dynamic systems as in Chapters 3 and 4 and in Appendix B, we encounter the real and *square* state matrix $A, n \times n$, which defines the stability properties of dynamic systems. Such properties depend on the eigenvalues and eigenvectors of A. We have seen in the previous paragraphs that the product $\boldsymbol{u} = A\boldsymbol{v}$ of a matrix $A, n \times n$ with a vector $\boldsymbol{v} \in \mathcal{R}^n$ produces another vector $\boldsymbol{u} \in \mathcal{R}^n$, which remains in the same vector space since the matrix is square. In general, $\boldsymbol{u} \neq \boldsymbol{v}$, because their norms (lengths, we always refer to the Euclidean norm $|\boldsymbol{u}| = \sqrt{\boldsymbol{u}^T\boldsymbol{u}}$, from the ancient Greek mathematician Euclide, around 300 BC) are different, formally $|\boldsymbol{u}| \neq |\boldsymbol{v}|$, and the vectors are not aligned, in which case we cannot write $\boldsymbol{u} = \lambda \boldsymbol{v}$, where λ is a real scalar. In other terms

$$\left|\boldsymbol{u}^T\boldsymbol{v} = |\boldsymbol{u}||\boldsymbol{v}|\cos\theta\right| < |\boldsymbol{u}||\boldsymbol{v}| \Leftrightarrow |\cos\theta| < 1. \tag{A.14}$$

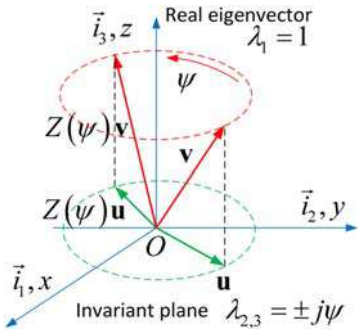


FIGURE A.3

Real eigenvector and invariant plane of $Z(\psi)$.

An eigenvector $\boldsymbol{v}$ is a real vector which premultiplied by A produces a vector $\lambda\boldsymbol{v}$ aligned with $\boldsymbol{v}$. We say that the *direction of* $\boldsymbol{v}$ is *invariant* upon transformation by A (or simply A-invariant). The scalar λ, which may have any sign, is the *eigenvalue* associated with $\boldsymbol{v}$. The *eigenvalue equation* is a homogeneous linear equation written as

$$(A - \lambda I)\boldsymbol{v} = 0, \tag{A.15}$$

where I denotes the identity matrix. In order to find a nonzero eigenvector $\boldsymbol{v} \neq 0$, the rank must be deficient, namely $\text{rank}(A - \lambda I) < n$, so that the square matrix $A - \lambda I$ possesses a nonzero nullspace. We have seen that deficient rank implies zero determinant, and therefore the following algebraic equation of degree n:

$$det(A - \lambda I) = \lambda^n + a_{n-1}\lambda^{n-1} + \ldots + a_1\lambda + a_0 = 0. \tag{A.16}$$

The above polynomial is known as the *characteristic polynomial* of A, and the equation is the characteristic (algebraic) equation of A. The roots $\lambda_k, k = 1, \ldots, n$ of the equation are the eigenvalues of A. Here we assume, for simplicity's sake, that the n eigenvalues are distinct; but in general, there may exist coincident eigenvalues. We consider two different cases.

1. *Real eigenvalues*: $\text{Re}\,\lambda_k = 0$, where Re denotes real part. A single real eigenvector $\boldsymbol{v}_k$, satisfying $(A - \lambda_k I)\boldsymbol{v}_k = 0$, exists, where, because of the homogeneous Eq. (A.15), the norm of the invariant vector is arbitrary and can be set equal to one: $|\boldsymbol{v}_k| = 1$.
2. *Complex conjugate eigenvalues:* $\lambda_k = |\lambda_k| exp\big(\pm j\big(\theta_k + \pi/2\big)\big)$, where θ_k is the angle from the imaginary axis, positive counterclockwise. The solution of Eq. (A.15) provides a pair of complex conjugate vectors, which are known as *complex eigenvectors*. Two questions arise: which is their geometric meaning? Which direction of $\mathscr{R}^n$ is A-invariant? The answer is that a single A-invariant direction does not exist any more, but an entire plane $\mathscr{P}$, spanned by the real and imaginary parts $\boldsymbol{a}_k = \text{Re}\,\boldsymbol{v}_k, \boldsymbol{b}_k = \text{Im}\boldsymbol{v}_k$ of the complex eigenvectors becomes A-invariant, in the sense that given a vector $\boldsymbol{u} = \alpha\boldsymbol{a}_k + \beta\boldsymbol{b}_k \in \mathscr{P}$, $A\boldsymbol{u} = \alpha A\boldsymbol{a}_k + \beta A\boldsymbol{b}_k \in \mathscr{P}$ remains in the plane. The proof is left to the reader.

The classical example is that of the Euler rotation $Z(\psi)$ [1] around say the z axis of Fig. A.1:

$$Z(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{A.17}$$

It is left to the reader to prove that the eigenvalues are the triple $\{1, j\psi, -j\psi\}$. The real eigenvector is the z axis [the Euler rotation axis which is invariant under the rotation of $Z(\psi)$], and the complex eigenvectors span the xy plane, where all the vectors are rotated by the rotation angle ψ but remain in the xy plane, which is therefore invariant under the rotation imposed by $Z(\psi)$ (see Fig. A.3).

Reference

[1] E. Canuto, C. Novara, L. Massotti, D. Carlucci, C. Perez Montenegro, Spacecraft Dynamics and Control: the Embedded Model Control Approach, Butterworth-Heinemann, Oxford, UK, 2018.

APPENDIX

Introduction to dynamic systems

B

B.1 State space equations: generalities

The concept of *state variable* derives from thermodynamics, where the macroscopic state of the matter is formulated through state variables, pressure P [MPa], volume $V\left[\mathrm{L}=\mathrm{dm}^3\right]$, and absolute temperature T [K]. Static relations between these variables are known as *equations of state*. See for instance the *ideal gas law* (6.5) and the *Van der Waals* equation (6.9) in Chapter 6.

As such, state variables are the *necessary and sufficient set* of time variables, collected in the vector $\boldsymbol{x}(t) = [x_1, \ldots, x_k, \ldots, x_n](t)$, for predicting the *free evolution (free response)* $\boldsymbol{x}(\boldsymbol{x}(t_0), t_0, t)$ of a dynamic system described by ordinary differential equations (ODE). The concept can be extended to partial differential equations. For instance, in quantum mechanics, the state $\Psi(\boldsymbol{v}, t)$ of a particle is a function not only of time t but also of the point $\boldsymbol{v} = [x, y, z]$ and is the solution of a partial differential equation known as the Schroedinger equation (see Appendix F).

The time evolution of a generic scalar state variable $x_k(t)$ is expressed by a *first-order differential equation*, like

$$\dot{x}_k(t) = f_k(x(t), t), x_k(t_0) = x_0, t \geq t_0, \tag{B.1}$$

where f_k is a differentiable function of the arguments which is locally bounded (in any arbitrary state interval) by a line (Lipschitz continuity). The evolution of the whole dynamic system is described by a set of n first-order equations (*the state* equations), n being the *order of the system:*

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), t), \boldsymbol{x}(t_0) = \boldsymbol{x}_0, t \geq t_0, \quad \boldsymbol{f} = [f_1, \ldots, f_k, \ldots, f_n], \tag{B.2}$$

where the vector $\boldsymbol{f}$ has properties similar to f_k.

An important case, known as *autonomous system*, is when $\boldsymbol{f}(\boldsymbol{x}(t))$ is time-independent, meaning that the free response $\boldsymbol{x}(\boldsymbol{x}(t_0), t)$ of Eq. (B.2) is *time-invariant*, being independent on the initial time t_0. In other terms, the shape of the free response remains the same, *whichever be the initial time*. Of course, the free response evolves in time but only because of the state evolution from the initial state $\boldsymbol{x}_0$. All the kinetic equations treated in Chapters 3 and 4 are autonomous. They become nonautonomous if either the rate constants are time-dependent through for instance the process temperature as in Section 3.8, or some exogenous time variable enters the equation. The reader should be warned that the process temperature might be taken as a further state variable, being the thermodynamic state of the process. Instead, environmental temperature should be taken as an exogenous variable.

The state trajectory is defined as the locus of the points $\boldsymbol{x}(t)$ in the state space, which in the case of order n is the vector space $\mathscr{R}^n$ of the real vectors of dimension n (see Appendix A). Only the second-order and third-order trajectories can be easily plotted.

An important point of the state space for autonomous equations is an *equilibrium point*, defined by

$$\boldsymbol{f}(\bar{\boldsymbol{x}}) = 0. \tag{B.3}$$

Several equilibrium points may exist in the case of nonlinear equations.

Two kind of exogenous variables (*input variables*) may be included in Eq. (B.2): (1) *command variables*, collected in the vector $\boldsymbol{u} = [u_1, \ldots, u_k, \ldots, u_{nu}]$, which can be manipulated by operators or control systems in order to regulate a dynamic process, (2) *disturbance variables*, collected into $\boldsymbol{d} = [d_1, \ldots, u_k, \ldots, d_{nd}]$, like for instance environmental temperatures, whose future evolution is *partly unknown*. The fact is that, in general, state prediction needs to know not only the initial state $\boldsymbol{x}_0$, but also exogenous variables, which being partly unknown may severely jeopardize the prediction fidelity. Thus we are forced to rewrite Eq. (B.2) as follows:

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{d}(t)), \boldsymbol{x}(t_0) = \boldsymbol{x}_0, t \geq t_0, \boldsymbol{f} = [f_1, \ldots, f_k, \ldots, f_n]. \tag{B.4}$$

The above equation turns out to become *autonomous* if the exogenous variables remain constant in time:

$$\boldsymbol{u}(t) = \boldsymbol{u}_0, \boldsymbol{d}(t) = \boldsymbol{d}_0, t \geq t_0. \tag{B.5}$$

This occurs, for instance, in the first equation of Eq. (3.14) in Chapter 3 (by forgetting the second equation), since $W = W_0$ is constant. Of course, if we consider both equations in Eq. (3.14), they are *autonomous* by construction, W being just a constant state variable.

B.2 Linear time invariant equations

B.2.1 Generalities

The solution or response of Eq. (B.4), either autonomous or not, can be explicitly computed only in few cases. The scalar Riccati equation in Eq. (3.24) (J. F. Riccati, 1676−1754), being one of these rare cases, will be solved below. As already mentioned in Chapter 3, the only alternative is the numerical integration by means of ODE solvers.

The only class which admits an explicit solution is that of the Linear Time Invariant (LTI) state equations, written in the following matrix form:

$$\begin{matrix} \dot{\boldsymbol{x}}(t) = A\boldsymbol{x}(t) + B\boldsymbol{u}(t) + B_d\boldsymbol{d}(t), \boldsymbol{x}(t) = \boldsymbol{x}_0 \\ \boldsymbol{y}(t) = C\boldsymbol{x}(t) \end{matrix}, \tag{B.6}$$

where $\boldsymbol{y}$ is the vector of the output variables, sometimes mentioned in Chapters 3 and 4. They usually denote measured variables or variables of interest (for instance to be graphically seen) which are combinations of state variables, and in our case, they are substance concentrations of interest. To simplify, we drop $\boldsymbol{d}$ from Eq. (B.6), but we leave $\boldsymbol{u}$, since we aim to provide the expression of the complete response, *free and forced*, as it was done in Eq. (3.15) of Chapter 3. Due to linearity,

the response is the superposition of the free response, only depending on the initial state, and of the forced response, only depending on the input/forcing variables, as follows

$$\boldsymbol{x}(t) = \exp(A(t-t_0))\boldsymbol{x}_0 \ \text{(free)} + \int_{t_0}^{t} \exp(A(t-\tau))\boldsymbol{B}\boldsymbol{u}(\tau)d\tau \text{(forced)}. \tag{B.7}$$

The integral is known as the *convolution integral* and it was already observed in Section 3.3.1 of Chapter 3, that is the sum of infinitesimal free responses starting at time τ with initial condition $Bu(\tau)d\tau$. The implication is that at any current time τ, a nonzero input signal of duration $d\tau$ changes the current state by adding a new infinitesimal free response. What is important to remark, is that the *input must be nonzero* and *must last a nonzero time interval*, although infinitesimal. Of course, the behavior descends from linearity.

The key parameter of the above response is the square real matrix $A, n \times n$, the *state matrix*, whereas the role of the input matrix B is that of scaling input signals and distributing them among equations. Matrix $B, n \times nu$, plays a key role in control system design [1], but not here.

The first property of state equations to be investigated is known as *stability*. In essence, we ask whether the response remains *bounded (stability)* or *not (instability)* during time. Since equations in Chapters 3 and 4 are autonomous, we restrict our investigation to the free response. Stability implies that the free response is bounded, whereas instability that diverges in magnitude. In the case of stability, we further ask whether the response decays in magnitude and asymptotically converges to zero (*asymptotic stability*, AS).

In the state equations of Chapter 3, AS applies to *irreversible reactions* where reactant concentration converges to zero (see Figures 3.2, 3.8, and 3.9). *Stability* applies to the total concentration of reactants and products which is conserved, and in reversible reactions to both reactants and products, whose concentrations tend to nonzero values. The mix of stability and AS is due to the presence of *conservation equations*, as already pointed out.

Stability criteria of LTI equations are intimately related to the eigenvalues of the state matrix A (see Appendix A). The eigenvalues of the square real matrix A are the solution of the characteristic equation

$$\det(\lambda I - A) = \lambda^n + a_{n-1}\lambda^{n-1} + \ \ldots \ + a_1\lambda + a_0 = 0, \tag{B.8}$$

whose roots $\lambda_k = \text{Re}\ \lambda_k \pm j\text{Im}\ \lambda_k, k = 1, \ldots, n$ may be real and *complex*. Whether complex, they occur in pair, root and conjugate. The pair of complex roots with negative real part is usually written as follows (see Appendix A)

$$\begin{gathered}\lambda_k = |\lambda_k|\exp\left(\pm j\left(\theta_k + \pi/2\right)\right) = |\lambda_k|(-\sin\theta_k \pm j\cos\theta_k) = \omega_k\left(-\zeta_k \pm j\sqrt{1-\zeta_k^2}\right) \\ \text{Re}\ \ \lambda_k = -\zeta_k\omega_k, \ \text{Im}\ \ \lambda_k = \sqrt{1-\zeta_k^2}\omega_k, \zeta_k = \sin\theta_k \geq 0, \omega_k = |\lambda_k| > 0\end{gathered}. \tag{B.9}$$

The dimensionless coefficient $\zeta_k = \sin\theta_k \geq 0$ is known as the *damping coefficient*. The complex eigenvalue magnitude $\omega_k = |\lambda_k|$ is known as the *angular frequency* and the unit is $\left[\text{rad/s}\right]$ (*rad* is the symbol of radian), like the unit of angular rates. We can write $\omega_k = 2\pi f_k$, where f_k is the *Fourier frequency* measuring the number of periods (cycles, complete turns) per second with unit [Hz = cycles/s]. It is of utmost importance to distinguish between ω_k and f_k. For instance, $f_k = 50$ Hz, a typical frequency of mains voltage, corresponds to $\omega_k \simeq 314$ rad/s. Since ω_k is the magnitude of complex eigenvalues, eigenvalues are assigned the same unit, namely $\left[\text{rad/s}\right]$.

To simplify, we assume that the roots of Eq. (B.8) are each other *distinct*. In this case, the necessary and sufficient conditions for stability and AS are as follows.

1. *marginal stability:* Re $\lambda_k \leq 0, k = 1, \ldots, n$ (nonpositive real part),
2. *asymptotic stability:* Re $\lambda_k < 0, k = 1, \ldots, n$ (negative real part).

In both cases $\tau_k = |\text{Re } \lambda_k|^{-1}$ is known as *time constant*, its unit being [s]. Instability occurs when at least one eigenvalue has positive real part. When A is *upper or lower triangular*, eigenvalues can be read on the main diagonal. We prefer the term *marginal stability* to the simpler *stability*, as it indicates that we are at the boundary of stability. Real eigenvalues with negative real part can be ordered with decreasing time constants $\infty > \tau_1 \geq \tau_2 \geq \ldots$. The largest time constant τ_1, corresponding to the real part closest to zero, defines the *dominant eigenvalue*, since the shape of the free response, for $t \to \infty$, converges to $\boldsymbol{v}_1 \exp(-t/\tau_1) = \boldsymbol{v}_1 \exp(\lambda_1 t)$, where $\boldsymbol{v}_1$ is the eigenvector associated with λ_1 and defined by the eigenvector homogeneous equation

$$(\lambda_1 I - A)\boldsymbol{v}_1 = 0. \tag{B.10}$$

The previous equation being homogeneous, implies that the eigenvector $\boldsymbol{v}_1$ is defined up to the magnitude, which therefore can be taken to be unitary: $|\boldsymbol{v}_1| = 1$.

Stability properties of LTI systems are *large-scale properties* since they apply to the whole state space. This is due to linearity and to the fact that the equilibrium point $\overline{\boldsymbol{x}}$ is unique and equal to zero, unless zero eigenvalues exist (conservation equations). But also in the latter case, the equilibrium point can be shifted to the origin by defining the new state $\delta\boldsymbol{x} = \boldsymbol{x} - \overline{\boldsymbol{x}}$.

Before the advent of numerical computers, solution of algebraic equations like Eq. (B.8) of degree greater than two was a hard problem (also in the case of three and four degrees). Luckily AS only requires to know whether the eigenvalues [the roots of the characteristic equation Eq. (B.8)] have or not negative real parts. The issue was solved by E.J. Routh (1831–1907) in 1876 by proposing an iterative method only based on the coefficients of Eq. (B.8). Here we provide the criteria of the second and third-degree equations, leaving the generic method to Wikipedia [2]:

$$\begin{array}{l} \text{second degree, } \lambda^2 + a_1\lambda + a_0 = 0 \Rightarrow a_1 > 0, a_0 > 0 \\ \text{third degree, } \lambda^3 + a_2\lambda^2 + a_1\lambda + a_0 = 0 \Rightarrow a_2 > 0, a_0 > 0, a_2 a_1 > a_0 \end{array}. \tag{B.11}$$

The third-degree criterion was employed in Chapter 4, Section 4.4.

B.2.2 Examples of linear time invariant dynamic systems

Pure integrator: $\dot{x}(t) = bu(t), x(t_0) = x_0$. The state matrix is $A = 0 \to \lambda_1 = 0$, implying stability and conservation law; the initial state is conserved under $u(t) = 0$. Total response:

$$x(t) = x_0 + \int_{t_0}^{t} u(\tau)d\tau. \tag{B.12}$$

The equilibrium point $\overline{x}$ of the autonomous equation $\dot{x}(t) = 0, x(t_0) = x_0$ is any and may be taken equal to the initial state; it can be shifted to the origin by defining the new state $\delta x = x - \overline{x}$.

AS equation in parallel to a conservation law as in Eq. (3.10) of Chapter 3: the first equation is AS, the second equation is marginally stable:

$$\dot{\boldsymbol{x}}(t) = \begin{bmatrix} -k & 0 \\ 0 & 0 \end{bmatrix} \boldsymbol{x}(t),\ \boldsymbol{x}(t_0) = \boldsymbol{x}_0 \rightarrow \lambda_1 = -k, \lambda_2 = 0 \tag{B.13}$$

The free response holds: $x_1(t) = \exp(-k(t-t_0))x_{10}, x_2(t) = x_{20}$.

