Heino Prinz

Numerical Methods for the Life Scientist

Binding and Enzyme Kinetics Calculated with GNU Octave and MATLAB

Springer

Numerical Methods for the Life Scientist

Heino Prinz

Numerical Methods for the Life Scientist

Binding and Enzyme Kinetics Calculated with GNU Octave and MATLAB

The source code of all programs is available on extras.springer.com

Springer

Dr. Heino Prinz
Max-Planck-Institut für molekulare Physiologie
AG Biochemische Analytik
Otto-Hahn-Str. 11
44227 Dortmund
Germany
heino.prinz@mpi-dortmund.mpg.de

ISBN 978-3-642-20819-5 e-ISBN 978-3-642-20820-1
DOI 10.1007/978-3-642-20820-1
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2011933466

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

This textbook gives an introduction to numerical methods. They are key tools for the calculation of reaction schemes and for the development of functional models in life sciences. The most exciting of this is model development, but it probably cannot be taught in a book. It is surprisingly personal. I will illustrate some aspects together with an acknowledgment of the scientists who impressed me most.

Multiple theoretical approaches: I was fortunate to study physics at Bonn University at a time when Wolfgang Paul gave his main lectures on “Experimentalphysik”. The lectures themselves were spectacular, but the emphasis was laid on the variety of possible interpretations of the same experimental findings. For the topic of gravity, for example, Wolfgang Paul included geocentric astronomy and concluded that one cannot rule out angels carrying planets in retrograde loops around the earth, but that the assumption of gravity is so much simpler. For an experimental physicist the observation itself is the result. Theoretical models are only limited by the creativity of the analyst. All are valid when they can be verified experimentally.

Courage and creativity: Friedrich Cramer was my PhD supervisor at the Max-Planck Institute in Göttingen. He had published a structural model for tRNA based on a map of reactive groups. When I entered his group, this model just had been disproven, so that he was criticized for having published a preliminary model. Model building is part of a scientific discourse, and one sometimes needs courage to publish new ideas. Friedrich Cramer was a very creative, sometimes artistic, and sometimes philosophical personality. He has strongly influenced me and many other scientists in Göttingen.

Additional low-affinity binding: In the beginning of the 1980s, we had employed a newly synthesized fluorescent cholinergic agonist for molecular studies of cholinergic excitation. For a detailed pharmacological characterization of this drug Alfred Maelicke and I traveled to Göttingen, where we had convinced Bert Sakmann and Erwin Neher to perform a series of electrophysiological studies at different ligand concentrations. They confirmed that the ligand acted as a pure agonist at low concentrations, but cautioned that it changed channel open times and thereby acted as a local anesthetic at high concentrations on the same receptor. I as a physicist who had calculated a model for cholinergic excitation found this particularly disturbing, but they had observed such concentration dependency before. It is well known that the physiological action of any compound depends on its dose, but before this moment of truth I had thought that different target proteins were responsible for the different interactions at different concentrations. Erwin Neher and Bert Sakmann were the first to perform functional studies on one single molecule. Independent of this, multiple interactions of ligands at the same target molecule may be quite common as discussed in Sect. 4.8. Unfortunately, additional low-affinity binding spoils the elegance of any mechanistic model. Additional sites are at odds with many published crystal structures and rigid docking programs used in drug design.

Open-minded observations and scientific communication: One cannot just sit down and write a new theory without experimental evidence. One cannot even plan for new functional models. Sometimes it just happens: One summer evening we were sitting in a beer garden when Jörg Striessnig, an Innsbruck pharmacologist, mentioned a discrepancy between a theoretical prediction and a common observation in the lab. He had observed that the addition of a second ligand to an existing ligand–receptor complex resulted in dissociation rate constants, which increased with the concentration of the second ligand. This observation corresponds to the German saying "viel hilft viel" (the more you take, the more it helps), and therefore "feels" all right, but, of course, it contradicts first-order dissociation kinetics. In the end, we calculated the observation with the assumption of overlapping sites as shown in Sect. 5.5. One really should talk science in relaxing environments.

Quantitative plausibility check: It is a good idea to present kinetic data to a critical audience before publication. Some questions can be as basic as: "When you have an equilibrium dissociation constant below nM, how can you have free protein?" The numerical values of rate constants imply scientific information which is evident to an experienced scientist. I have often admired Roger Goody for his overview.

Different scientific approaches: Scientists are individuals, who may have entirely different approaches to the same question. These differences can lead to individual animosities, but they also can lead to fruitful cooperation. Cooperation can be encouraged when the person in charge has a natural human understanding coupled to clear scientific goals. Herbert Waldmann is one scientist with such management qualities and I am happy to work in his department.

Dortmund, Germany Heino Prinz

Contents

Chapter 1
Introduction

The analysis of quantitative data in biochemistry usually requires the calculation of theoretical models. Their analytical solutions are functions given as explicit formulas

$$y = f(x)$$

The independent variable x may be the concentration of a substrate or the time, and the computed value y may be the activity of an enzyme, or intrinsic fluorescence of a receptor, or any other parameter which can be computed from a theory. Finding such an analytical function f(x) often has been a lengthy process. For complex reaction schemes, it is easier to resort to numerical methods. These are instructions sent to a computer to do the calculations.

$$x \rightarrow \boxed{\begin{array}{c}\text{Computer running}\\ \text{a program with a}\\ \text{set of instructions}\end{array}} \rightarrow y$$

Mathematically, these are also functions [1], even though they are not based on explicit formulas. The set of computer instructions, or "algorithms" [2], assigns one value y to each independent variable x. The output therefore is equivalent to any other function y = f(x). It may be understood with the "back box" concept [3], whereby computer and its program are considered an unknown entity (black box) which gives reproducible and accurate output y for a given input x. The calculations described in this book are not concerned with details of the black box. They simply use it. The resulting function is no formula, but is shown as a graph.

Only three types of numerical methods [4] are covered here, namely one method to solve equations for binding equilibria, one method to calculate differential equations for binding kinetics and one method to fit the data. Once the principles are understood, it is easier to calculate complex reactions than to understand them with complex simplifications.

DOI 10.1007/978-3-642-20820-1_1, © Springer-Verlag Berlin Heidelberg 2011

Enzyme kinetics, binding kinetics and equilibrium binding have been studied for more than 100 years. These topics are well presented in many modern textbooks [5–11]. Their analysis is based on analytical solutions [12] and therefore is limited to a given number of readily calculated reaction schemes. Numerical methods overcome these limitations. They employ computers so that Chaps. 4–8 should be read with an accompanying computer.

Programming is done in GNU Octave, a high-level language which is relatively easy to learn and which includes simple graphical output. GNU Octave is available for most computer platforms (Linux, windows, Mac, Sun, OS/2) and free to us all [13]. It had originally been developed as companion software for a textbook on reactor design [14]. Now it is continuously enhanced and refined by the GNU community [15]. It is compatible to MATLAB®, a very popular commercial high-level computer language [16]. MATLAB® is well supported by MathWorks™, its proprietor, who also offers excellent training courses. Numerous lectures or tutorials for MATLAB® and/or GNU Octave are available in many spoken languages on the Internet. In some universities it belongs to the curriculum of the mathematics departments. GNU Octave requires no financial investment and seems to be the ideal software to get started. Readers with a MATLAB® license may prefer the MATLAB® environment and should run the MATLAB® versions of the programs.

GNU Octave is introduced step by step. The installation of the program is described in the fourth chapter, and some basic commands are explained there. From then on the language is applied to more than 50 practical examples of increasing complexity. All sample programs, including the MATLAB® versions, can be found in http://www.mpi-dortmund.mpg.de/misc/numericalmethods/. Each sample program leads to a plot. Such a plot gives a relation of dependent and independent values and is a graphical description of the numerical function $x \rightarrow y$. An explicit formula is not required, and the plot illustrates the underlying mechanism. In the end the readers and their computers should be able to calculate and fit any reaction scheme, be it as complex as they like.

Exporting Octave programs to MATLAB® requires minor changes because the Octave functions `lsode` and `leasqr` are similar but not identical to their MATLAB® equivalents. The differences are explained when the functions are used. The numerical methods introduced in this textbook are based on algorithms available in most computer languages. When speed is an issue, one may wish to use a high-level language with a fast compiler. One important example should be mentioned; facsimile [17] is a dedicated program package based on the Fortran Harwell Subroutine Library. It has been developed specifically for the calculation of chemical reactions and is available under the GNU public license [13]. Unfortunately, Fortran is not directly compatible with Octave or MATLAB.

The book is organized as follows: The second chapter covers the basics, namely first- and second-order reactions. It explains how binding equilibria can be calculated from sets of ordinary equations, and how binding kinetics is calculated from sets of differential equations. Analytical solutions of these equations are limited to some examples, the most basics of which are summarized in Chap. 3. Chapter 4 gives an introduction to GNU Octave. Chapter 5 covers equilibrium binding. The first sample program, `EQ1.m`, is explained line by line, introducing all Octave commands as they come along. It compares a numerical and an analytical solution of simple reversible binding. From then on, the reaction schemes become more and more complex, covering numeric solutions for more than one site, more than one ligand and allosteric interactions. Chapter 6 deals with binding kinetics and its calculations on the basis of differential equations. The sample programs focus on the discrimination of molecular models. Enzyme kinetics is covered in Chap. 7, where initial velocities and steady-state approximations are compared to progress curves and numerical solutions of differential equations. Chapter 8 explains and discusses the methods and limits of data fitting.

Different scientists may wish to use the book differently. A physicist may skim through Chap. 2 to summarize the basics and then proceed to Chap. 3, which explains common analytical methods in biochemical binding studies and enzyme kinetics. He or she may skim through Chap. 4 before using the sample programs in the following chapters as a tutorial for enzyme kinetics and Octave alike. A chemist may skip Chap. 2, but should begin with Chap. 3, where simple reversible reactions in biochemistry are summarized. A Biologist should begin with Chap. 4 and focus on the principles of the computer language. Independent of the individual approach, Chaps. 2 and 3 are written as reference for the sample programs provided in Chaps. 5–8.

There are some *practical aspects* worth mentioning: Octave code and Octave file names, line numbers, etc. will be set in `Courier` font throughout the book; the units throughout the book are μM for the concentrations and seconds for the time; a forward rate constant of 0.1 in program code thus translates to 0.1 $\mu M^{-1}\ s^{-1}$, which is equal to $10^5\ M^{-1}\ s^{-1}$.

All sample programs are free as Open Access under the Creative Common Attribution License [18]. Readers are encouraged to copy, distribute and modify the programs as they like. They may be taken as seeds for further developments and the readers are encouraged to share new programs at http://www.mpi-dortmund.mpg.de/misc/numericalmethods/. Just send the new programs together with a short description as an e-mail attachment to heino.prinz@mpi-dortmund.mpg.de. A teaching course on numeric methods consists of a series of five lectures (2–3 h each) covering Chaps. 4–8. It may be arranged with the author.

Acknowledgments This book has been written in the Max-Planck-Institute for molecular physiology in Dortmund and owes to its stimulating scientific environment. Part of this book was presented as a lecture to its IMPRS students http://www.imprs-cb.mpg.de/. They asked important questions so that an improved course could be taught to students and faculty of the Suranaree University of Technology http://www.sut.ac.th. This led to the development of a teaching course. Alexander Fieroch introduced me to the free world of GNU software, and thus provided the stimulus to write a textbook on numerical methods. He wrote the instructions to download GNU Octave for Mac and Linux operating systems, which is provided in the appendix.

References

1. Ruthing D (1984) Some definitions of the concept of function from Bernoulli, Joh. to Bourbaki, N. Math Intel 6(4):72–77
2. Rogers H (1987) Theory of recursive functions and effective computability. The MIT Press, Cambridge, MA
3. Beizer B (1995) Black-box testing: techniques for functional testing of software and systems. Wiley, New York
4. Leader JJ (2004) Numerical analysis and scientific computation. Addison Wesley, Reading, MA
5. Gutfreund H (1995) Kinetics for the life sciences. Receptors, transmitters and catalysts. Cambridge University press, Cambridge
6. Segel IH (1993) Enzyme kinetics: behavior and analysis of rapid equilibrium and steady-state enzyme systems. Wiley Classical Library, New York
7. Copeland RA (2005) Evaluation of enzyme inhibitors in drug discovery: a guide for medicinal chemists and pharmacologists. Wiley-VCH, Weinheim
8. Copeland RA (2000) Enzymes: a practical introduction to structure, mechanism, and data analysis. Wiley, New York
9. Purich DL (2010) Enzyme kinetics: catalysis & control: a reference of theory and best-practice methods. Elsevier, London
10. Cook PF, Cleland WW (2007) Enzyme kinetics and mechanism. Garland Science, New York
11. Leskovac V (2003) Comprehensive enzyme kinetics, Kindle Edition. Amazon
12. Michaelis L (1912) Einführung in die Mathematik für Biologen und Chemiker. Springer, Berlin
13. GNU general public license: http://www.gnu.org/licenses/gpl.html
14. Rawlings JB, Ekerdt JG (2002) Chemical reactor analysis and design fundamentals. Nob Hill Publishing, Madison, WI
15. GNU Octave Homepage: http://www.gnu.org/software/octave/
16. MATLAB Homepage: http://www.mathworks.com/products/matlab/
17. Facsimile: http://www.facsim.org/ Powell MJD (1979) 25 years of theoretical physics 1954–1979: Chapter XVIII: numerical analysis. A special publication by Harwell Research Laboratory of UKAEA
18. http://creativecommons.org/licenses/by/3.0/

Chapter 2
The Basics

Chemical reactions of the first and second order are the basis of all binding kinetics, binding equilibria and enzyme kinetics. Kinetic experiments must be calculated from a set of differential equations, whereas equilibrium binding studies can be calculated from a set of ordinary equations. Chapter 2 provides guidelines to write such equations for any reaction scheme. This is independent of the solution, be it analytical or numerical.

2.1 Equations for First and Second Order Reactions

When a ligand L binds to a receptor R (2.1), the probability of interaction is proportional to both concentrations. The rate of product formation d[LR]/dt therefore is the product of these concentrations times a constant, which is called the rate constant k. Rate constants are denoted as lower case letters.

$$\text{Second order association}: \ \mathrm{L + R \rightarrow LR} \tag{2.1}$$

$$\text{Second order rates}: \ \mathrm{d[LR]/dt = k_1 \cdot [L] \cdot [R]} \tag{2.2}$$

$$\mathrm{d[L]/dt = -k_1 \cdot [L] \cdot [R]} \tag{2.3}$$

$$\mathrm{d[R]/dt = -k_1 \cdot [L] \cdot [R]} \tag{2.4}$$

These are differential equations. They have to be written for the concentration changes of all components of the reaction, for L, R and LR. Note that we follow the convention and write concentrations in square brackets. This convention cannot be upheld in computer code, where a square bracket cannot be part of a variable name. In octave and MATLAB, square brackets define a vector or matrix.

Once the complex LR has been formed, it may or may not dissociate. The dissociation rate is proportional to the concentration of the complex.

$$\text{First order dissociation}: \quad \mathrm{LR} \rightarrow \mathrm{L} + \mathrm{R} \tag{2.5}$$

$$\text{First order rates}: \quad \mathrm{d[LR]/dt} = -\mathrm{k}_{-1} \cdot \mathrm{[LR]} \tag{2.6}$$

$$\mathrm{d[L]/dt} = \mathrm{k}_{-1} \cdot \mathrm{[LR]} \tag{2.7}$$

$$\mathrm{d[R]/dt} = \mathrm{k}_{-1} \cdot \mathrm{[LR]} \tag{2.8}$$

The negative sign in (2.6) indicates that the concentration of LR is decreased upon the decay of the complex. Reaction schemes (2.1) and (2.5) together with the basic reaction rates (2.2)–(2.4) and (2.6)–(2.8) contain all the physics required to understand this textbook. Complex biochemical reactions result from bimolecular association and monomolecular dissociation and combinations of these steps.

The above differential equations have been solved analytically, but let us use this example to understand numerical solutions of differential equations: Mathematically, a differential quotient dx/dt is the limiting value of the difference quotient Δx/Δt for infinitesimal small differences. Computer programs use a practical version of this definition: Calculate the differences Δx of all concentrations for a small time interval Δt, add these differences to the original values and do the calculations for the next time interval. Repeat these steps until the time range of interest is covered. That is all. For ordinary differential equations (they can be written as dx/dt = f(x, t), like all combinations of first and second order reactions) octave provides the function `lsode`, a general procedure to solve them. The function `lsode` will be covered in Chap. 6, but let us discuss the general principle with the simple example of a reversible reaction:

$$\mathrm{L} + R \underset{\mathrm{k}_{-1}}{\overset{\mathrm{k}_1}{\rightleftharpoons}} \mathrm{LR} \tag{2.9}$$

Reversible binding (2.9) leads to the formation of the complex LR and to a corresponding decrease of L and R. The difference quotients are:

$$\Delta\mathrm{[L]}/\Delta\mathrm{t} = -\mathrm{k}_1 \cdot \mathrm{[L]} \cdot \mathrm{[R]} + \mathrm{k}_{-1} \cdot \mathrm{[LR]} \tag{2.10}$$

$$\Delta\mathrm{[R]}/\Delta\mathrm{t} = -\mathrm{k}_1 \cdot \mathrm{[L]} \cdot \mathrm{[R]} + \mathrm{k}_{-1} \cdot \mathrm{[LR]} \tag{2.11}$$

$$\Delta\mathrm{[LR]}/\Delta\mathrm{t} = \mathrm{k}_1 \cdot \mathrm{[L]} \cdot \mathrm{[R]} - \mathrm{k}_{-1} \cdot \mathrm{[LR]} \tag{2.12}$$

When all rate constants and all initial concentrations are known, the concentration changes Δ[LR], Δ[L] and Δ[R] can be calculated from (2.10)–(2.12) for any given time interval Δt.

Let us assume that the experiment simply consists of mixing receptor and ligand to total concentrations R0 and L0, respectively. At time zero, the concentrations are [R] = R0, [L] = L0, [LR] = 0. After the small time interval Δt, these concentrations are [R] = R0 + Δ[R], [L] = L0 + Δ[L] and [LR] = 0 + Δ[LR].

For the next time interval Δt these new concentrations have to be inserted in (2.10)–(2.12) in order to compute the new concentration changes, and so forth. These are very basic repetitive operations, ideally suited for a computer.

Equilibrium is reached for reversible reactions (2.9) when the dissociation rate (1.6) is equal to association rate (2.2)

$$k_1 \cdot [L] \cdot [R] = k_{-1} \cdot [LR] \tag{2.13}$$

This leads to (2.14) with the equilibrium dissociation constant K_D1. Equilibrium constants $K_1 = k_1/k_{-1}$ and equilibrium dissociation constants $K_D1 = k_{-1}/k_1$ are denoted with uppercase letters. In life science and in this textbook, equilibrium dissociation constants K_D are used rather than equilibrium constants K. They correspond to the ligand concentration where half of the binding sites for this K_D are occupied.

$$K_D1 = k_{-1}/k_1 = [L] \cdot [R]/[LR] \tag{2.14}$$

$$[LR] = [L] \cdot [R]/K_D1 \tag{2.15}$$

Bound ligand and free ligand concentrations must add up to the total ligand concentration L0. The same must hold for all components of bound and free receptors which add up to the total receptor concentration R0:

$$R0 = [R] + [L] \cdot [R]/K_D \tag{2.16}$$

$$L0 = [L] + [L] \cdot [R]/K_D \tag{2.17}$$

Equations (2.16) and (2.17) are two equations with the two unknowns [L] and [R]. They follow the pattern: “The total concentration of each molecule must be the sum of free and bound concentrations”. This pattern leads to *n* equations of *n* unknowns. Even the most complex equilibria can be calculated from these equations. Analytical solutions for such general equations involving numerous complexes and numerous ligands may be difficult or impossible to find, but numerical solutions are relatively easy with the help of a suitable algorithm. Again, as has been discussed for differential equations, the main task for the scientist lies in writing the equations.

2.2 Equilibrium Binding to Two Sites

The binding of ligands to two sites of a receptor is often described with reaction scheme (2.18), where a ligand may bind to two sites. Only one singly bound complex LR and one doubly bound L_2R are considered in this scheme. The scheme can be extended to incorporate more binding sites, always with only one complex and one equilibrium dissociation constant for each sequential binding step

$$2L + R \underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} L + LR \underset{k_{-2}}{\overset{k_2}{\rightleftharpoons}} L_2R \tag{2.18}$$

Scheme (2.18) is ideally suited for the analysis of equilibrium binding studies, since it involves a minimal number of constants. It is, however, not a plausible scheme. Generally, one would expect the ligand to have a choice for binding to one or to the other site, before the second ligand binds to the non-occupied site. Such a mechanism of accessible sites is shown in scheme (2.19) where the first ligand may bind to the "left" (LR) or "right" (RL) site, before a second ligand leads to the formation of the fully occupied receptor LRL.

$$\begin{tikzcd}
 & L + LR \arrow[dl, shift left, "k_{-1}"] \arrow[dr, shift left, "k_3"] & \\
2L + R \arrow[ur, shift left, "k_1"] \arrow[dr, shift left, "k_2"] & & LRL \arrow[ul, shift left, "k_{-3}"] \arrow[dl, shift left, "k_{-4}"] \\
 & L + RL \arrow[ul, shift left, "k_{-2}"] \arrow[ur, shift left, "k_4"] &
\end{tikzcd} \tag{2.19}$$

Reaction scheme (2.19) has sometimes been called "diamond model" because of its shape. It illustrates the most general model for two site binding. Sites are considered to be independent when the ligand bound to one site has the same affinity independent of the occupation of the other site. This statement translates to $K_D1 = K_D4$ and $K_D2 = K_D3$. The sites are equivalent when their affinity is the same, i.e. $K_D1 = K_D2$. For equivalent sites, binding is regarded to be cooperative, when $K_D1 > K_D3$, so that the second ligand binds with a higher affinity. For $K_D1 < K_D3$ and equivalent sites, the binding mechanism is called anticooperative.

One should note that reaction scheme (2.19) shows a coupled equilibrium and that the affinity of the ternary complex LRL must be independent of the path by which it was formed.

$$\text{Upperpath}: [LR] = [L] \cdot [R]/K_D1 \quad [LRL] = [LR] \cdot [L]/K_D3 \tag{2.20}$$

$$\text{Lowerpath}: [RL] = [L] \cdot [R]/K_D2 \quad [LRL] = [RL] \cdot [L]/K_D4 \tag{2.21}$$

If we combine all four equations, we get

$$[L] \cdot [L] \cdot [R]/(K_D1 \cdot K_D3) = [L] \cdot [L] \cdot [R]/(K_D2 \cdot K_D4) \tag{2.22}$$

And thus

$$K_D1 \cdot K_D3 = K_D2 \cdot K_D4 \tag{2.23}$$

One equilibrium dissociation constant of a coupled equilibrium can therefore be calculated from the other ones. This principle holds for all coupled equilibria, since the affinity of a complex must be independent of the path by which it was formed. The concentrations of the complexes LR, RL and LRL, written in (2.20)–(2.22) are functions of free concentrations [L] and [R]. The total concentrations R0 and L0 must therefore be equal to

$$\mathrm{R0} = [\mathrm{R}] + [\mathrm{L}] \cdot [\mathrm{R}]/\mathrm{K_D1} + [\mathrm{L}] \cdot [\mathrm{R}]/\mathrm{K_D2} + [\mathrm{L}] \cdot [\mathrm{L}] \cdot [\mathrm{R}]/(\mathrm{K_D1} \cdot \mathrm{K_D3}) \tag{2.24}$$

$$\mathrm{L0} = [\mathrm{L}] + [\mathrm{L}] \cdot [\mathrm{R}]/\mathrm{K_D1} + [\mathrm{L}] \cdot [\mathrm{R}]/\mathrm{K_D2} + 2 \cdot [\mathrm{L}] \cdot [\mathrm{L}] \cdot [\mathrm{R}]/(\mathrm{K_D1} \cdot \mathrm{K_D3}) \tag{2.25}$$

Note that the complex LRL contains two bound ligand molecules. This is taken into account by the factor 2 in (2.25). For reaction scheme (2.18), the corresponding sets of equations are given in (2.26) and (2.27):

$$\mathrm{R0} = [\mathrm{R}] + [\mathrm{L}] \cdot [\mathrm{R}]/\mathrm{K_D1} + [\mathrm{L}] \cdot [\mathrm{L}] \cdot [\mathrm{R}]/(\mathrm{K_D1} \cdot \mathrm{K_D2}) \tag{2.26}$$

$$\mathrm{L0} = [\mathrm{L}] + [\mathrm{L}] \cdot [\mathrm{R}]/\mathrm{K_D1} + 2 \cdot [\mathrm{L}] \cdot [\mathrm{L}] \cdot [\mathrm{R}]/(\mathrm{K_D1} \cdot \mathrm{K_D2}) \tag{2.27}$$

Note that $\mathrm{K_D2}$ in reaction scheme (2.18) indicates the occupation of the second ligand to form the fully saturated complex LRL, whereas $\mathrm{K_D2}$ in reaction scheme (2.19) denotes the binding of one ligand to an independent second site. Equations (2.24) and (2.25) can be re-written as (2.28) and (2.29).

$$\mathrm{R0} = [\mathrm{R}] + [\mathrm{L}] \cdot [\mathrm{R}] \cdot (1/\mathrm{K_D1} + 1/\mathrm{K_D2}) + [\mathrm{L}] \cdot [\mathrm{L}] \cdot [\mathrm{R}]/(\mathrm{K_D1} \cdot \mathrm{K_D3}) \tag{2.28}$$

$$\mathrm{L0} = [\mathrm{L}] + [L] \cdot [\mathrm{R}] \cdot (1/\mathrm{K_D1} + 1/\mathrm{K_D2}) + 2 \cdot [\mathrm{L}] \cdot [\mathrm{L}] \cdot [\mathrm{R}]/(\mathrm{K_D1} \cdot \mathrm{K_D3}) \tag{2.29}$$

They are identical when $1/\mathrm{K_D1}$ for scheme (2.18) equals $(1/\mathrm{K_D1} + 1/\mathrm{K_D2})$ from scheme (2.19) and $\mathrm{K_D1} \cdot \mathrm{K_D2}$ from scheme (2.18) equals $\mathrm{K_D1} \cdot \mathrm{K_D3}$ from scheme (2.19).

Equilibrium binding studies therefore cannot distinguish between the sequential scheme (2.18) and a more plausible scheme of accessible sites (2.19). It will be shown in Sect. 6.5 that a sequential binding mechanism (2.18) can be identified with a kinetic "chase" experiment. But only a *kinetic* experiment with two different ligands allows the distinction between first and second bound ligand.

2.3 Equilibrium Binding to Any Number of Sites

These conclusions can be generalized: Equilibrium binding studies can only determine one effective affinity for each occupation of sites, no matter how many individual complexes can be identified for each monoliganded, diliganded, triliganded, etc. occupation. If we know the intrinsic affinities of each individual binding site at each step, we can compute the simplified affinities of a sequential scheme. This reasoning does not work in the opposite direction: Since equilibrium binding studies can only determine one effective equilibrium dissociation constant for the different occupation of sites, intrinsic equilibrium dissociation constants for the different sites cannot be deduced without further assumptions.