AS equation driven by a conservation law as in Eq. (3.14) of Chapter 3: the first equation is AS, the second equation is marginally stable:

$$\dot{\boldsymbol{x}}(t) = \begin{bmatrix} -k & k_2 \\ 0 & 0 \end{bmatrix} \boldsymbol{x}(t),\ \boldsymbol{x}(t_0) = \boldsymbol{x}_0 \rightarrow \lambda_1 = -k, \lambda_2 = 0 \tag{B.14}$$

The total response can be found in Eq. (3.16).

Series of AS equations in parallel with a conservation law as in Eq. (3.34) of Chapter 3:

$$\dot{\boldsymbol{x}}(t) = \begin{bmatrix} -k_1 & 0 & 0 & 0 \\ k_1 & -k_2 & 0 & 0 \\ 0 & k_2 & -k_3 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \boldsymbol{x}(t),\ \boldsymbol{x}(t_0) = \boldsymbol{x}_0 \rightarrow \lambda_k = -k_k, k = 1, 2, 3, \lambda_4 = 0 \tag{B.15}$$

The last state variable holds $x_4 = W$ as in Eq. (3.35).

Second-order system with imaginary eigenvalues: $\lambda_{k,k+1} = \pm j\omega$, where ω $[\mathrm{rad/s}]$ is the natural angular frequency:

$$\dot{\boldsymbol{x}}(t) = \begin{bmatrix} 0 & -k_1 \\ k_2 & 0 \end{bmatrix} \boldsymbol{x}(t),\ \boldsymbol{x}(t_0) = \boldsymbol{x}_0 \Rightarrow \omega = \sqrt{k_1 k_2} \tag{B.16}$$

A similar equation was found in Eq. (4.29) of Chapter 4, by writing the perturbation of the Lotka–Volterra equation from the equilibrium point. The reader is asked to prove that the periodic free response with period $P = 2\pi/\omega$ holds

$$\begin{aligned} x_1(t) &= x_{10}\cos(\omega t) - kx_{20}\sin(\omega t), \quad k = \sqrt{k_1/k_2} \\ kx_2(t) &= x_{10}\sin(\omega t) + kx_{20}\cos(\omega t) \end{aligned}. \tag{B.17}$$

Eq. (B.17) is a *conservation equation*. Indeed, eigenvalues have zero real part. What is conserved is the weighted Euclidean norm of the state vector, namely

$$r^2 = |\boldsymbol{x}(t)|^2_{\{1,k\}} = x_1(t)^2 + (kx_2(t))^2 = x_{10}^2 + (kx_{20})^2, \tag{B.18}$$

where the state variables have been scaled by the pair $\{1, k\}$. The state equation is only *marginally stable*, and the free response does not converge to the equilibrium point but remains bounded, and the plane trajectory $\{x_1, kx_2\}$ is a circle of radius r [see Eq. (B.18)] around the origin. This type of closed trajectory (known as *closed orbit*) is different from *limit cycles* to be defined in Section B.3. In fact, it is an *isolated trajectory*, and each initial state has its own trajectory where the free response remains confined. Fig. B.1 adapted from [3], shows the orbits of the nonlinear Lotka–Volterra equations around the equilibrium point at the intersection of the gray color lines. They are accompanied by the orbits of the perturbed Eq. (B.16), starting at the same initial conditions. The nonlinear orbits progressively deform as the orbit size increases.

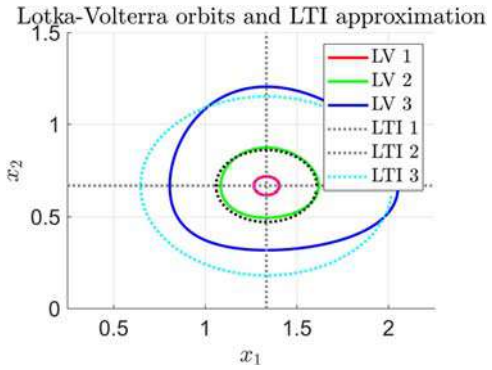


FIGURE B.1

Lotka–Volterra (LV) closed orbits and linear time invariant (LTI) closed orbits.

B.3 Perturbation equations and local stability of nonlinear systems

B.3.1 Perturbation equations

The free response of a LTI equation which is AS, namely, the eigenvalues of the state matrix A have negative real part, tends to an equilibrium point which is unique and equal to the state space origin. In fact A is invertible and $0 = A\overline{\boldsymbol{x}} \Rightarrow \overline{\boldsymbol{x}} = 0$.

In the case of autonomous nonlinear systems, large-scale stability properties cannot be easily demonstrated and in general the state space partitions into different stability regions depending on the number and location of equilibrium points. The stability pattern may be very complicated, since also equilibrium trajectories (*limit cycles*) may exist, which, if stable, capture the neighboring trajectories, as in Figure 4.12, left, and Figure 4.13, left, of Chapter 4. Limit cycles are nonisolated closed trajectories where other trajectories converge to (stable limit cycle) or depart from (unstable limit cycle). To be simple, we restrict our analysis to stability around single equilibrium points.

The method is to write the perturbed equation around the equilibrium $\overline{\boldsymbol{x}}$ in terms of the perturbation $\delta\boldsymbol{x} = \boldsymbol{x} - \overline{\boldsymbol{x}}$. In the general case, given the autonomous equation

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}),\ \boldsymbol{x}(t_0) = \boldsymbol{x}_0, \tag{B.19}$$

we write

$$\delta\dot{\boldsymbol{x}}(t) = A(\overline{\boldsymbol{x}})\delta\boldsymbol{x}(t),\ \delta\boldsymbol{x}(0) = \delta\boldsymbol{x}_0,\ A(\overline{\boldsymbol{x}}) = \partial\boldsymbol{f}(\overline{\boldsymbol{x}})/\partial\boldsymbol{x}, \tag{B.20}$$

where $A(\overline{\boldsymbol{x}})$ is the Jacobian matrix (C.G. Jacobi, 1804–51) of the vector function $\boldsymbol{f}(\boldsymbol{x})$ with respect to the components of $\boldsymbol{x}$, computed at the equilibrium point. Stability of the perturbed trajectories around the equilibrium points depend on the eigenvalues of $A(\overline{x})$. The following statements hold:

1. *Asymptotic stability:* the equilibrium point is AS if all the eigenvalues have negative real part. This means that, given a small perturbation $\delta\boldsymbol{x}_0$, the trajectory will converge to the equilibrium point, and the free response of Eq. (B.20) will converge to zero.

2. *Instability*: The perturbed trajectory does not converge to the equilibrium point if at least one eigenvalue with positive real part exists.
3. *Eigenvalues with zero real part*: if at least one eigenvalue exists with zero real part, nothing can be said about stability. We need to employ other means.

Sometimes as in Section 3.6 of Chapter 3, perturbation is taken with respect to nonequilibrium points. A further significant case will be mentioned in Section B.3.3. In the case of Section 3.6 of Chapter 3, the perturbed equation was made to include a suitable constant forcing term to reestablish equilibrium as follows

$$\delta\dot{\boldsymbol{x}}(t) = A(\overline{\boldsymbol{x}})\delta\boldsymbol{x}(t) + \overline{\boldsymbol{u}},\ \ \delta\boldsymbol{x}(0) = \delta\boldsymbol{x}_0,\ \ A(\overline{\boldsymbol{x}}) = \partial\boldsymbol{f}(\overline{\boldsymbol{x}})/\partial\boldsymbol{x},\ \ \overline{\boldsymbol{u}} = \boldsymbol{f}(\overline{\boldsymbol{x}}). \tag{B.21}$$

An equilibrium point of this kind is known as *forced equilibrium*, to be distinguished from the equilibrium points of Eq. (B.3), known as *natural equilibriums*.

The perturbed equation (3.28) of Chapter 3 is AS. The perturbed equation of Eq. (3.40) is derived here as an example. Let us rewrite Eq. (3.40) in a simplified form, free of the conservation equations, by defining $x = [A], D_0 = [D]_0$:

$$\begin{aligned} \dot{x}(t) &= -2k_1x^2 - k_2x + k_2S, x(0) = x_0 \\ \dot{S}(t) &= -k_3D_0S(t) + k_3D_0x(t), S(0) = S_0 \end{aligned} \tag{B.22}$$

Instead of the equilibrium point $\overline{x} = 0, \overline{S} = 0$ which is reached asymptotically, we consider the short-term equilibrium $\overline{x} = A_{\text{ShortTerm}}$ in Eq. (3.34), which, we remark, is not an equilibrium point of the whole equation and therefore requires an additional term as follows:

$$\begin{bmatrix} \delta\dot{x} \\ \delta\dot{S} \end{bmatrix}(t) = \begin{bmatrix} -(4k_1\overline{x} + k_2) & k_2 \\ k_3D_0 & -k_3D_0 \end{bmatrix}\begin{bmatrix} \delta x \\ \delta S \end{bmatrix}(t) + \begin{bmatrix} 0 \\ k_3D_0(\overline{x} - S_0) \end{bmatrix}, \tag{B.23}$$

where

$$\delta x = x - \overline{x}, \delta S = S - S_0. \tag{B.24}$$

As already said, the constant forcing term $k_3D_0(\overline{x} - S_0)$ is due to the fact that the short-term equilibrium is not a natural equilibrium but a *forced equilibrium*. We are interested in the eigenvalues, whose approximated expression is as follows

$$\begin{aligned} \lambda_1 &\simeq -(4k_1A_{\text{ShortTerm}} + k_2) \simeq -0.165 \Rightarrow \tau_1 = |\lambda_1|^{-1} \simeq 6.06\text{s} \\ \lambda_2 &\simeq 4k_1A_{\text{ShortTerm}}k_3D_0/|\lambda_1| \simeq -0.00151 \Rightarrow \tau_2 = |\lambda_2|^{-1} \simeq 662\text{s} \end{aligned}, \tag{B.25}$$

and in the eigenvector $\boldsymbol{v}_{1,\text{ShortTerm}} \simeq [-1, 0]$ of the dominant eigenvalue λ_1 (the dashed black line in Fig. B.2). The time constants in Eq. (B.25) are close to short-term (ST) and long-term (LT) time constants $\tau_{\text{ShortTerm}}$ and τ_3 in Section 3.6.1 of Chapter 3. The long-term equilibrium is the origin, which leads to the perturbation equation

$$\begin{bmatrix} \delta\dot{x} \\ \delta\dot{S} \end{bmatrix}(t) = \begin{bmatrix} -k_2 & k_2 \\ k_3D_0 & -k_3D_0 \end{bmatrix}\begin{bmatrix} \delta x \\ \delta S \end{bmatrix}(t). \tag{B.26}$$

The eigenvalues hold

$$\lambda_1 = 0,\ \ \lambda_2 = -(k_2 + k_3D_0) = -0.042\ \text{rad/s}. \tag{B.27}$$

The fact that one eigenvalue is zero implies that the AS dynamics is first-order. The dominant eigenvector $\mathbf{v}_{1,\text{LongTerm}}$ associated with λ_1 holds $\mathbf{v}_{1,\text{LongTerm}} = \left[-1/\sqrt{2}, -1/\sqrt{2}\right]$ (the dashed cyan

line in Fig. B.2). Fig. B.2 shows a pair of trajectories converging to their long-term equilibrium concentrations: $(0, 0)$ for the pair $\{[A], S\}$, and $(0, 1.2)$ for the pair $\{[A], [C]\}$. The former trajectory includes the dominant eigenvectors of the Eq. (B.23) (ST equilibrium) and Eq. (B.27) (LT equilibrium). The eigenvectors become the asymptotes of the trajectories as they approach the AS equilibrium points (known as *stable nodes*, see below).

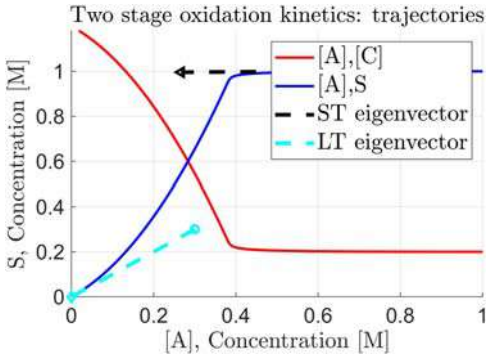


FIGURE B.2

Trajectories of the pairs $[A], [C]$ (red) and $[A], S$ (blue) converging to long-term AS equilibriums. The latter trajectory includes the dominant eigenvectors of the ST and LT equilibriums.

B.3.2 Second-order equilibrium points

State space trajectories of second-order systems allow their behavior to be visualized in the neighborhood of equilibrium points. Let us consider the second-order version of the perturbation Eq. (B.21):

$$\begin{bmatrix} \delta\dot{x}_1 \\ \delta\dot{x}_2 \end{bmatrix}(t) = \begin{bmatrix} \partial f_1\ (\bar{\boldsymbol{x}})/\partial x_1 & \partial f_1\ (\bar{\boldsymbol{x}})/\partial x_2 \\ \partial f_2\ (\bar{\boldsymbol{x}})/\partial x_1 & \partial f_2\ (\bar{\boldsymbol{x}})/\partial x_2 \end{bmatrix} \begin{bmatrix} \delta x_1 \\ \delta x_2 \end{bmatrix}(t) = A(\bar{\boldsymbol{x}})\delta\boldsymbol{x}(t). \tag{B.28}$$

The pair of *nullclines* employed in Section 4.6.2 of Chapter 4 have been obtained from the locus of points of the two equations $f_1(\boldsymbol{x}) = 0$, $f_2(\boldsymbol{x}) = 0$, whose intersections define the equilibrium points. Five types of the second-order equilibrium points exist, depending on the pair of eigenvalues $\{\lambda_1(\bar{\boldsymbol{x}}), \lambda_2(\bar{\boldsymbol{x}})\}$. The different types can be distinguished from the sign and value of the determinant $\det A = \lambda_1\lambda_2$ and of the trace $\mathrm{tr}A = \lambda_1 + \lambda_2$.

1. *Stable node*: real and negative eigenvalues, $\mathrm{tr}A < 0, (\mathrm{tr}A)^2 > 4\ \det A$ (see Fig. B.2). The node is asymptotically approached along the dominant eigenvector.
2. *Unstable node:* real and positive eigenvalues, $\mathrm{tr}A > 0, (\mathrm{tr}A)^2 > 4\ \det A$ (see Fig. B.4).
3. *Saddle point:* real and opposite sign eigenvalues, $\det A < 0$. No state equation in this textbook has a second-order saddle point.
4. *Stable spiral (focus):* complex eigenvalues with negative real part, $\mathrm{tr}A < 0, (\mathrm{tr}A)^2 > 4\ \det A$. Since eigenvectors are complex, no privileged direction exists to approach the AS equilibrium point, which is reached along a spiral, in the plane spanned by real and imaginary parts of the eigenvectors, in this case the 2D state space. Given the complex eigenvalues

$\lambda_{1,2} = -\left(\zeta \pm j\sqrt{1-\zeta^2}\right)\omega_n, \zeta > 0$, the spiral radius $r(t) = |x(t)|$ is proportional to $\exp(-\zeta\omega_n(t-t_0))$ (see Fig. B.3, left).

5. *Unstable spiral:* complex eigenvalues with positive real part, $\mathrm{tr}A < 0, (\mathrm{tr}A)^2 < 4\,\det A$ (see Fig. B.3, right).

All the five types of equilibrium points belong to the class of *hyperbolic equilibrium points*, defined as the equilibrium points whose perturbation Eq. (B.21) has no eigenvalue with zero real part. It has been proved [4] that nonlinear trajectories around hyperbolic equilibrium points are equivalent in the topological sense to the trajectory of the perturbation equation (they can be transformed into each other). The property implies that hyperbolic equilibrium points are robust, in other terms, converging or departing trajectories are insensitive to small perturbations. The property is related to the already mentioned fact that the eigenvalues of the perturbed equation decide asymptotic stability and instability of the nonlinear trajectories around the equilibrium point, unlike the case of eigenvalues with zero real part.

Fig. B.3, left, shows a stable spiral focus from the Oregonator model in Section 4.6.3 of Chapter 4. The equilibrium point has been selected on the $\dot{x}$ nullcline by fixing $f = 0.52$. The initial perturbation from the equilibrium is visible. Fig. B.3, right, shows an unstable spiral focus (in cyan color) leading to a stable limit cycle (in blue color), extracted from the Case 2 of the Briggs–Rauscher kinetics in Section 4.5 of Chapter 4.

Fig. B.4 shows an unstable node from the Oregonator model in Section 4.6.3 of Chapter 4. The equilibrium point has been selected on the $\dot{x}$ nullcline by fixing $f = 1$.

Second-order systems exhibit also equilibrium trajectories, which occur when the eigenvalue real part is zero. Two cases must be distinguished.

1. *Zero eigenvalues*, $\lambda_{1,2} = 0$. The case corresponds to the seminal and *nontrivial* Newton's state equation (I. Newton, 1642–1726)

$$\begin{bmatrix} \dot{x} \\ \dot{v} \end{bmatrix}(t) = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} x \\ v \end{bmatrix}(t) + \begin{bmatrix} 0 \\ b \end{bmatrix}u(t), \begin{bmatrix} x \\ v \end{bmatrix}(t_0) = \begin{bmatrix} x_0 \\ v_0 \end{bmatrix}. \tag{B.29}$$

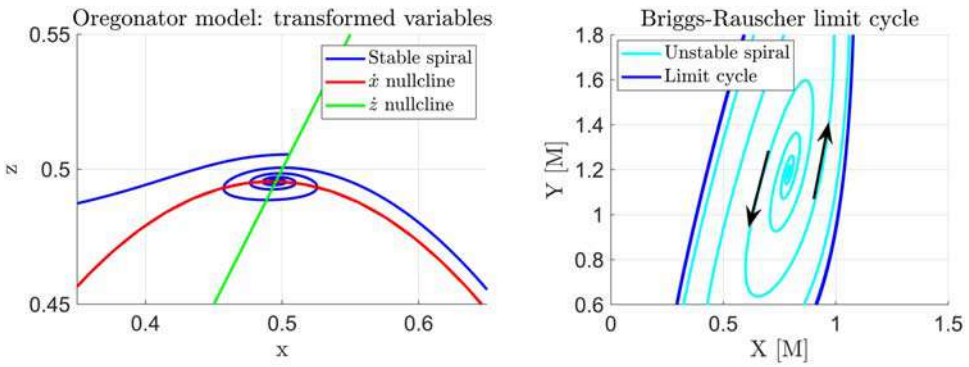


FIGURE B.3

(Left) A stable spiral around the equilibrium point (Oregonator model). (Right) Unstable spiral leading to a limit cycle (Briggs–Rauscher kinetics).

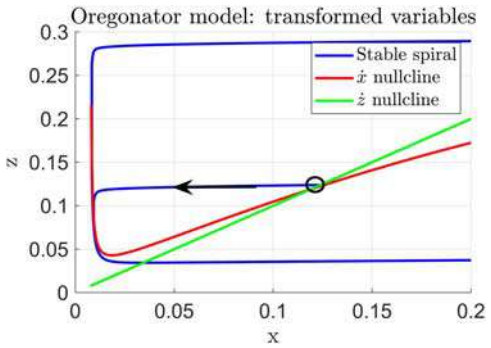


FIGURE B.4

Oregonator model: unstable node (black circle) leading to a stable limit cycle.

The equilibrium point is any point $\overline{\boldsymbol{x}} = [x_0, 0]$ (zero velocity), but the equilibrium is *unstable*, since under $u(t) = 0$, any small perturbation v_0 of the velocity triggers an *unbounded trajectory* along a line passing through $\boldsymbol{x}_0 = [x_0, v_0]$ and parallel to the x axis:

$$x(t) = x_0 + v_0(t - t_0). \tag{B.30}$$

It is left to the reader to prove that *only a single-real eigenvector* exists equal to $\boldsymbol{v}_1 = [1, 0]$, parallel to the unbounded trajectory lines.