Only if we assume that affinities of all *n* individual binding sites are equivalent, can intrinsic equilibrium dissociation constants $K_D i_{intrinsic}$ for the *i*th steps be calculated from the effective (sequential) equilibrium dissociation constants (2.30).

$$K_D i_{intrinsic} = ((n - i + 1)/i) \cdot K_D i_{sequential} \qquad (2.30)$$

As an example, let us assume two step sequential binding sites with $K_D 1_{sequential} = K_D 2_{sequential} = 10\ \mu M$ according to reaction scheme (2.18). These equilibrium dissociation constants of scheme (2.18) translate to $K_D 1_{intrinsic}$ ($=K_D 2_{intrinsic}$) $= 20\ \mu M$ and $K_D 3_{intrinsic}$ ($=K_D 4_{intrinsic}$) $= 5\ \mu M$ of reaction scheme (2.19). Therefore, identical equilibrium dissociation constants for each step of a sequential mechanism (like (2.18)) translate into cooperative binding when intrinsic equilibrium dissociation constants of accessible sites are computed.

Calculating equilibrium binding curves from any given reaction scheme is quite straight-forward. One simply writes one equation (2.31) for each molecule of the reaction

$$\text{Total concentration} = \text{free concentration} + \text{bound concentration} \qquad (2.31)$$

This corresponds to (2.24) and (2.25). The bound concentration is the sum of all molecules bound to all complexes. The set of (2.31) can be solved numerically. These equations need not be linear, and there may be more than one solution. A reasonable initial estimate ensures that the right solution is found. This is discussed in Chap. 5.

2.4 Writing Differential Equations for Two Site Binding

Differential equations can readily be solved with numerical methods, but they have to be set up first. For example, analyzing reaction scheme (2.18) involves the calculation of four concentrations, namely [R], [L], [LR], and [L_2R]. The four differential equations for the four concentration changes are given in (2.32)–(2.35)

$$d[R]/dt = -k_1 \cdot [R] \cdot [L] + k_{-1} \cdot [LR] \qquad (2.32)$$

$$d[L]/dt = -k_1 \cdot [R] \cdot [L] + k_{-1} \cdot [LR] - k_2 \cdot [LR] \cdot [L] + k_{-2} \cdot [L_2R] \qquad (2.33)$$

$$d[LR]/dt = k_1 \cdot [R] \cdot [L] - k_{-1} \cdot [LR] - k_2 \cdot [LR] \cdot [L] + k_{-2} \cdot [L_2R] \tag{2.34}$$

$$d[L_2R]/dt = k_2 \cdot [LR] \cdot [L] - k_{-2} \cdot [L_2R] \tag{2.35}$$

Note the algebraic signs. Since all concentrations and rate constants are positive, a decrease in concentration, that is a negative rate, always is indicated by a negative sign. Note (2.34), where LR is decreased by the dissociation part of reaction 1 and the association part of reaction 2. Any second order reaction involves three components of the reactions. Therefore, reaction 1 with k_1 and k_{-1} affects the concentration changes of R, L and LR in (2.32), (2.33) and (2.34). Likewise, reaction 2 with k_2 and k_{-2} involves the components L, LR and L_2R and thus affects (2.33), (2.34) and (2.35). For more complex schemes, such a check that a given rate constant must be involved in three (or two, for reversible first order) reactions helps in debugging a program. One can use the "find" function included in any text editor to make sure that each rate constant of a reversible reaction of the second order appears three times.

Differential equations for reaction scheme (2.19) require the calculation of five concentrations, namely [R], [L], [LR], [RL], and [LRL]. This is shown in equations (2.36)–(2.40):

$$d[R]/dt = -k_1 \cdot [R] \cdot [L] + k_{-1} \cdot [LR] - k_2 \cdot [R] \cdot [L] + k_{-2} \cdot [RL] \tag{2.36}$$

$$\begin{aligned} d[L]/dt = &- k_1 \cdot [R] \cdot [L] + k_{-1} \cdot [LR] - k_3 \cdot [LR] \cdot [L] + k_{-3} \cdot [LR] \\ &- k_2 \cdot [R] \cdot [L] + k_{-2} \cdot [LR] - k_4 \cdot [RL] \cdot [L] + k_{-4} \cdot [LRL] \end{aligned} \tag{2.37}$$

$$d[LR]/dt = k_1 \cdot [R] \cdot [L] - k_{-1} \cdot [LR] - k3 \cdot [LR] \cdot [L] + k_{-3} \cdot [LRL] \tag{2.38}$$

$$d[RL]/dt = k_2 \cdot [R] \cdot [L] - k_{-2} \cdot [RL] - k_4 \cdot [LR] \cdot [L] + k_{-4} \cdot [LRL] \tag{2.39}$$

$$d[LRL]/dt = k_3 \cdot [LR] \cdot [L] - k_{-3} \cdot [LRL] + k_4 \cdot [LR] \cdot [L] - k_{-4} \cdot [LRL] \tag{2.40}$$

The more complex a reaction scheme becomes, the more and longer differential equations have to be computed. Each reversible reaction corresponds to two products of rate constant and component of the reactions. Note that the free ligand L is involved in all four reactions, so that there are eight products in (2.37). There are two reactions involving the free receptor R in reaction scheme (2.19), so that there are four products in (2.36). Likewise, two pathways for the dissociation of the complex LRL are reflected in four products in (2.40).

One important restriction must be considered: Reaction scheme (2.19) contains a closed loop which corresponds to coupled equilibria calculated in (2.23). This reduces the number of independent rate constants from 8 (4 reversible reactions) to 7. Equation (2.23), written for rate constants instead of equilibrium constants, translates into:

$$k_1 \cdot k_3 \cdot k_{-2} \cdot k_{-4} = k_2 \cdot k_4 \cdot k_{-1} \cdot k_{-3} \tag{2.41}$$

Again (2.41) ensures consistency, just like (2.23). It may be interpreted as a rule for circular reactions: The product of rate constants in one (clockwise) direction of reaction scheme (2.19) must be the same as the product of rate constants in the other (counterclockwise) direction. One of the rate constants in a closed loop therefore can be calculated from the other ones.

2.5 Writing Differential Equations for Any Reaction Scheme

Writing differential equations for a complex reaction scheme may look complicated, but it does not require much fantasy. First, one counts all complexes and free ligands. Then one has to write one, and possibly a long one, differential equation for the concentration change of each of these components. For reversible reactions in a fixed volume in solution, these concentration changes can only be expected from second or first order reactions. Each second order reaction has to be considered in the concentration changes of all three components of the reaction. There may be first order conformational changes which involve only two components (one for each conformation) or first order decay which only involves one component, provided the product is irrelevant and the decay is not reversible. Closed circles in reaction schemes have to fulfill the criterion of (2.41), namely that the products of all rate constants in one direction must be the same as the products of rate constants in the other direction.

That is all. But when differential equations are written as part of a program code, no typing error is allowed. It helps to use a "find" function in a program editor and look for all the rate constants individually. Any rate constant involved in a bimolecular reversible reaction must appear thrice. Whenever a given concentration appears in the differential equation of this concentration, the algebraic sign in front of the accompanying rate constant must be negative. For reversible reactions, the number of products in each differential equation must be even. Those little controls may help.

2.6 Analytical and Numerical Solutions

Only for the simplest cases, the sets of equations described above can be solved analytically. But when an analytical solution is found, it is precise and reliable for all feasible concentrations. Finding analytical solutions needs a lot of effort, but calculating them can be done from one formula with simple spread sheets or pocket calculators. Some of the most important analytical solutions are covered in Chap. 3.

Numerical methods are different. They are well established, but they depend on a computer and a program to run them with. The algorithms are based on approximations. The results are the same within reasonable errors, but they are not identical to analytical formulas. Figure 5.2 illustrates this with one example. All calculations done by computers are limited by the precision of the stored variables. When small differences of large numbers approach the precision of those large numbers, they become unreliable. Octave and MATLAB typically issue warnings when the internal precision is not sufficient. In general, whenever stochastic results are computed with numerical methods, one should repeat the calculations with other parameters (concentrations or rate constants). Use a computer, but keep checking it and do not develop unconfined trust!

Chapter 3
Classical Analytical Solutions

Reversible binding to one site can be calculated with analytical solutions. On the basis of these formulas, data can be transformed to appear linear. Straight lines of equilibrium-binding studies in double reciprocal plots or of enzyme kinetics in Lineweaver-Burk plots or of dissociation kinetics in half logarithmic plots indicate simple mechanisms. Deviations from these are discussed in detail as cooperative or independent sites. Logistic functions commonly used to calculate dose–response curves, only correspond to a binding mechanism when the Hill coefficient is one. At the end of this chapter, the reader should be able to identify simple biochemical reactions.

3.1 Analytical Solutions for Equilibrium Binding

Simple reversible binding (scheme (2.9)) under equilibrium conditions may be calculated from (2.15) to (2.17). Computing [R] from (2.16) and inserting it into (2.15) results in the well-known hyperbolic binding function:

$$[LR] = [L] \cdot R0/([L] + K_D) \tag{3.1}$$

3.1.1 Direct Binding Curves

Figure 3.1 shows a binding curve corresponding to (3.1) with a receptor concentration of 100 μM, a K_D value of 10 μM and a maximum free ligand concentration of 100 μM. It is characterized by a linear, maximal concentration increase at low free ligand concentration, followed by a steady decrease in its slope, leading to an asymptotic approach to its maximal value at high ligand concentrations.

Almost all figures in this textbook show the calculations with Octave programs, and the names of the program files (like `ana1.m`) are indicated in brackets after the title. The program files are listed as source code and are supplied in the supplementary material. They are not the topic of Chap. 3, but may serve later as independent exercises.

The maximal bound ligand is equal to the total receptor concentration R0 = 100 μM, and is indicated as a solid line in Fig. 3.1. The concentration of free ligand which leads to 50% saturation of receptor (50 μM bound ligand) is equal to the equilibrium dissociation constant K_D. This is also indicated as a solid line in Fig. 3.1.

Figure 3.1 shows the calculation with a receptor concentration ten times the equilibrium dissociation constant. Usually, binding curves are measured with receptor concentrations around the K_D value. This is not relevant for the shape of the curve when bound ligand is plotted versus the *free* ligand concentration (Fig. 3.1 and (+) in Fig. 3.2). When bound ligand is plotted versus *total* ligand concentration ((o) in Fig. 3.2), the curves differ, depending on receptor concentration and affinity. For extremely high affinities (low K_D value) or for irreversible reactions, the ligand binds quantitatively until saturation is reached. This yields a straight line with a sharp bend, the solid line in Fig. 3.2. The bend marks the total receptor concentration R0.

K_D values can be determined directly from 50% binding only when binding is plotted versus the free ligand concentration (Fig. 3.1). When plotted versus total concentration (Fig. 3.2), the curves may differ considerably, depending on the receptor concentration. In most cases the receptor concentration is in the order of

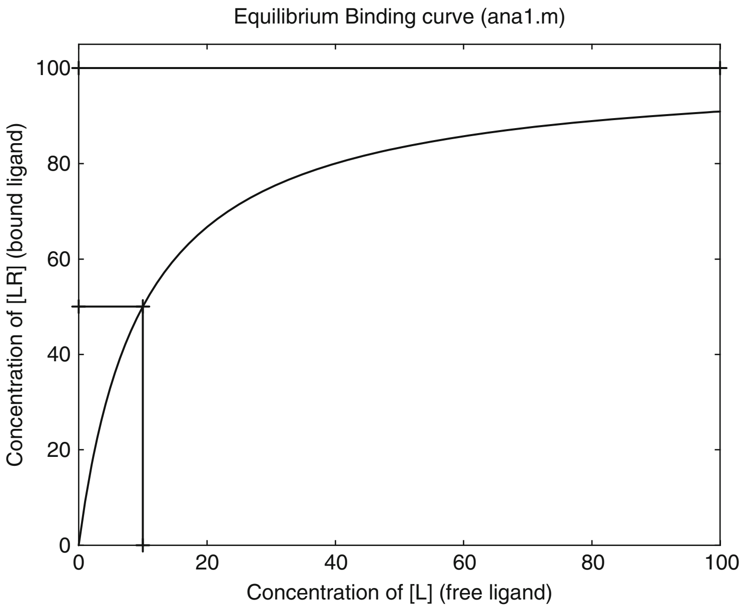


Fig. 3.1 Equilibrium-binding curve (bound ligand vs. free ligand), calculated with R0 = 100 μM and K_D = 10 μM. The maximal bound ligand R0 and half-maximal bound ligand are indicated by solid lines, giving the asymptote and the K_D values, respectively

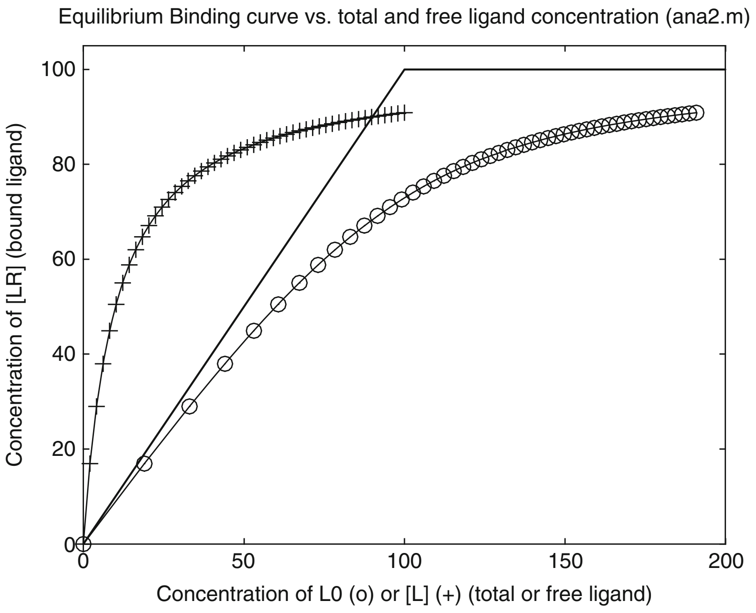


Fig. 3.2 Equilibrium-binding curve vs. total and free ligand concentration. R0 = 100 μM, K_D = 10 μM. Bound ligand [LR] is plotted vs. free (+) and total (o) ligand concentration. Analytical solutions are calculated with `ana2.m`. The solid line corresponds to the limiting case of extremely high affinity or irreversible binding, where all ligands are bound and no free ligand is available until the receptor is saturated. The sharp bend therefore is observed at L0 = R0 = 100 μM

magnitude lower than the ligand concentration. For these common cases, bound ligand is only a tiny fraction of the total ligand concentration so that the difference between total and free ligand concentration can be disregarded. The concentrations of pharmacological receptors are much more than thousand times lower than the administered drugs so that 50% occupation of these receptors is expected when the total drug concentration (at the target tissue) equals to the K_D value. If a physiological effect is caused only by the binding of a drug to the receptor molecule [1], then half-maximal binding to this receptor will lead to a half-maximal effect. The effective drug concentration for 50% activity (EC_{50}) must therefore be equal the equilibrium dissociation constant K_D of the drug to the receptor.

3.1.2 Double Reciprocal Plots

If one takes the inverse values of both sides of (3.1), it becomes:

$$1/[LR] = (1/[L]) \cdot (K_D/R0) + 1/R0 \tag{3.2}$$

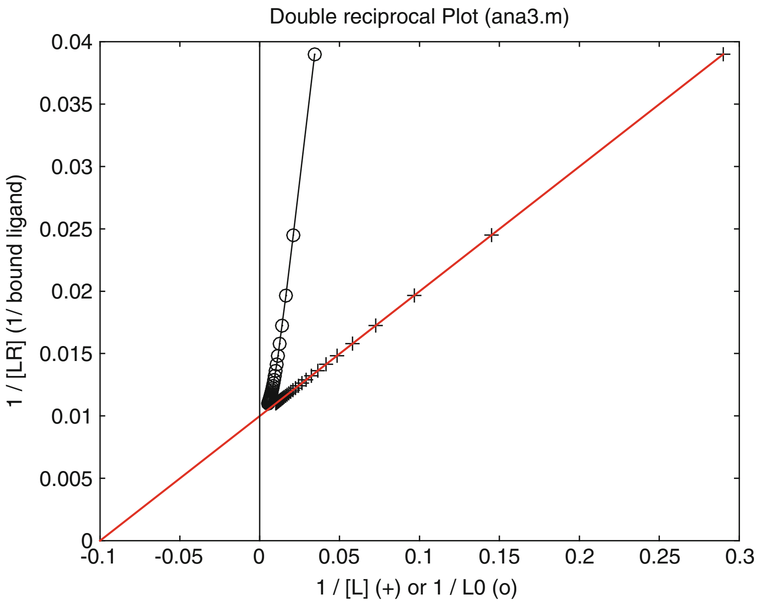


Fig. 3.3 Double reciprocal Plot with R0 = 100 μM, K_D = 10 μM (same data as shown in Fig. 3.2). 1/[LR] is plotted vs. 1/free (+) and 1/total (o) ligand concentration. The theoretical straight line is extrapolated to (0, $-1/K_D$)

Therefore, when 1/[LR] is plotted versus 1/[L], the resulting curve should be a straight line with a slope of K_D/R0 and the intersection of 1/R0 at the y-axis. If the line is extrapolated toward negative values, the intersection with the x-axis will be at $-1/K_D$. This is shown in Fig. 3.3.

Indeed the double reciprocal plot of bound and free ligand (indicated by the symbol (+)) is linear. Many functions appear linear, or almost linear, in a double reciprocal plot (compare Figs. 3.4 and 3.5). A double reciprocal plot of bound ligand versus total ligand ((o) in Fig. 3.2) also appears linear. If this curve would inadvertently be fitted to a straight line, the resulting K_D values and receptor concentrations might be far too high.

3.1.3 Comparison of Scatchard Plots and Double Reciprocal Plots

Double reciprocal plots (Fig. 3.3) sometimes appear linear, even when the plotted function is not a simple binding curve (o). Another transformation of (3.1) is shown in (3.3)

$$[LR]/[L] = -[LR]/K_D + R0/K_D \tag{3.3}$$

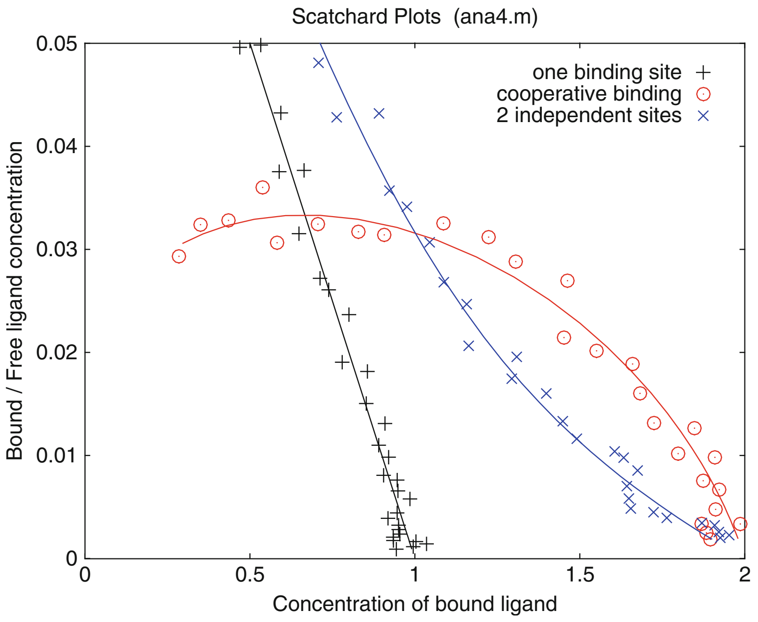


Fig. 3.4 Scatchard Plots. R0 = 1 μM. (+) Binding to one site with K_D = 10 μM. (o) cooperative binding to two equivalent sites (scheme 1.19, $K_D2 = K_D1$), K_D1 = 100 μM, K_D3 = 10 μM. (x) Two independent binding sites [scheme (2.19), $K_D3 = K_D2$] with K_D1 = 10 μM, K_D2 = 100 μM. The theoretical curves plotted are shown as *solid lines*. A random noise of 5% was assumed for both free and bound ligand. Data points were calculated from logarithmic distributions of ligand concentrations

Plotting bound ligand divided by free ligand versus the bound ligand concentration will give a straight line with a negative slope $-1/K_D$. This is called a Scatchard plot [2–5] and shown in Fig. 3.4. The corresponding double reciprocal plot is shown in Fig. 3.5.

Scatchard plots are more sensitive or responsive than double reciprocal plots. Higher sensitivity also means that experimental errors influence the shape of the curve to a larger extent. This becomes obvious when theoretical curves (solid lines shown in Figs. 3.4 and 3.5) are supplemented with "experimental" data corresponding to a 5% random noise.

The data points shown in Figs. 3.4 and 3.5 were calculated from ligand concentrations distributed in a logarithmic scale. Logarithmic distributions help to get more significant information with the same number of data points. This can be seen when a double reciprocal plot with linear distribution of data points (Fig. 3.3) is compared with the same type of plots and a logarithmic distribution (Fig. 3.5). Linear distributions plotted in a double reciprocal scale simply do not have enough data points at low ligand concentrations. Experimental noise of these sparsely distributed points influences the shape of the curve disproportionally. Transformed data such as generated from double reciprocal or Scatchard plots

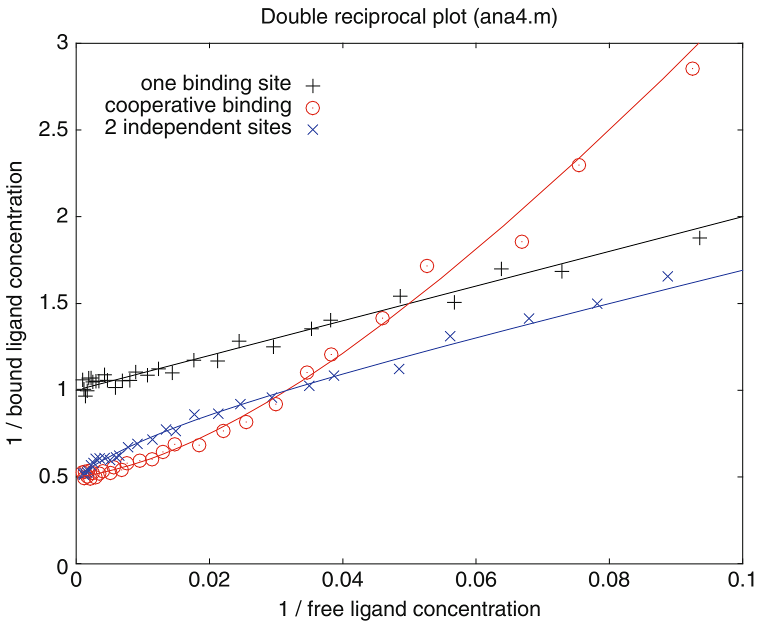


Fig. 3.5 Double reciprocal plot (same data as shown in Fig. 3.4) R0 = 1 μM. (+) Binding to one site with K_D = 10 μM. (o) Cooperative binding to two sites, K_D1 = 100 μM, K_D2 = 10 μM. (x) Two independent binding sites with K_D1 = 10 μM, K_D2 = 100 μM. The theoretical curves plotted are shown as *solid lines*. A random noise of 5% was assumed for both free and bound ligand. Data points were calculated from logarithmic distributions of ligand concentrations

generally should not be used for data fitting because the experimental errors are not proportional to the transformed data values. Fits to the original data are more reliable. They require nonlinear fitting routines which are covered in Chap. 8.

A simple binding curve (3.1) gives a straight line in a Scatchard plot ((+) in Fig. 3.4). The interpolation to the x-axis gives the concentration of maximal bound ligand. For one binding site on the receptor molecule, this is identical to the receptor concentration R0. If there are two binding sites on the receptor, this value increases by the factor 2. These sites may have different affinities for the same ligand. If one binding site is characterized by an equilibrium dissociation constant of 10 μM, and the other by 100 μM, the data show a marked curvature (x) in the Scatchard plot (Fig. 3.4), whereas the same data almost appear linear in a double reciprocal plot (Fig. 3.5). This is the reason why Scatchard plots have been used to distinguish between cooperative (o) and noncooperative (x) binding. It should be mentioned that a Scatchard Plot is the same as an Eadie-Hofstee plot when y- and x-axes are exchanged.

The comparison of Figs. 3.4 and 3.5 shows that Scatchard plots are superior to detect systematic deviations from simple binding equilibrium. Double reciprocal plots always have a tendency to appear linear.

3.1.4 Lineweaver-Burk Plots

Enzymes are biopolymers which catalyze a biochemical reaction. In the simplest case, an enzyme E binds a substrate S and forms the complex ES. The product P is formed in the enzyme. Both the product formation and dissociation are first-order processes. They may be combined to one first-order catalytic rate constant k_{cat} so that reaction scheme (3.4) may be used.

$$S + E \underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} ES \xrightarrow{k_{cat}} E + P \tag{3.4}$$

When the dissociation rate constant k_{-1} is much larger than k_{cat}, only a fraction of ES dissociates to E + P. When substrate is in large excess over E, only a tiny fraction of S can bind and only a small part of this can be catalyzed to form P. In this case, the initial equilibrium of S + E and ES is maintained for quite a while until a significant amount of S has been used up by the catalysis. This initial part of such a reaction is called “steady-state equilibrium” [6]. It can be computed just like any other equilibrium [7], with an equilibrium dissociation constant $K_m = (k_{-1} + k_{cat})/k_1$. K_m is called the Michaelis constant and steady-state enzyme kinetics of reaction scheme (3.4) are referred to as “Michaelis-Menten kinetics” [5, 8–13].

The rate of product formation follows from first-order dissociation (2.6)

$$v = d[P]/dt = k_{cat} \cdot [ES] \tag{3.5}$$

The enzyme concentration E0 usually is more than a factor of hundred smaller than the substrate concentration so that in the beginning of the reaction the free ligand concentration does not differ significantly from the total ligand concentration ([L] = L0). This gives a steady-state equilibrium, and the concentration of bound substrate [ES] can be computed from (3.1). Substituting L with S0, R0 with E0, and K_D with K_m leads to

$$v_0 = d[P]/dt = k_{cat} \cdot S0 \cdot E0/(S0 + K_m) \tag{3.6}$$

The maximal velocity is

$$v_{max} = k_{cat} \cdot E0 \tag{3.7}$$

and (3.6) becomes

$$v_0 = d[P]/dt = v_{max} \cdot S_0/(S_0 + K_m) \tag{3.8}$$

Equation (3.8) is called Michaelis-Menten equation [5, 8–13]. It is basically the same as (3.1), but it calculates the initial velocity of product formation (or substrate depletion) rather than ligand binding. The Michaelis constant is equal to the equilibrium dissociation constant of substrate binding when $k_{cat} << k_{-1}$. Other cases of steady-state approximations will not be discussed in this chapter, because numerical methods allow correct calculations. The program `enz2.m` (Fig. 7.3) calculates reaction scheme (3.4) without any restrictions imposed on the selection of rate constants.