2. *Imaginary eigenvalues*, $\lambda_{1,2} = \pm j\omega_n$. The case corresponds to Eq. (B.16) with $k_1 = k_2 = \omega_n$. The equation is marginally stable and the trajectories in the coordinates $\left[x, v/\omega_n\right]$ are circles (closed/periodic orbits like those in Fig. B.1) with their center in the origin. The radius $r = \sqrt{x_0^2 + \left(v_0/\omega_n\right)^2}$ is given by initial conditions, meaning that each initial condition has its own orbit, different from the other ones. Eigenvectors are imaginary, which is coherent with the absence of a privileged trajectory direction.

Both types of equilibriums are *nonhyperbolic equilibriums*, a class which includes closed/periodic orbits like the *limit cycles* of the Briggs–Rauscher and Belousov–Zhabotinsky kinetics (in the form of the Oregonator model) in Sections 4.5 and 4.6 of Chapter 4, but also chaotic orbits like the *strange attractor* [5] of the Lorenz equation [6] (E. Lorenz, 1917–2008), which has the vague form of a butterfly. As already mentioned in Section 4.1 of Chapter 4, also chemical reactions have been found leading to chaotic trajectories (not treated in the book). Nonhyperbolic equilibriums are very sensible to initial conditions and parameter perturbations.

B.3.3 Lyapunov exponents

Closed orbits and limit cycles are periodic trajectories in the state space. This implies that given the point $\boldsymbol{x}(t)$, the state vector after one or more periods kP repeats $\boldsymbol{x}(t)$, that is

$$\boldsymbol{x}(t + kP) = \boldsymbol{x}(t), k = 0, 1, 2, \ldots \tag{B.31}$$

Let us consider the generic nonlinear autonomous state equation $\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}), \boldsymbol{x}(t_0) = \boldsymbol{x}_0$ and the linear perturbation equation around a generic point $\boldsymbol{x}$ [not to be treated as an equilibrium point and therefore without the additional forcing term of Eq. (B.23)]

$$\delta\dot{\boldsymbol{x}}(t) = A(\boldsymbol{x}(t))\delta\boldsymbol{x}(t), \delta\boldsymbol{x}(t_0) = \delta\boldsymbol{x}_0, A(\boldsymbol{x}(t)) = \partial\boldsymbol{f}(\boldsymbol{x}(t))/\partial\boldsymbol{x}, \tag{B.32}$$

where $A(t) = A(\boldsymbol{x}(t))$, $n \times n$, is the Jacobian matrix computed at $\boldsymbol{x}$. Eq. (B.32) is known as the *tangent dynamic system* at $\boldsymbol{x}$. Now, if the state vector $\boldsymbol{x}(t)$ of the nonlinear system asymptotically converges to a stable limit cycle, one expects that the tangent Eq. (B.32), which is *linear but time varying* due to $\boldsymbol{x}(t)$ in the state matrix, tends to be *periodic*.

Unfortunately, the free response $\boldsymbol{x}(t, t_0) = \Phi(t, t_0)\boldsymbol{x}(t_0)$ of time varying linear systems cannot be expressed as in Eq. (B.7) by means of the state matrix exponential, except in the case of *linear periodic systems*, whose state matrix $A(t)$ satisfies $A(t + kP) = A(t)$, given the period P [1]. The matrix $\Phi(t, t_0)$ can be obtained by simulating n parallel state equations like Eq. (B.32), with unitary initial conditions defined by the identity matrix $I = [\boldsymbol{e}_1 \ldots \boldsymbol{e}_k \ldots \boldsymbol{e}_n]$, where $\boldsymbol{e}_k = [0, \ldots, 1, \ldots, 0]$, such that

$$\Phi(t, t_0) = [\boldsymbol{x}_1(t, t_0) \ldots \boldsymbol{x}_k(t, t_0) \ldots \boldsymbol{x}_n(t, t_0)], \quad \boldsymbol{x}_k(t, t_0) = \Phi(t, t_0)\boldsymbol{e}_k. \tag{B.33}$$

The orthonormal axes $\boldsymbol{e}_k$ of I are the Cartesian axes of a generic initial perturbation $\delta\boldsymbol{x}(t_0)$. We are interested to find how the axes are asymptotically transformed, namely rotated and their unitary length amplified/reduced by $\Phi(t, t_0)$. Asymptotic amplification/reduction is clearly related to stability properties. For instance, in the case of a 3D stable limit cycle (like those of the Briggs–Rauscher and Oregonator kinetics in Chapter 4), we are expecting a bounded perturbation tangent to the trajectory and vanishing perturbations in the orthogonal plane (see Fig. B.5). In geometric terms, the initial unitary 3D sphere is expected to be transformed into a degenerate 3D

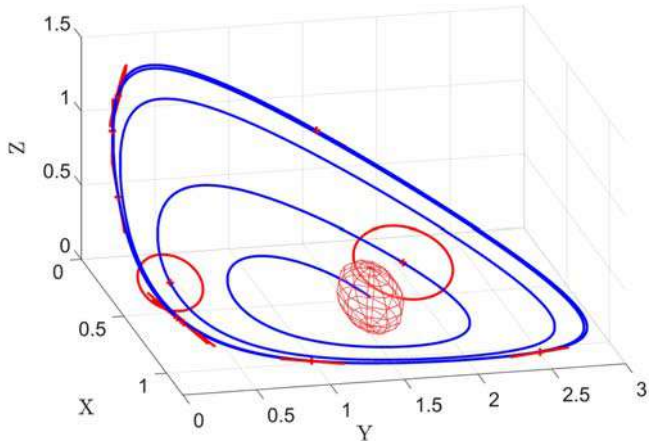


FIGURE B.5

Limit cycle of the Briggs–Rauscher kinetic (case 2) with initial sphere and progressively degenerating ellipsoids of perturbations.

ellipsoid made by a segment of nonzero length, tangent to the limit cycle. The picture is completed by a chemical kinetics converging to an asymptotically stable equilibrium, in which case the unit sphere is progressively shrunk to a single point. More complex situations occur when perturbations tend to become unbounded as in the already mentioned strange attractors [5].

Fig. B.5 shows the progressively degenerating 3D perturbation ellipsoids of the Briggs−Rauscher kinetics in Chapter 4 (case 2). We start from a unit sphere centered at the initial point perturbed from the unstable equilibrium. Along the unstable spiral fairly soon 3D ellipsoids degenerate into 2D ellipses which further degenerates into a single tangent segment as soon as the limit cycle is approached. The reason lies in the *Lyapunov exponents* to be explained below (A.M. Lyapunov, 1857−1908). They are plotted for the limit cycle of Fig. B.5 in Fig. B.6, left.

The time sequence of ellipsoids (in the generic case of dimension n) is defined by the orthogonal real eigenvectors and nonnegative real eigenvalues (see Appendix A) of the symmetric matrix

$$\begin{aligned} G(t,t_0) &= \Phi^T(t,t_0)\Phi(t,t_0) = V^T(t,t_0)S^2(t,t_0)V(t,t_0) \\ S^2(t,t_0) &= \mathrm{diag}\left(s_1^2 \geq \ldots s_k^2 \geq \ldots s_n^2\right),\ V^T(t,t_0)V(t,t_0) = I \end{aligned} \tag{B.34}$$

By assuming the existence of the limit

$$\begin{aligned} G_\infty &= \lim_{t\to\infty} G(t,t_0) = V_\infty^T S_\infty^2 V_\infty \\ S_\infty &= \mathrm{diag}\left(s_{1,\infty} \geq \ldots s_{k,\infty} \geq \ldots s_{n,\infty}\right), \end{aligned} \tag{B.35}$$

the square root eigenvalue $s_{k,\infty}$ can be expressed as the real eigenvalues of an exponential matrix, that is

$$s_{k,\infty} = \lim_{t\to\infty} \exp\left(\mu_k(t-t_0)\right) \to \mu_k = \lim_{t\to\infty} \log(s_k)/(t-t_0) \tag{B.36}$$

where $\mu_k, k = 1,\ldots,n$ is known as *Lyapunov exponent*, which may be negative (asymptotic stability), zero (marginal stability) or positive (instability).

There is no space to give an account of the Lyapunov exponent algorithm (see Reference [7]). We only mention that the pair of matrices $\{V_\infty, S_\infty\}$ are periodically computed via a QR factorization (see Appendix A) to build up a converging average of the exponents.

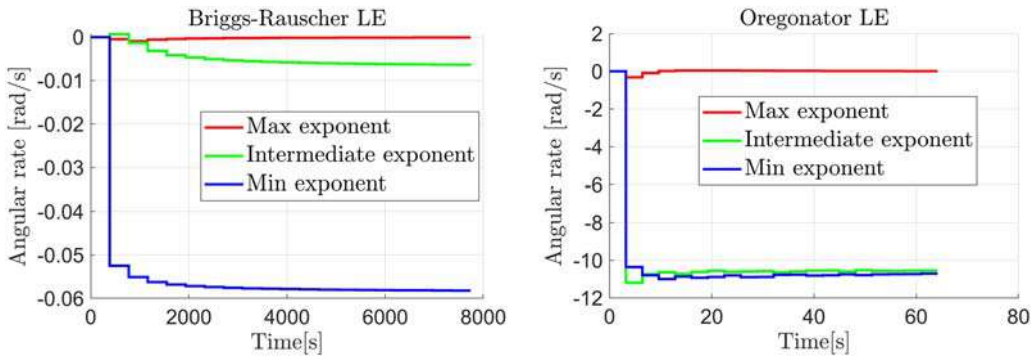


FIGURE B.6

Lyapunov exponents. (Left) Briggs−Rauscher case 2. (Right) Oregonator model ($f = 1$).

Table B.1 Lyapunov exponents of Briggs–Rauscher and Oregonator kinetics.

No.	Lyapunov exponent	Symbol	Unit	Value	Time constant [s], $1/\lvert\lambda_k\rvert$
Briggs–Rauscher kinetics, case 2, period $P = 387s$**, number of orbits** $N = 20$					
1	Maximum	λ_{max}	mrad/s	0.0805	$12400 > NP$
2	Intermediate	λ_{mid}	mrad/s	−6.39	156
3	Minimum	λ_{min}	mrad/s	−58.2	17.2
Oregonator, $f = 1$**, period** $P = 3.2s$**, number of orbits** $N = 20$					
1	Maximum	λ_{max}	rad/s	0.012	$83 > NP$
2	Intermediate	λ_{mid}	rad/s	−10.5	0.095
3	Minimum	λ_{min}	rad/s	−10.7	0.093

Since a stable *limit cycle* is a bounded periodic trajectory in the state space like the closed orbit of a second-order dynamic system with imaginary eigenvalues, we expect that the largest Lyapunov exponent $\mu_1 = \mu_{max}$ is equal to zero. We only mention the limit cycles of a *third-order* dynamic system like those of the Briggs–Rauscher and Oregonator kinetics in Sections 4.5 and 4.6 of Chapter 4. The largest Lyapunov exponent is equal to zero, the other two exponents are negative:

$$\mu_{max} = \mu_1 = 0, \mu_2 < 0, \mu_3 = \mu_{min} < 0. \tag{B.37}$$

We show in Fig. B.6 the converging plots of the Lyapunov exponents of the Briggs–Rauscher case 2 (left, see also Figure 4.12) and of the Oregonator case 1 (right, see Figure 4.19). Convergence to the estimated asymptotic limits in Table B.1 is fast in both cases. Because of the limit cycle, the maximum exponent is ideally zero, although numerically non zero in Table B.1 because of the finite time simulation. Let us remark as done in Table B.1, that the corresponding time constant is much larger than the simulated time interval NP, thus implying convergence to zero.

As a conclusion, the maximum Lyapunov exponent becomes *positive* in the case of chaotic trajectories (strange attractors) as in the already mentioned Lorenz system [6]. Eigenvalues with positive real part mean *instability*, which in this case reveals to be a subtle concept, since the region of a strange attractor in the state space is bounded, like in the case of a limit cycle. The fact is that the *traveled distance* between two points moving along two trajectories which start from slightly different initial points will diverge (instability), whereas the distance of the points in the state space remains bounded (Lagrange stability [1], J.-L. Lagrange, 1736–1813). Think of faster and slower runners along the closed path of a stadium. The difference of their traveled distances will diverge, whereas their distance in the stadium remains bounded.

B.3.4 Singular perturbation

Singular perturbation of a state equation is something different from the perturbations of the Section B.3.1, being concerned with different time scales hidden into equations. Concept and method were employed in Section 4.6 of Chapter 4 to reduce the order of a nonlinear equation. For simplicity's sake, concept and method will be explained with the help of a second-order LTI system. Consider the following state equation in matrix form

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix}(t) = \begin{bmatrix} -a & b \\ -c/\varepsilon & -d/\varepsilon \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}(t), \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}(0) = \begin{bmatrix} x_{10} \\ x_{20} \end{bmatrix}, \tag{B.38}$$

where $\varepsilon \ll 1$, all the coefficients are positive and the state matrix is AS, which is implied by $a + d/\varepsilon > 0$ and $ad + bc > 0$. By multiplying the second equation by ε, we can reach the conclusion that $\varepsilon \dot{x}_2(t) \simeq 0$, in other terms, that the second equation reaches equilibrium in a very short time τ_ε. This fact suggests that by neglecting the relevant initial interval, we may assume that $x_2 \simeq -cx_1/d, t > \tau_\varepsilon$, and reduce the state Eq. (B.38) to first order as follows:

$$\dot{x}_1(t) = -\left(a + bc/d\right)x_1(t), x_1(0) = x_{10}. \tag{B.39}$$

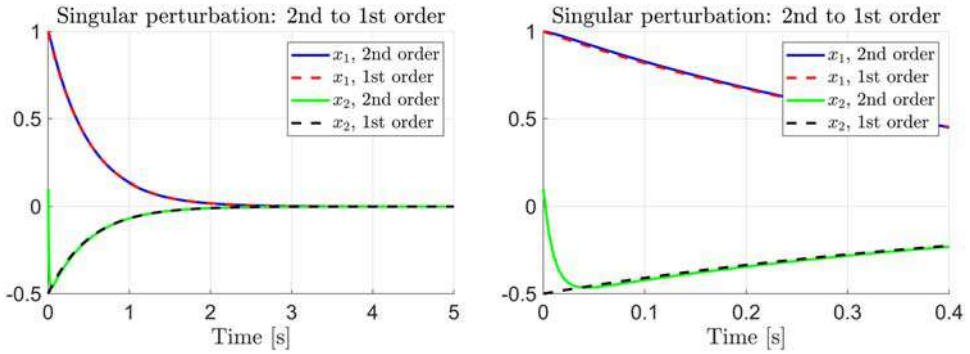


FIGURE B.7

(Left) The second-order (solid lines) and first-order responses (dashed lines). (Right) Enlargement on the neglected initial transient of x_2.

The time interval τ_ε follows from the approximate eigenvalues (the reader is asked to justify the approximation) of the state matrix in Eq. (B.38) and the relevant time constants:

$$\lambda_1 \simeq -\left(a + bc/d\right), \lambda_2 \simeq -d/\varepsilon, \tau_1 = |\lambda_1|^{-1}, \tau_2 = |\lambda_2|^{-1}. \tag{B.40}$$

It follows that $\tau_\varepsilon < 5\varepsilon/d$ and that, by assuming $a, b, c, d \ll \varepsilon^{-1}$, we can write $\tau_\varepsilon \ll |\lambda_1|^{-1}$ and neglect the initial transient of x_2 as shown in Fig. B.7, left. In the same figure, the second and first order free responses of Eqs. (B.38) and (B.39), respectively, are overlapped to show their coincidence except for the neglected initial transient (the green line in the left figure). The parameter values are

$$a = d = 1, b = 2, c = 0.5, \varepsilon = 0.01, x_{10} = 1, x_{20} = 0.1. \tag{B.41}$$

References

[1] E. Canuto, C. Novara, L. Massotti, D. Carlucci, C. Perez Montenegro, Spacecraft Dynamics and Control: the Embedded Model Control Approach, Butterworth-Heinemann, Oxford, UK, 2018.

[2] Wikipedia, *Routh-Hurwitz stability criterion*, from https://en.wikipedia.org/wiki/Routh%E2%80%93Hurwitz_stability_criterion.

[3] E. Canuto, D. Mazza, Introduction to population dynamics and resource exploitation, in ArXiv.org > q-bio, doc. ArXiv:2102.01205, January 2021.
[4] Wikipedia, *Hyperbolic equilibrium point*, https://en.wikipedia.org/wiki/Hyperbolic_equilibrium_point.
[5] Wikipedia, *Attractor*, https://en.wikipedia.org/wiki/Attractor#Strange_attractor.
[6] Wikipedia, *Lorenz system*, https://en.wikipedia.org/wiki/Lorenz_system.
[7] L. Dieci, M.S. Jolly, E.S. Van Vleck, Numerical techniques for approximating Lyapunov exponents and their implementation, J. Comput. Nonlinear Dyn. 6 (1) (2011) 1−7. Available from: https://doi.org/10.1115/1.4002088.

APPENDIX

Introduction to linear regression

C

C.1 Model and measurement errors

Statistics is intimately connected with the *inherent uncertainty* of real phenomena when measured by numerical instruments. These are capable of converting a physical output variable $y(t), t \geq t_0$ into a finite sequence of real numbers $\breve{y}(t_k)$ sampled at discrete times $t_k, k = 0, 1, 2, \ldots, N-1$. The interest is of relating the measured variable to other measurable variables $\{u_1(t), u_2(t), \ldots, u_m(t)\}$ (*predictor or input variables*) including time, in order to *predict* the output variable in *future times* $t > t_{N-1}$ or under *different conditions* than the measurement procedure. The essential objects needed are as follows.

1. An *instrument* has been designed and calibrated for the output variable and the measurement conditions, which latter are defined in terms of physical variables like temperature, pressure, humidity, stress, ...
2. A *mathematical model* which is provided by experimental sciences allows us to select suitable predictor variables and a known rule f for combining them in view of the desired prediction, $y(t) = f(u_1, u_2, \ldots, u_\mu, \ldots, u_m, t; \boldsymbol{a})$, where $\boldsymbol{a}$ is a finite vector of parameters fitting the rule to the measurements of the output variable. Predictor variables can be collected in the vector $\boldsymbol{u}(t)$.

The inherent uncertainty mentioned above affects the rule $f(\cdot)$ in two main ways.

1. The *parameter vector* $\boldsymbol{a}$ cannot be kept as perfectly known. The fact is that very often $\boldsymbol{a}$ is assumed to be either unknown or randomly distributed, in order to fit rule and measured data through the best selection of $\boldsymbol{a}$ itself. For simplicity's sake, we assume that $\boldsymbol{a}$ is unknown.
2. The *predictor variables* are insufficient to fit and predict the output also in a well defined and bounded range of experimental conditions. This is due to *complexity and unpredictable variability* of real phenomena, which fact may be expressed by physical models. As a consequence, one should add, at least implicitly, an unknown component $\delta\boldsymbol{u}$, whose effect δy on y, if sufficiently small in magnitude with respect to the expected magnitude of y, can be extracted from f, leading to the equation

$$y(t) = f(\boldsymbol{u}(t), t; \boldsymbol{a}) + \delta y(\boldsymbol{u}(t), \delta\boldsymbol{u}(t), t). \tag{C.1}$$

The correction δy is usually referred to as the *model error*.