When the initial velocity is plotted versus the substrate concentration, the resulting curve likewise is very similar to that shown in Figure 3.1. It is an equilibrium-binding curve attributed to the steady-state equilibrium of enzyme kinetics. Sometimes this type of plot itself is referred to as "enzyme kinetics" or "Michaelis-Menten kinetics." Just as shown in (3.2) for direct binding curves, taking the inverse of (3.8) will give

$$1/v_0 = (K_m/v_{max}) \cdot 1/S_0 + 1/v_{max} \tag{3.9}$$

Therefore, when 1/v0 is plotted versus 1/S0, the result will be a straight line with a slope of K_m/v_{max} and an intersection $1/v_{max}$ at the y-axis. The double reciprocal plot for enzyme kinetics is called Lineweaver-Burk plot [14] (Fig. 3.6).

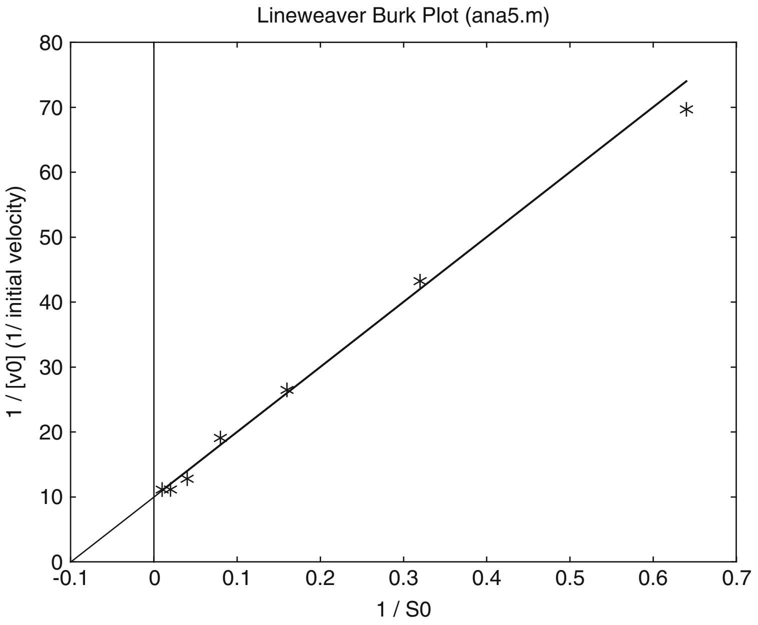


Fig. 3.6 Lineweaver-Burk plot. Double reciprocal plot of initial velocity and total substrate concentration. Initial velocity is calculated from (3.8) with $K_M = 10\ \mu M$, $E0 = 10\ nM$, $kcat = 10\ s^{-1}$. Intersection with the x-axis yields $-1/K_M$, the intersection with the y-axis $1/v_{max}$. The theoretical curve plotted is shown as a *solid line*. A random noise of 10% was assumed for the initial velocity. Data points were calculated from a twofold dilution series of the substrate concentration beginning with 100 μM

Figure 3.6 shows the calculation from substrate concentrations derived from a twofold dilution series. Section 8.2 explains why such a series ensures a minimal experimental error. Comparing Figs. 3.6 and 3.3 shows that such a dilution series is much more effective in covering a more significant data range. Only seven data points were required for the calculation shown in Fig. 3.6.

3.1.5 Dose–Response Curves and Hill Coefficients

When drug binding to a receptor [1] leads to a physiological response, the intensity of this response should follow the same curve as the binding curve (3.1). Such a dose–response curve usually is plotted in a logarithmic scale, where a hyperbolic function (Fig. 3.1) will look sigmoid ((+) in Fig. 3.7). In many cases, experimental dose–response curves do not follow (3.1). The maximal slope of the sigmoid dose–response curve often is larger than that expected from binding to one site. In this case, scheme (3.10), which had been used in 1910 for the binding of oxygen to hemoglobin [15], often is used as an approximation.

$$nL + R \rightleftharpoons L_nR \tag{3.10}$$

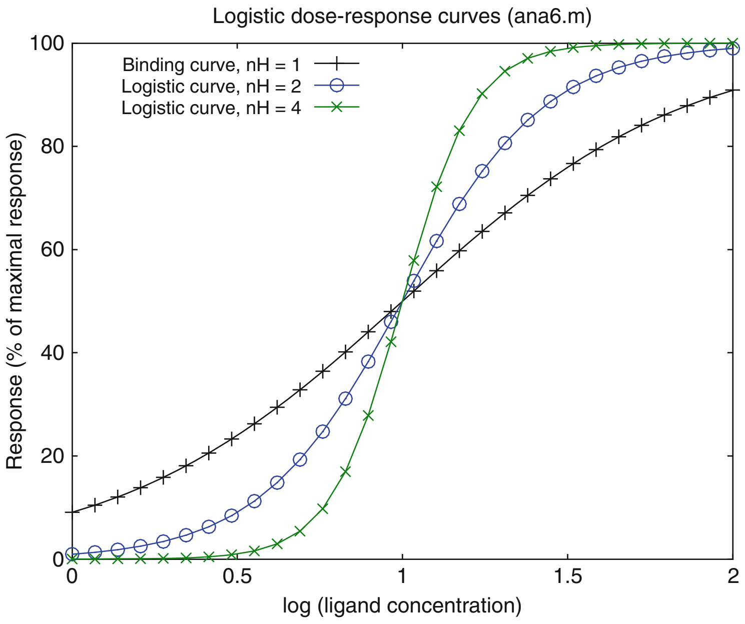


Fig. 3.7 Logistic dose–response curves. The binding curve (+) was calculated from (3.1), the logistic curves from (3.12) with $n = 2$ (o) and $n = 4$ (x). K_D or EC_{50} was 10 μM

Scheme (3.10) assumes simultaneous binding of n ligand molecules L to a receptor R, without intermediates LR, L_1R, L_2R,. . . and the like. Such simultaneous binding is unlikely in solution, but nevertheless (3.10) is commonly used for the calculation of dose–response curves. It would lead to a binding function (3.11)

$$[L_nR] = R0/(1 + (K_D/[L])^n) \tag{3.11}$$

K_D is the equilibrium dissociation constant for each step, n is called the Hill coefficient, and is usually denoted as n_H. Unlike n in (3.10) the Hill coefficient n in (3.11) need not be an integer. Mathematically (2.11) is a logistic function, which originally had been developed [16] for the description of population growth. Logistic functions are easy to calculate and easy to fit. They are often applied to dose–response curves as "4 parameter logistic equations" (3.12)

$$\text{Response} = (\text{Max} - \text{Min})/(1 + (EC_{50}/x)^n) \tag{3.12}$$

With x as the drug concentration and the four parameters, Max = maximal amplitude, Min = background, EC_{50} = ligand concentration for the half-maximal response and n = n_H = Hill coefficient, it has been pointed out that logistic functions do not correspond to any realistic binding scheme unless the Hill coefficient is one [17, 18].

When logistic functions (3.12) are calculated for different Hill coefficients and plotted versus the logarithm of the drug concentration, the result is a series of symmetric sigmoid curves (Fig. 3.7). Such curves are very useful for fitting, because the four parameters Max, Min, EC_{50} and n_H are usually not correlated (Sect. 8.1). Modification of n_H changes the shape of the curve. Its maximal slope thus can be varied independent of the inflection point (EC_{50}). Amplitude (Max) and background (Min) can also be varied independently.

From the logistic dose–response curves in Fig. 3.7, only the binding curve (+) with a Hill coefficient of 1 corresponds to a plausible reaction scheme (3.1). Experimental dose–response curves were found to be unsymmetrical when the maximal slope was steeper than for the binding curve (+) shown in Fig. 3.7 [18]. For these types of experimental curves, there is no generally accepted explanation [17, 18]. One new explanation for commonly observed steep dose–response curves has been published [19]; another one is calculated from multiple allosteric interactions (Fig. 5.12) and another one from irreversible reactions (Fig. 6.11).

3.2 Analytical Solutions for Binding Kinetics

The dissociation of ligand L from the receptor–ligand complex is a first-order reaction as described in (2.6). Rearranging this equation gives

$$d[LR]/[LR] = -k_{-1} \cdot dt \tag{3.13}$$

This can be integrated from the initial concentration $[LR]_0$ (the equilibrium concentration for $t = 0$ at the beginning of the dissociation reaction) to the concentration of [LR] at time t:

$$\int_{[LR]_0}^{[LR]} d[LR]/[LR] = -k_{-1} \cdot \int_0^t dt \tag{3.14}$$

The analytical solution is:

$$\ln([LR]) = \ln([LR]_0) - k_{-1} \cdot t \tag{3.15}$$

$$[LR] = [LR]_0 \cdot e^{-k_{-1} \cdot t} \tag{3.16}$$

Equation (3.15) states that the logarithm of [LR] should be a linear function of t. Dissociation can easily be measured when a large excess of a competing ligand is added to the complex LR. Since the forward reaction is blocked by the competitor, only the dissociation reaction with the rate constant k_{-1} should be observed. This type of experiment is extensively discussed and calculated in Sects. 6.4–6.6.

3.2.1 Exponential Decay

Equation (3.16) describes the exponential decrease of the complex [LR], and is shown in Fig. 3.8. Equation (3.16) is based on (2.6) and therefore can be applied whenever the decay rate of a substance is proportional to its concentration. Such a phenomenon is rather common in nature, with radioactive decay as the most prominent example.

The time which is required for the dissociation of half of the molecules is called "half life" and often is denoted with the Greek letter τ (tau). From (3.15) or (3.16) one can compute that τ is equal to ln2/k. It is independent of the initial concentration and therefore would be the same for the decay from 100% to 50% or from 50% to 25% and so forth. A dissociation reaction followed for more than five half-lives is shown in Fig. 3.7.

Measuring the half-life of a dissociation reaction is the simplest method for obtaining the rate constant of this reaction: $k = \ln 2/\tau$. It is also the simplest method to ensure that a reaction truly is of the first order: Subsequent half-lives must always give the same value, and one can begin to monitor a first-order reaction at any time.

3.2.2 Half-Logarithmic Plots

Half-logarithmic plots of simple exponential decays should be linear, as they follow (3.15). Experimental data often do not yield a linear half-logarithmic plot. In some cases the dissociation reaction is more complex. In other cases, fluctuation of the background signal may cause a deviation. Typically, one does not monitor absolute concentrations, but will follow the reaction by a signal (fluorescence, NMR, absorption,. . .) which is proportional to the concentration. Such a signal always will have a background, and in order to take the logarithm of the signal in Fig. 3.8, one has to take the logarithm of signal minus background. The background shown in Fig. 3.8 would be the signal at "infinite" time, which translates to the signal after at least ten half-lives. Fluctuations of the experimental points will lead to statistical fluctuations, but fluctuations of the background will lead to systematic deviations as shown in Fig. 3.9.

Drifts in the background can also be caused by typical experimental artifacts, such as bleaching, instability of a lamp, temperature increase and so forth. The half-logarithmic plot of ln [LR] vs. time (3.15) should be linear, but background variations may lead to distortions (Fig. 3.9). This may be one of the reasons why half-logarithmic plots have disappeared with the introduction of personal computers. Instead, one typically would fit (3.16) and add the background signal as an additional parameter. It should be stated, however, that the initial part of a

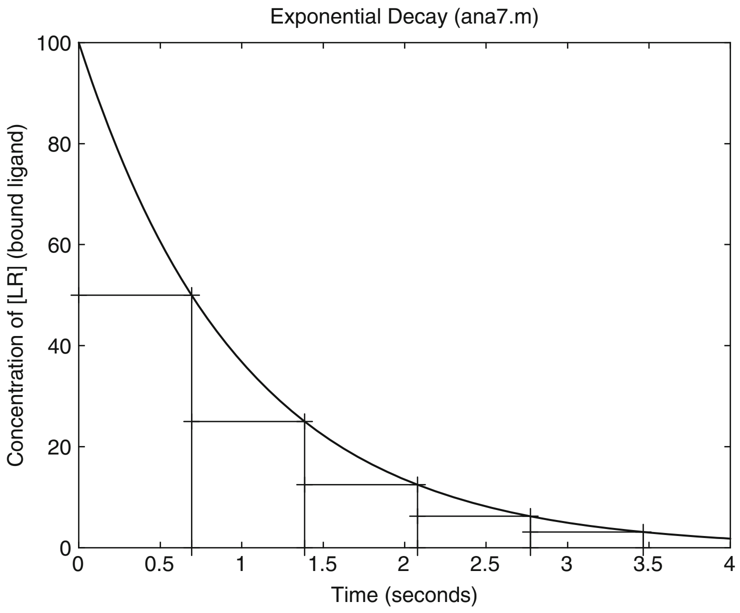


Fig. 3.8 Exponential Decay. Equation (3.16) is calculated with a rate constant $k_{-1} = 1\ s^{-1}$, initial concentration $[LR]_0 = 100$, $\tau = \ln 2$ s

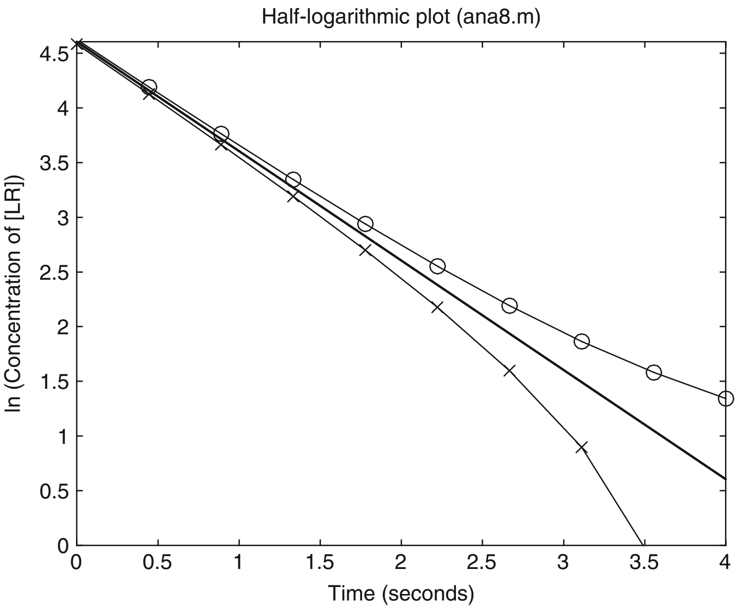


Fig. 3.9 Half-logarithmic plot of the reaction in Fig. 3.8. Equation (3.16) with $k_{-1} = 1\ s^{-1}$, $[LR]_0 = 100$ is calculated. (−) Logarithm of [LR]. (o) 2% background was added before the logarithm was taken, (x) −2% background

complex reaction often is identical to one simple exponential curve and that subsequent deviations may escape notice when background is used as a parameter for curve fitting.

3.2.3 Initial Velocity of Association Kinetics

A reversible reaction may involve conformational changes and may be quite complex, but the first part of an association reaction where a ligand L is added to free receptor R (only one binding site) is simple at the time point $t = 0$. When both reaction partners meet for the first time, their initial free concentrations are equal to the total concentrations:

$$v_0 = d[LR]/dt|_{t=0} = k_1 \cdot L0 \cdot R0 \tag{3.17}$$

Experimentally, the initial velocity is easy to measure, since it is the tangent to a kinetic experiment at zero time. At zero time, there is no bound ligand formed so that all back reactions can be ignored. When the concentration change of LR can be quantified and when the initial concentrations L0 and R0 are known, the rate constant k_1 of the initial part of the association reaction can be determined without curve fitting.

3.2.4 Pseudo First-Order Kinetics

The rate of a second-order reaction (2.1) is proportional to the concentration of both reactants (2.2). If one of these concentrations is kept constant, the rate is only proportional to the other. This results in a "pseudo" first-order reaction. If L0 is more than a factor of ten larger than R0, the free ligand concentration L0 of a reversible reaction (2.9) does not change significantly upon binding and therefore can be regarded as constant so that the differential equation for [LR] can be simplified:

$$\begin{aligned} d[LR]/dt &= k_1 \cdot L0 \cdot [R] - k_{-1} \cdot [LR] \\ d[LR]/dt &= k_1 \cdot L0 \cdot (R0 - [LR]) - k_{-1} \cdot [LR] \\ d[LR]/dt &= -(k_1 \cdot L0 + k_{-1}) \cdot [LR] + k_1 \cdot L0 \cdot R0 \end{aligned} \tag{3.18}$$

This is, apart from the constant $k_1 \cdot L0 \cdot R0$, the same as (3.13). Integration leads to

$$\ln([LR]) = \text{const} - (k_1 \cdot L0 + k_{-1}) \cdot t \tag{3.19}$$

This is the same as an exponential decay (3.15) and (3.16) with an observed rate constant k

$$k = k_1 \cdot L0 + k_{-1} \tag{3.20}$$

If one measures binding kinetics at different concentrations of L0 (the ligand in excess), one can obtain a series of pseudo first-order rate constants k. If one plots these constants versus the concentrations of excess ligand, the result is a straight line with a slope of k_1 and an intersection with the y-axis of k_{-1}. It should be noted that (3.20) is independent of the direction of the reaction. If the reaction is started from a complex LR, and if this complex is diluted by a large factor, then the observed dissociation also would follow first-order kinetics with the pseudo first-order rate constant k from (3.19).

3.2.5 Multiple Exponential Fits

In many cases, binding kinetics is complex and cannot be fitted to simple exponentials. Sometimes the following sum is used to fit the experimental data:

$$[LR] = A1 \cdot e^{-k1 \cdot t} + A2 \cdot e^{-k2 \cdot t} + A3 \cdot e^{-k3 \cdot t} + \ldots \tag{3.21}$$

This is a sum of exponential functions with amplitudes A1, A2, A3,... and rate constants k1, k2, k3,.... Almost all curves can be fitted with such a sum of exponential curves. The resulting amplitudes and rate constants are not very meaningful since generally there is no plausible reaction scheme which would lead to (3.21). Sometimes (3.21) is used simply as a way to describe the data. This is correct, as long as the whole ensemble of parameters (rate constants and amplitudes) is reported. One should note that the parameters of (3.21) generally are correlated so that individual rate constants cannot be extracted from a fit with (3.21), unless they are significantly more than one order of magnitude apart. Correlation is discussed in Sect. 8.1. Note that k1, k2, k3,. . . of a multi exponential fit in a kinetic experiment are real numbers, and that (3.21) does not describe an exponential Fourier series.

If an association reaction under pseudo first-order conditions does not give a single exponential, then the reaction is more complex. The kinetics of such a model can only be calculated from sets of differential equations and require numerical methods.

References

1. Langley JN (1905) On the reaction of cells and of nerve-endings to certain poisons, chiefly as regards the reaction of striated muscle to nicotine and to curare. J Physiol 33:374–413
2. http://de.wikipedia.org/wiki/Scatchard-Diagramm
3. http://en.dogeno.us/2004/03/scatchard-plot/
4. Voet DJ, Voet JG (1995) Biochemistry. John Wiley & Sons, New York
5. Gutfreund H (1995) Kinetics for the life sciences. Receptors, transmitters and catalysts. Cambridge University press, Cambridge
6. Segel LA, Slemrod M (1989) The quasi-steady-state assumption: a case study in perturbation. SIAM Rev 31:446–477. doi:10.1137/1031091
7. Michaelis L, Menten ML (1913) Die Kinetik der Invertin-Wirkung. Biochem Z 49:333–369
8. Segel IH (1993) Enzyme kinetics: behavior and analysis of rapid equilibrium and steady-state enzyme systems. Wiley Classical Library, New York
9. Copeland RA (2005) Evaluation of enzyme inhibitors in drug discovery: a guide for medicinal chemists and pharmacologists. Wiley-VCH, Weinheim
10. Copeland RA (2000) Enzymes: a practical introduction to structure, mechanism, and data analysis. Wiley, New York
11. Purich DL (2010) Enzyme kinetics: catalysis & control: a reference of theory and best-practice methods. Elsevier, London
12. Cook PF, Cleland WW (2007) Enzyme kinetics and mechanism. Garland Science, New York
13. Leskovac V (2003) Comprehensive enzyme kinetics, Kindle Edition. Amazon
14. Lineweaver H, Burk D (1934) The determination of enzyme dissociation constants. J Am Chem Soc 56:658–666
15. Hill AV (1910) The possible effects of the aggregation of the molecules of haemoglobin on its dissociation curves. J Physiol (Lond) 40:iv–vii
16. Verhulst PF (1845) Recherches mathématiques sur la loi d'accroissement de la population. Nouv mém de l'Academie Royale des Sci et Belles-Lettres de Bruxelles 18:1–41
17. Weiss JN (1997) The Hill equation revisited: uses and misuses. FASEB J 11:835–841
18. Prinz H (2010) Hill coefficients, dose-response curves and allosteric mechanisms. J Chem Biol 3:37–44
19. Prinz H, Schönichen A (2008) Transient binding patches: a plausible concept for drug binding. J Chem Biol 1:95–104

Chapter 4
Getting Started with Octave

GNU Octave is a free and open high-level language adequate for the calculation of reaction schemes. Its installation is described in detail and the basic structure is illustrated with the help of a very elementary tutorial. GNU Octave corresponds strongly to MATLAB®. Both are matrix-oriented computer languages. The sample programs often use matrices such as spreadsheets to calculate arrays of concentrations. At the end of this chapter, the reader should have installed GNU Octave and should be familiar with some basic features and the writing of simple programs.

4.1 Installation Instructions

GNU Octave is available in the Internet. It is freely redistributable under the terms of the GNU General Public License (GPL) [1] and ready to download at http://www.gnu.org/software/octave/. For windows, use the windows installer from http://octave.sourceforge.net/. This page from Octave Forge also provides assorted sets of Octave functions, called "Packages". We will need the package "Miscellaneous", the package "io", and the package "Optim". For windows version $\geq$ 3.2.3, the packages are selected as "Choose Components" when setup of the windows installer is executed. For other operating systems, the packages have to be downloaded separately and linked with the Octave command `pkg install`. Detailed procedures for Mac and Linux operating systems are described in the appendix. GNU Octave is continuously being developed, so that the actual installation routines may change.

The same is true for MATLAB. This language is also developed continually. Installation procedures may change, but a license has to be obtained before the program is installed. Trial licenses may be available. Note that the MATLAB Optimization Toolbox is required to solve nonlinear equations and to fit the data. A statistical analysis of the fitted parameters with covariance and correlation matrices requires functions such as `nlinfit` from the Statistics Toolbox. One needs an additional license for each toolbox.

Octave and MATLAB use very similar code. MATLAB or Octave or both often is taught in university courses for mathematicians, physicists, or engineers. Both languages are almost identical, so that someone who had learnt MATLAB will be able to write code in GNU Octave, and vice versa. MathWorks Inc. offers excellent seminars for MATLAB. The MATLAB programming environment is excellent, and its debugging features are better than Octave's. For larger programs, one may develop the programs in MATLAB and distribute them to the scientific community in Octave.

For Octave, there are numerous tutorials and help pages available in the internet. They are useful, but they should not be necessary for the understanding of the sample programs presented in this book. These programs are introduced step by step and all new commands are explained wherever they appear for the first time.

Notepad++ is a free windows editor available at http://notepad-plus-plus.org/. It is useful for writing Octave code, but other editors may be equivalent or even better. Useful editors should display line numbers, because error messages in Octave usually refer to line numbers. Notepad++ can be configured to MATLAB/Octave with the pull down tab Language/M/Matlab. Notepad++ then recognizes strings, comments, instruction words, etc., and displays them in different styles. These styles can be edited in Settings/Style Configurator/Matlab.

4.2 Typing the First Commands

GNU Octave is a modern high-level computer language, but looks "retro" for most life scientists. There are (almost) no pull down menus, no buttons to click, and only hidden help functions. The Spartan "terminal window" (Fig. 4.1), which appears when Octave is executed in a Microsoft environment, may come as a shock, but one can get used to it. There will be "cryptic" messages, a prompt (the ">" character), and a blinking cursor. Nothing else!

Do not get intimidated. Just try it out and type

```
A = 12*3
```

You will get the answer `A = 36` and again a prompt and a blinking cursor.

If you then type

```
A
```

Again you get the answer

```
A = 36
```

However, if you type

```
a
```

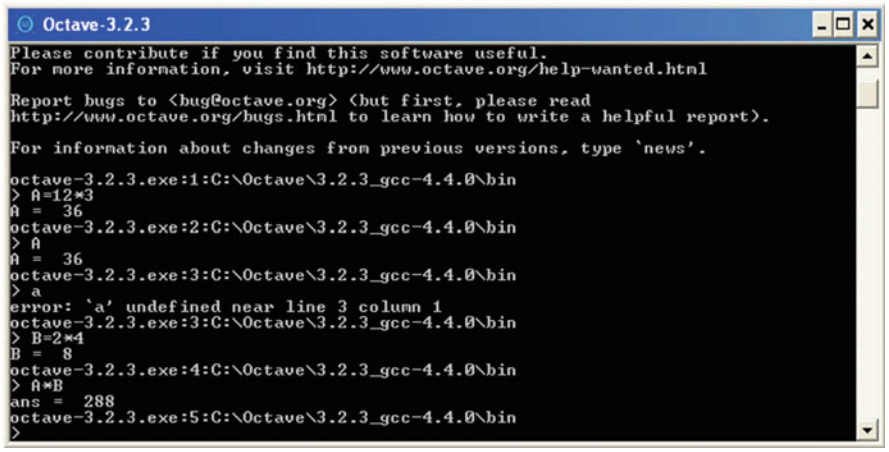
you get an error message. Octave distinguishes between uppercase and lowercase letters and does not "know" the variable `a`. If you type

```
B = 2*4;
```

You get no answer. The semicolon is used to denote the end of a command and prevents the direct output. If you then type `A*B`, (or `c = A*B`), you will get

```
ans = 288 (or c = 288)
```

which is the product 36*8.



```
Please contribute if you find this software useful.
For more information, visit http://www.octave.org/help-wanted.html

Report bugs to <bug@octave.org> (but first, please read
http://www.octave.org/bugs.html to learn how to write a helpful report).

For information about changes from previous versions, type `news'.

octave-3.2.3.exe:1:C:\Octave\3.2.3_gcc-4.4.0\bin
> A=12*3
A =  36
octave-3.2.3.exe:2:C:\Octave\3.2.3_gcc-4.4.0\bin
> A
A =  36
octave-3.2.3.exe:3:C:\Octave\3.2.3_gcc-4.4.0\bin
> a
error: `a' undefined near line 3 column 1
octave-3.2.3.exe:3:C:\Octave\3.2.3_gcc-4.4.0\bin
> B=2*4
B =  8
octave-3.2.3.exe:4:C:\Octave\3.2.3_gcc-4.4.0\bin
> A*B
ans =  288
octave-3.2.3.exe:5:C:\Octave\3.2.3_gcc-4.4.0\bin
>
```

Fig. 4.1 The Octave terminal window (Screenshot)

It should be emphasized that one cannot learn a computer language from a book without applying the language. It is mandatory to run Octave at this stage and one should "play" with it using different commands. Of course, all basic operations (+, −, *, /, ^) are the same as given on the numeric keyboard or on any calculator. Play with it now!

The arrows of the keyboard help to navigate in the Octave terminal window:

↑	Previous command
↓	Next command
←	Position the curser one character to the left
→	Position the curser one character to the right

Edit functions become available with a right mouse-click. This is the only pull down menu available. Copy consists of "Mark" followed by highlighting the characters to be copied, followed by "enter", the return key. The standard windows edit commands such as Ctrl + C or Ctrl + V do not work in the Octave terminal window. Ctrl + C is reserved as a key to terminate any program. Therefore, one has to use the right mouse-click for the edit functions.