Uncertainty effects are not limited to parameter and model uncertainty, but they directly affect the measurements $\breve{y}(t_k)$ of y and $\breve{\boldsymbol{u}}(t_k)$ of $\boldsymbol{u}$, as their hypothetical value is hidden by *measurements*

errors due to instrument and environment. The modern ways to guess, under some assumptions, hypothetical values are computer *simulations* of the phenomena under a designed uncertainty model. We are thus forced to rewrite Eq. (C.1) as follows,

$$\check{y}(t_k) = f(\check{\boldsymbol{u}}(t_k) - \tilde{\boldsymbol{u}}(t_k), t_k; a) + \delta y(\boldsymbol{u}(t_k) + \delta \boldsymbol{u}(t_k), t_k) + \tilde{y}(t_k), \tag{C.2}$$

where $\tilde{\boldsymbol{u}}(t_k)$ denotes the measurement error of the predictor variables and $\tilde{y}(t_k)$ that of the output variable. Instrument calibration aims to reduce measurement errors to be zero mean and unpredictable from measurement to measurement, which, it should be recalled, is not always feasible due to accumulating random drifts (not treated here, see Reference [1]). Neglecting drifts and confining uncalibrated components into the model error $\delta y(\cdot)$, measurement errors can be profitably assumed to be zero-mean and statistically independent, which implies that the error sequence can be modeled as a *discrete time white noise* with zero expected value, uncorrelated samples, and finite but not necessarily constant variance.

C.2 Linear regression and least squares estimation

The practical way to convert a generic rule $f(\cdot)$ into a rule which is *linear* in the parameters (*linear regression*) is to develop the variable part of $f(\cdot)$ into a finite series of linearly independent and possibly zero-mean scalar functions $f_\mu(\boldsymbol{u}(t))$, $\mu = 1, \ldots, m$. They may be further expanded, if needed, to reduce the model error component which depends on the input vector $\boldsymbol{u}$.

One of the possible expansions, the *polynomial*, was adopted in Section 8.2 of Chapter 8. Accordingly, Eq. (C.2) can be replaced by the linear relation

$$\begin{array}{l} \check{y}(t_k) = a_0 f_0 + \Delta y(\boldsymbol{u}(t_k), t_k) = a_0 f_0 + \sum_{\mu=1}^{m} a_\mu f_\mu(\boldsymbol{u}(t_k)) + \delta y(t_k) + \tilde{y}(t_k) \\ \Rightarrow \check{\boldsymbol{y}}_N = F_N(\boldsymbol{u})\boldsymbol{a} + \Delta \boldsymbol{y} + \tilde{\boldsymbol{y}}, f_0 = 1 \end{array}, \tag{C.3}$$

which has been rewritten, in the second row, in matrix form. If the variable part $\Delta y(\boldsymbol{u}(t_k), t_k)$ is zero mean, the constant part equals the measured mean of the output, namely

$$a_0 f_0 = \bar{y}_N = N^{-1} \sum_{k=0}^{N-1} \check{y}(t_k). \tag{C.4}$$

The model matrix $F_N = [\boldsymbol{f}_0 \boldsymbol{f}_1 \ldots \boldsymbol{f}_\mu \ldots \boldsymbol{f}_m]$, where $\boldsymbol{f}_\mu = \left[f_\mu(t_0), \ldots, f_\mu(t_k) \ldots, f_\mu(t_{N-1})\right]$, is sized $N \times (m+1)$. The parameter vector $\boldsymbol{a}$ (we are using the same notation of Section 8.2 in Chapter 8) is the unknown vector, to be found by minimizing a norm of the error $\boldsymbol{e}_N = \boldsymbol{y}_N - F_N(\boldsymbol{u})\boldsymbol{a}$. In the *least-squares method*, the norm is a weighted Euclidean norm as follows

$$\begin{array}{c} \hat{\boldsymbol{a}} = \operatorname{argmin} \left| \boldsymbol{e}_N = \check{\boldsymbol{y}}_N - F_N(\boldsymbol{u})\boldsymbol{a} \right|_W^2 \\ |\boldsymbol{e}_N|_W^2 = \boldsymbol{e}_N^T W \boldsymbol{e}_N, W \geq 0 \end{array}. \tag{C.5}$$

In a statistical framework, the weight matrix holds $W_N = \tilde{S}_N^{-2}$, where $\tilde{S}_N^2$ is the covariance matrix of the measurement errors, which, under the discrete time white noise assumption, becomes diagonal as follows $\tilde{S}_N^2 = \operatorname{diag}\left(\tilde{\sigma}^2(t_0), \ldots, \tilde{\sigma}^2(t_{N-1})\right)$.

Minimization in Eq. (C.5) can be interpreted by means of the Euclidean norms of the component signals.

1. The data variance, or *total sum of squares*, is defined in terms of the variable part of the measurements $\Delta y(k) = y(k) - \bar{y}_N$:

$$s_y^2 = \sum_{k=0}^{N-1} \left(\breve{y}(k) - \bar{y}_N\right)^2. \tag{C.6}$$

2. The *residual sum of squares* (RSS) is defined in terms of the estimated residuals $\hat{\boldsymbol{e}}_N = \breve{\boldsymbol{y}}_N - F_N(\boldsymbol{u})\hat{\boldsymbol{a}}$:

$$\hat{s}_{res}^2 = \hat{\boldsymbol{e}}_N^T \hat{\boldsymbol{e}}_N. \tag{C.7}$$

3. The *explained variance* $\hat{s}_y^2 = s_y^2 - \hat{s}_{res}^2$ is the difference between total and residual sum of squares.
4. The ratio between explained variance and total sum of squares is denoted by R^2, known as *R-squared* [see Eq. (8.12) in Chapter 8], and when closed to unit denotes good fitting between model and measurements:

$$R^2 = \frac{\hat{s}_y^2}{s_y^2} = 1 - \frac{\hat{s}_{res}^2}{s_y^2}. \tag{C.8}$$

5. The *adjusted R-squared* accounts for the explained and residual degrees of freedom (DF) as follows

$$\overline{R}^2 = \frac{(N-1)\hat{s}_y^2}{(N-m-1)s_y^2} = 1 - \frac{(N-1)\hat{\sigma}_{res}^2}{s_y^2} > R^2, \tag{C.9}$$

where $\hat{\sigma}_{res}^2$ is the normalized RSS [see Eq. (8.14) in Chapter 8]. $\overline{R}^2$, being larger than R^2, is less conservative in qualifying model and data fit.

The above variances are essential in building the *F-test of significance* to be detailed in Section C.3.3.

Under $\delta y(t_k) = 0$ (zero model error) and zero measurement error of the predictor variables, $\tilde{\boldsymbol{u}}(t_k) = 0$, the a priori covariance matrix P_N^2 of the estimation error $\tilde{\boldsymbol{a}} = \hat{\boldsymbol{a}} - \boldsymbol{a}$ can be computed and holds

$$P_N^2 = \left(F_N(\boldsymbol{u})^T \tilde{S}_N^{-2} F_N(\boldsymbol{u})\right)^{-1}. \tag{C.10}$$

It simplifies into

$$P_N^2 = \tilde{\sigma}^2 \left(F_N(\boldsymbol{u})^T F_N(\boldsymbol{u})\right)^{-1}, \tag{C.11}$$

under measurement errors with a stationary variance equal to $\tilde{\sigma}^2$. The covariance in Eq. (C.11) is referred to as *a priori* since it can be computed before the regression from the model matrix and the measurement error covariance. It is *a posteriori* estimated from the least squares residuals, as follows

$$\tilde{P}_N^2 = \hat{\sigma}_{res}^2 \left(F_N(\boldsymbol{u})^T F_N(\boldsymbol{u})\right)^{-1}. \tag{C.12}$$

In order that the previous inverse matrices exist, the model matrix $F_N(\boldsymbol{u})$ has to be full rank (see Appendix A). The diagonal element $p_\mu^2 = (P_N^2)_\mu$ of P_N^2 provides the a *priori variance* p_μ^2 of the estimation error $\tilde{a}_\mu$ of a_μ; the diagonal element $\tilde{p}_\mu^2 = (\tilde{P}_N^2)_\mu$of $\tilde{P}_N^2$ provides the a *posteriori variance* $\tilde{p}_\mu^2$, whose square root $\tilde{p}_\mu$ is known as *standard error* (SE, see Section 8.2.2).

It can be proved that the ratio between the estimation error $\tilde{a}_\mu = \hat{a}_\mu - a_\mu$ and the a priori standard deviation p_μ is normally distributed with zero mean and unit variance (*standard normal variable*), which fact can be written as

$$\frac{\hat{a}_\mu - a_\mu}{p_\mu} \sim N(0,1). \tag{C.13}$$

C.3 Model degrees of freedom and test of significance

C.3.1 Overfitting and poor fitting

Any linear model (C.3), and more generically any rule f, is facing two issues.

1. *Overfitting* corresponds to an excessive degree m of the expansion in Eq. (C.3), which implies that some parameters, say a_μ, will be estimated close to zero, thus declaring useless the relevant function $f_\mu(\boldsymbol{u})$. In practice, a high degree expansion is such to fit any variable component of the measured output, including components close to be interpreted as insignificant noise, although possessing some correlation. In fact, we should be aware that the least squares procedure not only estimates the expansion parameter vector a, sized $m+1$ (the estimate DF, including the constant a_0), but also the unknown vectorial error sum $\boldsymbol{e}_N = \delta\boldsymbol{y}(\boldsymbol{u}) + \tilde{\boldsymbol{y}}$, sized N, whose DF reduce to $N-m-1$. Thus, by increasing m, it likely occurs that some components of the model error (it may be a positive effect) and measurement errors (always a negative effect) enter and may degrade the estimated model. The error estimate $\hat{\boldsymbol{e}}_N = \check{\boldsymbol{y}}_N - F_N(\boldsymbol{u})\hat{\boldsymbol{a}}$ has been already referred to as *residual*. To this end, statistical tests like *t-* and *F-test* have been devised to discriminate between the hypothesis of null parameters (*null hypothesis*) and the alternative hypothesis of nonzero parameters (reduction of model error with the risk of overfitting). It can be proved that the ratio between the normalized RSS $\hat{\sigma}^2_{res}$ and the measurement error variance $\tilde{\sigma}^2$ is chi-square distributed with $N-m-1$ DF, which fact can be written as

$$\frac{\hat{\sigma}^2_{res}}{\tilde{\sigma}^2} \sim \chi^2_{N-m-1}. \tag{C.14}$$

2. *Poor fitting* is the opposite condition in which model error tends to become significant and its variance dominates the residuals, thus decreasing R^2. Under these conditions, least squares estimates tend to become biased, which implies that the standard normality in Eq. (C.13) is lost. In other terms, repeated estimation trials show that the mean values of the estimated coefficients deviate systematically from the expected value, and the residuals tend to become correlated, thus deviating from the discrete-time white noise assumption.

Finding a trade-off between overfitting and poor fitting is the key challenge of any experimental data explanation/fitting. Statistical tests have been designed to the purpose.

C.3.2 Student's *t*-test

The first statistical test of significance to be recalled here is the *t-test* (*t* stands for test) or *Student's t test*, which under the null hypothesis follows the *Student's distribution*, proposed by W.S. Gosset

(1876–1937) under the pseudonym of *Student*. The third column (`tStat`) in the table of Fig. 8.2 (Chapter 8) reports the *t-test* t_μ of each estimated component $\hat{a}_\mu$, the test being defined by

$$\begin{aligned} t_\mu &= \frac{\hat{a}_\mu}{\tilde{p}_\mu} = \frac{\hat{a}_\mu \tilde{\sigma}}{p_\mu \hat{\sigma}_{res}} \\ \tilde{p}_\mu &= p_\mu \frac{\hat{\sigma}_{res}}{\tilde{\sigma}} = \text{standard error(SE)} \end{aligned}, \tag{C.15}$$

where the *standard error* has been already defined in the Section C.2.

The second ratio in Eq. (C.12) is the ratio between Eq. (C.13), under the null hypothesis that $a_\mu = 0$ and the square root of Eq. (C.14). A random ratio between a standard normal and the square root of a chi-square variable follows the *Student's t probability distribution* with the same DF of the chi-square variable, in this case $\nu = N - m - 1$. The Student's probability density $f(t)$ is symmetric around the zero-mean value like a standard normal density. Actually the resemblance is even more striking, since the variance tends to be unitary for increasing DF, namely

$$\lim_{\nu \to \infty} \text{var(Student's } t) = \lim_{\nu \to \infty} \frac{\nu}{\nu - 2} = 1. \tag{C.16}$$

The test is operated as follows.

One expects that under the null hypothesis that $a_\mu = 0$, t_μ in Eq. (C.15) will be sufficiently small in magnitude, since the largest portion of the Student's t probability mass is located around zero. Of course, due to measurement errors, large magnitude outliers may occur but they are assigned a sufficiently small probability, say 0.05 (5%), which implies that by repeating the test 100 times, under the same conditions and null hypothesis, outliers will occur only in five cases. This suggests that the null hypothesis should be accepted when

$$|t_\mu| \leq t_{max}(\alpha), \; P\{|t| > t_{max}(\alpha)\} = \alpha. \tag{C.17}$$

The outlier probability α is known as the *level of significance* and is also known as the probability of the *type I error*, that is, the probability of rejecting the null hypothesis when TRUE. The threshold $t_{max}(\alpha)$ is known as the *significance threshold*. Of course, as soon as the measured t_μ becomes larger than the threshold, the actual probability $P\{|t| > t_\mu\}$ of rejecting the null hypothesis tends to be negligible. The corresponding probability is usually known as `pValue` (see the table of Fig. 8.2 in Chapter 8).

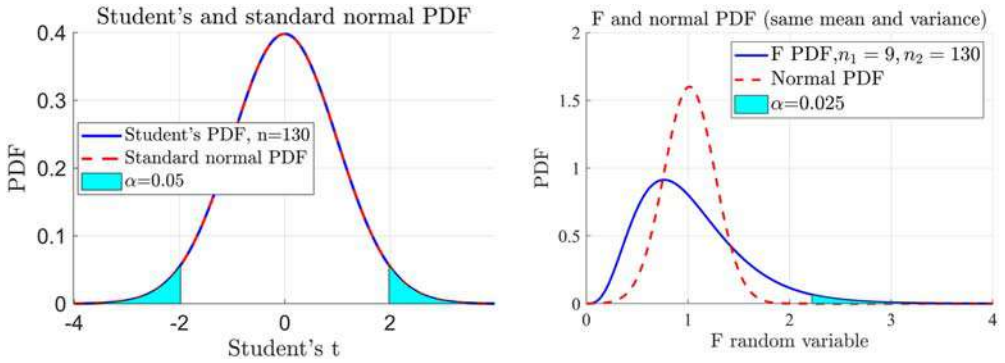


FIGURE C.1

(Left) Student's and standard normal PDF together with the null hypothesis rejection regions. (Right) The F probability density function compared with the normal PDF with same mean and variance.

As a result, when $|t_\mu| > t_{max}(\alpha)$, *the null hypothesis is rejected*, which implies that $a_\mu \neq 0$, a rather *vague information*. The threshold $t_{max}(\alpha)$ may be computed with the help of the *standard normal distribution*, because of the rapid convergence of the Student's t for increasing DF as in Fig. C.1, left. The magnitude of the lower and upper thresholds in Fig. C.1, left, holds:

$$t_{max}(0.05) \simeq 1.98. \tag{C.18}$$

The above vague information $a_\mu \neq 0$ may be improved by specifying an alternative hypothesis, like $a_\mu > a_{min} > 0$. This leads to the definition of the *type II error*, that is of rejecting the alternative hypothesis when TRUE, which is the same of accepting the null hypothesis when FALSE (the so called *false alarm*). The reader is addressed to specific textbooks [2].

C.3.3 *F*-test

The second kind of test is recognized as more effective in checking the significance of regression components, since it compares sums of squares (variances) through their ratio, for instance explained variances to RSS, the only case treated here. The test was proposed by G.W. Snedecor (1881−1974) and denoted by F in honor of the statistician Sir R.A. Fisher (1890−1962). The *F-test* is employed in the regression analysis of variance (ANOVA) as in Section 8.2.3 (Chapter 8). The null hypothesis is similar to that of the Student's t-test, but in this case, we can test whether an arbitrary set of parameters $A = \{a_1, a_2, \ldots\}$ is zero. Two different sets of parameters are contemplated as in Section 8.2.3.

1. The *null hypothesis* is that the whole set $A = \{a_\mu, \mu = 1, \ldots, m\}$ of the model parameters is zero, which implies that experimental data are significantly explained by their mean value. The F-test is the ratio between the normalized explained variance $\hat{s}_y^2/m$ and the normalized RSS $\hat{s}_{res}^2/(N-m-1)$, as follows [see Eq. (8.14) in Chapter 8]:

$$\begin{gathered} F_A = \frac{(N-m-1)\hat{s}_y^2}{m\hat{s}_{res}^2} \\ \hat{s}_y^2 = s_y^2 - \hat{s}_{res}^2 \end{gathered} \tag{C.19}$$

   Under the null hypothesis, the sums of squares in the numerator and denominator are chi-squared distributed with m and $N-m-1$ DF, respectively, which leads the normalized ratio to be *F-distributed* with $\{n_1 = m, n_2 = N-m-1\}$ DF. Mean and variance of the relevant PDF hold:

$$\begin{gathered} \text{mean value} = \frac{N-m-1}{N-m-3} \simeq 1 \\ \text{variance} \simeq \frac{2}{m}, N-m-1 \gg m > 4 \end{gathered}. \tag{C.20}$$

   The PDF tends to a normal distribution $N\left(1, \sqrt{2/m}\right)$ as soon as $N \to \infty, m \to \infty$. Thus, for $m \gg 1$, the PDF becomes narrower around the mean value, implying that the null hypothesis tends to be unlikely because of the large DF of the explained variance. In order to fix the *level of significance* α, that is the probability that the null hypothesis is rejected when TRUE (type I error), we observe that, unlike the Student's t-test, F-test is nonnegative, and therefore we have to consider the right queue of the PDF. By denoting the relevant PDF with $f_F(F; n_1, n_2), F \geq 0$,

where the pair $\{n_1, n_2\}$ denotes numerator and denominator DF, the level of significance defines the *significance threshold* $F_{max}(\alpha)$ as follows

$$\int_{F_{max}(\alpha)}^{\infty} f_F(\varphi; n_1, n_2) d\varphi = P\{F \geq F_{max}(\alpha; n_1, n_2)\} = \alpha. \tag{C.21}$$

By recalling Eq. (C.18), the *normal (but not standard) approximation*, if restricted to the right queue and therefore to $\alpha = 0.025$, corresponds to the threshold

$$\begin{array}{l} F_{\max}(0.025; n_1, n_2) \simeq 1 + 1.96\sqrt{2/m} \simeq 1.9 \\ m = 9, N = 140,\ n_1 = m,\ n_2 = N - m - 1 \end{array}. \tag{C.22}$$

The reader should be aware that under $m = 9$ (not a very large value), normality assumption looks not appropriate, especially because of the right queue extension which is much larger than the relevant normal PDF, as Fig. C.1, right, shows. The PDF of the F-test is skewed on the right side, meaning that a nonnegligible probability mass exists where the normal probability mass is close to zero. This implies that the actual threshold will be larger than the value in Eq. (C.22). The value can be checked with the Matlab function `finv(P, n1, n2)`, which computes the inverse of the probability function, namely the abscissa F of the F cumulative probability function given the probability `P`, and the numerator and denominator DF `n1, n2`. The following value

$$F_{max}(0.025; 9, 130) = finv(0.975, 9, 130) \simeq 2.2, \tag{C.23}$$

shows a deviation greater than 10% from the value in (C.22). The null hypothesis, meaning *insignificant model*, will be rejected when the above threshold will be overcome, under the 0.025 probability of type I error.

2. The null hypothesis is that a single model parameter $a_\mu, \mu = 1, ..., m$ is equal to zero, and that the relevant function $f_\mu(\boldsymbol{u})$ is of no help in explaining experimental data. In this case, the test becomes [see Eq. (8.15) in Chapter 8]

$$F_\mu = \frac{(N - m - 1)\hat{s}_\mu^2}{\hat{s}_{res}^2} \tag{C.24}$$

and the PDF $f_F(F; 1, N - m - 1)$ has the following mean value and variance

$$\begin{array}{l} \text{mean value} = \dfrac{N - m - 1}{N - m - 3} \simeq 1 \\ \text{variance} \simeq 2, N - m - 1 \gg 4 \end{array}. \tag{C.25}$$

Now, the threshold $F_{max}(\alpha; n_1, n_2)$ increases with respect to Eq. (C.22), since by restricting the test to a single parameter (DF $n_1 = 1$), the null hypothesis, in other terms assuming zero parameter, becomes more likely to occur. In fact, we find

$$F_{max}(0.025; 1, 130) = finv(0.975, 1, 130) = 5.14. \tag{C.26}$$

C.3.4 *p*-value

In the tables of Section 8.2.3, the experimental statistical tests (either Student's t-test $\hat{t}$ or F-test $\hat{F}$) were accompanied by the so called `pValue`. By denoting a generic statistical test with N

experimental data by $\hat{t}_N(\alpha; DF)$, and its PDF under the null hypothesis with $f_t(t; DF)$, where DF denotes the relevant degrees of freedom, the `pValue` is defined by

$$p\text{-value}\left(\hat{t}_N\right) = \int_{\hat{t}_N}^{\infty} f_t(\tau; DF) d\tau. \tag{C.27}$$

The `pValue` is the a-priory probability of observing, from experimental data and under the null hypothesis, values of the statistical test $\hat{t}_N(\alpha; DF)$ larger than the current observed value. Under a p-value $\ll \alpha$, the null hypothesis appears to be highly implausible given the small probability of finding test values of this order of magnitude. In other terms, under a TRUE null hypothesis, the observed test appears as an improbable outlier, to be discarded together with the null hypothesis.

References

[1] E. Canuto, C. Novara, L. Massotti, D. Carlucci, C. Perez, Montenegro, *Spacecraft Dynamics and Control: the Embedded Model Control Approach*, Butterworth-Heinemann, Oxford, UK, 2018.

[2] H. Cramer, *Mathematical Methods of statistics*, Princeton University Press, 1945.

APPENDIX

Introduction to Matlab Simulink D

D.1 Simulink pane and library

To start, let us select `Simulink model` (last row) from the `New` menu (the third column) of the `Matlab Home pane`. In the `New` pane, select `Blank model`, which opens the Simulink blank model pane in Fig. D.1, where to draw the desired block diagram (model).

The elements of the block diagram can be extracted from the `Simulink Library` by opening the `Library Browser`, which is pointed in Fig. D.1 by an upward arrow. Fig. D.2 shows the `commonly used blocks` of the library, where the arrow points to the `integrator block`, which is the core of dynamic systems. Simulink blocks possess input and output arrows. Pointing with the mouse to an arrow, a cross appears to be dragged for connecting output to input arrows of different blocks.

D.2 The integrator block

The integrator block corresponds to the elementary state equation:

$$dx(t)/dt = u(t), x(0) = x_0, \tag{D.1}$$

which, when integrated, provides the expression

$$x(t) = x_0 + \int_0^t u(\tau)d\tau, \tag{D.2}$$

where $u(t)$ is the input signal to be added to the block input port as shown below.

Fig. D.3 shows a simple block diagram corresponding to the state equation:

$$dx(t)/dt = -kx(t) + u_0, x(0) = x_0. \tag{D.3}$$

The integrator block has been endowed with an `external initial state` x_0 which is provided by a `constant block`, dragged from the library, where the constant `x0` has been inscribed after having opened the block. The block is opened by a double click of the mouse left button. The input of the block is twofold: (1) a constant signal `u0` and (2) a negative feedback consisting of the gain `k` inscribed in the `gain block` which is connected to the integrator output and whose output is subtracted from the constant input in the `add block` (dragged from `math operations`). The block

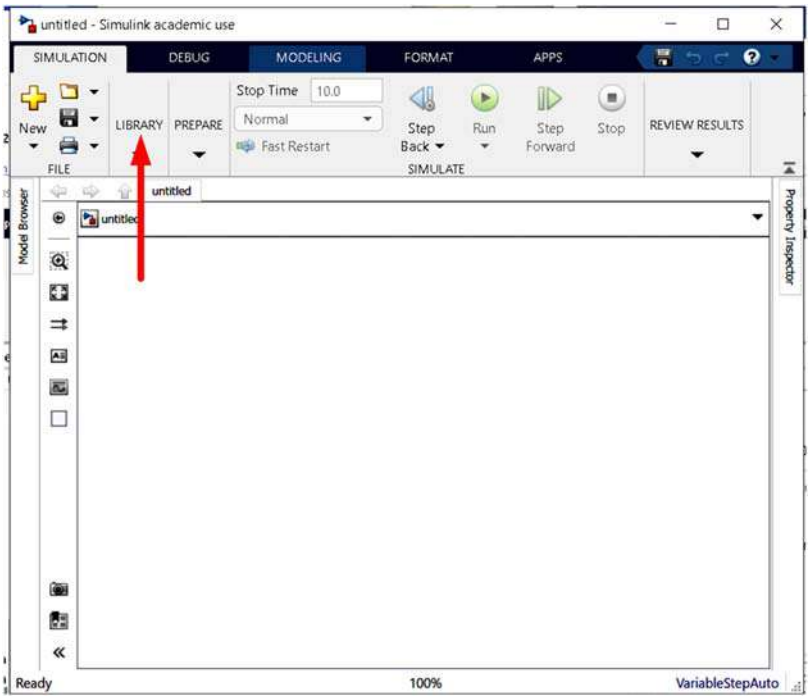


FIGURE D.1

Simulink blank model pane.

inscribed with `out.x` is a recording block (dragged from `sinks` under the name `To Workspace`), where the samples of the state variable `x` are recorded. The simplest recording mode is `array` to be selected in the opened block. One has also to inscribe the variable name `x`. The name `out.x` is created by Matlab® Simulink, and `out` is the given name of the recorded variables in the `Matlab Workspace` at the end of the simulation (see the script in Section D.4).

What about the values of the three parameters `u0, x0, k`? They must be provided by a `Matlab script`, say `First.m` which launches the `Simulink model`. The latter must be saved with a name, for instance `FirstSim`. The model is classified by Matlab as of type `slx`.

D.3 The configuration parameters

We also add the block diagram in Fig. D.3 with the *sampling time*, which is done by adding (1) the `clock` block (top of Fig. D.3) from the item `sources` of the library, (2) a recording block to record the sequence of time instants $t_i = iT_s, i = 0, \ldots, N$ with the name `t` (Matlab will give it the name `out.t`), and (3) the `display` (from the item `sinks` of the library) to monitor the current time t_i during simulation.

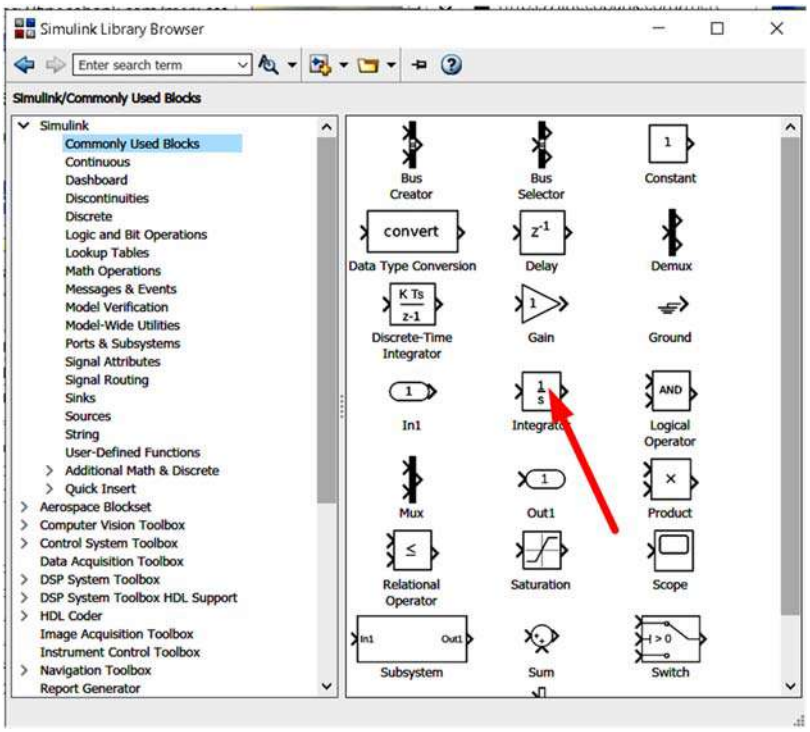


FIGURE D.2

The integrator block with initial state, input, recording, and clock blocks.

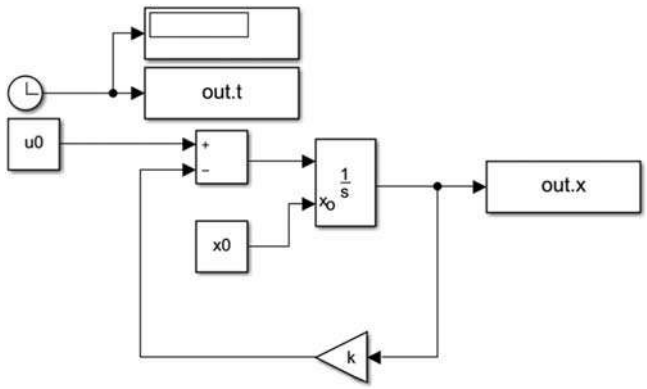


FIGURE D.3

Block library opened in the `commonly used blocks`.

We have to decide which is the simulation time unit T_s, named `Ts`, which must be smaller than the minimum time constant, in this case $T_s \simeq 0.1/k$. It is a good practice to insert `Ts` in all the recording blocks `To Workspace`, in the bottom row `sampling time`, which by default is set to `-1`.

Before writing the script, we have to tell Simulink the following information:

1. the integrator engine (`ODE solver`), in this case `ode4`,
2. the initial and final times $t_0 = 0$ and $t_N = NT_s$ under the names `tInit, tEnd`, and the time unit (`sampling time`) `Ts`,
3. the type of integrating step, here chosen as `Fixed-step`.

The names must be inserted in the `Configuration parameters` pane, which is opened by clicking on `Model Settings` (in the menu `Modelling` of the model pane). Their values must be assigned by appropriate statements of the script (see Section D.4). The `Configuration parameters` pane is shown in Fig. D.4, together with red boxes indicating the five rows where to either insert variable names or make a choice.

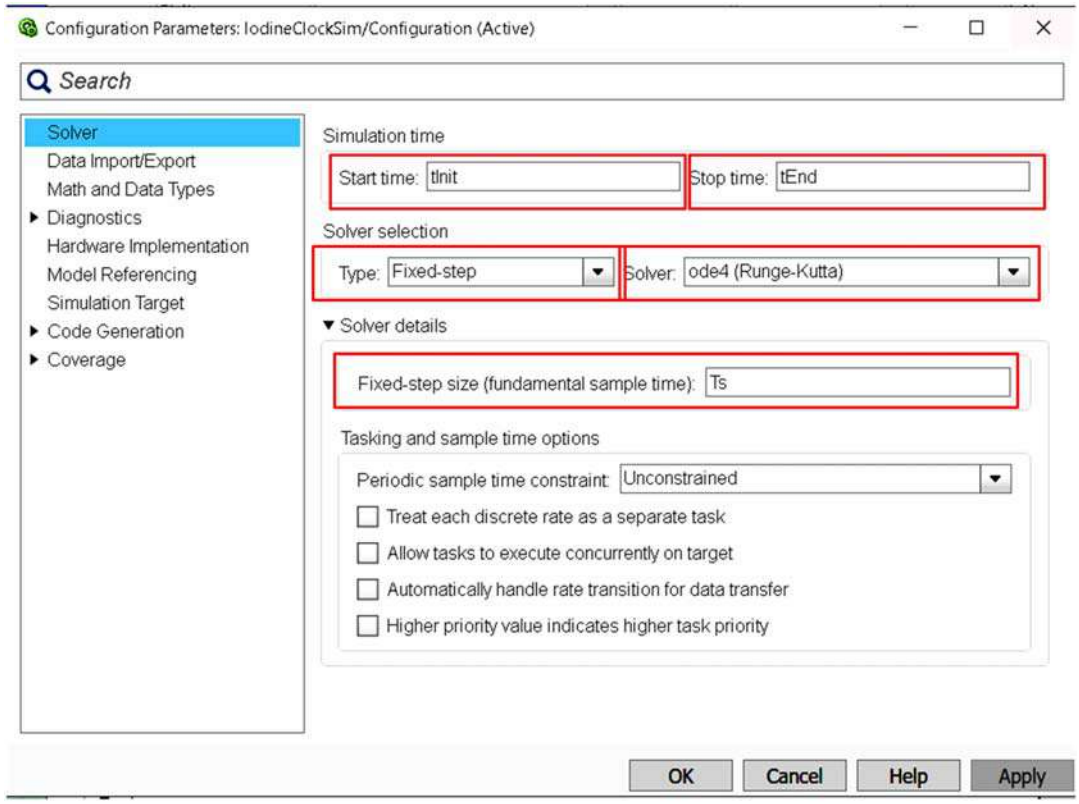


FIGURE D.4

The pane of `Configuration parameters`.

D.4 The script and graphical output

The script, driving the time simulation of a Simulink block diagram (model), is suggested to be subdivided into four parts.

1. *Initialization* clears the `Matlab workspace`, where you can find list and size of all the variables, and sets some graphical properties: see the script `InitChem` below.
2. *Parameter values*. Parameters, initial conditions, and timing data are defined.
3. *Integration run*, via the Matlab function `sim()`.
4. *Graphical output*. The time profiles of the variables collected by the recording blocks `To Wokspace` are plotted.

Below you can find the script `First.m` of the model `FirstSim.slx` in Fig. D.3.

```
%% Script for FirstSim model
%% 1) Initialization
InitChem;
%% 2) Parameters and timing
k=0.1; % 1/s tau=10 s
tau=1/k;
u0=0.2; % constant command
x0=1; % initial state
xinf=u0/k; % steady state
% Ts as a multiple of 1 s (Warning: not always fits)
tInit=0;
Ts=floor(0.1/k);
tEnd=10*tau; % tEnd must be multiple of Ts
inttEnd=floor(tEnd/Ts);
tEnd=inttEnd*Ts;
%% 3) Integration (simulation) launch
% all the recorded variables are collected for later used in out.mat
out=sim('FirstSim');
%% 4) Graphical output
% state
t=out.t;
x=out.x;
figure('name','state x'); hold on; grid on;
plot(t,x,'b');
xlabel('Time [s]');
ylabel('Variable [unit]');
title('Time profiles');
legend('State x','FontSize',20);
ylim([0 xinf*1.5]);
hold off;
```

D.5 Initialization script

The initialization script `InitChem` clears the workspace and the graphical figures. It sets graphical properties and colors.

It defines some scales, and you can select between variable and repeated random generation.

```
%% Chemistry with Matlab
%% Initialization
clear;clc;format compact; % variables
close all; % figures
st=fclose('all'); % files
%% Graphical properties
set(groot,'DefaultLineLineWidth',2)
set(groot,'DefaultAxesFontSize',16)
set(groot,'DefaultFigureWindowStyle','docked') % 'normal'=separate figures
set(groot,'defaultAxesFontName','Arial')
set(groot,'defaultAxesXGrid','on')
set(groot,'defaultAxesYGrid','on')
set(groot,'defaultTextFontWeight','normal')
set(groot,'defaultTextInterpreter','Latex')
set(groot,'defaultLegendInterpreter','Latex')
set(groot,'defaultLegendFontSize',20)
% colors
color(1)='r';
color(2)='g';
color(3)='b';
color(4)='k';
color(5)='c';
color(6)='m';
color(7)='y';
color(8)='r';
color(9)='g';
color(10)='b';
grey=[0.5 0.5 0.5];
% style
style={'- ','--',': ','-.'}; % use char(style(i)) to drop blanks
%% scales
deg2rad=pi/180;
KiloScale=1e-3;
MegaScale=1e-6;
milliScale=1000;
centiScale=100;
microScale=1e6;
nanoScale=1e9;
picoScale=1e12;
%% random nummber
rng('shuffle'); % different at code start
%rng('default'); % same at code start
```

D.6 View of a 3D graphical plot

The textbook includes several 3D plots of surfaces and curves produced by different Matlab functions. Matlab allows to define the view point of a 3D graphical plot through a pair of angles, azimuth φ, and elevation δ given in angular degree units. The reader should pay attention to Fig. D.5, where the Matlab 3D figure axes are shown together with the spherical coordinates $\{\varphi, \delta\}$ of the viewpoint unit vector. Cartesian axes and the view point unit vector are defined by azimuth and elevation. The azimuth angle is counterclockwise positive starting from the $-y$ axis (in red color, the negative second axis), In this way, the viewpoint orthogonal to xz plane is directed from the origin along $-y$. The viewpoints of the xy and yz planes are also indicated. A good viewpoint may often be (45 degrees and 25 degrees) close to that in Fig. D.5. A viewpoint is defined by a statement like `view([45 25])` to be inserted after the `figure` statement.

Users may change the viewpoint from the figure pane by enabling the `Rotate 3D` button.

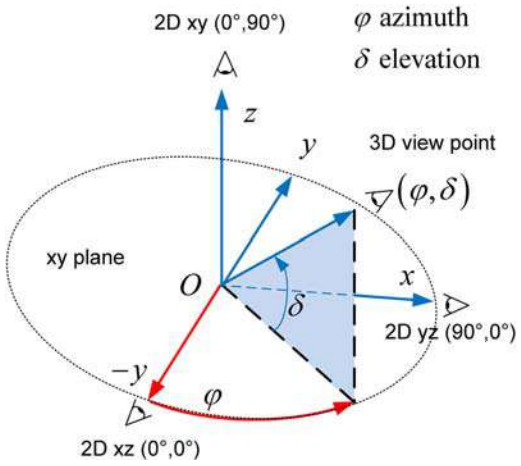


FIGURE D.5

Matlab 3D graphical plot. Cartesian axes and the view point unit vector defined by azimuth and elevation.

D.7 Matlab ODE solvers

Chapters 3 and 4 are concerned with the integration of ordinary differential equations (ODE) by means of Matlab ODE solvers. ODE solvers are called either directly by statements like:

```
[t,A_calc] = ode45(first,timespan,A0);
```

where a specific ODE solver, `ode45`, is specified, or through a Simulink model, like `IrrevFirstOrderSim.slx`, via statements like:

```
out = sim('IrrevFirstOrderSim');
```

In the latter case, the ODE solver must be selected in the pane of `Configuration Parameters` shown in Fig. D.4.

Although Matlab provides several solvers together with brief suggestions (see the help page `Choose an ODE solver`), it may be of interest to briefly recall the principles and include a simple comparison test.