Typing **`help`** followed by the name of a command will show help information for the specified command. Usually, this help information is bigger than the terminal window, and an information line is printed in the lower left corner of the terminal window, giving the numbers of the displayed lines and the commands to navigate: (f)orward, (b)ack, and (q)uit

Typing **`doc`** followed by an expression gives access to the GNU Octave Manual. In this case, navigation is done with the keyboard. Ctrl + C will terminate the doc session and return to the terminal window.

4.3 Writing a Program

Open your text editor and type these lines:

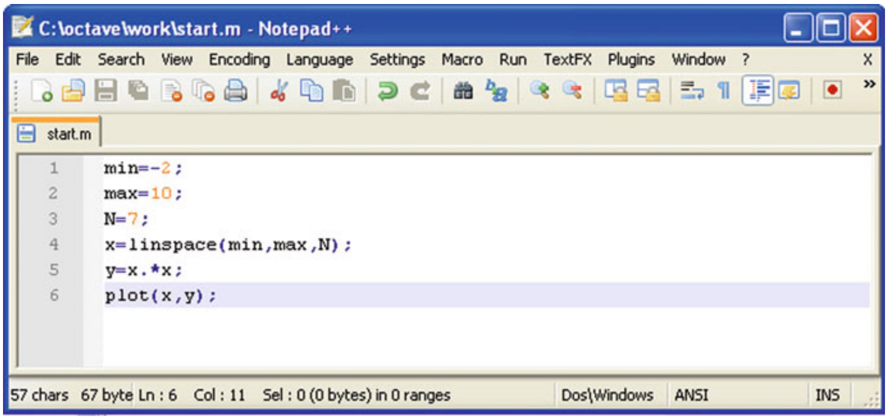
```
1  min=-2;
2  max=10;
3  N=7;
4  x=linspace(min,max,N);
5  y=x.*x;
6  plot(x,y);
```

The line numbers are supplied by the editor, as shown in Fig. 4.2. They must not be entered separately. Line numbers are useful for discussing and debugging the program. If they are not shown in your editor, try the help function. All editors should be able to display line numbers.

Do not forget that Octave is case sensitive. Once you have written the program, you should give it a name like "`start.m`" and save it with that name. All programs and functions in octave must have the ending "`.m`" This ending is derived from **m**ATLAB. Save the program into a working directory, such as C:/work or C:/octave/work.

Then start the program GNU Octave. Once the terminal window appears, enter it and type `cd C:/work` (or the name of your favorite working directory). The command `cd` **c**hanges the **d**irectory to your folder. Once the prompt and the blinking cursor appears, type the name of the program which you have just written (without the ending `.m`), such as "`start`" in your terminal window. You will see a new window and the plot shown in Fig. 4.3.

Now let us discuss the program `start.m` line by line. The first line assigns the number -2 to the variable `min`. The equals sign (=) in Octave and in any other computer language is not symmetric. It takes the value of the right side and assigns it to the variable on the left side. For example, if `a` = `1` and `b` = `2`, then the


C:\octave\work\start.m - Notepad++
File Edit Search View Encoding Language Settings Macro Run TextFX Plugins Window ?
start.m

```
1  min=-2;
2  max=10;
3  N=7;
4  x=linspace(min,max,N);
5  y=x.*x;
6  plot(x,y);
```

57 chars 67 byte Ln : 6 Col : 11 Sel : 0 (0 bytes) in 0 ranges Dos\Windows ANSI INS

Fig. 4.2 The Notepad ++ editor window (Screenshot)

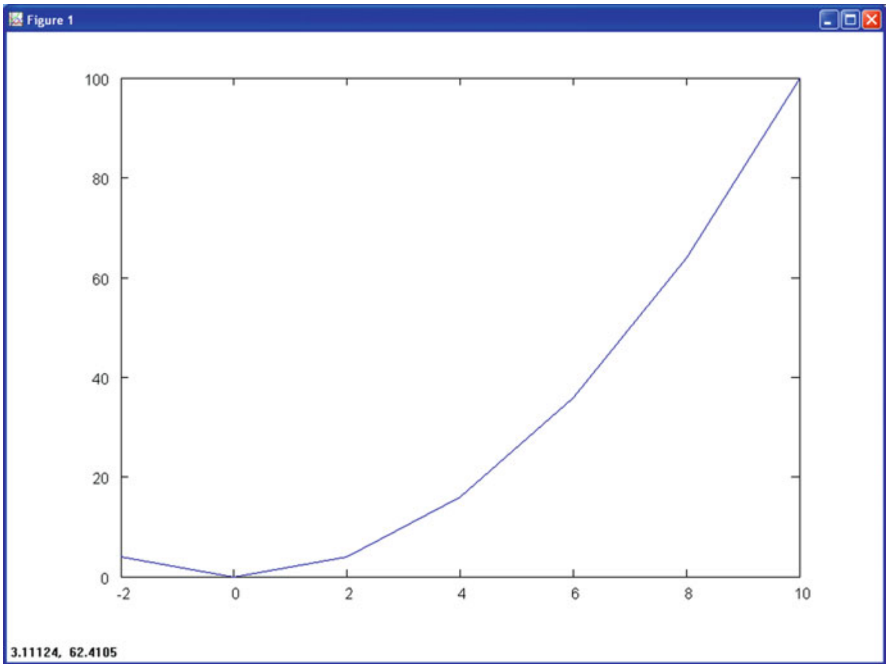


Fig. 4.3 A plot generated by `start.m`. (Screenshot) The command plot(x,y) – in its most basic variety – shows the data as dots connected with straight lines. The numbers in the lower right corner correspond to the x and y coordinates of the cursor position (not visible in the screenshot)

command `a = b` will result in `a = b = 2`, whereas `b = a` will result in `a = b = 1`.

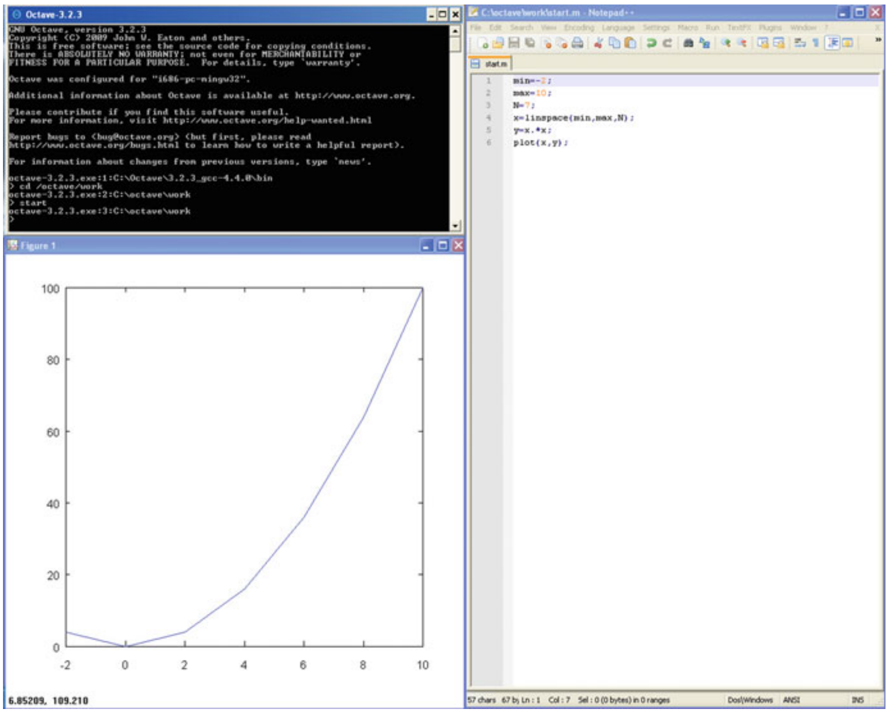
Note that all lines end with a semicolon. If you omit the semicolon, the variable is also sent to the terminal window. Try it out! The first three lines of `start.m` therefore simply assign numbers to the variables `min`, `max`, and `N`. It is always a good idea to define all variables and assign them to numbers in the beginning of the program, since they are easy to find there. `linspace` in line 4 is an interesting function: It creates the array `x` of `N` points, equally spaced linearly between `min` and `max`. You can see the N = seven components if you type "`x`" in the terminal window. In fact, you can see the assigned value of any variable if you type in the variable name. Try it out with `min` or `N`. Line 5 uses the operator `.*` (the character dot followed by an asterisk for the multiplication sign). As explained in Sect. 4.6, this denotes element-by-element multiplication of vectors (arrays are vectors in Octave). When the seven elements of `y` ($=x^2$) are plotted versus the seven elements of `x`, using the `plot` command in line 6, the result is a hyperbola. It will appear as "Figure 1" in a separate window.

The Octave plot function internally uses gnuplot, another software freely distributable under the GPL. Information is available at www.gnuplot.info. The sample programs shown below will increase in their complexity and will use more and more features of gnuplot. Titles, axes, legends, and texts will be used, so that all information will be stuffed into the plot outputs. Independent help pages, textbooks, and tutorials are available for gnuplot. Just as for Octave, external help is useful, but should not be required for the understanding of the sample programs here.

4.4 Setting Up a Work Environment for Octave: Arrangement of Editor, Terminal, and Plot Window

Unlike MATLAB, Octave does not come with a cohesive work environment containing all windows of interest. It is a good idea to arrange the relevant windows, so that one can edit, run, and see the results of a program in one glance. Figure 4.4 shows such an arrangement.

Once the program `start.m` had been executed by typing `start` in the terminal window, it can be run again with ↑, the up arrow followed by the return key. Try it out. If the program `start.m` is modified (replace, for example, the number `2` in line 1 with `10`), it has to be saved again before it can be executed. Note that the icon for the command "save" in Notepad++ turns red when a file has been modified, but not saved.


Octave-3.2.3
Figure 1
100
80
60
40
20
0
-2
0
2
4
6
8
10
6.85209, 109.210
C:\octave\work\start.m - Notepad++
File Edit Search View Encoding Language Settings Macro Run TextFX Plugins Window ?
start.m
57 chars 67 bytes Ln : 1 Col : 7 Sel : 0 (0 bytes) in 0 ranges Dos\Windows ANSI INS

```
GNU Octave, version 3.2.3
Copyright (C) 2009 John W. Eaton and others.
This is free software; see the source code for copying conditions.
There is ABSOLUTELY NO WARRANTY; not even for MERCHANTABILITY or
FITNESS FOR A PARTICULAR PURPOSE.  For details, type `warranty'.

Octave was configured for "i686-pc-mingw32".

Additional information about Octave is available at http://www.octave.org.

Please contribute if you find this software useful.
For more information, visit http://www.octave.org/help-wanted.html

Report bugs to <bug@octave.org> (but first, please read
http://www.octave.org/bugs.html to learn how to write a helpful report).

For information about changes from previous versions, type `news'.

octave-3.2.3.exe:1:C:\Octave\3.2.3_gcc-4.4.0\bin
> cd /octave/work
octave-3.2.3.exe:2:C:\octave\work
> start
octave-3.2.3.exe:3:C:\octave\work
>
```

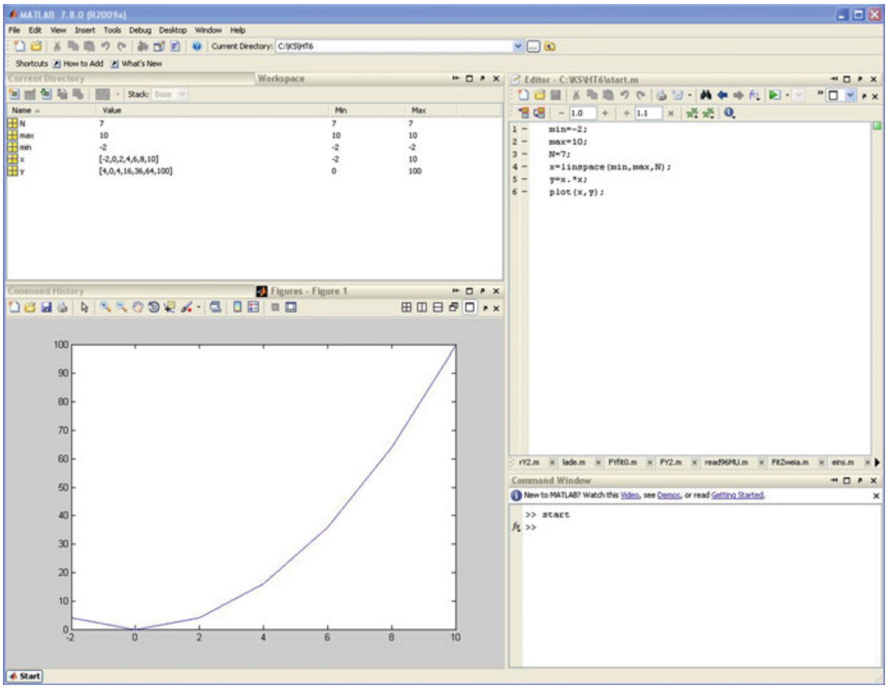
```
1  min=-2;
2  max=10;
3  N=7;
4  x=linspace(min,max,N);
5  y=x.*x;
6  plot(x,y);
```

Fig. 4.4 Arrangement of terminal, editor, and plot windows (Screenshot)

It is possible to copy single commands, or lines, or sections of a program from the text editor to the terminal window, with Ctrl + C, but remember the unique environment of the terminal window (right mouse-click and select "Paste" for the paste command).

4.5 The MATLAB Environment

The MATLAB environment (Fig. 4.5) looks slightly different. There are pull down menus, there are good help functions, and there are good debugging features. Still, the setup is practically the same along with the language. The "Terminal Window" from GNU Octave is called "Command Window" in MATLAB. It can be found in the lower right corner, and it uses standard edit commands (Ctrl + C, etc.). In MATLAB, all windows can be customized. A separate "Workspace" window (upper left window) is useful, because the values of all variables are listed there. There are many other user-friendly features of MATLAB, such as a separate help window, but they are not covered here. MATLAB comes with extensive documentation, but the basic features are almost [2] the same: One still has to write a program in MATLAB/Octave code, and one has to write it line by line.



Name	Value	Min	Max
N	7	7	7
max	10	10	10
min	-2	-2	-2
x	[-2,0,2,4,6,8,10]	-2	10
y	[4,0,4,16,36,64,100]	0	100

```
min=-2;
max=10;
N=7;
x=linspace(min,max,N);
y=x.*x;
plot(x,y);
```

New to MATLAB? Watch this Video, see Demos, or read Getting Started.

```
>> start
>>
```

Fig. 4.5 One arrangement of MATLAB windows (Screenshot)

4.6 Arrays, Vectors, and Matrices

The name MATLAB is derived from MATrix and LABoratory. It is a matrix-oriented language, and Octave has the same properties. Any variable initially is regarded as a matrix, and a matrix with one column is a column vector, a matrix with one row is a row vector, and a matrix with one element is a scalar. This may sound complicated, but it really is simple once you understand the concept. Matrices are written in square brackets.

Arrays of concentrations, such as the substrate concentrations of an experiment, may be regarded as vectors. If one uses seven concentrations like 1, 2, 4, 10, 20, 40, and 100 μM for an experiment, these concentrations would be entered as a statement such as `C0 = [1, 2, 4, 10, 20, 40, 100]` in Octave. Square brackets define a matrix (in our case, a matrix with one row, which is a row vector). When the elements are separated with a comma, it is a row vector, and when the elements are separated with a semicolon, it is a column vector. The third concentration of the array C0 is denoted as `C0(3)` in Octave. If you type this in the terminal window, you will get the answer `ans = 4`.

Let us consider two vectors, a row vector `a = [1,2,3]` and a column vector `b = [4;5;6]` Octave is well suited for vector algebra, but the sample programs listed in this book will use vectors mostly as arrays of concentrations. For these, element-by-element multiplication is a useful operation, written as `.*` (dot followed by an asterisk). The operation requires that the vectors contain the same numbers of rows and columns. If you typed `a.*b` with the examples above, you would get the error message "nonconformant arguments", since **a** is a row vector, whereas **b** is a column vector. The transpose operator is denoted as the character′ (the single quote or prime symbol). It transposes a row vector into a column vector and vice versa. Therefore, the product `a.*b′` will give the row vector `[a1*b1, a2*b2, a3*b3]` for the above example. Try it out.

Note that the usual matrix multiplication is denoted by the operator "*" (asterisk). In the above example, `a*b` is the matrix multiplication of the row vector `a` times the column vector `b`. The resulting scalar product is the number `a1*b1+a2*b2+a3*b3`, which in our example is 32.

A matrix may be written as a column vector of row vectors or a row vector of column vectors. `M = [a; a; a]`, for example, would be a 3X3 matrix where all three rows contain the same elements a1, a2, and a3. The elements of a matrix can be addressed directly as `M(Nrow,Ncolumn)` with `Nrow` and `Ncolumn` denoting the index number for the row and the column, respectively. Vectors can be regarded as matrices with one row or one column, respectively. For the row vector **a**, the element `a2` may be addressed as `a2 = a(2) = a(1,2)`, and for the column vector **b**, the third element might be addressed as `b3 = b(3) = b(3,1)`.

At this stage, it is best to try out your own examples directly in the octave window. If you have run the program `start.m` before, you may want to type y' and see the result. Or type `M = [x; 2*x; 3*x]` Or type `5*3` and get `ans = 15`. `ans` is the variable name (short for "answer") for the result of the last octave calculation where you did not supply a variable name. If you then type `ans* 3` afterward, you will get `ans = 45`. Just try it out and play with it!

4.7 A Very Elementary Tutorial for GNU Octave and MATLAB

It is easier to learn languages from practical exercises than from books alone. The same is true for computer languages. One needs practical exercises. While we do not want to limit the creativity of the reader, the commands listed below can be used as a very elementary tutorial. If the commands (set in `courier` font) on the left are typed directly line by line, either in the Octave terminal window or in the MATLAB command window, the results will appear immediately.

`cd`	**c**hange **d**irectory like `cd C:/octave/work`
`b=5`	assign the value of 5 to the variable b (`5=b` gives an error)
`a=b`	assign the value of b to the variable a
`a=[1,2;3,4]`	Square brackets denote a matrix or vector; comma for row, semicolon for column. `a` is a 2x2 matrix.
`a(1,2)`	will give `ans=2`
`a(2,1)`	will give `ans=3`
`d=a(:,1)`	Column vector of the first column of the matrix `a`. It will contain the elements `1` and `3`. The colon denotes the full range.
`e=a(2,:)`	row vector of the second row of the matrix `a`. It will contain the elements `3` and `4`.
`f=e*d`	Scalar product of row vector e and column vector d. The number `f` is `15`.
`g=5*d`	multiply a vector d with a number. g is a column vector with the elements 5 and 15.
`c=g.*d`	Element by element multiplication (same dimension of matrices or vectors). Note that g and d both are column vectors of two elements. The resulting vector c is a column vector with the elements 5 and 45.
`i=c'`	transpose of a vector (row to column or column to row). i is a row vector with the elements 5 and 45.
`h='Hallo'`	String of characters (enclosed with apostrophes). Strings of characters will be required for output.
`w='World'`	A second string of characters
`t=[h,' ',w]`	row vector, three strings = 11 characters. `t = Hallo World`
`x=linspace(0,10,11)`	row vector of 11 elements linear from 0 to 10, the result is `x = 0,1,2,3,4,5,6,7,8,9,10`
`y=x.*x`	vector with elements $y_i = x_i^2$. The result is the row vector `y = 0,1,4,9,16,25,36,49,64,81,100`
`plot(x,y)`	plot of y versus x
`text(1,50,t)`	write text inside a plot(`x`, `y` position, depending on the scale, `t` = character string)
`title(h)`	assign a title h to the plot
`xlabel(w)`	assign a label w to the x-axis.

The result of the typing should be a plot giving a parabola, a text within the plot, Hallo as a title, and a label "World" for the *x*-axis. If all these commands are understood, this tutorial has been finished successfully.

Some commands which may also be useful:

`;`	Semicolon marks the end of a command and prevents output to the terminal window.
`Ctrl-C`	interrupts any program
`mkdir`	makes a new directory like `mkdir C:/octave/work`
`ls`	lists all files within the current directory
`path`	shows the path, where Octave looks for functions
`addpath`	adds a directory to the path (for storing and retrieving your own functions)
`exit`	exit GNU Octave and close all Octave windows

4.8 Recommended Literature for Octave/MATLAB

This will conclude a first elementary introduction to octave. There is extensive literature and helpful information to be found abundantly in the Internet. Both GNU Octave and MATLAB are living languages, which present their current developments in the Internet, so that written textbooks may seem to be obsolete. There are noteworthy exceptions: Cleve Moler has written an introductory course on numerical methods [3], which gives the necessary background for numerical methods applied here. Individual chapters can be downloaded from Mathworks.com [4]. MATLAB comes with extensive documentation and help functions. For GNU octave, its manual [5] is a useful piece of hardware. It gives an overview and can be used as an excellent reference for each command. It is a useful investment, although its contents are shown directly in the Octave terminal window with the command `doc`. There is a very active GNU Octave community, so that there are more than 326,000 Google hits for "Octave Tutorial". Some of these seem to be particularly useful [6–8], and most tutorials cover MATLAB as well. MATLAB is at least as popular as Octave, with 571,000 Google hits for "MATLAB Tutorial". Tutorials may be available in any spoken language worldwide.

Octave and MATLAB have a broad application spectrum, mainly focused on science and engineering. Only a few of these features are required for the calculation of reaction schemes. They are introduced one by one as they come along to solve practical problems.

References

1. http://www.gnu.org/software/octave/license.html
2. http://en.wikibooks.org/wiki/MATLAB_Programming/Differences_between_Octave_and_MATLAB

3. Moler C (2010) Numerical computing with MATLAB. Society for Industrial Mathematics, Philadelphia, PA
4. http://www.mathworks.com/moler/chapters.html
5. Eaton JW, Bateman D, Hauberg S (2008) GNU octave manual version 3. Network Theory Ltd, London
6. http://smilodon.berkeley.edu/octavetut.pdf
7. http://en.wikibooks.org/wiki/Octave_Programming_Tutorial
8. http://srl.informatik.uni-freiburg.de/downloadsdir/Octave-Matlab-Tutorial.pdf

Chapter 5
Equilibrium Binding

Binding equilibria of n compounds can be calculated from n equations of n unknowns. These are solved numerically with the Octave/MATLAB subroutine `fsolve`. Writing program code for this task is introduced step by step. The output of all programs is given in a graphic format so that binding mechanisms of increasing complexity can be visualized. Allosteric interactions of subunits and multiple allosteric interactions at one target molecule are calculated as practical examples. At the end of this chapter, the reader should be able to calculate equilibrium binding for any reaction scheme involving any number of ligands, inhibitors and binding sites.

5.1 Solving Nonlinear Equations for Equilibrium Binding

Sections 2.2 and 2.3 describe the sets of nonlinear equations derived from equilibrium-binding schemes. In Octave and MATLAB, these equations are solved with the function `fsolve`, which in turn is based on the MINIPACK subroutine hybrid [1]. MATLAB users must obtain the "Optimization toolbox," which includes `fsolve`.

The set of equations to be solved has to be written as an array of n equations with n unknowns. In Octave/MATLAB, such an array is treated as a vector, and the set of equations can be written as a vector function

$$\mathbf{F}(\mathbf{x}) = 0 \tag{5.1}$$

F and **x** are vectors of the same size. The function `fsolve` requires an initial estimate for the unknowns, a vector **xo**. It then calculates the set of equations as F (**x0**). The result will not be zero, but `fsolve` varies the unknowns **x** in several steps until (5.1) is solved within the required mathematical precision.

This general strategy has to be illustrated with an example; reversible binding to one site leads to (2.16) and (2.17) for equilibrium binding. These equations can be re-written as elements of the function (5.1):

DOI 10.1007/978-3-642-20820-1_5, © Springer-Verlag Berlin Heidelberg 2011

$$\mathrm{F(1) = x(1) + x(1) \cdot x(2)/KD1 - R0 = 0} \tag{5.2}$$

$$\mathrm{F(2) = x(2) + x(1) \cdot x(2)/KD1 - L0 = 0} \tag{5.3}$$

Remember, x(i) denotes the ith element of the vector of unknowns **x**, with x(1) = [R] and x(2) = [L]. Likewise, the initial estimates will also be defined as one vector **x0**. When R0 and L0 are similar, the free concentrations are estimated to be half of the total concentrations:

$$\mathrm{x0(1) = 0.5 \cdot R0} \tag{5.4}$$

$$\mathrm{x0(2) = 0.5 \cdot L0} \tag{5.5}$$

Note: When the ligand concentration is much larger than the receptor concentration, equation (5.6) becomes a better estimate for the free ligand concentration:

$$\mathrm{L0 >> R0 : \quad x0(2) = L0 - 0.5 \cdot R0} \tag{5.6}$$

Having explained the basic concept of iterative techniques used in `fsolve`, this function can now be applied in a sample program. The function fsolve is called with the statement: `x = fsolve('name',x0)`, where `'name'` is the name of the function for the set of (5.2) and (5.3). `x0` and `x` are column vectors of the same length. Remember, Octave and MATLAB are matrix-oriented languages, and all variables may be matrices, vectors or scalars.

5.2 Equilibrium Binding to One Site (**`EQ1.m`**)

The first sample program is `EQ1.m`, a program which calculates reversible binding to one site with analytical and numeric methods. The central function, called from `fsolve`, is called `EQ1F.m`. It is stored with this name in the octave work directory. It mainly contains (5.2) and (5.3) written in Octave code.

```
1      %% Equilibrium Binding to one site
2      function F = EQ1F(x)
3      global KD1 R0 L0;
4      F(1) = x(1) + x(1)*x(2)/KD1 -R0; % x(1)=[R]
5      F(2) = x(2) + x(1)*x(2)/KD1 -L0; % x(2)=[L]
```

The first line is merely a description of the function. It is written as a "comment." Comments are defined as lines beginning with the percent sign `%`. In MATLAB, two of these signs (line 1) define the beginning of a program section. The `%` sign may not only appear at the beginning, as illustrated in lines 4 and 5. In this case, the comment begins at the `%` sign and ends at the end of the line. The `function` statement (line 2) defines the name of the function (EQ1F). The function names in this textbook are derived from the names of the main program, supplemented by the letter F. The name of a **function** must be the same as the name of the **file** in which it is stored (EQ1F.m). The letter F in line 2 defines the **variable** name which is returned from the function. The components of the vector F are calculated in lines 4 and 5. They correspond to (5.2) and (5.3). The global statement in line 3 allows data exchange between the function and the main program.

The main program (EQ1.m) also begins with a short description given as comment lines.

```
1      %% Program description
2      % EQ1:
3      % Equilibrium Binding of ligand L to receptor R
```

The next section of EQ1.m is the definition of all relevant parameters. It is usually a good idea to define parameters at the beginning of a program. Computer code is read from top to bottom so that any variable has to be defined before it is used. When all variables are defined together in consecutive lines, it is easy to identify them for debugging or modifying the program.

```
6      %% Definition of parameters
7      clear;
8      filename='EQ1';
9      global KD1 R0 L0;
10     R0=1;
11     KD1=10;
12     MaxL0=100;
13     N=30;
14     VL0=linspace(0, MaxL0, N);
```

Clear in line 7 is a command which is useful at the beginning of any program, since all user-defined variables are cleared from the memory. Octave remembers all parameters, even after a program has successfully been terminated. For example, if you run EQ1.m by typing EQ1 in the octave terminal window, you will see the plot of Fig. 5.1 and the > character in the terminal window. When you then type R0 in the terminal window, you will get R0 = 1 as an answer. Liberating memory with the clear command helps to keep the slate clear.

The string variable filename defined in line 8 will be used later in the program, in lines 30, 36, 37, 38 and 40. A string variable is defined by the apostrophes in line 8. The statement global in line 9 corresponds to the same statement in line 3 of the function EQ1F and allows transfer of the specified parameters. Lines 10–13 simply assign numbers to variable names. linspace in line 14 returns the row vector VL0 with N linearly spaced elements between 0 and MaxLo. These 30 ligand concentrations will be used for plotting the data. Variable names beginning with V emphasize that the variable is a vector.

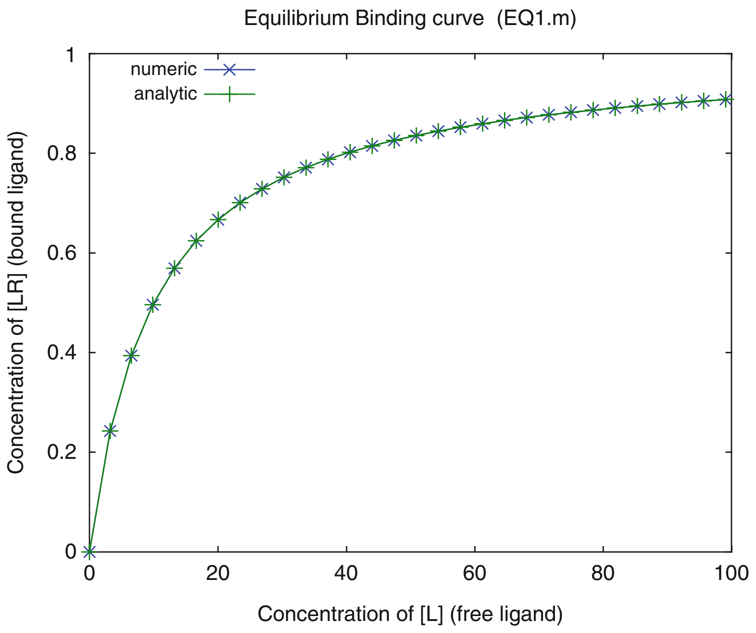


Fig. 5.1 Equilibrium-binding curve. Bound ligand versus free ligand is shown for 1 µM receptor and an equilibrium dissociation constant of 10 µM. The values were computed with the program `EQ1.m` as numeric approximations (x) and analytical solutions (+)

```
15    %% Beginning of the program
16    for i=1:N
17    L0=VL0(i);
18    x0(1)=0.5*R0;
19    x0(2)=0.5*L0;
20    x = fsolve('EQ1F',x0');
21    R=x(1);
22    L=x(2);
23    VLR(i)=L*R/KD1;
24    VL(i)=L;
25    end;
26    LR_analytical=(VL*R0)./(KD1+VL);
```

The main program basically contains a loop running from line `16` to line `25`. Loops are sets of commands which are executed over and over, until a condition is fulfilled. The simplest condition is the number of repeats, which is `N` in line `16`. The command `for i = 1:N` states that the index `i` begins at 1 and is increased by its default value of 1 until `i = N`. For each `i`, a different total ligand concentration `L0` (line `17`) is used from `VL0`, the vector of initial concentrations which had been created in line `14`. Lines `18` and `19` give the initial estimates and correspond to (5.4) and (5.5). Line `20` calls the function `fsolve`. It requires the name of the subroutine function 'EQ1F' where the equations are written. Note that the name'EQ1F' is a string of characters as defined by single-quote ('). fsolve also requires the vector x0 of initial estimates as an argument. The result in line 20 is the vector x. In lines 21 and 22, the elements of the vector x are translated into the chemical notation as free receptor R and free ligand L, respectively. Line 23 defines VLR as the vector of complexes LR, employing (2.15). VL (line 24) simply gives the vector of free ligand concentration L. The loop ends with an end statement. Octave allows different end statements like endfor, endif, endwhile and endfunction. Since MATLAB does not support this, we do not make use of this Octave feature.

The loop creates the vectors VLR and VL of complex and free ligand, respectively. Calculating the analytical solution in octave does not require a loop. Line 26 is octave code and looks identical to (3.1), but note that VL is a vector of concentrations L. In line 26 all concentrations of bound ligand LR for all free ligand concentrations L are calculated with one statement. The operator ./ (dot followed by forward slash) in line 26 denotes an element-by-element division. It corresponds to the element-by-element multiplication explained in Sect. 4.7.

Once all relevant concentrations are calculated, they can be plotted with the plot command given in line 28. The first argument in the plot function is the x-value, in our case the free concentration vector VL and the second is the corresponding y-value, the bound ligand vector VLR. The style (included in quotes) specifies line (−) and symbol (x). A second pair of x and y coordinates is supplied for the analytical solution (LR_analytical). This is specified by another symbol (+). Lines 29 and 30 are remarkable for two reasons. First, the continuation marker (three dots) at the end of a line indicates that the next line belongs to the same statement. In this case, lines 29 and 30 are pasted to a single title function. The second remarkable feature of the title function is the string which is given as an argument. Every string may be regarded as a row vector of characters. In Octave, any row vector is defined by its elements, separated by commas, in square brackets (Sect. 4.7). Therefore ['Equilibrium Binding curve (',filename,'.m)'] is a row vector of the string 'Equilibrium Binding curve (', followed by the parameter string filename (defined in line 8), followed by the string '.m)'.

```
27    %% Plot the data
28    plot(VL,VLR,'-x',VL,LR_analytical,'-+');
29    title(['Equilibrium Binding curve  (',...
30    filename,'.m)'])
31    axis([0, MaxL0, 0, R0])
32    xlabel('Concentration of [L] (free ligand)')
33    ylabel('Concentration of [LR] (bound ligand)')
34    legend('numeric','analytic',2);
```

The same procedure (row vector of strings) is used in the print and save commands in lines 36, 37, 38 and 40. Unfortunately, Octave version 3.2.4 gives a message implicit conversion from matrix to string whenever a print command (line 36 and 37) is executed. Such a warning is not given when the same program is run in MATLAB or older versions of Octave. Therefore, please ignore the message `implicit conversion from matrix to string` displayed in the Octave Terminal window of version 3.2.4.

The statement `axis` (line `31`) requires a row vector of min and max values for the x- and y-axes, respectively. The `legend` function (line `34`) inserts a legend into the plot, thus also relates to the `plot` function. The two strings correspond to the two data pairs in the plot command of line `28.` A third argument is a number (2, in this case), which defines the quadrant in which the legend will appear.

```
35    %% Output to files
36    print([filename,'_plot.jpg'],'-djpeg');
37    print([filename,'_plot.emf'],'-dmeta');
38    save([filename,'_all.txt']);
39    out=[VL0',VL',VLR',LR_analytical'];
40    save('-ascii',[filename,'_dat.txt'],'out');
```

The last lines of the program `EQ1.m` give examples for different output formats. The print command in lines `36` and `37` is used to save the graphic output in two graphic formats, namely .jpg (line `36`) and .emf (line `37`). The later format (Microsoft-enhanced metafile) was used for all figures in this textbook. All parameters can be saved with the save command in line `38`. The data might be retrieved at any time with a corresponding `load` command. Octave (not MATLAB) stores the resulting data file `EQ1_all.txt` in an ASCII format which can be accessed with any text editor. In both computer languages, the data can be stored in ASCII format as a spreadsheet. This is done with the option `-ascii` in line `40`, but one can save only a single matrix with this command. Therefore, the matrix `out` was generated in line 39 as a row of the column vectors of interest. Figure 5.1 is generated when `EQ1` is typed in the Octave terminal or MATLAB command window. As mentioned before, please ignore the two messages `implicit conversion from matrix to string.` They appear when Octave version 3.2.4 is used.

How to modify the sample program. All programs in this book are examples, which can and should be modified so that they may be used as templates for other problems. One possible modification of `EQ1.m` is shown in the program `EQ2.m`, which calculated equilibrium binding to two sites. But there are numerous other possibilities. Variables can be changed in lines 10–14. The `plot` may be modified in line `28`, and title, legends and labels in lines `29–34`. Try it out! For example, if you want to show your data as a double reciprocal plot, you simply modify line `28`. Note, however, that one cannot simply use a division for vectors. One has to use element-by-element division (./):

```
28 plot(1./VL,1./VLR,'-x',1./VL,1./LR_analytical,'-+');
```

If you only changed line `28` and run `EQ1` in the octave terminal window, you will get an empty plot, because you forgot to modify the axis. One may remove line 31 and the plot function will find its own default axes. Instead of removing a line, it is easier to define it as a comment by adding the % (percent) character at the beginning of line 31 to define it as a comment:

```
31 %axis([0, MaxL0, 0, R0])
```

And ... voila, a double reciprocal plot appears when you enter EQ1 in the terminal window. Of course, the axis labels are still wrong, but play with it. One can easily modify everything.

The program EQ1b.m is a modification of EQ1.m for the calculation of the difference between numerical and analytical solutions. As discussed in Sect. 2.6, numerical solutions are approximations, whereas analytical solutions always are correct. For Octave 3.2.3 and its default parameters the differences between numerical and analytical solutions of EQ1.m are shown in Fig. 5.2.

The differences were computed with EQ1b.m, a program which differs from EQ1.m mainly in lines 27 and 29.

```
27    Dif=VLR-LR_analytical;
28    %% Plot the data
29    plot(VL,Dif,'-x');
```

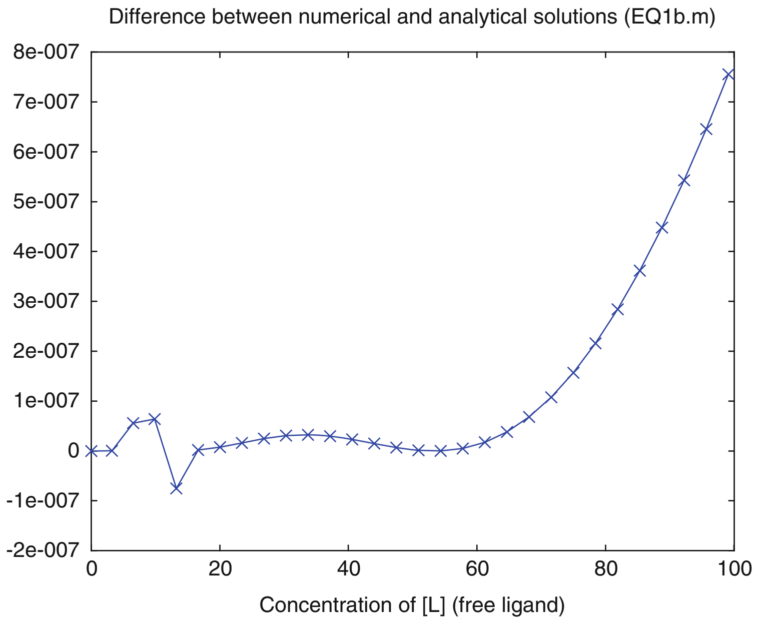


Fig. 5.2 Differences between numerical and analytical solutions. Bound and free ligand concentrations shown in Fig. 5.1 were calculated for 1 unit (μM) receptor and an equilibrium dissociation constant of 10 units (μM). The differences of the numerical and analytical solution as calculated with Octave version 3.2.3 are plotted versus the free ligand concentration. Note the scale of the y-axis, which merely covers a total range of 10^{-6} units, corresponding to 1 pm, which is not visible clearly in Fig. 5.1

The differences between analytical and numerical solutions can, of course, be neglected for all practical reasons. They are more than a million-fold lower than the calculated values for bound ligand. If the accuracy of numerical solutions is an issue, it can be optimized in MATLAB with the `optimset` command. Figure 5.2 is shown here as a caveat. There may be a situation where numerical solutions give unexpected results. We all have a tendency to believe in computers, but numerical artifacts must not be ignored. If they are suspected, they can be detected by switching between Octave and MATLAB or with modifying the optimization parameters or even running the same program on a different computer platform.

5.3 Equilibrium Binding to Two Sites (**`EQ2.m`**)

Reaction scheme (2.19) shows ligand binding to a receptor where two binding sites are accessible. The corresponding set of equations is given in (2.24) and (2.25). Obviously, they differ from the set of equations calculated in `EQ1.m` so that the function `EQ1F` has to be modified. A modified program should have a new name, and `EQ2F` is appropriate. Equations (2.24) and (2.25) translate to:

```
1      %% EQ2F: Binding to two sites on a receptor
2      function F = EQ2F(x)
3      global KD1 KD2 KD3 R0 L0;
4      R=x(1);
5      L=x(2);
6      F(1)=R+R*L/KD1+R*L/KD2+R*L*L/(KD1*KD3)-R0;
7      F(2)=L+R*L/KD1+R*L/KD2+2*R*L*L/(KD1*KD3)-L0;
```

Please note a programming technique, which is unnecessary for the computer, but makes life much easier for the programmer. When the equations become more complex, it is a good idea to replace `x(1)` by `R` and `x(2)` by `L` in lines `6` and `7`. Of course, `R` and `L` have to be assigned, and this is done in lines `4` and `5` before they are used for the equations.

The main program is stored as `EQ2.m`. Again the concentrations of complexes are calculated inside a loop for each ligand concentration in lines `23`–`29`, and the plot command in line `32` can use the vectors generated within the loop.

```
8      filename='EQ2';
9      global KD1 KD2 KD3 R0 L0;

15     MaxL0=100;
16     N=30;
17     VL0=linspace(0, MaxL0, N);
```

```
18    %% Beginning of the program
19    for i=1:N
20    L0=VL0(i);
21    x0(1)=0.5*R0;
22    x0(2)=0.5*L0;
23    x = fsolve('EQ2F',x0);
24    R=x(1);
25    L=x(2);
26    VLR(i)=L*R/KD1;
27    VRL(i)=L*R/KD2;
28    VLRL(i)=L*L*R/(KD1*KD3);
29    VL(i)=L;
30    end;
31    %% Plot the data
32    plot(VL,VLR,'-x',VL,VRL,'-+',VL,VLRL,'-*');
```

The `plot` command is used to plot the concentrations of all complexes. This is done in line `32`. The `legend` in line `38` (not shown here) is changed accordingly.

Figure 5.3 shows the concentrations of LR, RL and LRL as a function of the free ligand concentration. Note that the concentrations of the monoliganded complexes LR and RL only differ by the same factor at all ligand concentrations. This is a general property of all equilibrium-binding curves, where complexes with the same number of ligands only differ by a given factor at all ligand concentrations. The complex LRL initially shows a sigmoid increase. This has nothing to do with cooperativity. It simply results from concentration dependence of the plot, where monoliganded complexes LR and RL are favored at lower ligand concentrations than the saturated complex LRL.

How to modify the sample program. The parameters in the program `EQ2.m` are specified in lines `9–12`, and can easily be modified. For extremely low receptor concentrations, one may consider (5.6) and replace `x0(2) = 0.5*L0;` in line `29` with `x0(2) = L0-0.5*R0;`

One practical modification is shown in `EQ2b.m`: Most binding studies do not measure the concentration of complexes, as in Fig. 5.3, but the concentration of bound ligand. The concentration C_B of bound ligand is

$$C_B = RL + LR + 2 \cdot LRL \tag{5.7}$$

To plot this, the following lines are modified:

```
2     % EQ2b:
8     filename='EQ2b';
31    CB=VLR+VRL+2*VLRL;
33    plot(VL,VLR,'-x',VL,VRL,'-+',VL,2*VLR,'-*',
      VL,CB,'-ok');
38    ylabel('Bound ligand')
39    legend('LR','RL','2*LRL','Bound',2);
```

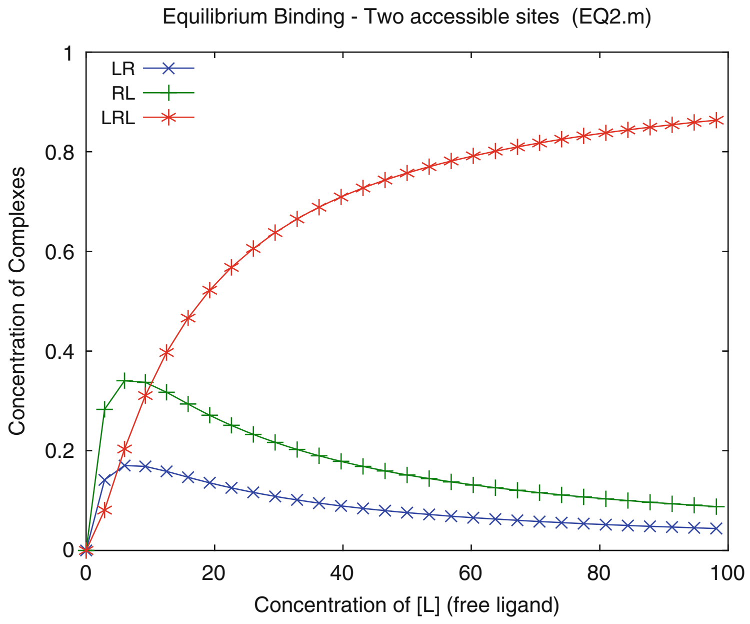


Fig. 5.3 Equilibrium binding to two independent sites. For reaction scheme (2.19), the concentrations of complexes LR (x) RL (+) and LRL (*) are plotted versus the free ligand concentration. The calculated affinities are KD1 = KD3 = 5 µM for the first and KD2 = KD4 = 10 µM for the second site

Lines `2` and `8` are required when the name of the program is changed. It is advisable to rename a program every time it is modified. Line `31` is a translation of (5.7) into Octave code. Line `33` is the modified `plot` command. Note that it is possible to do simple operations like `2* VLR` within the `plot` command, and also note that it is possible to define colors in the plot style (defined with quotes). The three symbols of `'-ok'` at the end of line 40 translate into `-`: solid line `o`: point style as circle and `k`: color black. The colors are specified in Octave as `'k'` black, `'r'` red, `'g'` green, `'b'` blue, `'m'` magenta, `'c'` cyan and `'w'` white. Figure 5.4 is a diagram of bound ligand rather than a diagram of the different complex concentrations. Of course, the bound ligand of the complex LRL is 2·LRL.

5.4 Equilibrium Binding to Two Sites in the Presence of Inhibitor (`EQ3.m`)

When a second ligand can also bind to the receptor, the reaction scheme looks complicated, although the underlying mechanism is not. In most cases, the second ligand is an inhibitor so that the second ligand is denoted as I in scheme (5.8). Just

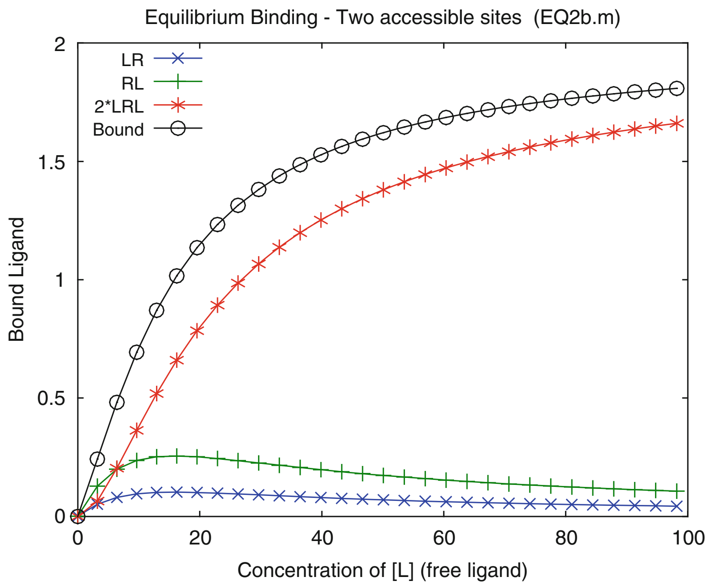


Fig. 5.4 Ligand binding to a receptor with two sites. For reaction scheme (2.19), the concentration of bound ligand (0) is plotted versus the free ligand concentration. The contribution of the complexes LR (x) RL (+) and LRL (*) to the total binding curve is shown. The affinities in (1.19) are $K_D1 = 50$ µM, $K_D2 = 20$ µM, $K_D3 =$ µM and $K_D4 = 12.5$ µM

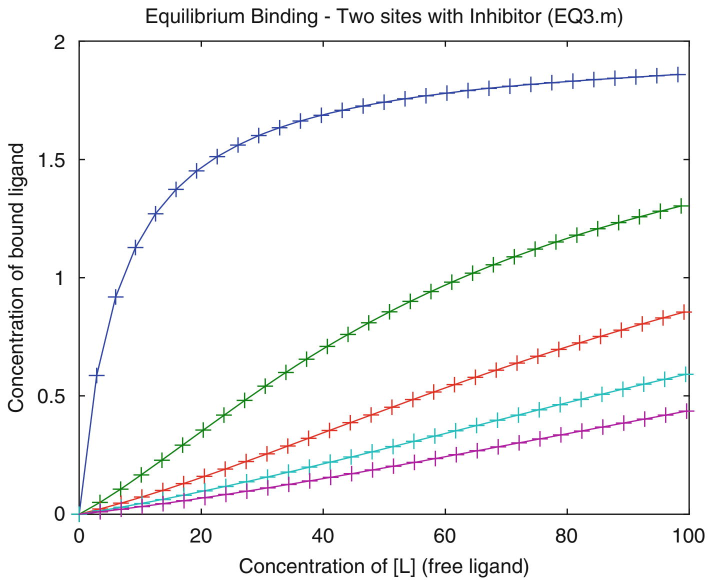
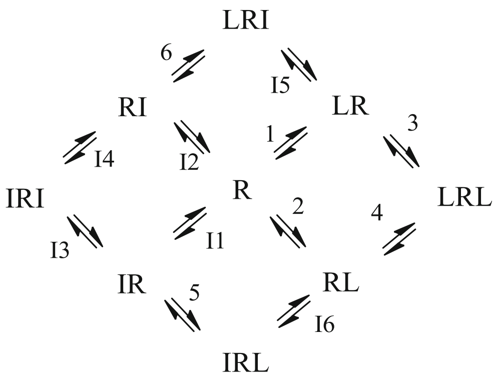


Fig. 5.5 Equilibrium binding – two sites with inhibitor. The concentration of bound ligand was calculated from reaction scheme (5.8) with KD1 = 10 µM, KD2 = 5 µM, KD3 = 5 µM, KI1 = 3 µM, KI2 = 5 µM, KI3 = 3 µM, KI5 = 10 µM and KI6 = 10 µM. The total inhibitor concentrations are (from top to bottom) 0, 25, 50, 75 and 100 µM. The total receptor concentration is 1 µM

like L in scheme (2.19), the inhibitor I may bind to two sites. In reaction scheme (5.8), the assignment of the sites to the respective affinities is consistent with scheme (2.19) so that the equilibrium dissociation constant K_I1 for the inhibitor concerns the "left" site, just like the equilibrium dissociation constant K_D1 for the ligand. There are two mixed complexes, LRI and IRL. LRI can either be formed by the addition of L to RI (equilibrium dissociation constant K_D6) or by the addition of I to LR, with the equilibrium dissociation constant K_I5.



(5.8)

To simplify the reaction scheme, free ligands are not shown explicitly in (5.8). Moreover, the reactions are not assigned with their rate constants, but simply with numbers. The letter I in front of the reaction number indicates that inhibitor is added at this reaction step.

There are four coupled equilibria, i.e. four closed circles in (5.8). Similar to (2.22) the affinities of the three additional ternary complexes must be independent of the reaction pathway so that in addition to (2.23), (5.9)–(5.11) must hold. This reduces the number of independent variables from 12 to 8 independent equilibrium dissociation constants.

$$K_D1 \cdot K_D3 = K_D2 \cdot K_D4 \tag{2.23}$$

$$K_I1 \cdot K_I3 = K_I2 \cdot K_I4 \tag{5.9}$$

$$K_D2 \cdot K_I6 = K_I1 \cdot K_D5 \tag{5.10}$$

$$K_D1 \cdot K_I5 = K_I2 \cdot K_D6 \tag{5.11}$$

The dependent equilibrium dissociation constants are calculated in lines `18–21` of the main program `EQ3.m`.

```
22    KD4=KD1*KD3/KD2;
23    KI4=KI1*KI3/KI2;
24    KD5=KD2*KI6/KI1;
25    KD6=KD1*KI5/KI2;
```

These four equilibrium dissociation constants are not required in EQ3F.m and therefore need not be included in the global declaration in line 2 of EQ3F.m and line 12 of EQ3.m. The main difference to EQ2.m is the additional ligand I which translates to the additional independent variable x(3) in line 5 and to the additional equation F(3) in lines 11–12 of EQ3F.m:

```
1     function F = EQ3F(x)
2     global KD1 KD2 KD3 KI1 KI2 KI3 KI5 KI6 R0 L0 I0;
3     R=x(1);
4     L=x(2);
5     I=x(3);
6     F(1)=R+R*L/KD1+R*L/KD2+R*L*L/(KD1*KD3)...
7     +R*I/KI1+R*I/KI2+R*I*I/(KI1*KI3)...
8     +R*L*I/(KD1*KI5)+R*L*I/(KD2*KI6)-R0;
9     F(2)=L+R*L/KD1+R*L/KD2+2*R*L*L/(KD1*KD3)...
10    +R*L*I/(KD1*KI5)+R*L*I/(KD2*KI6)-L0;
11    F(3)=I+R*I/KI1+R*I/KI2+2*R*I*I/(KI1*KI3)...
12    +R*L*I/(KD1*KI5)+R*L*I/(KD2*KI6)-I0;
```

The three equations with the three unknowns R, L and I in lines 6–12 follow the pattern described in (2.31). For each of these equations, all (free, bound and total) concentrations involving the respective unknown (R, L or I) of scheme (5.8) have to be included. Each of the complexes required for the bound concentration is calculated by the law of mass action from free receptor R, free ligand L and free inhibitor I concentration. Note that bound ligand calculated from LRL in line 9 involves the factor 2, since two ligand molecules are bound in that complex. The same is true for IRI in line 11.