D.7.1 Principles of ODE solvers

We only consider for simplicity's sake *fixed-step* solvers. In the *variable-step* solvers, the default time unit T_s may be diminished for keeping integration errors below the required accuracy or enlarged for reducing computing time. Let us consider the scalar state equation

$$\dot{x}(t) = f(x(t), u(t), t), x(t_0) = x_0, \tag{D.4}$$

where $u(t)$ is an input variable which is known at current time t. Integration from t to $t + h$ via the mean value theorem provides the implicit equation

$$x(t + h) = x(t) + f(x(t + \theta), u(t + \theta), t + \theta)h, \tag{D.5}$$

where $0 \leq \theta \leq h$ is unknown.

The simplest way to solve the implicit equation is the Euler method (L. Euler, 1707−83), which by assuming $\theta = 0$ provides the discrete-time equation

$$\hat{x}(t + h) = \hat{x}(t) + f(\hat{x}(t), u(t), t)h, \hat{x}(t_0) = x_0, h = T_s, \tag{D.6}$$

where $h = T_s$ is the integration time unit, and all the variables on the right side are known at the current time t. The integrated variable has been marked with a hat for indicating that is different from the unknown true state $x(t)$, $\tilde{x}(t) = x(t) - \hat{x}(t)$ being the integration error. The method amounts to keep the *integrand constant* during the integration time unit.

A natural extension proceeds from the observation that Eq. (D.6) provides an estimate (actually a *prediction*) of $x(t + h)$, which could be employed for *correcting* the same prediction as follows

$$\bar{x}(t + h) = \hat{x}(t + h) + k(f(\hat{x}(t + h), u(t), t + h) - f(\hat{x}(t), u(t), t))h, \tag{D.7}$$

where the correction gain k must be designed. The gain value is found by rewriting Eq. (D.7) in the usual way

$$\bar{x}(t + h) = \hat{x}(t) + kf(\hat{x}(t + h), u(t), t + h)h + (1 - k)f(\hat{x}(t), u(t), t)h, \tag{D.8}$$

and by ascertain that the optimal gain is $k = 1/2$, which value follows from the linear interpolation of the trapezoidal rule as opposed to the constant integrand of Eq. (D.6) (see Fig. D.6, left). The numerical integration, known as Heun method (K. Heun, 1859−1929), which combines the prediction Eq. (D.6) and the correction Eq. (D.7), is the simplest case of *predictor-corrector methods*, like the Adams-Bashforth-Moulton solver (`ode113`, variable-step [1]). The complete pair of equations is the following:

$$\begin{aligned}\hat{x}(t+h) &= \hat{x}(t) + f(\hat{x}(t), u(t), t)h, \hat{x}(t_0) = x_0, \quad \text{prediction} \\ \bar{x}(t+h) &= \hat{x}(t+h) + \frac{1}{2}(f(\hat{x}(t+h), u(t), t+h) - f(\hat{x}(t), u(t), t))h, \quad \text{correction}\end{aligned}. \tag{D.9}$$

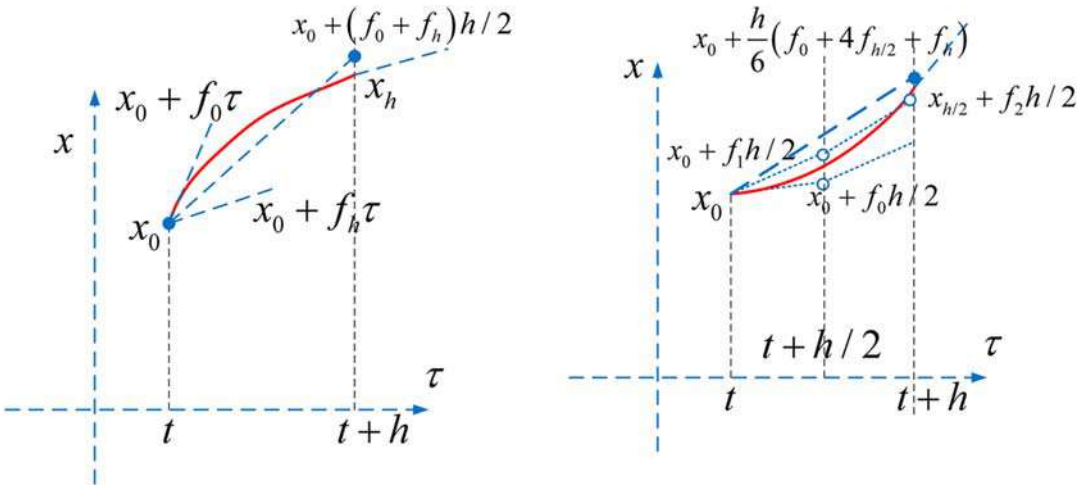


FIGURE D.6

Integration rules. (Left) Heun (trapezoidal) rule. (Right) Runge–Kutta (Simpson) rule.

The Matlab solvers which are based on the Heun's method are `ode2` (fixed step) and `ode23t`, which is variable-step.

Let us remark that u has been assumed unpredictable, which is the worst-case assumption, since the relevant prediction error may worsen the integration error and in extreme cases it may force the solver to halt due to overflow of the computed derivatives, with the following halt message (in red color on the `Command Window`).

```
Derivative of state '1' in block 'OregonatorTestSim/OregonX/Integrator' at time
2.71625 is not finite. The simulation will be stopped. There may be a
singularity in the solution. If not, try reducing the step size (either by
reducing the fixed step size or by tightening the error tolerances)
```

Actually, the previous halt was due to a large time unit, $T_s = 5$ ms, which made unstable the discrete-time equation of the solver [like that in Eq. (D.9)]. A time unit $T_s < 1$ms allowed the simulation to be successful as mentioned in Section D.7.2.

Let us rewrite the Heun's Eq. (D.8) in the simplified form (the input u has been neglected):

$$\begin{aligned}x(t+h) &= x(t) + \frac{h}{2}(f_0(t) + f_h(t+h)) \\ f_0(t) &= f(x(t), t), f_h(t+h) = f(x(t) + f(x(t), t)h)\end{aligned}, \tag{D.10}$$

where the integration increment $\Delta x(t) = x(t+h) - x(t)$ exploits the average of the estimated initial and final derivatives.

Eq. (D.10) can be extended to include the state derivative at intermediate points. The classical method is the Runge−Kutta method (P.O. Runge, 1777−1810, M.W. Kutta, 1867−1944) of fourth-order (RK4) in which the derivative $f_{h/2}$ at the midpoint $t+h/2$ is computed by the average:

$$\begin{aligned} f_{h/2}(t+h/2) &= \frac{1}{2}\left(f\left(x_1(t+h/2)\right)+f\left(x_2(t+h/2)\right)\right) \\ x_1(t+h/2) &= x(t)+hf_0(t)/2 \\ x_2(t+h/2) &= x(t)+hf\left(x_1(t+h/2)\right)=x(t)+hf_1(t+h/2) \end{aligned} \quad . \tag{D.11}$$

The derivatives in Eq. (D.11) are computed in two midpoints, extrapolated from the initial and mid point derivatives $f_0(t)$ and $f_1(t+h/2)$, respectively. The final expression follows the integration Simpson rule (T. Simpson, 1710−61, see Fig. D.6, right):

$$\begin{aligned} x(t+h) &= x(t)+\frac{1}{6}\left(f_0(t)+4\left(f_1(t+h/2)+f_2(t+h/2)\right)+f_h(t+h)\right) \\ f_h(t+h) &= f\left(x_2(t+h/2)+hf_2(t+h/2)\right) \end{aligned} \quad . \tag{D.12}$$

The RK4-based Matlab solvers are `ode4`, which is fixed step, and `ode45`, which is variable step [1].

D.7.2 Matlab solver comparison

The comparison concerns two fixed-step and one variable-step solver. The reference solver has been chosen to be the fixed-step eighth-order `ode8` (of the family of Runge−Kutta solvers). The solvers to be compared are `ode4` (fixed step) and `ode45` (variable step). The variable-step solver is treated as a fixed-step solver

1. by calling the solver from the statements

```
solver={'ode8','ode45','ode45'};
...
out=sim('OregonatorTestSim', 'Solver', solver{i});
```

2. by choosing `Fixed-step` and by setting the `Fixed-step size` to `Ts`, in the pane of `Configuration Parameters` (see Fig. D.4). The choice of the solver can be any, since it will be superseded by the previous call to `sim`.

The chosen model is the Oregonator Case 1 with $f=1$ of Chapter 4. The time unit T_s has been chosen to be compatible with the smallest time constant τ_{min} of the perturbation state matrix $A(\boldsymbol{x}(t))=\partial \boldsymbol{f}(\boldsymbol{x}(t))/\partial \boldsymbol{x}$ computed at any point $\boldsymbol{x}(t)$ of the trajectory (see Section B.3.3 of Appendix B). Having found the range $\tau_{min}\simeq 1\sim 10$ ms which depends on the point, we chose $T_s=0.5$ ms. A value >1ms, as previously mentioned, may force the solver to halt.

Fig. D.7 shows the simulation discrepancy $\tilde{\boldsymbol{x}}_{84}=\boldsymbol{x}_8-\boldsymbol{x}_4$ between `ode8` and `ode4` (in red color) and $\tilde{\boldsymbol{x}}_{85}=\boldsymbol{x}_8-\boldsymbol{x}_{45}$ between `ode8` and `ode45` (in blue color). Only the first and third coordinates $\{x,z\}$ of the transformed Oregonator variables are graphically plotted. The discrepancy $\tilde{\boldsymbol{x}}_{84}$ can be ascribed to `ode4` integration errors, whereas $\tilde{\boldsymbol{x}}_{85}=\boldsymbol{x}_8-\boldsymbol{x}_5$, being very small, can be ascribed to both solvers.

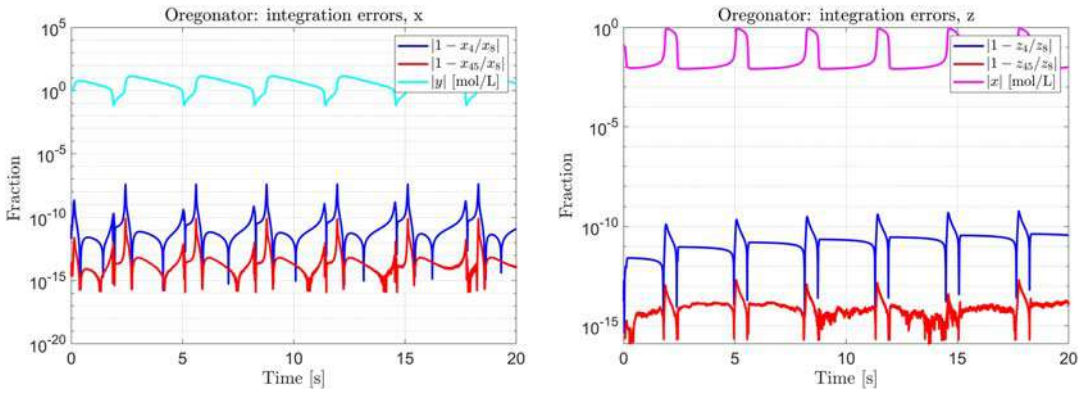


FIGURE D.7

Discrepancies between ODE solvers to be ascribed to integration errors. Oregonator case 1. (Left) variable x. (Right) variable z.

Table D.1 Fractional RMS of the discrepancies between ODE solvers.

No	Solver compared to `ode8`	Oregonator transformed variable $(\times 10^{-10})$			Simulation time [s]	Computing time [s]
		x	y	z		
1	`ode4`	2.5	0.59	0.49	20	0.50
2	`ode45`	0.0029	0.00075	0.00015	20	0.74
1	`ode8`	NA	NA	NA	20	1.5

Each coordinate is accompanied by the time profile of the other coordinate entering the relevant state Eq. (4.47) of Chapter 4, namely y for the coordinate x and x for the coordinate z. They are partly acting as an unpredictable input signal u especially during their fast jumps, as they look clearly synchronized with the peak integration errors.

The discrepancy between `ode8` and `ode45` is so close to the floating-point machine accuracy, $\varepsilon \simeq 2.2 \times 10^{-16}$, that you can, especially in Fig. D.7, right, observe the random drift of numerical errors.

Table D.1 reports the fractional RMS of the integration discrepancies of the three components of $\tilde{\boldsymbol{x}}_{84} = \boldsymbol{x}_8 - \boldsymbol{x}_4$ and $\tilde{\boldsymbol{x}}_{85} = \boldsymbol{x}_8 - \boldsymbol{x}_{45}$. The fractional RMS is defined for a generic component x of the error vector $\tilde{\boldsymbol{x}}_{84}$ by

$$s_{x,84} = \sqrt{\frac{\sum_{i=0}^{N-1} (x_8(i) - x_4(i))^2}{\sum_{i=0}^{N-1} x_8^2(i)}}. \tag{D.13}$$

The same definition applies to $\tilde{\boldsymbol{x}}_{85} = \boldsymbol{x}_8 - \boldsymbol{x}_{45}$.

Clearly, `ode45` performs as the best, at least in the current comparison, since integration errors are abated by a factor better than 1000 with respect to `ode4` at the price of a 50% longer computing time.

Reference

[1] L.F. Shampine, M.W. Reichelt, The MATLAB ODE suite, SIAM J. Sci. Comput. 18 (1997) 1–2.

APPENDIX

Table of seawater coefficients

E

The Matlab Scripts `SeaWaterSurfChemV3.m` and `SeaWaterDeepV3.m` are driven by different sets of coefficients entering the experimental expression of dissociation constants of the substances which dissolve in seawater. They were derived from Reference [1]. Such constants depend on water temperature T[K], pressure P[MPa], salinity S[g/kg], and the fractional volume V[ppmv] of the carbon dioxide, CO_2, in the dry air. They are collected in different sheets of the excel file `SeaWaterV4.xlsx`. The printout of such coefficients is performed by the script `SWInputDataPrintout`, which is called by the previously mentioned main scripts, and is reported below. Also, the printout tables in the `Command Window` are reported as a reference.

E.1 The printout script

Printout is triggered by `logInput = 1` in the initialization section of the main script.

```
%% 4) Possible printout of input data

if logInput==1
    disp('4) Printout of input data');
    fprintf('SeaWater:  Standard salinity  \n');
    fprintf('   Species     Concentration  Charge \n');
    for i=1:nrow
        fprintf('%12s', cell2mat(symbolS(i)));
        fprintf(' %12.5e %10.0f \n',SSTab{i,1},SSTab{i,2});
    end

    %% Carbonatic

    fprintf('SeaWater: Carbonatic species\n');
    fprintf('SeaWater: Temperature coefficients');
    fprintf('\nVariable      ');
    for i=1:5, fprintf('%12s', cell2mat(symbolT(i))); end
    for k=1:ncolT
        sprint=TTab.Properties.VariableNames(k);
        fprintf('%14s ',cell2mat(sprint));
        for i=1:5
            if  TTab{i,k}==0 || abs(TTab{i,k})==1,
                fprintf('%12.1f', TTab{i,k});
```

```
            else fprintf('%12.4e', TTab{i,k}); end
        end
        fprintf('\n');
    end
    % Salinity
    fprintf('SeaWater: Salinity coefficients');
    for k=1:ncolS
        sprint=STab.Properties.VariableNames(k);
        fprintf('\n%14s ',cell2mat(sprint));
        for i=1:5
            if  STab{i,k}==0 || abs(STab{i,k})==1,
                fprintf('%12.1f', STab{i,k});
            else  fprintf('%12.4e', STab{i,k}); end
        end
    end
    fprintf('\n');
    % Pressure
    fprintf('SeaWater: Pressure coefficients');
    for k=1:ncolP-1
        sprint=PTab.Properties.VariableNames(k);
        fprintf('\n%14s ',cell2mat(sprint));
        for i=1:5
            if PTab{i,k}==0 || abs(PTab{i,k})==1,
                fprintf('%12.1f', PTab{i,k});
            else fprintf('%12.4e', PTab{i,k}); end
        end
    end
    fprintf('\n');

    %% Other species

    fprintf('SeaWater: other species\n');
    fprintf('SeaWater: Temperature coefficients');
    fprintf('\nVariable      ');
    for i=6:nrowT,     fprintf('%12s', cell2mat(symbolT(i))); end
    for k=1:ncolT
        sprint=TTab.Properties.VariableNames(k);
        fprintf('%14s ',cell2mat(sprint));
        for i=6:nrowT
            if  TTab{i,k}==0 || abs(TTab{i,k})==1,
                fprintf('%12.1f', TTab{i,k});
            else  fprintf('%12.4e', TTab{i,k}); end
        end
        fprintf('\n');
    end
```

```
% Salinity
fprintf('SeaWater: Salinity coefficients');
for k=1:ncolS
    sprint=STab.Properties.VariableNames(k);
    fprintf('\n%14s ',cell2mat(sprint));
    for i=6:nrowT
        if  STab{i,k}==0 || abs(STab{i,k})==1,
            fprintf('%12.1f', STab{i,k});
        else  fprintf('%12.4e', STab{i,k}); end
    end
end
fprintf('\n');
% Ionic strength
fprintf('SeaWater: ionic coefficients');
for k=1:ncolI
    sprint=ITab.Properties.VariableNames(k);
    fprintf('\n%14s ',cell2mat(sprint));
    for i=6:nrowT
        if  ITab{i,k}==0 || abs(ITab{i,k})==1,
            fprintf('%12.1f', ITab{i,k});
        else  fprintf('%12.4e', ITab{i,k}); end
    end
end
fprintf('\n');
% Pressure
fprintf('SeaWater: Pressure coefficients');
for k=1:ncolP-1
    sprint=PTab.Properties.VariableNames(k);
    fprintf('\n%14s ',cell2mat(sprint));
    for i=6:nrowT
        if  PTab{i,k}==0 || abs(PTab{i,k})==1,
```

```
                fprintf('%12.1f', PTab{i,k});
            else fprintf('%12.4e', PTab{i,k}); end
        end
    end
    fprintf('\n');
    % Fugacity and H2O vapor pressure
    fprintf('SeaWater: fugacity, water vapor pressure\n');
    fprintf('\nVariable      ');
    for i=1:nrowF, fprintf('%12s', cell2mat(symbolF(i))); end
    fprintf('\n');
    for k=1:ncolF
        sprint=FTab.Properties.VariableNames(k);
        fprintf('%15s ',cell2mat(sprint));
        for i=1:nrowF
            if  FTab{i,k}==0 || abs(FTab{i,k})==1,
                fprintf('%12.1f', FTab{i,k});
            else fprintf('%13.4e', FTab{i,k}); end
        end
        fprintf('\n');
    end
end
```

E.2 Numerical tables

E.2.1 Standard salinity concentrations and electrical charge

```
SeaWater:  Standard salinity
   Species     Concentration  Charge
      Cl^{-}  5.45860e-01         -1
      Na^{+}  4.69060e-01          1
       K^{+}  1.02100e-02          1
     Sr^{2+}  9.00000e-05          2
     Mg^{2+}  5.28200e-02          2
   SO_4^{2-}  2.82400e-02         -2
     Ca^{2+}  1.02800e-02          2
      Br^{-}  8.40000e-04         -1
 B(OH)_4^{-}  4.20000e-04         -1
       F^{-}  7.00000e-05         -1
```