The binding curves are calculated with arrays of 30 ligand and 5 inhibitor concentrations as defined in lines 26–31. These concentrations are varied in a nested loop with the index i varying between 1 and 30 and the index k varying between 1 and 5. The loops run between lines 33 and 54.

```
11    filename='EQ3';
12    global KD1 KD2 KD3 KI1 KI2 KI3 KI5 KI6 R0 L0 I0;

22    KD4=KD1*KD3/KD2;
23    KI4=KI1*KI3/KI2;
24    KD5=KD2*KI6/KI1;
25    KD6=KD1*KI5/KI2;
26    MaxL0=100;
27    N=30;
28    MaxI0=100;
29    NI=5;
30    VL0=linspace(0, MaxL0, N);
31    VI0=linspace(0, MaxI0, NI);
```

```
33    for i=1:N
34    for k=1:NI
35    L0=VL0(i);
36    I0=VI0(k);

44    VLR(i,k)=L*R/KD1;
45    VRL(i,k)=L*R/KD2;
46    VLRL(i,k)=L*L*R/(KD1*KD3);
47    VL(i,k)=L;
48    VIR(i,k)=I*R/KI1;
49    VRI(i,k)=I*R/KI2;
50    VIRI(i,k)=I*I*R/(KI1*KI3);
51    VI(i,k)=I;
52    VLRI(i,k)=L*I*R/(KD1*KI5);
53    VIRL(i,k)=L*I*R/(KD2*KI6);
54    end;  end;

56    BoundL=VRL+VLR+2*VLRL+VIRL+VLRI;
57    plot(VL,BoundL,'-+');
```

The concentrations of all eight complexes as well as the free ligand and inhibitor concentrations are calculated in lines `44–53` within the loop. The bound ligand `BoundL` can then be calculated outside the loop. Note that the concentrations generated inside the loop are not just vectors (arrays) of concentrations at different total ligand concentration, but now are matrixes (spreadsheets) of concentrations at different total ligand and total inhibitor concentration. The elements of these matrices are identified with the indices `i` and `k`. For spreadsheets, rows are typically addressed with numbers and columns with letters. A matrix element VRL(3,2) therefore would correspond to the cell B3 in a spreadsheet. For readers familiar with spreadsheets, this relation may help to identify the matrix elements.

It has been mentioned in Chap. 4 that every variable in Octave is treated by default as a matrix. The statement in line `56` therefore is matrix addition, so each element of the spreadsheet `VRL` is added to each corresponding element of `VLR` and so forth. This leads to rather elegant programming code. Likewise, the plot command in line `56` looks elegant and simple, but it plots five columns of bound ligand calculated for five different inhibitor concentrations versus the corresponding columns of substrate concentration. The results are shown in Fig. 5.5.

Increasing the number of ligands and/or the number of sites and/or the number of conformational states can be done in two steps. First, one has to make a list of all the expected complexes and calculate them as products of their free concentrations divided by the respective equilibrium dissociation constants, just like those in lines `6–8` of `EQ3F.m`. With an increasing number of complexes it may be more complicated to draw a reaction scheme than to write down these equations. The second step consists of writing one equation of the type (2.31) for each ligand. The resulting set of equations `F(x)` has to be solved by `fsolve` in the main program.

This procedure can get as complex as one likes, but always follows the same pattern.

The main advantage of numerical methods and a high-level programming language consists in its flexibility. One can calculate the concentration dependence of any complex or any set of complexes. For example, fluorescence resonance energy transfer (FRET) is specific for the interaction of ligand and inhibitor and can be employed to study its mechanism [2]. Part of the signal often is proportional to the concentrations of ternary complexes, where both types of ligands are bound to the receptor simultaneously. Modifying the program `EQ3.m` and plotting the ternary complexes versus the substrate concentration is performed with the program `EQ3b.m` and the statements below:

```
56     mix=VLRI+VIRL;
57     MA=max(mix);
58     plot(VL,mix,'-o');
61     axis([0, MaxL0, 0, max(MA)])
```

In line `56` the mixed ternary complexes IRL and LRI are added, because they show the same dependence on inhibitor and ligand concentration. The function `max()` in lines `57` and `61` is worth mentioning. For a matrix, it returns the maximal values for each column in a matrix. Therefore, the result of line `56` is a row vector `MA` with the maximal values of `mix` calculated at the five different inhibitor concentrations. For a vector argument, the function `max` in line `61` simply returns the maximal value so that `max(MA)` is the maximal value of all ternary complexes at all inhibitor concentrations. The resulting plot is shown in Fig. 5.6.

The concentrations of the ternary complexes IRL and LRI alike depend on both the ligand and the inhibitor concentrations. At high ligand concentration, the ligand competes with the inhibitor and forms LRL, and at high inhibitor concentrations, the inhibitor mainly binds as IRI. Therefore, the concentration dependence of the ternary complexes is not easy to predict and has to be calculated. Note the circles on the x-axis. They should correspond to the concentration of ternary complexes in the absence of inhibitor and should all be zero. With the numerical solution returned from the Octave algorithm, some were slightly negative and thus could not be plotted within the axes defined in line 61. This problem had been addressed in Fig. 5.2 and discussed in Sect. 2.6.

5.5 Allosteric Interactions of Subunits (**`EQ4.m`**)

The term "allosteric" is used differently in the literature [3]. For most cases today, it states that a ligand may bind to a protein and elicit a conformational change. This change may then change the properties of the protein at another ("allosteric") site. In contrast, the concerted model for oxygen binding to hemoglobin of Monod, Wyman and Changeux [4] ("MWC" model) concerns the allosteric interactions of

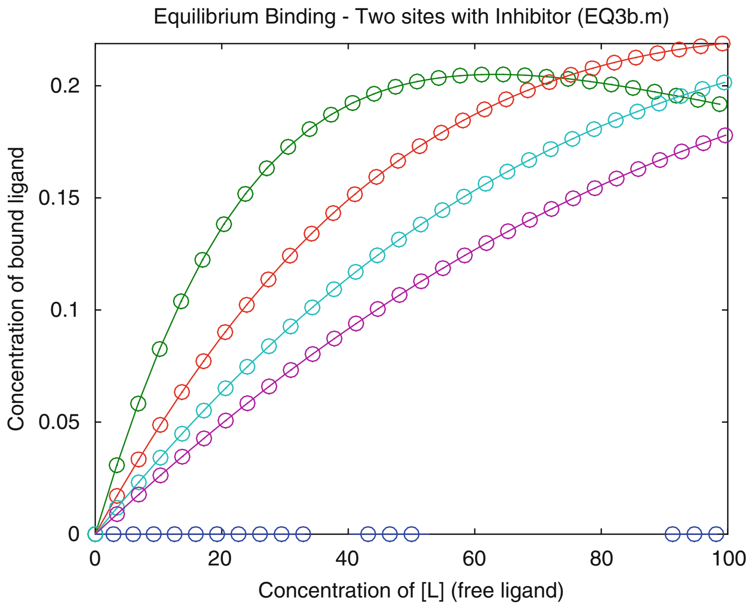


Fig. 5.6 Concentration of ternary complexes IRL + LRI calculated from reaction scheme (5.8) with KD1 = 10 µM, KD2 = 5 µM, KD3 = 5 µM, KI1 = 3 µM, KI2 = 5 µM, KI3 = 3 µM, KI5 = 10 µM and KI6 = 10 µM. The total inhibitor concentrations are (from top to bottom) 25, 50, 75 and 100 µM. The total receptor concentration is 1 µM

protein subunits. It assumes that the four hemoglobin subunits may exist in two conformational states (R and T) even in the absence of ligand as shown in reaction scheme (5.12).

$$\begin{tikzcd} L + R \arrow[r, shift left, "k_1"] \arrow[d, shift left, "k_2"] & LR \arrow[l, shift left, "k_{-1}"] \arrow[d, shift left, "k_4"] \\ L + T \arrow[r, shift left, "k_3"] \arrow[u, shift left, "k_{-2}"] & LT \arrow[l, shift left, "k_{-3}"] \arrow[u, shift left, "k_{-4}"] \end{tikzcd} \tag{5.12}$$

These conformational states traditionally are named R (for relaxed) and T (for tense). In the absence of ligand, the subunits are predominantly in the R conformation ([R] > [T] translates to $K_D2 > 1$ in scheme (5.12)). If the ligand L has a higher affinity for T ($K_D3 < K_D1$) then the equilibrium of these conformational states is shifted upon binding. The binding equilibrium of reaction scheme (5.12) is calculated from the set of equations given in `EQ4a.m`:

```
1      %% Equilibrium Binding to two conformations
2      function F = EQ4aF(x)
3      global KD1 KD2 KD3 R0 L0;
4      R=x(1);
5      L=x(2);
6      T=R/KD2;
7      F(1)=R+T+R*L/KD1+T*L/KD3-R0;
8      F(2)=L+R*L/KD1+T*L/KD3-L0;
```

Line `6` calculates the equilibrium concentration of `T` so that this conformational state can then be used in lines `7` and `8`. These equations again calculate the sums of all (free, bound and total) receptor and all (free, bound and total) ligand concentrations. The program `EQ4a.m` solves equations `7` and `8` and calculates a binding curve as a Scatchard plot (Fig. 5.7)

Figure 5.7 illustrates a general principle of equilibrium binding; no matter how many conformations are there, they will always be in equilibrium at any given ligand concentrations. The concentrations of conformational states differ by the same factor at all ligand concentrations in equilibrium studies. Consequently, since only one binding site is involved, the Scatchard plots are linear for the bound ligand ([LT] + [LR]), as well as for the complexes [LR] and [LT]. Reaction scheme (5.12) shows coupled equilibria with two conformational states, but only one site. The cooperative

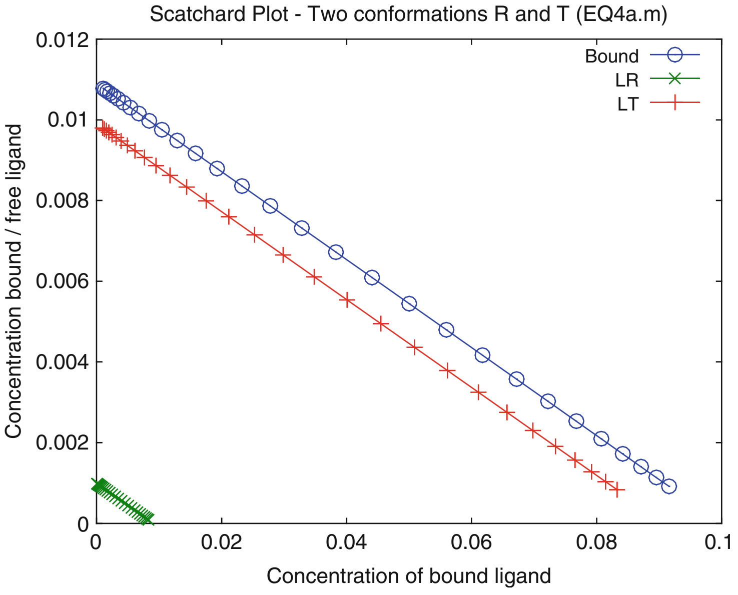


Fig. 5.7 Scatchard Plot: Equilibrium binding to a receptor in two conformational states, R and T. Reaction scheme (5.12) was calculated with KD1 = 100 μM, KD2 = 100 μM, KD3 = 100 nM and a receptor concentration of R0 = 100 nM. The ligand L0 concentration was varied from 0.1 to 100 nM. Total bound ligand (o) as well as the two single bound complexes LR (x) and LT(+) are shown in the Scatchard diagram

binding, which is the main point of the MWC model [4], is not contained within scheme (5.12). For this, an allosteric interaction of subunits is required.

For simplicity, we will assume that the subunits R and T can form dimers (not tetramers, as calculated with the MWC model for hemoglobin). These dimers may be formed independently of the occupation of the monomers with ligand so that 16 dimers have to be considered. Four of these (RR, RT, TR and TT) are formed in the absence of ligand. Depending on the symmetry of the molecule, the dimer TR may be different from the dimer RT so that both dimers have to be distinguished.

$$\begin{array}{cccc}
R+R \underset{K_D5}{\rightleftharpoons} RR & R+T \underset{K_D6}{\rightleftharpoons} RT & T+R \underset{K_D7}{\rightleftharpoons} TR & T+T \underset{K_D8}{\rightleftharpoons} TT \\
LR+R \underset{K_D5}{\rightleftharpoons} LRR & LR+T \underset{K_D6}{\rightleftharpoons} LRT & LT+R \underset{K_D7}{\rightleftharpoons} LTR & LT+T \underset{K_D8}{\rightleftharpoons} LTT \\
R+LR \underset{K_D5}{\rightleftharpoons} RRL & R+LT \underset{K_D6}{\rightleftharpoons} RTL & T+LR \underset{K_D7}{\rightleftharpoons} TRL & T+LT \underset{K_D8}{\rightleftharpoons} TTL \\
LR+LR \underset{K_D5}{\rightleftharpoons} LRRL & LR+LT \underset{K_D6}{\rightleftharpoons} LRTL & LT+LR \underset{K_D7}{\rightleftharpoons} LTRL & LT+LT \underset{K_D8}{\rightleftharpoons} LTTL
\end{array} \tag{5.13}$$

The allosteric monomer–dimer interaction is usually regarded to be independent of the occupation of the monomers so that four equilibrium dissociation constants K_D5–K_D8 are sufficient to calculate dimer formation in (5.13). For cooperative binding, one would expect K_D6 and K_D7 to be large so that the dimers of the type RT and TR are unlikely. The elements R, T, LR and LT of the dimer formation in reaction scheme (5.13) depend on the ligand concentration L as shown in (5.12). The reaction schemes (5.12) and (5.13) therefore have to be merged for calculating all equilibrium complexes. The function `EQ4F.m` becomes rather complex, but the principle behind is simple. For example (5.13) states that the concentration of LTRL is

$$[LTRL] = [LT] \cdot [LR]/K_D7 \tag{5.14}$$

from (5.12)

$$[LR] = [L] \cdot [R]/K_D1 \tag{5.15}$$

$$[LT] = [L] \cdot [T]/K_D3 \tag{5.16}$$

Therefore:

$$[LTRL] = [L] \cdot [L] \cdot [R] \cdot [T]/(K_D1 \cdot K_D3 \cdot K_D7) \tag{5.17}$$

All other complexes are calculated with the same principle, applying the law of mass action. Function EQ4F looks complex, because all 16 complexes of (5.13) have to be considered.

```
1     %% Equilibrium Binding to two conformations
2     function F = EQ4F(x)
3     global KD1 KD2 KD3 KD5 KD6 KD7 KD8 R0 L0;
4     R=x(1);
5     L=x(2);
6     T=x(1)/KD2;
7     F(1)=R+T+2*R*R/KD5+2*R*T/KD6+2*T*R/KD7...
8     +2*T*T/KD8+R*L/KD1+T*L/KD3…
9     +2*L*R*R/(KD1*KD5)+2*L*R*T/(KD1*KD6)...
10    +2*L*T*R/(KD3*KD7)+2*L*T*T/(KD3*KD8)...
11    +2*L*L*R*R/(KD1*KD1*KD5)...
12    +2*L*L*R*T/(KD1*KD3*KD6)...
13    +2*L*L*T*R/(KD1*KD3*KD7)...
14    +2*L*L*T*T/(KD3*KD3*KD8)-R0;
15    F(2)=L+R*L/KD1+T*L/KD3...
16    +L*R*R/(KD1*KD5)+L*R*T/(KD1*KD6)...
17    +L*T*R/(KD3*KD7)+L*T*T/(KD3*KD8)...
18    +2*L*L*R*R/(KD1*KD1*KD5)...
19    +2*L*L*R*T/(KD1*KD3*KD6)...
20    +2*L*L*T*R/(KD1*KD3*KD7)...
21    +2*L*L*T*T/(KD3*KD3*KD8)-L0;
```

The factor 2 is required when there are two receptor subunits or two ligand molecules contained within a complex. In the main program (EQ4.m), the bound ligand is calculated from the sum of all these complexes:

```
29    VLR(i)=L*R/KD1;
30    VLT(i)=L*T/KD3;
31    VL(i)=L;
32    VLRR(i)=L*R*R/(KD1*KD5);
33    VLRT(i)=L*R*T/(KD1*KD6);
34    VLTR(i)=L*R*T/(KD3*KD7);
35    VLTT(i)=L*T*T/(KD3*KD8);
36    VLRRL(i)=L*L*R*R/(KD1*KD1*KD5);
37    VLRTL(i)=L*L*R*T/(KD1*KD3*KD6);
38    VLTRL(i)=L*L*T*R/(KD1*KD3*KD7);
39    VLTTL(i)=L*L*T*T/(KD3*KD3*KD8);
40    end;
41    Bound=VLR+VLT+VLRR+VLRT+VLTR+VLTT...
42    +2*VLRRL+2*VLRTL+2*VLTRL+2*VLTTL;
```

The resulting Scatchard plot (Fig. 5.8) not only shows the total bound ligand from lines 41 to 42, it also includes the ligand bound in the complexes LT and LTTL, as stated in lines 44 and 45.

```
44    plot(Bound,Bound./VL,'-o',VLT,VLT./VL...
45    ,'-x',2*VLTTL,2*VLTTL./VL,'-+');
```

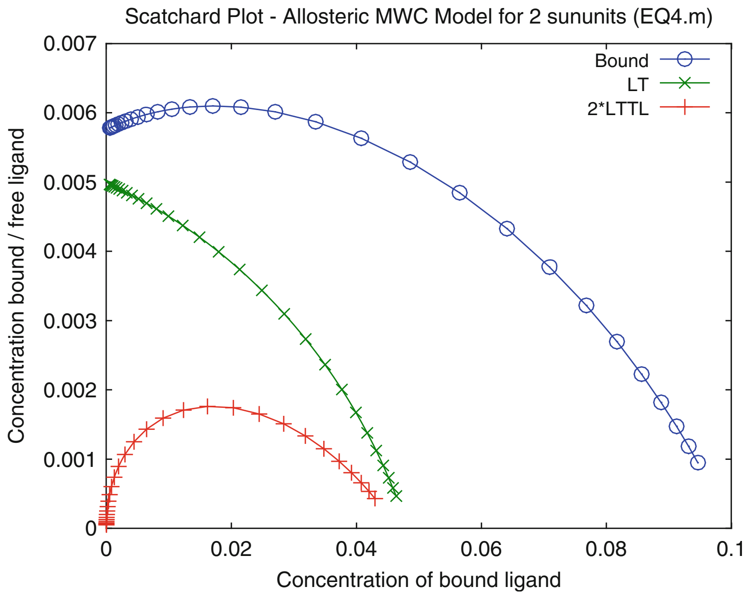


Fig. 5.8 Scatchard Plot: Allosteric Model. Reaction scheme (5.13) with (5.12) was combined in `EQ4F.m` and calculated with KD1 = 100 μM, KD2 = 100 μM, KD3 = 100 nM and a receptor concentration of R0 = 100 nM. RR, RT, TR and TT interactions were calculated with equilibrium dissociation constants of 100 nM, 100 μM, 100 μM and 100 nM, respectively. The ligand concentration L0 was varied from 0.1 to 100 μM. Total bound ligand (o), [LT] (x) and 2·[LTTL] (+) are shown in the Scatchard diagram

Figure 5.8 shows a curved Scatchard plot not only for the totally bound ligand, but also for the main components. This clearly shows cooperative binding which is induced by the allosteric interaction of subunits (5.13). Comparing Fig. 5.8 to Fig. 5.7 illustrates that cooperative binding can only be observed when more than one binding site is involved. In the case of the MWC model [4], the number of sites corresponds to the number of interacting subunits, not to the number of ligand-binding sites on one subunit.

How to modify program EQ4.m. The parameters can be changed in lines 10–17. As soon as the equilibrium dissociation constants of dimer formation K_D5–K_D8 are equal, the cooperativity is lost. It may be interesting to calculate the ratios of R/T, of RL to TL and of LRRL to LTTL as a function of ligand concentrations. It certainly will help to understand the intrinsic properties of the MWC model.

5.6 Allosteric Activators and Inhibitors (`EQ5.m`)

As mentioned before, there are many other interpretations of the word "allosteric," mainly referring to allosteric interactions between ligands and proteins, not to the allosteric interactions between subunits [4] as described above. A dedicated review

[6] explains the different usage of the term. Here we do not discuss molecular interpretations, but calculate reaction schemes instead. The simplest feasible scheme (5.18) may be derived from the following reasoning; the word “allosteric” is used when one wants to state that another (an “allosteric”) ligand may influence the binding of the main ligand without direct competition. In this case, the binding of the allosteric effector would influence the conformation of the receptor molecule and either enhance the activity of the primary ligand or decrease it. In the first case, the effector would be called an “activator,” and in the second case it is an inhibitor. Interaction between subunits is not required. In the simplest form, a corresponding reaction scheme is depicted in (5.18), where L is the ligand, R the receptor and I the allosteric effector.

$$\begin{tikzcd}
L + R \arrow[r, shift left, "k_1"] \arrow[d, shift left, "k_2", "+I"'] & LR \arrow[l, shift left, "k_{-1}"] \arrow[d, shift left, "k_4", "+I"'] \\
L + RI \arrow[r, shift left, "k_3"] \arrow[u, shift left, "k_{-2}"] & LRI \arrow[l, shift left, "k_{-3}"] \arrow[u, shift left, "k_{-4}"]
\end{tikzcd} \tag{5.18}$$

Scheme (5.18) corresponds to equilibrium binding in the presence of inhibitor. If, however, $K_D4 < < K_D2$, the “inhibitor” (the effector I), will stabilize LRI. I then increases the affinity of L to RI (coupled equilibrium (5.11)) so that I is an allosteric activator and no inhibitor. The set of equations for this scheme is given in EQ5F.m:

```
function F = EQ5F(x)
global KD1 KD2 KD3 R0 L0 I0;
R=x(1);
L=x(2);
I=x(3);
F(1)=R+R*L/KD1+R*I/KD2...
+R*L*I/(KD2*KD3)-R0;
F(2)=L+R*L/KD1...
+R*L*I/(KD2*KD3)-L0;
F(3)=I+R*I/KD2...
+R*L*I/(KD2*KD3)-I0;
```

When the bound ligand is calculated in EQ5.m as the sum of LR and LRI in reaction scheme (5.14), the Scatchard plot (Fig. 5.9) can be calculated. It is linear for all effector concentrations, indicating that the effector changes the affinity, but that it cannot lead to cooperativity. Figure 5.9 was calculated with $K_D1 = 10\ \mu M$ and $K_D3 = 1\ \mu M$ so that the allosteric effector I acts as an activator. For $K_D1 < K_D3$, it acts as an inhibitor, because the ligand binding to RI has a lower affinity than the ligand binding to the free receptor R.

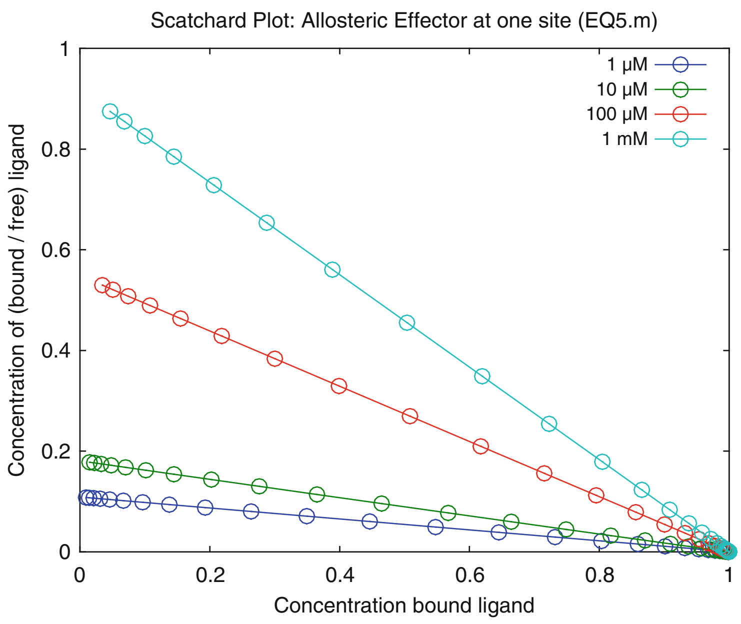


Fig. 5.9 Scatchard plot in the presence of different concentrations of an allosteric effector. Reaction scheme (5.18) was calculated with KD1 = 10 µM, KD2 = 100 µM, KD3 = 1 µM and a receptor concentration of 1 µM. The effector concentration is varied from 1 (*bottom line*) to 1,000 µM (*top line*), indicating an increase in apparent affinity with the effector concentration

Figure 5.9 shows a linear Scatchard plot for an allosteric activator. A linear Scatchard plot would also be observed for $K_D1 < K_D3$, i.e. for an allosteric inhibitor. This concludes that cooperative binding is not related to allosteric mechanism per se. Cooperative effects can only be expected when a ligand or an inhibitor or an activator binds to more than one site of a receptor molecule. For the MWC model, the additional sites are generated by allosteric interactions between subunits, leading to multiple sites on the holoreceptor.

5.7 Dose–Response Curves (`EQ6.m`)

Dose–response curves are basically equilibrium-binding curves, plotted in a logarithmic scale (Sect. 3.1.5). The response (be it enzymatic activity, the opening of an ion channel, the cellular activity or any physiological response) is regarded as the result of agonist binding to a receptor. In a logarithmic scale, any equilibrium-binding curve and any dose–response curve looks sigmoid (Figs. 3.7 and 5.10).

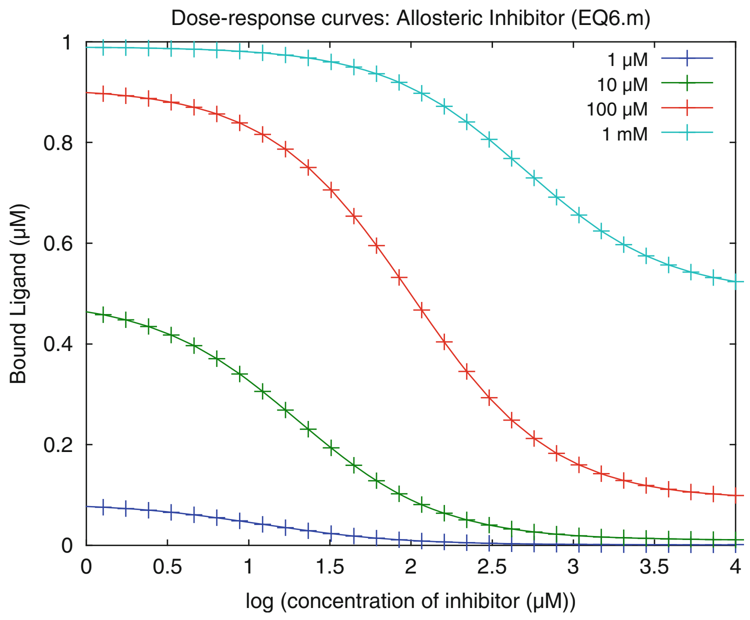


Fig. 5.10 Dose–response curves (bound ligand vs. log inhibitor concentration) for different agonist concentrations. Reaction scheme (5.18) was calculated with KD1 = 10 µM, KD2 = 10 µM, KD3 = 1 mM and a receptor concentration of 1 µM. The ligand concentration is varied from 1 (*bottom line*) to 1,000 µM (*top line*) in four steps

For screening campaigns, the response resulting from agonist binding simply is used as a signal, and the inhibition or stimulation of this signal as a function of drug concentration is investigated. Most compounds in screening campaigns act as inhibitors so that the activity typically is reduced with increasing drug (inhibitor) concentration. This type of dose–response curve is shown in Fig. 5.10, calculated for different ligand concentrations of reaction scheme (5.18). The equations for this scheme are given in the function `EQ5F.m`. The parameters used for Fig. 5.10 imply inhibition, whereby the allosteric inhibitor decreases the affinity for the ligand, but does not block it completely. In terms of reaction scheme this corresponds to a factor 100 between the affinity of the agonist (L) in to R in the presence (KD3) or absence (KD1) of inhibitor.

The extrapolations of the response to zero inhibitor concentrations shown in Fig. 5.10 reflect the formation of LR at the different ligand concentrations, whereas the extrapolations to infinite inhibitor concentrations reflect the formation of LRI.

In screening campaigns, one usually does not plot bound ligand in absolute scales, but shows the data as percent of maximal–minimal signal. This type of signal manipulation is performed in lines `42–55` of the program `EQ6b.m`.

```
11    filename='EQ6b';
12    global KD1 KD2 KD3 R0 L0 I0;

18    N=4;
19    MaxI0=100;
20    NI=30;
21    VL0=logspace(0, 3, N);
22    VI0=logspace(0, 4, NI);
23    %% Beginning of the program
24    for i=1:N

41    I0=0;
42    x = fsolve('EQ1F',x0);
43    VLRmax(i)=x(1)*x(2)/KD1;
44    temp=KD1;
45    KD1=KD3;
46    x = fsolve('EQ1F',x0);
47    VLRmin(i)=x(1)*x(2)/KD1;
48    KD1=temp;
49    end;
50    boundL=VLR+VLRI;
51    lVI=log10(VI(1,:));
52    for i=1:N
53    resp(i,:)=100*(boundL(i,:)...
54    -VLRmin(i))./(VLRmax(i)-VLRmin(i));
55    end;
```

For zero inhibitor concentrations, the maximal concentration `VLRmax` (line `43`) is calculated from the function `EQ1F.m` with the equilibrium dissociation constant `KD1`. For infinite inhibitor concentrations, `VLRmin` (line `47`) is calculated from the same function (`EQ1F.m`, binding to one site in the absence of an inhibitor) with the equilibrium dissociation constant `KD3`. `EQ1F` expects `KD1` as the parameter name to be transferred via the `global` statement so that `KD3` has to be renamed `KD1` in line `45`. The variable `temp` is used in line `44` to save the value of `KD1` and restore it in line `48`. With `VLRmax` and `VLRmin`, the response can then be calibrated as percent values in lines `53` and `54`.

With these modifications, the dose–response curves from Fig. 5.10 all have the same shape, as shown in Fig. 5.11. They are shifted along the x-axis in agreement with competition mechanisms, where the apparent K_D for an inhibitor is increased with the ligand concentration. The shape of these dose- or inhibitor–response curves corresponds exactly to simple binding curves, which look sigmoid in a logarithmic scale, as shown in Fig. 3.7. The Hill coefficient of all these curves is one. Again, the allosteric model of (5.18) cannot explain the results of screening campaigns, where dose–response curves often show steeper maximal slopes, corresponding to Hill coefficients larger than one [7].

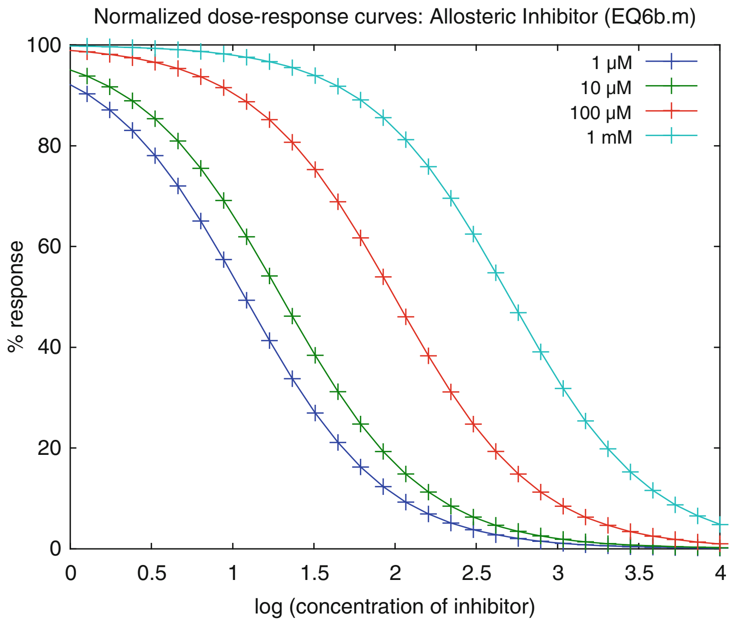


Fig. 5.11 Normalized dose–response curves (% activity vs. inhibitor concentration) for different ligand concentrations. Reaction scheme (5.18) was calculated with KD1 = 10 µM, KD2 = 10 µM, KD3 = 1 mM and a receptor concentration of 1 µM. The ligand concentration is varied from 1 (*bottom line*) to 1,000 µM (*top line*)

5.8 Multiple Allosteric Inhibition (`EQ7.m`)

Dose–response curves are measured under equilibrium condition, or at least are intended to be measured under equilibrium conditions (but note Fig. 5.11). When any concentration of a complex is measured under equilibrium conditions as a function of other concentrations, its shape is only influenced by the number of binding sites, not by the number of conformations. Therefore, inhibitor–response curves under equilibrium conditions with a steeper maximal slope than shown in Fig. 5.11 can only be envisioned when more than one inhibitor-binding site is involved in the process. Reaction scheme (5.19) depicts the simplest of these cases.

A ligand may bind specifically to one site (the active site) of a receptor, but there may also be different sites for inhibitors. If the binding of one or more of inhibitors leads to loss of the active conformation of the receptor, then the ligand may lose its affinity to the active site. In the case of enzymes, loss of the active conformation corresponds to loss of activity.

Denaturation via conformational changes is rather trivial. Heating of proteins leads to conformational changes and typically to inactivation. Denaturation by the binding of molecules is also not so very new, since protons, ionic detergents, salts or organic solvents, etc. all bind to sites different from the ligand-binding site and lead to denaturation and inactivation of proteins. These are unspecific allosteric mechanisms. Drugs identified in screening campaigns are specific for their target proteins, yet they may act via conformational changes at multiple sites. Reaction scheme (5.19) shows an example for a maximum of six multiple allosteric sites.

$$\begin{tikzcd}
R \arrow[rr, rightleftharpoons, "+L", "K_D1"'] \arrow[d, shift left, rightleftharpoons, "+I", "K_I1"'] & & LR & & & \\
RI \arrow[r, rightleftharpoons, "+I", "K_I2"'] & RI_2 \arrow[r, rightleftharpoons, "+I", "K_I3"'] & RI_3 \arrow[r, rightleftharpoons, "+I", "K_I4"'] & RI_4 \arrow[r, rightleftharpoons, "+I", "K_I5"'] & RI_5 \arrow[r, rightleftharpoons, "+I", "K_I6"'] & RI_6
\end{tikzcd} \tag{5.19}$$

Reaction scheme (5.19) need not imply a mechanism with exactly six sites. It can be used to calculate inhibitor binding to one site when only K_I1 is near the inhibitor concentration, and the other equilibrium dissociation constants for the inhibitor are orders of magnitude larger. Likewise, it can be used to calculate inhibition with 2, 3, 4, 5 and a maximum of six sites, depending on the chosen constants.

The set of equations from (5.19) are calculated with the function `EQ7F.m`, listed below. The binding of inhibitors in scheme (5.19) is a simplified sequential mechanism. The sequential equilibrium dissociation constants K_I correspond to the minimum of parameters for the respective number of sites and therefore are useful for the calculations. With the assumption of accessible and equivalent sites, intrinsic equilibrium dissociation constants can be calculated with the help of (2.30).

```
function F = EQ7F(x)
global KD1 KI1 KI2 KI3 KI4 KI5 KI6 R0 L0 I0;
R=x(1);
L=x(2);
I=x(3);
F(1)=R+R*L/KD1+R*I/KI1+R*I*I/(KI1*KI2)...
+R*I*I*I/(KI1*KI2*KI3)...
+R*I*I*I*I/(KI1*KI2*KI3*KI4)...
+R*I*I*I*I*I/(KI1*KI2*KI3*KI4*KI5)...
+R*I*I*I*I*I*I/(KI1*KI2*KI3*KI4*KI5*KI6)-R0;
F(2)=L+R*L/KD1-L0;
F(3)=I+R*I/KI1+2*R*I*I/(KI1*KI2)...
+3*R*I*I*I/(KI1*KI2*KI3)...
+4*R*I*I*I*I/(KI1*KI2*KI3*KI4)...
+5*R*I*I*I*I*I/(KI1*KI2*KI3*KI4*KI5)...
+6*R*I*I*I*I*I*I/(KI1*KI2*KI3*KI4*KI5*KI6)-I0;
```

The main program, `EQ7.m`, initially defines $K_I1 = 100$ μM and assumes that all other equilibrium dissociation constants for the inhibitor (K_I) take the extremely high value of 1 M (lines `16–21`). This combination of parameters allows the calculation of inhibition with one inhibitor site. This is done in the loop in lines `25–26`. In line `37` the command `KI2 = KI1` sets `KI2 = 100` μM so that the next loop in lines `37–49` can calculate inhibition with two inhibitor sites and so forth. The program gets rather long because six of these loops have to be executed. This type of programming does not look elegant, but copy and paste is fast and effective.

```
25    for k=1:NI
26    I0=VI0(k);
36    end;
37    KI2=KI1;
38    for k=1:NI
39    I0=VI0(k);
49    end;
50    KI3=KI1;
...
```

Figure 5.12 shows inhibitor–response curves for 1, 2, 3 and 6 sites with the same equilibrium dissociation constant of 100 μM. Of course, the shapes of these curves not only depend on the number of inhibitor sites, but also on their relative affinities.

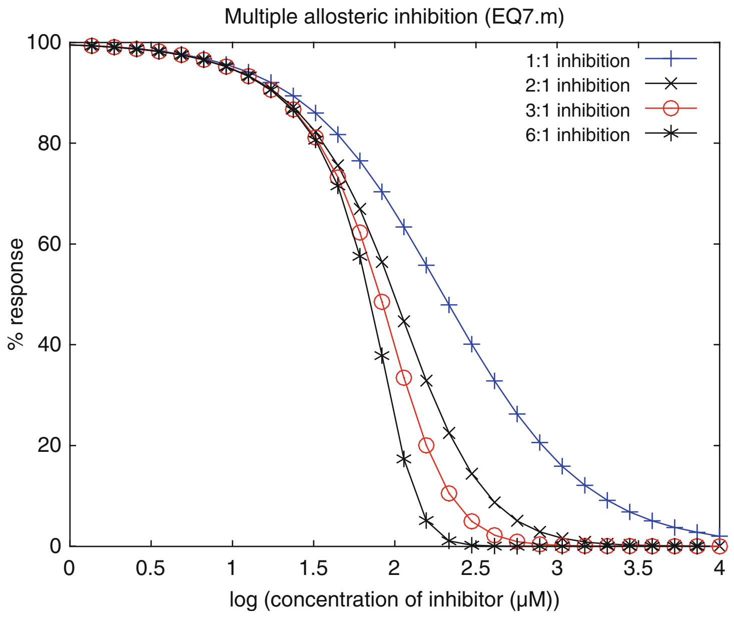


Fig. 5.12 Multiple allosteric inhibition. Dose–response curves for relative activity vs. inhibitor concentration were calculated for 1, 2, 3 and 6 multiple allosteric inhibitor sites. Receptor concentration R0 = 1 μM, ligand (agonist) concentration L0 = 10 μM and KD1 = 10 μM. All equilibrium dissociation constants from scheme (5.19) were calculated as 100 μM with (+) 1, (x) 2, (o) 3 and (*) six binding sites for the inhibitor

Different equilibrium dissociation constants for the inhibitors can be set in lines 16–21 of EQ7.m.

The calculated curves shown in Fig. 5.12 are not symmetric. They differ markedly from symmetric logistic functions (Fig. 3.7) routinely used for the analysis of dose–response curves [8]. The asymmetric shape is a general feature of compound binding to multiple sites: At low concentrations, only a small fraction of the sites is occupied, mostly in the form of singly bound receptor. For dose–activity curves this results in relatively low increases of inhibition with the inhibitor concentration. Multiple bound receptors will only appear at higher inhibitor concentrations, where the concentration of free (active) receptor then rapidly decreases.

Reaction scheme (5.19) or dose–response curves shown in Fig. 5.12 do not prove allosteric interactions. The same scheme and the same dose–response curves have been interpreted isosterically as transient binding patches [9]. We can conclude that dose–response curves with Hill coefficients larger than one cannot be calculated from binding equilibria with one inhibitor or activator binding site alone, be it an allosteric mechanism or not. For our own studies with chitinase inhibition, we had found Hill coefficients different from one, looked again at our X-ray structures and identified two inhibitor-binding sites [10]. Whenever Hill coefficients larger than one are found, there must be reasons for it. Fitting dose–response curves to plausible reaction schemes should be possible with the programs presented here or with enhanced modifications. Additional evidence (like isothermal calorimetry or crystal structure) will help to refine the model and lead to a consistent understanding of the mechanisms involved.

References

1. More JJ, Garbow BS, Hillstrom KE (1980) User guide for MINIPACK-1. Argonne National Laboratory Report ANL-80-74
2. Berger W, Prinz H, Striessnig J, Kang H-C, Haugland R, Glossmann H (1994) Complex molecular mechanism for dihydropyridine binding to L-type Ca^{2+}-channels as revealed by fluorescence resonance energy transfer. Biochemistry 33:11875–11883
3. http://en.wikipedia.org/wiki/Allosteric_regulation
4. Monod J, Wyman J, Changeux JP (1965) On the nature of allosteric transitions: a plausible model. J Mol Biol 12:88–118
5. Verhulst PF (1845) Recherches mathématiques sur la loi d'accroissement de la population. Nouv mém de l'Academie Royale des Sci et Belles-Lettres de Bruxelles 18:1–41
6. Fenton AW (2008) Allostery: an illustrated definition for the 'second secret of life'. Trends Biochem Sci 33:420–425
7. NIH Data base: http://www.ncbi.nlm.nih.gov/sites/entrez. Chem BioAssay
8. Guidelines for standardized dose-response curves: http://www.ncgc.nih.gov/guidance/manual_toc.html
9. Prinz H, Schönichen A (2008) Transient binding patches: a plausible concept for drug binding. J Chem Biol 1:95–104
10. Pantoom S, Vetter IR, Prinz H, Suginta W (2011) Potent family-18 chitinase inhibitors: X-ray structures, affinities and binding mechanisms. J Biol Chem, 286:24312–24323

Chapter 6
Binding Kinetics

Differential equations are solved by calculating the difference quotient and adding the computed concentration differences to the initial values. This is a typical task for a computer. For GNU Octave, the function lsode is used as a universal solver for ordinary differential equations, while a selection of different solvers is available in MATLAB®. They are all used with a similar syntax. The main challenge for the scientist consists in transforming reaction schemes into differential equations. From these, association or dissociation kinetics is computed by selecting different initial concentrations. Sample programs for lag-phase, facilitated dissociation, sequential binding mechanisms or irreversible inhibitors are explained. At the end of this chapter, the reader should be able to write and solve differential equations for any reaction scheme, be it as complex as desired.

6.1 Solving Differential Equations in GNU Octave and MATLAB

Numeric methods are ideally suited for the calculation of differential equations. Mathematically, the differential quotient dx/dt is the limiting value of the difference quotient Δx/Δt for infinitesimal small differences. Computer programs simply use their power of repetitive commands: Calculate the differences Δx of all concentrations for a small time interval Δt, add these differences to the original values and do the calculations for the next time interval, until the desired time range is covered.

This procedure is called a numerical solution of differential equations. For ordinary differential equations ("ODE"s, they can be written as dx/dt = f(x, t); all feasible reaction schemes fall into this category) Octave and MATLAB provide procedures to solve them. For Octave, the solver is the function `lsode`. It is based on ODEPAC [1] a collection of Fortran solvers. In MATLAB, there is a family of similar solvers called `ode23, ode45, ode113, ode15s, ode23s, ode23t, ode23tb`.

All of these solvers require initial concentrations, because all differences Δx have to be calculated from starting values. Such initial concentrations are no estimates, and therefore should not be confused with x0 in fsolve. In Octave and MATLAB, the solvers require that the differential equations are supplied as functions. Again (like in `fsolve` in Chap. 5), the set of equations (6.1) to be solved is written as a vector, whereby each element of this vector is a differential equation.

$$\Delta\mathbf{x} = \mathbf{f}(\mathbf{x}) \cdot \Delta t \tag{6.1}$$

with **Δx**, **f** and **x** as vectors. Since the solvers have their own algorithms to optimize the values for Δt, the functions for Octave and MATLAB solvers are written without Δt as

$$\Delta\mathbf{x} = f(\mathbf{x}) \tag{6.2}$$

Both, for Octave and for MATLAB solvers, these vectors are column vectors. In GNU Octave, the solver is called with the command

```
M=lsode('name_of_function',x0,t);
```
(6.3)

In MATLAB, the corresponding command is

```
[T,M]=ode45('name_of_function',t,x0);
```
(6.4)

`ode45` is a versatile function from the MATLAB library of solvers. For both computer languages, t is a vector of time points for which the differential equations have to be solved, and x0 is a column vector of all initial concentrations. M is the matrix of the results. Its rows correspond to the time points, and the columns to the different complexes. Therefore, calculating concentration changes of 3 molecules at 30 time points will give a matrix of 30 rows and 3 columns. T in (6.4) is basically the same vector as t, but whereas t may be a column or a row vector, the resulting T in MATLAB always is a column vector. The main (and trivial) difference between GNU Octave and MATLAB, however, is the order of arguments in the functions [(6.3) (6.4)]. This corresponds to reversed orders of arguments in the function name, when called either from lsode in Octave or from ode45 in MATLAB. In GNU Octave, the basic syntax is:

```
function dx=name_of_function(x,t)
dx=f(x)
```
(6.5)

In MATLAB, the corresponding function has to have the arguments exchanged:

```
function dx=name_of_function(t,x)
dx=f(x)
```
(6.6)

This is all, but unfortunately it implies that Octave functions cannot be called from `ode45` in MATLAB, and that MATLAB functions cannot be called from `lsode` in GNU Octave. Therefore, all programs involving differential equations and all corresponding functions have to be supplied in two versions, one for each programming language. The MATLAB versions are marked as `nameM.m` and stored in the MATLAB directory, whereas the Octave versions are not distinguished and listed as `name.m`, stored in the Octave directory.

6.2 Kinetics of Ligand Binding to One Site (`kin1.m`)

These general considerations have to be substantiated with a simple example. Reversible binding of the ligand L to a receptor R [reaction scheme (2.9)] leads to the formation of the complex LR and to a corresponding decrease of L and R. The difference quotients are:

$$\Delta[\mathrm{L}]/\Delta \mathrm{t} = -\mathrm{k}_1 \cdot [\mathrm{L}] \cdot [\mathrm{R}] + \mathrm{k}_{-1} \cdot [\mathrm{LR}] \tag{6.7}$$

$$\Delta[\mathrm{R}]/\Delta \mathrm{t} = -\mathrm{k}_1 \cdot [\mathrm{L}] \cdot [\mathrm{R}] + \mathrm{k}_{-1} \cdot [\mathrm{LR}] \tag{6.8}$$

$$\Delta[\mathrm{LR}]/\Delta \mathrm{t} = \mathrm{k}_1 \cdot [\mathrm{L}] \cdot [\mathrm{R}] - \mathrm{k}_{-1} \cdot [\mathrm{LR}] \tag{6.9}$$

These equations correspond to lines 5–7 in the Octave function `kin1F.m`. The command `zeros` in line 4 defines a column vector `dx` with 3 elements. The numerical value (zero) is not important at this stage, and line 4 is only used to define the dimensions of the output `dx`.

```
1     %% reversible binding
2     function dx=kin1F(x,t)
3     global k1 km1;
4     dx=zeros(3,1);
5     dx(1)=-k1*x(1)*x(2)+km1*x(3); %x(1)=L
6     dx(2)=-k1*x(1)*x(2)+km1*x(3); %x(2)=R
7     dx(3)= k1*x(1)*x(2)-km1*x(3); %x(3)=LR
```

For MATLAB, x and t are exchanged in their position in line 2:

```
2              function dx=kin1FM(t,x)
```

Everything else remains the same. Note that `x(1)` = [L], `x(2)` = [R] and `x(3)` = [LR]. The differential equations `kin1F` are solved with `lsode` in line `26` of the main program `kin1.m`:

```
23    t=linspace(0, tmax, N);
24    x1=[L1;R1;LR1];
25    %% Main Program
26    M=lsode('kin1F',x1,t);
27    L=M(:,1);
28    R=M(:,2);
29    LR=M(:,3);
30    %% Plot
31    plot(t,L,'-+',t,R,'-x',t,LR,'-*');
```

The time points t are defined in line 23, the initial concentrations x1 as a column vector in line 24. The **initial** concentrations for all reactants are denoted as `L1`, `R1` and `LR1`. We may avoid the more common L0, R0 and LR0 at this stage, because the zero has been used for **total** concentrations in Chap. 5. Note that the time points given in the row vector `t` (lines `23`, `26` and `31`), only are those time points, for which the calculated values are **shown**, and not all the infinitesimal small time steps which are computed internally.

The solution of the differential equations is given as a matrix M, which is a row of column vectors for all the computed concentrations. These concentrations are assigned to their symbols in lines `27`–`29`. The colon (`:`) in line `27` translates to the command: "Take all rows (the colon character stands for the whole range) from column 1 of the matrix M and assign it to the new column vector L". The length of these new column vectors `L`, `R` and `LR` is the same as the number of time points `t`. The results are shown in Fig. 6.1.

Running the program `kin1.m` in MATLAB only requires that line `26` is changed. The MATLAB version of `kin1.m` is named `kin1M.m`, and line `26` of the MATLAB version is the following:

```
26    [T,M]=ode45('kin1FM',t,x1);
```

The resulting matrix M in MATLAB has the same structure as in Octave, so that no other lines have to be modified in the MATLAB version.

Figure 6.1 shows the association of ligand L (+) and receptor R (x) as a function of time. Both of these free concentrations decrease, while the concentration of bound ligand LR (*) increases. Unlike pseudo-first order kinetics described in Sect. 3.2.4, numerical methods allow this reaction to be computed at any ratio of ligand to receptor, even when it is almost 1:1, as shown here.

How to modify the sample program. Just by changing the initial concentrations, dissociation kinetics can be computed with the same program which had been used for association kinetics. The program `kin1b.m` gives such an example, with the initial concentrations defined in lines `18`–`20`.

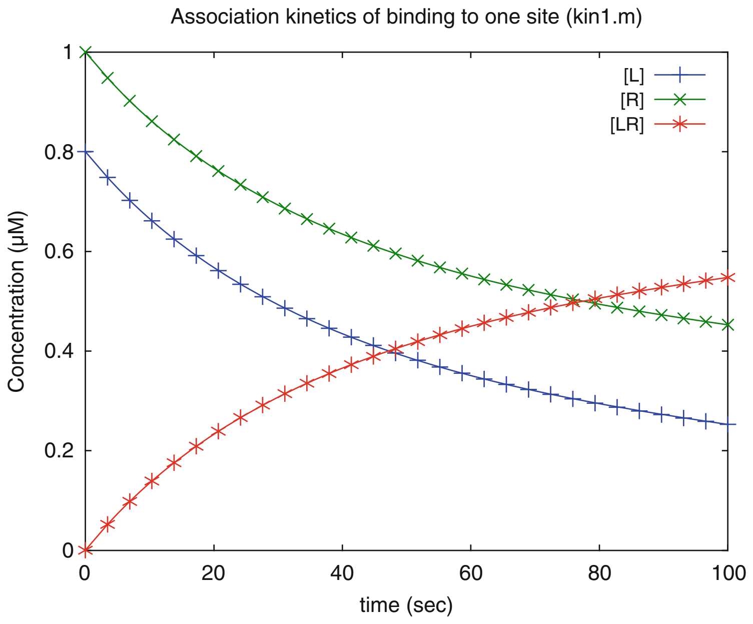


Fig. 6.1 Association kinetics of binding to one site. Reaction scheme (2.9) is calculated with $k_1 = 0.02\ \mu M^{-1}\ s^{-1}$ and $k_{-1} = 0.001\ s^{-1}$. The initial ligand and receptor concentrations are 0.8 and 1 µM, respectively. Free ligand L (+), free receptor R (x) and the complex LR (*) are shown as a function of time

```
filename='kin1b';
global k1 km1;
k1=0.02;
km1=0.01;
L1=0;
R1=0.1;
LR1=1;
tmax=100;
N=30;
t=linspace(0, tmax, N);
x1=[L1;R1;LR1];
%% Main Program
M=lsode('kin1F',x1,t);
L=M(:,1);
R=M(:,2);
LR=M(:,3);
%% Plot
plot(t,L,'-+',t,R,'-x',t,LR,'-*');
```

The resulting plot is shown in Fig. 6.2. It corresponds to a hypothetical dissociation experiment from a complex LR1 = 1.0 μM, with free ligand concentration L1 = 0 and free receptor concentration R1 = 0.1 μM. Kinetic experiments are calculated from initial concentrations, and the initial concentrations of all reaction partners have to be defined. This contrasts to the equilibrium binding studies of Chap. 5, where all equilibrium concentrations of all complexes are computed only from the total concentrations. The total concentrations in `kin1b.m` are L0 = L1 + LR1 = 1.0 μM and R0 = R1 + LR1 = 1.1 μM.

Figure 6.2 shows the increase in free ligand (+) and free receptor (x) concentrations together with the expected concentration decrease of the complex (*). The relatively large amplitude results from the assumption that the free ligand concentration is zero at the beginning of the reaction. This is an exercise to demonstrate that the same differential equations (`kin1F`) are used to calculate association (Fig. 6.1) and dissociation (Fig. 6.2) alike. It does not correspond to a real dissociation experiment, where the initial concentrations would have to be computed from an initial equilibrium.