E.2.2 Coefficients of carbonatic species

1) Temperature and salinity coefficients

```
SeaWater: Carbonatic species
SeaWater: Temperature coefficients
Variable              H^{+}     (OH)^{-}      H_2CO_3     HCO_3^{-}     CO_3^{2-}
         Scale          0.0          1.0          1.0 -2.3026e+00 -2.3026e+00
      Constant          0.0  1.4898e+02 -1.6781e+02 -1.2634e+02 -9.0183e+01
           1/T          0.0 -1.3847e+04  9.3452e+03  6.3208e+03  5.1437e+03
        log(T)          0.0 -2.3652e+01  2.3358e+01  1.9568e+01  1.4613e+01
             T          0.0          0.0          0.0          0.0          0.0
SeaWater: Salinity coefficients
       sqrt(S)          0.0 -5.9770e+00          0.0  5.5930e+00  1.3397e+01
     sqrt(S)/T          0.0  1.1867e+02          0.0 -2.2575e+02 -4.7286e+02
 sqrt(S)log(T)          0.0  1.0495e+00          0.0 -8.7151e-01 -2.1563e+00
      sqrt(S)T          0.0          0.0          0.0          0.0          0.0
      sqrt(S)S          0.0          0.0          0.0          0.0          0.0
    sqrt(S)S/T          0.0          0.0          0.0          0.0          0.0
             S          0.0 -1.6150e-02  2.3517e-02  2.8845e-02  1.2193e-01
           S/T          0.0          0.0          0.0 -4.7610e+00 -1.9036e+01
       Slog(T)          0.0          0.0          0.0          0.0          0.0
            ST          0.0          0.0 -2.3656e-04          0.0          0.0
          ST^2          0.0          0.0  4.7036e-07          0.0          0.0
            S^2          0.0          0.0          0.0 -6.3880e-05 -3.8362e-04
          S^2/T          0.0          0.0          0.0
```

2) Pressure and temperature coefficients

Pressure P is assumed to be given in MPa, which implies the following value of the universal gas constant: $R = 8.3144$ J/(mol K). Coefficients do not change from those assuming that the pressure is given in bar = 100 kPa and consequently $R = 83.144$ cm^3 bar/(mol K).

```
SeaWater: Pressure coefficients
         P/(RT)          0.0  2.5600e+01          0.0  2.5500e+01  1.5820e+01
       TcP/(RT)          0.0 -2.3240e-01          0.0 -1.2710e-01  2.1900e-02
     Tc^2P/(RT)          0.0  3.6246e-03          0.0          0.0          0.0
       P^2/(RT)          0.0 -2.5650e-02          0.0 -1.5400e-02  5.6500e-03
     TcP^2/(RT)          0.0  3.9700e-04          0.0  4.3850e-04 -7.3750e-04
```

E.2.3 Other species

1) Temperature and salinity coefficients

SeaWater: other species					
SeaWater: Temperature coefficients					
Variable	HSO_4^{-}	CaCO_3(Cal)	CACO_3(Arg)	B(OH)_3	HF
Scale	1.0	2.3026e+00	2.3026e+00	1.0	1.0
Constant	1.4133e+02	-1.7191e+02	-1.7194e+02	1.4802e+02	-1.2641e+01
1/T	-4.2761e+03	2.8393e+03	2.9033e+03	-8.9669e+03	1.5902e+03
log(T)	-2.3039e+01	3.1093e+01	3.1093e+01	-2.4434e+01	0.0
T	0.0	-7.7993e-02	-7.7993e-02	0.0	0.0
SeaWater: Salinity coefficients					
sqrt(S)	0.0	-7.7712e-01	-6.8393e-02	1.3719e+02	0.0
sqrt(S)/T	0.0	1.7834e+02	8.8135e+01	-2.8905e+03	0.0
sqrt(S)log(T)	0.0	0.0	0.0	-2.5085e+01	0.0
sqrt(S)T	0.0	2.8426e-03	1.7276e-03	5.3105e-02	0.0
sqrt(S)S	0.0	4.1249e-03	5.9415e-03	0.0	0.0
sqrt(S)S/T	0.0	0.0	0.0	1.7280e+00	0.0
S	0.0	-7.7110e-02	-1.0018e-01	1.6214e+00	0.0
S/T	0.0	0.0	0.0	-7.7942e+01	0.0
Slog(T)	0.0	0.0	0.0	-2.4740e-01	0.0
ST	0.0	0.0	0.0	0.0	0.0
ST^2	0.0	0.0	0.0	0.0	0.0
S^2	0.0	0.0	0.0	0.0	0.0
S^2/T	0.0	0.0	0.0	-9.9600e-02	0.0

2) Ionic strength coefficients (temperature and salinity)

SeaWater: ionic coefficients					
sqrt(I)	3.2457e+02	0.0	0.0	0.0	1.5250e+00
sqrt(I)log(T)	-4.7986e+01	0.0	0.0	0.0	0.0
sqrt(I)/T	-1.3856e+04	0.0	0.0	0.0	0.0
I	-7.7154e+02	0.0	0.0	0.0	0.0
log(T)I	1.1472e+02	0.0	0.0	0.0	0.0
I/T	3.5474e+04	0.0	0.0	0.0	0.0
sqrt(I)I/T	-2.6980e+03	0.0	0.0	0.0	0.0
I^2/T	1.7660e+03	0.0	0.0	0.0	0.0
log(Sn)	1.0	0.0	0.0	0.0	1.0

3) Pressure and temperature coefficients

Pressure P is assumed to be given in MPa and consequently $R = 8.3144$ J/(mol K). Coefficients of the kind $P/(RT)$ do not change from those given under a pressure in bar $= 100$ kPa and $R = 83.144$ cm^3 bar/(mol K). Coefficients of the kind $P^2/(RT)$ are obtained by 10 times multiplying those given under pressure in bar $= 100$ kPa and $R = 83.144$ cm^3 bar/(mol K).

```
SeaWater: Pressure coefficients
         P/(RT)   1.8030e+01  4.8760e+01  4.6000e+01  2.9480e+01  9.7800e+00
       TcP/(RT)  -4.6600e-02 -5.3040e-01 -5.3040e-01 -1.6220e-01  9.0000e-03
     Tc^2P/(RT)  -3.1600e-04         0.0         0.0 -2.6080e-03  9.4200e-04
       P^2/(RT)  -2.2650e-02 -5.8800e-02 -5.8800e-02 -1.4200e-02 -1.9550e-02
     TcP^2/(RT)   4.5000e-04  1.8460e-03  1.8460e-03         0.0  2.7000e-01
```

E.2.4 Fugacity and H_2O vapor pressure

Temperature and salinity coefficients

```
SeaWater: fugacity, water vapor pressure
Variable      CO2 fugacity     H2O pressure
           Scale    1.0132e-01          1.0
        Constant    1.1805e+01  4.6784e+01
             1/T   -1.5213e+03 -6.7451e+03
          log(T)          0.0  -4.8489e+00
               T   -3.2796e-02          0.0
              T2    3.1653e-05          0.0
               S          0.0  -5.4400e-04
```

Reference

[1] F.J. Millero, Thermodynamics of the carbon dioxide system in the oceans, Geochim. Cosmochim. Acta 59 (1995) 661–677.

APPENDIX

The Schroedinger equation

F

F.1 Introduction

F.1.1 Time-dependent equation

The time-dependent form of the 3D Schroedinger equation (E. Schroedinger, 1887−1961) is the following [1]

$$j\bar{h}\frac{\partial\Psi(t)}{\partial t} = H(t)\Psi(t) \quad [\mathrm{J}] \quad , \tag{F.1}$$

where $j = \sqrt{-1}$ is the imaginary unit (the letter j instead of i is typical of electrical engineering), $\bar{h} = h/(2\pi) = 1.054 \times 10^{-34}$ Js is the reduced Planck constant (M. Planck, 1858−1947), H[J] is a Hamiltonian operator (W.R. Hamilton, 1805−65), sum of the kinetic energy T (the symbol should not be confused with that of absolute temperature, widely used in other parts of the book) and the potential energy V of a particle of mass m[kg] at a point $\boldsymbol{v} = [x, y, z]$:

$$H = -\left(\frac{\bar{h}^2}{2m}\right)\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}\right) + V = -\left(\frac{\bar{h}^2}{2m}\right)\nabla^2 + V = T + V. \tag{F.2}$$

$\Psi(\boldsymbol{v}, t)$ is the dimensionless wave function. Eq. (F.1) can be rewritten as

$$j\bar{h}\frac{\partial\Psi(\boldsymbol{v}, t)}{\partial t} = -\left(\frac{\bar{h}^2}{2m}\right)\nabla^2\Psi(\boldsymbol{v}, t) + V\Psi(\boldsymbol{v}, t), \tag{F.3}$$

where ∇^2 is the Laplacian (P.-S. Laplace, 1749−1827) in Cartesian (R. Descartes, 1596−1650) coordinates. The kinetic energy expression is typical of quantum mechanics and is the scalar product of the momentum operator $p = j\bar{h}\nabla$[kg m/s], where $\nabla = \left[\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}\right]$ is the gradient vector. The kinetic energy is thus defined by

$$T = \frac{p^T p}{2m} = -\left(\frac{\bar{h}^2}{2m}\right)\nabla^2. \tag{F.4}$$

F.1.2 Derivation

Eq. (F.1) looks different from wave equations in mechanics and electromagnetism, which asks for some justification.

1. The wave function $\Psi(\boldsymbol{v},t)$ being a wave in time should be defined as a complex function with magnitude and argument, that is

$$\Psi(\boldsymbol{v},t)=\left|\Psi(\boldsymbol{v},t)\right|\exp(j\arg\Psi(\boldsymbol{v},t)),\ \left|\Psi(\boldsymbol{v},t)\right|=\Psi(\boldsymbol{v},t)\Psi^{*}(\boldsymbol{v},t), \tag{F.5}$$

   where asterisk means conjugate.
2. We assume that the wave propagates at *constant magnitude*, which fact can be expressed by assuming *unitary magnitude* multiplied by a scale factor. If the wave propagates from the initial time $t=0$ under unitary magnitude, the wave at any successive time can be obtained by transforming the initial wave through a *unitary operator*, which preserves the magnitude.
3. A *unitary operator* of this sort must work in the complex plane and must be expressed as the *complex exponential of a Hermitian operator* $H(\boldsymbol{v},t)=H^{*}(\boldsymbol{v},t)$ (C. Hermite, 1822−1901):

$$\Psi(\boldsymbol{v},t)=\exp\left(-\frac{j}{\bar{h}}H(\boldsymbol{v},t)\Psi(\boldsymbol{v},t_0)\right),\ \exp\left(-\frac{j}{\bar{h}}H(\boldsymbol{v},t)\right)\exp\left(\frac{j}{\bar{h}}H^{*}(\boldsymbol{v},t)\right)=I, \tag{F.6}$$

   where I is the identity operator, such that

$$\Psi(\boldsymbol{v},t)\ \Psi^{*}(\boldsymbol{v},t)=\Psi(\boldsymbol{v},t_0)\ \Psi^{*}(\boldsymbol{v},t_0). \tag{F.7}$$

4. A Hermitian operator is the generalization of a symmetric matrix S, which given two vectors $\{\boldsymbol{v},\boldsymbol{w}\}$ satisfies the scalar product relationship $\boldsymbol{v}^{T}S\boldsymbol{w}=(S\boldsymbol{w})^{T}\boldsymbol{v}=\boldsymbol{w}^{T}S\boldsymbol{v}$ or in bracket form $\langle\boldsymbol{v},S\boldsymbol{w}\rangle=\langle S\boldsymbol{w},\boldsymbol{v}\rangle$. A typical Hermitian operator is the Hamiltonian operator in Eq. (F.3), since it is invariant under complex conjugation. A further key property of symmetric matrices and Hermitian operators is that all their *eigenvalues are real* and their *eigenvectors (eigenfunctions* in the case of *Hermitian operators) are each other orthogonal*. More important for quantum mechanics, the eigenvalues of Hermitian operators are *countable*, in other terms, each eigenvalue is associated to an *integer number*.
5. The first-order expansion of the complex exponential in Eq. (F.6) around a generic time t in Eq. (F.6) provides

$$\Psi(\boldsymbol{v},t+\Delta t)=\left(I-\frac{j}{\bar{h}}\Delta t\ H(\boldsymbol{v},t)\right)\Psi(\boldsymbol{v},t). \tag{F.8}$$

   Convergence under an infinitesimal time interval yields the Schroedinger equation:

$$j\bar{h}\lim_{\Delta t\to 0}\frac{\Psi(\boldsymbol{v},t+\Delta t)-\Psi(\boldsymbol{v},t)}{\Delta t}=j\bar{h}\frac{\partial\Psi(\boldsymbol{v},t)}{\partial t}=H(\boldsymbol{v},t)\Psi(\boldsymbol{v},t). \tag{F.9}$$

F.1.3 Time-independent equation

If the potential V is *time independent*, time and space separate and the solution of Eq. (F.2) is the linear combination of complex waves:

$$\Psi(\boldsymbol{v},t)=\sum\nolimits_{n=0}^{\infty}c_n\psi_n(\boldsymbol{v})exp\left(-j\frac{E_n t}{\bar{h}}\right), \tag{F.10}$$

where E_n[J] is an energy level, and the stationary spatial waves $\psi_n(\boldsymbol{v})$ satisfy the stationary Schroedinger equation

$$\left(\frac{-\bar{h}^2}{2m}\nabla^2+V\right)\psi(\boldsymbol{v})=E\psi(\boldsymbol{v})\Leftrightarrow H(\boldsymbol{v})\psi(\boldsymbol{v})=E\psi(\boldsymbol{v}), \tag{F.11}$$

where E is a generic energy level. The form of Eq. (F.11) is exactly the form of an eigenvalue/eigenfunction equation. The real number E is an eigenvalue and its value being countable, can be as well indicated by $E_n, n = 0, 1, \ldots$. The wave function $\psi(\boldsymbol{v})$ or better $\psi_n(\boldsymbol{v})$ is the eigenfunction (also eigenstate) associated to E_n. The Eq. (F.11), if solvable, says that eigenfunctions exist which, when transformed by the Hamiltonian $H(\boldsymbol{v})$, provide, up to the scalar E_n, the same eigenfunction.

We should remark that although the imaginary unit j has disappeared, there is no reason to constrain $\psi(\boldsymbol{v})$ to be real. However, aiming to atom's orbitals, we are interested in *real solutions*.

Coming back to the time-dependent Eq. (F.3), it describes the time evolution (transient) from an initial wave until the stationary wave is reached. The same occurs to linear dynamic systems (see Appendix B) in which certain input functions are repeated, once the transient is vanished, by the output functions up to gains playing the role of eigenvalues. Such functions are nothing else than complex trigonometric (or harmonic) functions in time, like $u(t) = \exp(j\omega t)$, which are repeated by the output up to a complex eigenvalue $P(j\omega)$ which depends on the properties of the dynamic system. Let us observe that also in this case, it is advantageous not to restrict to real functions. This means that the output $y(t)$ which converges to $y(t) = P(j\omega)\exp(j\omega t) = P(j\omega)u(t)$ can be rewritten by separating magnitude and argument as follows:

$$y(t) = P(j\omega)\exp(j\omega t) = \left|H(j\omega)\right|\exp(j(\omega t + \arg P(j\omega))). \tag{F.12}$$

The difference is that linear dynamic operators are in general not Hermitian, and therefore their eigenvalues are complex and uncountable. The eigenfunctions are time harmonic functions, whereas the stationary Schroedinger eigenfunctions are spatial harmonic functions, and precisely, they contain, as a factor, the spherical harmonic functions, the 2D harmonics of the functions defined on the unit sphere.

F.2 Solution of the stationary equation

F.2.1 Passing to spherical coordinates

Let us assume that the potential energy $V(\boldsymbol{v}) = V(r)$ is central, function of the distance $r = |\boldsymbol{v}|$ from the atom's nucleus center. In this case, the spherical coordinates $\{r, \varphi, \theta\}$, namely radius, azimuth, and polar angle, are the natural choice and the Laplacian operator becomes

$$\nabla^2 = \frac{1}{r^2}\left(\frac{\partial}{\partial r}\left(r^2\frac{\partial}{\partial r}\right) + \frac{1}{\sin\theta}\frac{\partial}{\partial\theta}\left(\sin\theta\frac{\partial}{\partial\theta}\right) + \frac{1}{\sin^2\theta}\frac{\partial^2}{\partial\phi^2}\right). \tag{F.13}$$

Replacement in Eq. (F.11) provides the new equation:

$$\frac{-\bar{h}^2}{2mr^2}\left(\frac{\partial}{\partial r}\left(r^2\frac{\partial\psi}{\partial r}\right) + \frac{1}{\sin\theta}\frac{\partial}{\partial\theta}\left(\sin\theta\frac{\partial\psi}{\partial\theta}\right) + \frac{1}{\sin^2\theta}\frac{\partial^2\psi}{\partial\phi^2}\right) + V(r)\psi(\mathbf{v}) = E\psi(\mathbf{v}), \tag{F.14}$$

where we abuse the notation $\boldsymbol{v} = \{r, \varphi, \theta\}$. We look for a factorized solution, product of the radial wave $R(r)$ depending on r and of the angular wave $Y(\varphi, \theta)$ only depending on $\{\varphi, \theta\}$. By inserting $\psi = RY$ into Eq. (F.14), we obtain:

$$\frac{-\bar{h}^2}{2mr^2}\left(Y\frac{\partial}{\partial r}\left(r^2\frac{\partial R}{\partial r}\right) + \frac{R}{\sin\theta}\frac{\partial}{\partial\theta}\left(\sin\theta\frac{\partial Y}{\partial\theta}\right) + \frac{R}{\sin^2\theta}\frac{\partial^2 Y}{\partial\phi^2}\right) + V(r)RY = ERY. \tag{F.15}$$

Dividing by RY and separating radial and angular functions provide the form:

$$\left(\frac{1}{R}\frac{\partial}{\partial r}\left(r^2\frac{\partial R}{\partial r}\right)\right) - \frac{2mr^2}{\hbar^2}(V(r)-E) = \frac{-1}{Y}\left(\frac{1}{\sin\theta}\frac{\partial}{\partial\theta}\left(\sin\theta\frac{\partial Y}{\partial\theta}\right) + \frac{1}{\sin^2\theta}\frac{\partial^2 Y}{\partial\phi^2}\right). \tag{F.16}$$

The terms on either side must be equal to the same constant in order to satisfy equality whichever be the pair $\{R, Y\}$. The important result to become clear in the solution is that the constant must be a positive integer, which can be expressed as $l(l+1)$, with l integer.

From Eq. (F.16), we obtain two separate equations, radial and angular:

$$\begin{array}{l} \dfrac{1}{R}\dfrac{\partial}{\partial r}\left(r^2\dfrac{\partial R}{\partial r}\right) - \dfrac{2mr^2}{\hbar^2}(V(r)-E) = l(l+1) \\ \dfrac{1}{Y}\left(\dfrac{1}{\sin\theta}\dfrac{\partial}{\partial\theta}\left(\sin\theta\dfrac{\partial Y}{\partial\theta}\right) + \dfrac{1}{\sin^2\theta}\dfrac{\partial^2 Y}{\partial\phi^2}\right) = -\,l(l+1) \end{array}. \tag{F.17}$$

The simplest equation is the angular one, as it does not depend on the potential V, whose form has yet to be defined.

F.3 Angular waves

F.3.1 Variable separation

Also the angular equation can be further separated, by factoring Y as $Y(\varphi,\theta) = \Theta(\theta)\Phi(\varphi)$ and multiplying by $\sin^2\theta$, which provides

$$\frac{\sin\theta}{\Theta}\frac{\partial}{\partial\theta}\left(\sin\theta\frac{\partial\Theta}{\partial\theta}\right) + l(l+1)\sin^2\theta + \frac{1}{\Phi}\frac{\partial^2\Phi}{\partial\varphi^2} = 0. \tag{F.18}$$

Again, we have two terms, depending on different angles, which must be constant to ensure equality to zero. In this case, the constant, which turns to be a positive integer, is denoted by m^2. The pair of equations are as follows

$$\begin{array}{l} \sin\theta\dfrac{\partial}{\partial\theta}\left(\sin\theta\dfrac{\partial\Theta}{\partial\theta}\right) = \left(-\,l(l+1)\sin^2\theta + m^2\right)\Theta \\ \dfrac{\partial^2\Phi}{\partial\varphi^2} + m^2\Phi = 0 \end{array}. \tag{F.19}$$

Both equations are second-order, but the former is nonlinear.

F.3.2 Azimuth equation

The *azimuth equation* is the classical oscillator equation, whose solution can be expressed in complex form as

$$\Phi(\varphi) = A_0\exp(\pm jm\varphi) = A_0(\cos(m\varphi) \pm j\sin(m\varphi)), \tag{F.20}$$

where the scale factor holds $A_0 = 1/\sqrt{2\pi}$ in order that $\Phi(\varphi)$ has unitary norm and m may possess any sign. Since the above solutions are each other conjugate, real solutions exist and are computed as combinations of the above components:

$$\begin{aligned}\Phi_c(\varphi) &= (\exp(jm\varphi) + \exp(-jm\varphi))/\left(2\sqrt{2\pi}\right) = \cos(m\varphi)/\sqrt{2\pi} \\ \Phi_s(\varphi) &= (\exp(jm\varphi) - \exp(-jm\varphi))/\left(2j\sqrt{2\pi}\right) = \sin(m\varphi)/\sqrt{2\pi}\end{aligned}. \tag{F.21}$$

To guarantee that $\Phi(\varphi)$ is periodic, namely that $\Phi(\varphi) = \Phi(\varphi + 2k\pi), k = \ldots -1, 0, 1 \ldots$, the constant m must be integer and unsigned as already anticipated:

$$m = \ldots -1, 0, 1, \ldots. \tag{F.22}$$

Let us remark that the norm of the real solutions for $|m| \geq 1$ is not unitary but holds $1/\sqrt{2}$.

F.3.3 Polar angle equation

The polar angle equation

$$\sin\theta \frac{\partial}{\partial\theta}\left(\sin\theta \frac{\partial\Theta}{\partial\theta}\right) = \left(-l(l+1)\sin^2\theta + m^2\right)\Theta \tag{F.23}$$

has a more intricate solution, namely

$$\Theta(\theta) = AP_l^m(\theta), \tag{F.24}$$

where the function $P_l^m(\theta)$ is the *associated Legendre function* (not polynomial, A.-M. Legendre, 1752−1833) defined by

$$\begin{aligned}P_l^m(x) &= (-1)^m\left(1-x^2\right)^{m/2}\left(\frac{d}{dx}\right)^m P_l(x),\ -1 \leqslant x \leqslant 1 \\ P_l(x) &= \frac{1}{2^l l!}\left(\frac{d}{dx}\right)^l \left(x^2-1\right)^l,\ -1 \leqslant x \leqslant 1\end{aligned}, \tag{F.25}$$

and by $x = \cos\theta$. The second row is the Rodrigues formula (O. Rodrigues, 1795−1851) of the Legendre polynomials $P_l(x = \cos\theta)$, which says that l must be nonnegative and that $m > l$, otherwise the associate Legendre function would become zero. Low-degree Legendre polynomials are as follows:

$$\begin{aligned}P_0 = 1, P_1 = x, P_2 = \frac{1}{2}\left(3x^2-1\right), P_3 = \frac{x}{2}\left(5x^2-3\right) \\ P_4 = \frac{1}{8}\left(35x^4-30x^2+3\right), P_5 = \frac{x}{8}\left(63x^4-70x^2+15\right)\end{aligned}. \tag{F.26}$$

Their profiles as a function of θ are plotted in Fig. F.1.

Using the abbreviations $c = \cos\theta, s = \sin\theta$, the low-degree associated Legendre functions have the following expressions:

$$\begin{aligned}P_0^0 = 1, P_1^1 = -s, P_1^0 = c \\ P_2^2 = 3s^2, P_2^1 = -3sc, P_2^0 = \frac{1}{2}\left(3c^2-1\right) \\ P_3^3 = -15s\left(1-c^2\right), P_3^2 = 15s^2c, P_3^1 = \frac{-3}{2}s\left(5c^2-1\right), P_3^0 = \frac{1}{2}c\left(5c^2-3\right)\end{aligned}. \tag{F.27}$$

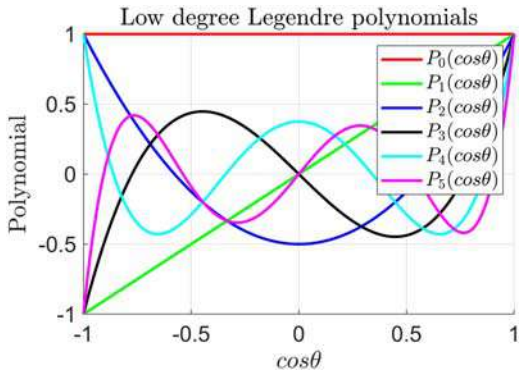


FIGURE F.1

Low-degree Legendre polynomials.

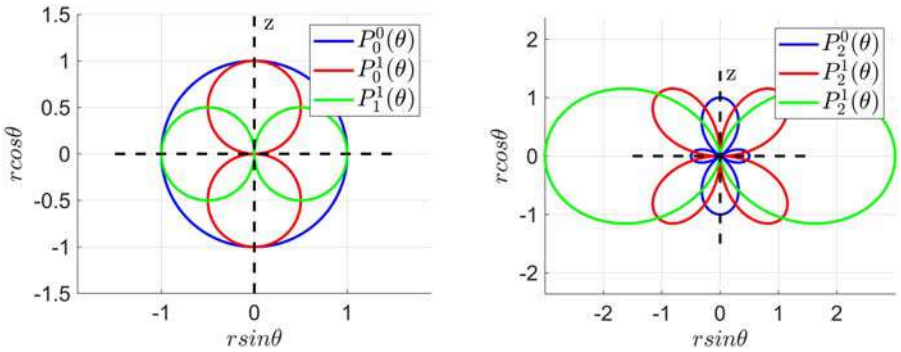


FIGURE F.2

Associated Legendre functions. (Left) $P_0^0(\theta)$ and $P_1^m(\theta), m = 0, 1$. (Right) $P_2^m(\theta), m = 0, 1, 2$.

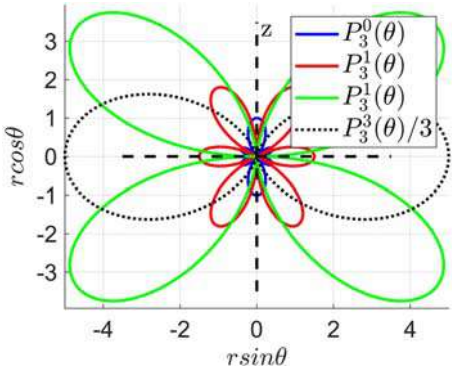


FIGURE F.3

Associated Legendre functions: $P_3^m(\theta), m = 0, 1, 2, 3$.

The functions in Eq. (F.27) are plotted in Figs. F.2 and F.3 in polar coordinates. In other terms, the abscissa (horizontal axis coordinate, say x) and the ordinate z are defined by:

$$x = r\sin\theta, z = r\cos\theta, r = \left|P_l^m(\cos\theta)\right|, \tag{F.28}$$

where θ is the polar angle as in Fig. 1.1 of Chapter 1, and r is the magnitude of the associated Legendre function in the direction defined by θ. Any real angular wave function $Y(\varphi,\theta) = \Theta(\theta)\Phi(\varphi)$ is nothing else than a 2D surface obtained by rotating a normalized associated Legendre function about the vertical axis and scaled by $\cos(m\phi)$ and $\sin(m\phi)$ as in Figs. 1.2−1.4 and 1.6 of Chapter 1.

The function $P_3^0(\theta)$ in Fig. F.3 has been scaled by 1/3.

The *angular functions* $Y = \Theta\Phi$ when normalized on the unit sphere as in Fig. F.3 are known as *spherical harmonics*, are each other orthogonal and are indicated in the complex form by

$$Y_l^m(\varphi,\theta) = N(l,m)\exp(jm\varphi)P_l^m(\cos\theta), \quad N(l,m) = \sqrt{\frac{(2l+1)(l-m)!}{4\pi(l+m)!}}, \tag{F.29}$$

where $N(l,m)$ is the normalization factor. If the real and imaginary parts of the exponential in Eq. (F.29) are used, their scale factor must be multiplied by $\sqrt{2}$ in order to keep normalization. The low-order real angular waves are listed in Table 1.1 of Chapter 1.

F.4 Radial function of the hydrogen atom

We start from the first equation in Eq. (F.17), rewritten as

$$\frac{\partial}{\partial r}\left(r^2\frac{\partial R}{\partial r}\right) = \frac{2mr^2}{h^{-2}}\left(V(r) - E + \frac{\overline{h}^2}{2mr^2}l(l+1)\right)R. \tag{F.30}$$

It is convenient to apply the transformation

$$R = \frac{u}{r}, \quad \frac{\partial}{\partial r}\left(r^2\frac{\partial R}{\partial r}\right) = r\frac{d^2u}{dr^2}, \tag{F.31}$$

which changes Eq. (F.30) into

$$\frac{d^2u}{dr^2} = \frac{2m}{\overline{h}^2}\left(V(r) - E + \frac{\overline{h}^2}{2mr^2}l(l+1)\right)u. \tag{F.32}$$

The hydrogen atom equation requires that $V(r)$ is replaced by the potential energy of the electron of charge $-e$ and mass m_e, given by the Coulomb law (C.A. de Coulomb, 1736−1806) as follows

$$V(r) = \frac{-e^2}{4\pi\epsilon_0 r}, \lim_{r\to\infty}V(r) = 0. \tag{F.33}$$

The hydrogen atom's radial equation is found to be

$$\frac{d^2u}{dr^2} = \left(\frac{-2m_e e^2}{4\pi\epsilon_0\overline{h}^2 r} - \frac{2m_e}{\overline{h}^2}E + \frac{l(l+1)}{r^2}\right)u. \tag{F.34}$$

The solution is studied for bound states, such that $E < V(r\to\infty) = 0$, which implies that

$$\kappa^2(E) = \frac{-2m_e}{\overline{h}^2}E > 0\left[\mathrm{m}^{-2}\right]. \tag{F.35}$$

By defining the dimensionless radius

$$\rho = \kappa(E)r, \rho_0(E) = \frac{m_e e^2}{2\pi\epsilon_0 \bar{h}^2 \kappa(E)}, \tag{F.36}$$

Eq. (F.34) simplifies to the second-order autonomous differential equation:

$$\frac{d^2u}{d\rho^2} = \left(1 - \frac{\rho_0(E)}{\rho} + \frac{l(l+1)}{\rho^2}\right)u. \tag{F.37}$$

The solution of Eq. (F.37) is not immediate, but it can be expressed by the following product:

$$\begin{gathered} u(\rho) = \rho^{l+1}\exp(-\rho)p(\rho), p_0 = p(\rho = 0) \neq 0 \\ \lim_{\rho\to\infty} \rho^{l+1}p(\rho) = p_\infty < \infty \\ \lim_{\rho\to 0} u(\rho) = p_0\rho^{l+1} \\ \lim_{\rho\to\infty} u(\rho) = p_\infty \exp(-\rho) \end{gathered} \tag{F.38}$$

The above limits imply that the second factor in Eq. (F.38) is the asymptotic solution of Eq. (F.37) for $\rho \to \infty$:

$$\begin{gathered} \frac{d^2u}{d\rho^2} = u \Rightarrow u(\rho\to\infty) = p_\infty \exp(-\rho) + 0 \times \exp(\rho) \\ u(0) = p_\infty, \quad \frac{du}{d\rho}(0) = -p_\infty \end{gathered}. \tag{F.39}$$

The first factor is the asymptotic solution for $\rho \to 0$:

$$\begin{gathered} \frac{d^2u}{d\rho^2} = \frac{l(l+1)}{\rho^2}u \Rightarrow u(\rho\to 0) = p_0\rho^{l+1} + 0 \times \rho^l \\ u(0) = p_0, \quad \frac{du}{d\rho}(0) = (l+1)p_0 \end{gathered}. \tag{F.40}$$

The asymptotic solution applies also for $l = 0$, since in this case $u(\rho) = p_0\rho$. The second terms of both asymptotic solutions in Eqs. (F.39) and (F.40) must be zero, since they, being divergent, would impede the radial solution to possess a finite integral for $0 \leq \rho < \infty$ and thus to be normalized. Essential for finding the complete solution is the limit $\lim_{\rho\to\infty} \rho^{l+1}p(\rho) = p_\infty < \infty$, which states that the polynomial $p(\rho)$ is of finite degree as follows:

$$p(\rho) = \sum_{k=0}^{N-1} c_k \rho^k. \tag{F.41}$$

With some effort, by replacing $u(\rho) = \rho^{l+1}\exp(-\rho)p(\rho)$ in Eq. (F.37), one finds the recursive expression of the polynomial coefficients

$$c_{k+1} = \frac{2(k+1+l) - \rho_0(E)}{(k+1)(k+2l+2)}c_k, k = 0, 1, \ldots, N-1, \tag{F.42}$$

where c_0 is found by the normalization of the radial function (see below). By imposing $c_{N+1} = 0$, we find the finite polynomial degree

$$2(N+l+1) = \rho_0(E) \Rightarrow \rho_0(E_n) = 2(N+l+1) = 2n, \tag{F.43}$$

where $\rho_0(E_n)$ is made integer by selecting the energy level E_n, and n is the principal quantum number. The energy level holds from Eqs. (F.35) and (F.36):

$$E_n = \frac{-\overline{h}^2\kappa^2}{2m_e} = \frac{-\overline{h}^2}{2m_e}\left(\frac{m_e e^2}{2\pi\epsilon_0\overline{h}^2\rho_0}\right)^2 = -\frac{1}{n^2}\frac{\overline{h}^2}{2m_e}\left(\frac{m_e e^2}{4\pi\epsilon_0\overline{h}^2}\right)^2 = \\ = -\frac{1}{n^2}\frac{\overline{h}^2}{2m_e}\frac{1}{a_0^2} = \frac{E_1}{n^2}, n = 1, 2, \ldots \tag{F.44}$$

and corresponds to the famous *Bohr formula* (N. Bohr, 1885−1962), which anticipated the Schroedinger equation by 13 years. The parameter $a_0 = 0.0529$ nm is the *Bohr radius*. The energy level $E_1 = -13.6$ eV, where 1 eV $= 0.160 \times 10^{-18}$ J, is the *ground state*, in other terms it is the energy to impart for ionizing a hydrogen atom. It follows also that $\rho = r(a_0 n)^{-1}$.

From Eqs. (F.31) and (F.38), we obtain the radial function

$$R_{nl}(r) = \frac{1}{r}\left(\frac{r}{a_0 n}\right)^{l+1}\exp\left(\frac{-r}{a_0 n}\right)p(r) \\ p(r) = \sum_{k=0}^{N-1} c_k\left(\frac{r}{a_0 n}\right)^k, c_{k+1} = \frac{2(k+1+l-n)}{(k+1)(k+2l+2)}c_k \tag{F.45}$$

Let us consider the case $n = 1, l = 0$, which implies $c_0 \neq 0, c_k = 0, k \geq 1$. From Eq. (F.45), we find:

$$R_{10}(r) = R(r, 1s) = \frac{1}{a_0}\exp\left(\frac{-r}{a_0}\right)c_0. \tag{F.46}$$

Normalization follows Eq. (1.19) in Chapter 1. We normalize the radial probability density, that is

$$\int_0^\infty R_{10}^2(r)r^2 dr = 1 \Rightarrow c_0 = \frac{2}{\sqrt{a_0}} \rightarrow R(r, 1s) = 2\frac{1}{a_0^{3/2}}\exp\left(\frac{-r}{a_0}\right), \tag{F.47}$$

where the atomic number $Z = 1$ (for the hydrogen atom) does not appear, unlike in Table 1.2 of Section 1.1.3. In Section 1.4, the radial probability density $f_r(r, ns) = R(r, ns)^2 r^2$ is graphically shown, which, for $n = 1$, holds

$$f_r(r, 1s) = \frac{4r^2}{a_0^3}\exp\left(-2\frac{r}{a_0}\right). \tag{F.48}$$

One can easily check that the peak value corresponds to $\overline{r}_{10} = a_0$, that is to the Bohr radius. The radial functions for $n \leq 4$ are reported in Table 1.2 of Chapter 1.

Reference

[1] D.J. Griffiths, D.F. Schroeter, Introduction to Quantum Mechanics, Third Edition, Cambridge University Press, 2018.

Index

Note: Page numbers followed by "*f*" and "*t*" refer to figures and tables, respectively.

D

I

K

L

M

N

T

V

W

Z

FUNDAMENTAL CHEMISTRY WITH MATLAB

A guide to using MATLAB scripts, models, and algorithms to explore the fundamentals and applications of key topics in chemistry

Key Features

- Provides practical examples of using the MATLAB platform to explore contemporary problems in chemistry
- Outlines the use of MATLAB Simulink to produce block diagrams for dynamic systems, such as in chemical reaction kinetics
- Heavily illustrated with supportive block diagrams and both 2D and 3D MATLAB plots throughout

MATLAB offers huge capabilities for data processing, graphical presentation, and modeling, making it an important tool for chemists in navigating their increasingly data-driven field. *Fundamental Chemistry with MATLAB* highlights how this functionality can be used to explore the fundamentals and applications of key topics in chemistry.

After an introduction to MATLAB, this book goes on to provide examples of its application in both fundamental and developing areas of chemistry, from atomic orbitals, chemical kinetics, and gaseous reactions, to clean coal combustion and ocean equilibria among others. Complimentary scripts and datasets are provided to support experimentation and learning, with scripts thoroughly outlined enabling the reader to modify them to fit their particular needs.

Drawing on the experience of its expert authors, *Fundamental Chemistry with MATLAB* is a practical guide for all those in chemistry who are interested in harnessing the scripts, models, and algorithms of the MATLAB platform to further enhance their use, understanding, and visualization of data.

About the Authors

Daniele Mazza was a professor of Chemistry and Materials Science at the Politecnico di Torino. His main research fields were the electrical and crystallochemical properties of ionic conductors and ceramic superconductors.

He is author or coauthor of more than 80 publications in international journals about solid-state chemistry, crystal structure determination, X-ray diffraction, and modelization of ionic movement in crystalline lattices. In addition, he is author of three books in chemistry. His current research interests focus on environmental and climatic issues, in particular ocean chemistry and CO_2 equilibria in seawater.

Enrico Canuto has taught Automatic Control for more than 40 years at Politecnico di Torino, Turin, Italy. He developed and applied the Embedded Model Control methodology for the design and implementation of digital control systems. He contributed to data reduction of the European astrometric mission Hipparcos, concluding with the publication of the Hipparcos star Catalogue of 120,000 stars. He was active in the design of spacecraft control systems, contributing to the European GOCE mission, to other forthcoming missions, and to instruments for space qualification like the Nanobalance thrust-stand. In the last 10 years he has collaborated with the Center for Gravity Experiments, Huazhong University of Science and Technology, Wuhan, and the Tianqin Centre, Sun-Yat-Sen University, Zhuhai, China, in the field of scientific space missions. He has authored one book and published over 150 papers.

ISBN 978-0-323-91341-6

9 780323 913416

ELSEVIER elsevier.com/books-and-journals