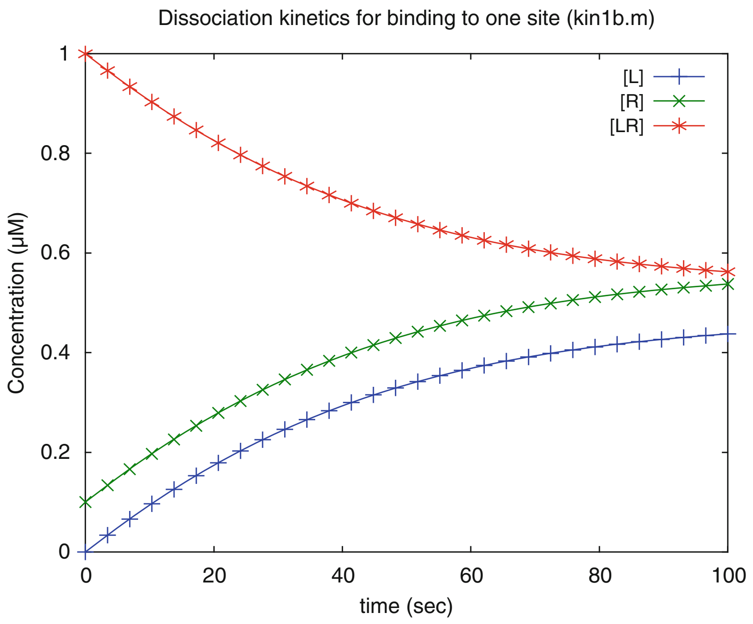


Fig. 6.2 Dissociation kinetics of binding to one site. Reaction scheme (2.9) is calculated with $k_1 = 0.02\ \mu M^{-1}\ s^{-1}$ and $k_{-1} = 0.01\ s^{-1}$. The initial complex and receptor concentrations are 1 and 0.1 μM, respectively. The ligand concentration at zero time is zero

6.3 Binding to One Site Followed by a Conformational Change of the Receptor (**kin2.m**)

In most cases, binding of a ligand will lead to conformational changes of the target protein. Such conformational changes cannot be detected in equilibrium binding studies, but often are detected in kinetic experiments. In the simplest case, they follow reaction scheme (6.10):

$$\mathrm{L + R} \underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} \mathrm{LR} \underset{k_{-2}}{\overset{k_2}{\rightleftharpoons}} \mathrm{LR}^* \qquad (6.10)$$

Scheme (6.10) can be translated into the four differential equations (6.11)–(6.14). Four different components of the reactions (L, R, LR and LR*) are involved, so that four differential concentration changes have to be computed.

$$\mathrm{d[L]/dt} = -k_1 \cdot [\mathrm{L}] \cdot [\mathrm{R}] + k_{-1} \cdot [\mathrm{LR}] \qquad (6.11)$$

$$\mathrm{d[R]/dt} = -k_1 \cdot [\mathrm{L}] \cdot [\mathrm{R}] + k_{-1} \cdot [\mathrm{LR}] \qquad (6.12)$$

$$\mathrm{d[LR]/dt} = k_1 \cdot [\mathrm{L}] \cdot [\mathrm{R}] - k_{-1} \cdot [\mathrm{LR}] - k_2 \cdot [\mathrm{LR}] + k_{-2} \cdot [\mathrm{LR}^*] \qquad (6.13)$$

$$\mathrm{d[LR*]/dt} = k_2 \cdot [\mathrm{LR}] - k_{-2} \cdot [\mathrm{LR}^*] \qquad (6.14)$$

These equations are solved with the octave program kin2.m. The function kin2F.m contains the differential equations (6.11)–(6.14). The function has the same name as the main program, completed with the letter F. The complex LR* is named LRS (LR star) in octave code because special characters cannot be used in variable names. The function kin2F.m uses L, R, LR and LRS in lines 9–12 for the sake of clarity. These variable names have to be translated from the vector components x(i) in lines 5–8. Likewise, the computed concentration changes have to be translated into the vector notation dx(i) in lines 13–16. This ensures together with line 4 that the output of the function kin2F.m is a column vector of four elements.

```
%% One site, conformational change
function dx=kin2F(x,t)
global k1 km1 k2 km2;
dx=zeros(4,1);
L=x(1);
R=x(2);
LR=x(3);
LRS=x(4);
dL=-k1*L*R+km1*LR;
dR=-k1*L*R+km1*LR;
dLR=k1*L*R-km1*LR-k2*LR+km2*LRS;
dLRS=k2*LR-km2*LRS;
dx(1)=dL;
dx(2)=dR;
dx(3)=dLR;
dx(4)=dLRS;
```

Association kinetics is calculated from `kin2F.m` when the initial free concentrations `R1` and `L1` are set equal to the total concentrations `R0` and `R0`. Then, the main characteristics of the conformational change LR → LR* become obvious in Fig. 6.3. The final product [LR*] is formed only after the initial binding

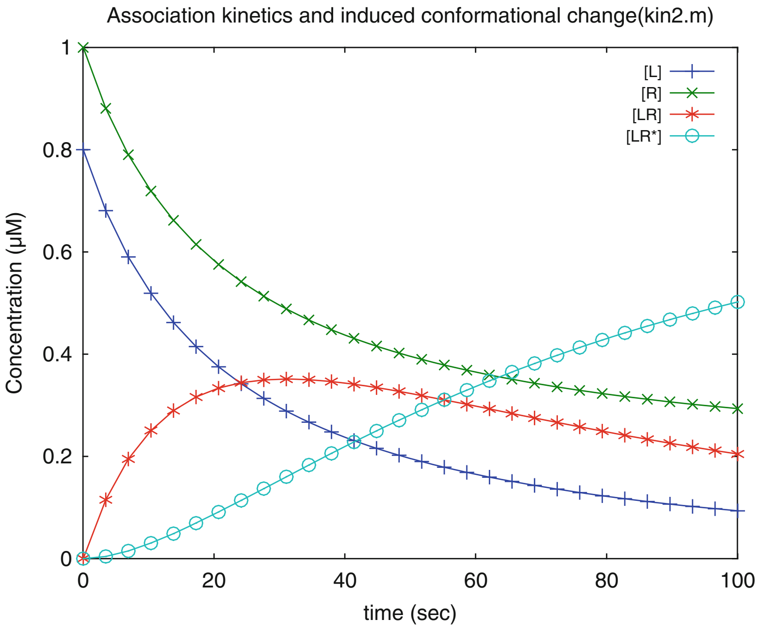


Fig. 6.3 Association kinetics and induced conformational change. Reaction scheme (6.10) is calculated with $k_1 = 0.05\ \mu M^{-1}\ s^{-1}$, $k_{-1} = 0.001\ s^{-1}$, $k_2 = 0.02\ s^{-1}$ and $k_{-2} = 0.002\ s^{-1}$. The initial ligand and receptor concentrations are 0.8 and 1 µM, respectively

process. Its concentration as a function of time (o) in Fig. 6.3 follows a sigmoid curve. This time phase is called the "lag-phase", because the formation of LR* lags behind the decrease of L and R. Transient complexes such as [LR] sometimes can only be inferred from such a lag-phase of the final product.

Figure 6.3 shows that the complex LR is formed transiently. Its decrease correlates with the increase of LR*, the final product. The decrease of L and R resembles exponential decay, but analyzing this as a sum of exponentials would be deceptive.

How to modify the sample program. Rather than choosing arbitrary concentrations for the calculation of a hypothetical dissociation experiment, we will now calculate dissociation observed after dilution. The initial concentrations given in lines `20`–`23` and the rate constants in lines `16`–`19` therefore are not modified. Instead, the association reaction of Fig. 6.3 is calculated until equilibrium is reached at `tmax` = `10,000` s (line `24`). After that, dissociation is initiated by diluting the solution by a factor 100. For this, all concentrations have to be divided by the same factor, as shown in lines `30`–`33`. These diluted concentrations are the initial concentrations of the dissociation kinetics. The modified program is called `kin2.m`.

```
14    filename='kin2b';
15    global k1 km1 k2 km2;
16    k1=0.05;
17    km1=0.001;
18    k2=0.02;
19    km2=0.002;
20    L1=0.8;
21    R1=1;
22    LR1=0;
23    LRS1=0;
24    tmax=10000;
25    N=30;
26    t=linspace(0, tmax, N);
27    x1=[L1;R1;LR1;LRS1];
28    %% Main Program
29    M=lsode('kin2F',x1,t);
30    L2=M(N,1)/100;
31    R2=M(N,2)/100;
32    LR2=M(N,3)/100;
33    LRS2=M(N,4)/100;
34    x2=[L2;R2;LR2;LRS2];
35    M=lsode('kin2F',x2,t);
36    L=M(:,1);
37    R=M(:,2);
38    LR=M(:,3);
39    LRS=M(:,4);
40    %% Plot
41    plot(t,L,'-+',t,R,'-x',t,LR,'-*',t,LRS,'-o');
```

The last time point t(N) of the association reaction is used for the calculation of the initial concentrations of the dilution experiment, and the respective equilibrium concentrations are `L2` = `M(N,1)`, `R2` = `M(N,2)`, `LR2` = `M(N,3)` and `LRS2` = `M(N,4)`. Check it by typing M in the Octave terminal window. The vector `x2` consists of these concentrations calculated at the end of the association reactions. It is used for the initial concentrations of the dissociation kinetics in line `35`.

Alternatively, one could calculate the binding equilibria with `fsolve` from a set of nonlinear equations as shown in Chap. 5, and then take these values as the initial concentrations of the dilution kinetics. Both methods are equivalent, provided that a real equilibrium is reached. But one should note that for most experiments the incubation times are known, even when it is not clear if the equilibrium has been reached. Therefore, calculating kinetics for a defined long time (like in `kin2b.m`) usually corresponds to the experimental set-up.

Figure 6.4 shows dissociation kinetics induced by dilution. It corresponds to a shift from one equilibrium to another. The new equilibrium has 100-fold lower concentrations for all components of the reaction, so that decrease of LR* (o) is expected. Of course, the increase in free receptor (x) and ligand (+) concentration in Fig. 6.3 is calculated **after** the initial 100-fold dilution step. The transient concentration change observed for LR is not visible any more, since the equilibrium between LR and LR* is exactly K_D2, for the old and for the new equilibrium

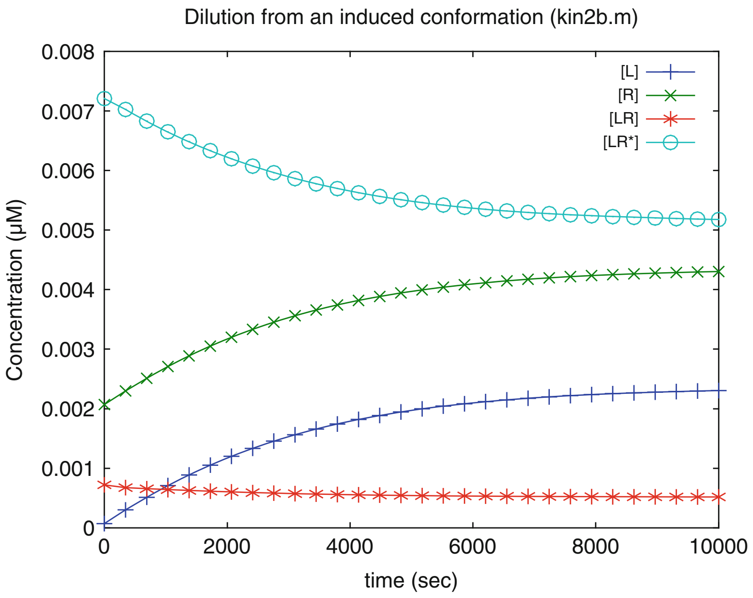


Fig. 6.4 Dilution from an induced conformation. Reaction scheme (6.10) is calculated with $k_1 = 0.05\ \mu M^{-1}\ s^{-1}$, $k_{-1} = 0.001\ s^{-1}$, $k_2 = 0.02\ s^{-1}$ and $k_{-2} = 0.002\ s^{-1}$. The initial ligand and receptor concentrations had been 0.8 and 1 µM, respectively. The reactions initially were calculated to proceed for 10,000 s. After that time, all concentrations were diluted by a factor of 100 and calculated from then on as shown

alike. This can be verified by typing `LRS./LR` in the octave or in the MATLAB command window after `kin2b` has run. Comparing Figs. 6.4 and 6.3 reveals that the dissociation kinetics is not simply the reverse association kinetics.

6.4 Dissociation Kinetics: Chase with Inhibitor (**`kin3.m`**)

Dilution experiments are easy to perform, but the results often are noisy. Dilution typically leads to a decrease of signal by the dilution factor, so that one may wish to study the reaction with a high dilution factor. But high dilution results in low signal and bad signal to noise ratio. Low dilution results in high signal but small signal change. This dilemma can be avoided when a competing ligand is added in large excess. For a reversible reaction (2.9) large excess of inhibitor removes all free receptor, so that only a first order dissociation of the bound ligand LR with the rate constant k_{-1} should be observed. This type of "chase" experiments can be calculated from scheme (6.15)

$$\begin{tikzcd} R \arrow[r, shift left, "+L\quad k_1"] \arrow[d, shift left, "+I\ k_{i1}"] & LR \arrow[l, shift left, "k_{-1}"] \\ IR \arrow[u, shift left, "k_{-i1}"] & \end{tikzcd} \tag{6.15}$$

Written in Octave code, the differential equations for competitive binding to one site (6.15) are part of the function `kin3F.m`:

```
global k1 km1 k2 km2;
dL=-k1*L*R+km1*LR;
dR=-k1*L*R+km1*LR-ki1*I*R+kim1*IR;
dLR=k1*L*R-km1*LR;
dI=-ki1*I*R+kim1*IR;
dIR=ki1*I*R-kim1*IR;
```

For calculating a chase experiment, one has to calculate the initial equilibrium of ligand and receptor (2.9) first. For the program `kin3.m`, this is done with the function `EQ1F.m` described in Sect. 5.2. The dissociation kinetics is initiated by the addition of inhibitor. The decrease of [LR] (bound ligand) is shown in Fig. 6.5 for different inhibitor concentrations. Lines `15–45` of `kin3.m` are listed below. The first part up to line `27` defines the parameters. Note line `15`: It defines `global` parameters for two functions: The function `EQ1F.m`, which is used to calculate the binding equilibrium, only needs `KD1, R0 and L0`. For the differential equations defined in `kin3F.m` the additional variables `k1, km1, ki1` and `kim1` are listed in the `global` statement of line `15`. This illustrates that any variable which is listed in the `global` statement of the main program can be used in the `global` statement of any function, independent of the order in which it appears.

```
15    global KD1 k1 km1 ki1 kim1 R0 L0;
16    R0=1;
17    L0=100;
18    KD1=10;
19    k1=0.001;
20    KI1=100;
21    ki1=k1;
22    km1=KD1*k1;
23    kim1=KI1*ki1;
24    tmax=100;
25    N=30;
26    t=linspace(0, tmax, N);
27    I00=linspace(100, 1000, 4);
28    %% Main Program
29    x0(1)=0.5*R0;
30    x0(2)=0.5*L0;
31    x = fsolve('EQ1F',x0);
32    R1=x(1);
33    L1=x(2);
34    LR1=L1*R1/KD1;
35    IR1=0;
36    for k=1:4
37    I1=I00(k);
38    x1=[L1;R1;LR1;I1;IR1];
39    M=lsode('kin3F',x1,t);
40    LR(:,k)=M(:,3);
41    end;
42    LR1=LR(:,1);
43    LR2=LR(:,2);
44    LR3=LR(:,3);
45    LR4=LR(:,4);
```

Lines `23 and 24` illustrate an important point which will be addressed in more detail in Sect. 8.1: Differential equations for reaction schemes are calculated from forward and backward rate constants. However, these rate constants usually are correlated when used for data fitting. The most important parameter is their quotient, either the equilibrium dissociation constant $K_D = k_-/k_+$ or its reciprocal value, the equilibrium constant. In life sciences, the equilibrium dissociation constant is more common, since it immediately gives the concentration of half-maximal saturation at equilibrium conditions. Therefore, the K_D values and association rate constants are taken as parameters, and the dissociation rate constants are calculated from these in lines `22` and `23`. Likewise, competing ligands may have similar association rates to the ligands themselves. Equal association rate constants

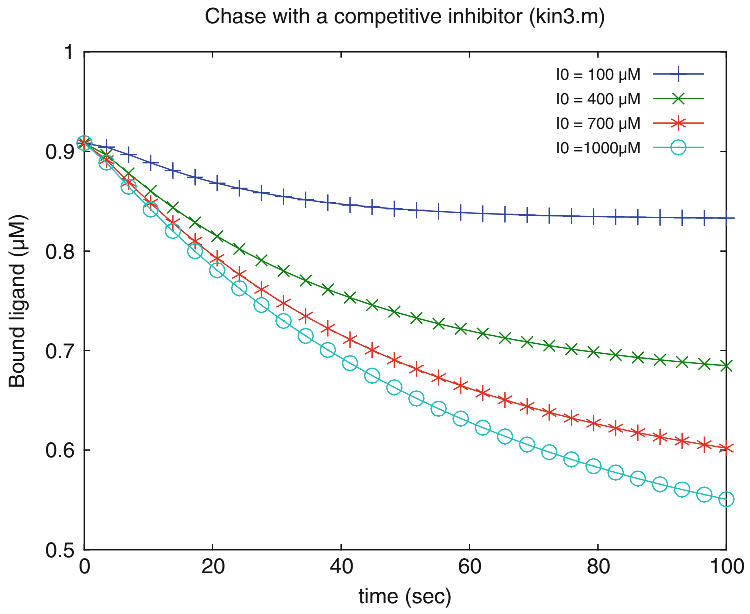


Fig. 6.5 Chase with a competitive inhibitor. Reaction scheme (6.15) is calculated with KD1 = 10 μM, KI1 = 100 μM, $k_1 = k_{i1} = 0.001\ \mu M^{-1}\ s^{-1}$. The initial concentrations are L0 = 100 μM, R0 = 1 μM. The inhibitor concentrations are 100 (+), 400 (x), 700 (*) and 1,000 μM (o)

are stated line 21. Changes in the numerical value for k1 (defined in line 19) will then also affect ki1.

The initial equilibrium is calculated with fsolve in line 31 from total ligand and receptor concentrations L0 and R0. From this the initial free and bound ligand and receptor concentrations L1, R1, LR1 are derived in lines 32–34. The initial concentration of bound inhibitor IR1 is zero (line 35). Four total inhibitor concentrations are defined with the help of linspace(100, 1000, 4) in line 27. Each of these is used as an initial free (=total) inhibitor concentration I1 in line 37 within the for loop (lines 36–41). The subsequent concentration changes are computed from the differential equations with the function lsode in line 39. The concentration of the bound ligand [LR] as a function of time is given in the third column of the result matrix M. For each k within the loop, a column vector for [LR] is added to the matrix LR. The result is a matrix LR with four columns for each total inhibitor concentration. Type LR in the octave window and you can see its structure. Lines 42–45 (after the loop) compute four concentration arrays (vectors) for the bound ligand [LR] at the four different inhibitor concentrations from the matrix M.

Figure 6.5 shows a decrease of bound ligand with time after the addition of inhibitor. Note that the final concentration of bound ligand decreases with increasing inhibitor concentration. This is an expected feature of competitive inhibition.

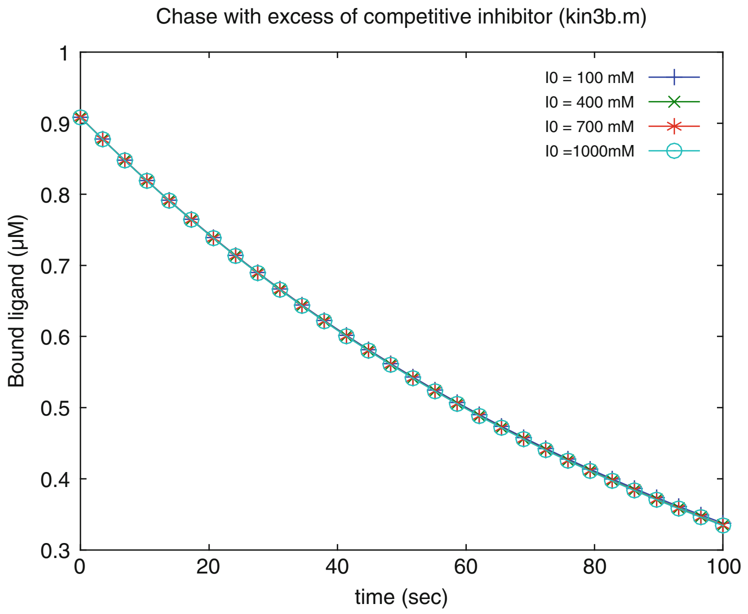


Fig. 6.6 Chase with excess of competitive inhibitor. Reaction scheme (6.15) is calculated with KD1 = 10 μM, KI1 = 100 μM, $k_1 = k_{i1} = 0.001\ \mu M^{-1}\ s^{-1}$. The initial concentrations are L0 = 100 μM, R0 = 1 μM. The inhibitor concentrations are 100 (+), 400 (x), 700 (*) and 1,000 mM (o)

How to modify the sample program. Again, it may be interesting to do the calculations with different inhibitor concentrations and/or different rate constants. Increasing the inhibitor concentrations by three orders of magnitude gives the dissociation kinetics of Fig. 6.6. The rate determining step at high inhibitor concentrations is the first order dissociation from [LR] to [L] + [R].

Figure 6.6 shows the dissociation kinetics after addition of high excess (100 mM to 1 M) of inhibitor. All these curves are identical. The final value of [LR] at infinite times will approach zero, since all receptor will be bound to inhibitor and form the complex IR. This type of reaction is expected, and the rate constant of the resulting first order reaction is k_{-1}.

6.5 Facilitated Dissociation (`kin4.m`)

Inhibitors which only bind to the inhibitor binding site (6.15) are called competitive inhibitors. In many cases, inhibitors may bind to entirely different or to overlapping sites. Independent of the structural model, a ternary complex LRI of ligand, receptor and inhibitor may be formed according to reaction scheme (6.16)

$$\begin{tikzcd}
L + R \arrow[r, shift left, "k_1"] \arrow[d, shift right, "+I"] & LR \arrow[l, shift left, "k_{-1}"] \arrow[d, shift right, "+I"] \\
L + RI \arrow[u, shift right, "k_{-i1}"] \arrow[r, shift left, "k_2"] & LRI \arrow[l, shift left, "k_{-2}"] \arrow[u, shift right, "k_{-i2}"]
\end{tikzcd} \tag{6.16}$$

Note that reaction scheme (6.16) is similar (5.18), but that the rate constants for ligand and inhibitor are systematically differentiated. Reaction scheme (6.16) shows a closed loop, so that the ternary complex LRI may be formed by different pathways. Its energy must be the same, so one of the four equilibrium dissociation constants can be calculated from the three others in (6.17)

$$K_D1 \cdot K_I2 = K_D2 \cdot K_I1 \tag{6.17}$$

The differential equations for scheme (6.16) can be written directly in Octave code

```
dL=-k1*L*R+km1*LR-k2*L*IR+km2*LRI;
dR=-k1*L*R+km1*LR-ki1*I*R+kim1*IR;
dLR=k1*L*R-km1*LR-ki2*I*LR+kim2*LRI;
dI=-ki1*I*R+kim1*IR-ki2*I*LR+kim2*LRI;
dIR=ki1*I*R-kim1*IR-k2*L*IR+km2*LRI;
dLRI=k2*L*IR-km2*LRI+ki2*I*LR-kim2*LRI;
```

When a chase experiment (dissociation after addition of inhibitor) is calculated from these equations, the initial concentrations of all components of the reaction must be known. The initial equilibrium is calculated from scheme (6.16) without inhibitor, and this corresponds to simple reversible binding (2.9), so that the function `EQ1F.m` can be used as in `kin3.m`.

For comparison, `kin4.m` uses the same rate constants and concentrations as `kin3.m`. For the additional reactions involving the ternary complex LRI, we can choose three parameters: The two forward rate constants k_2 and k_{i2} and one equilibrium dissociation constants K_D2 or K_I2 [note (6.17)]. If we assume that all forward rate constants are the same, this leaves the affinity of the ternary complex as the only significant parameter. For overlapping sites, this affinity should be several orders of magnitude lower than the affinity of the corresponding binary complex. The calculations performed in `kin4.m` assume that this difference is three orders of magnitude.

Compared to competitive inhibition (Figs. 6.5 and 6.6), Fig. 6.7 shows a dramatic increase in dissociation rates. It reflects the opening of a new dissociation pathway for L from LR to LRI to IR and has been named "facilitated dissociation" [2]. Note that not only the amplitudes, but also the curvature of the dissociation curves increase with the inhibitor concentration. Facilitated dissociation as

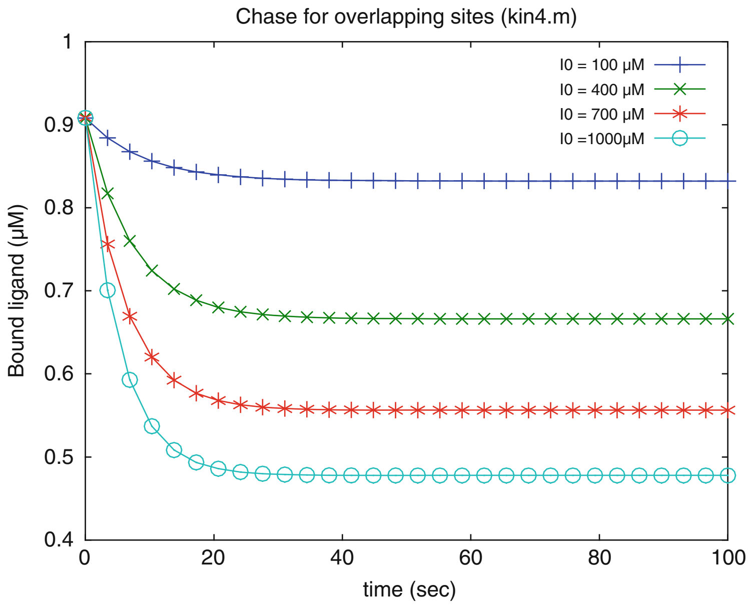


Fig. 6.7 Chase for overlapping sites. Reaction scheme (6.16) is calculated with KD1 = 10 µM, KI1 = 100 µM, KI2 = 100 mM, $k_1 = k_{i1} = k_2 = k_{i2} = 0.001\ \mu M^{-1}\ s^{-1}$. The initial concentrations are L0 = 100 µM, R0 = 1 µM. The inhibitor concentrations are I0 = 100 (+), 300 (x), 700 (*) and 1,000 (o) µM. The ternary complex LRI has a low affinity, which may be explained by overlapping sites

calculated here is has been reported before [3, 4]. It can be expected when the ternary complex has a very low affinity. This may either result from overlapping sites [3] or from a specific molecular mechanism involving considerable conformational changes [4]. The limiting rate constant at extremely high inhibitor concentrations is k_{-2}, for the dissociation of LRI to RI. Facilitated dissociation may escape our notice, because it corresponds to "common sense". We seem to expect that a ligand dissociates faster the more competitors are added, but of course first order dissociation on its own cannot be faster than $[LR]\cdot e^{-k\cdot t}$, the first order dissociation of the ligand shown in Fig. 6.6. Facilitation of this reaction by an inhibitor requires at least a ternary complex.

How to modify the sample program. It may be interesting to calculate the dissociation over a wider concentration range of inhibitor or for different inhibitor affinities, but most importantly for different affinities of the ternary complex. This easily is done by modifying lines `16–27` of `kin4.m`. The results can be understood with the help of reaction scheme (6.16): The formation of LRI from IR is a second order reaction and its rate increases with the inhibitor concentration. The dissociation of LRI to IR is a first order reaction. Therefore, the rate-limiting step for facilitated dissociation is k_{-2}, which is related to the affinity of the ternary complex LRI.

6.6 Sequential Binding and Obstructed Dissociation (`kin5.m`)

When ligands bind to a receptor, one usually assumes that the sites are randomly accessible like in reaction scheme (2.19). This need not be the case. For example, when two sites are located inside a pit or a channel, the ligand which binds first may not be able to dissociate before the second ligand has been released [reaction scheme (2.18)]. Equilibrium measurements cannot distinguish between random and sequential mechanisms, since only the numerical values of the apparent dissociation constants would change in accordance with (2.30). Kinetic measurements can, however, detect such a sequential mechanism when a chase experiment is performed [5]. In the presence of inhibitor, the reaction scheme of a sequential mechanism (2.18) has to be extended as shown in reaction scheme (6.18).

$$\begin{tikzcd}
 & R \arrow[r, shift left, "+L\quad k_1"] \arrow[d, shift left, "+I\quad k_{i1}"] & LR \arrow[l, shift left, "k_{-1}"] \arrow[r, shift left, "+L\quad k_2"] \arrow[d, shift left, "+I\quad k_{i3}"] & LLR \arrow[l, shift left, "k_{-2}"] \\
LIR \arrow[r, shift left, "k_{-3}"'] & IR \arrow[u, shift left, "k_{-i1}"] \arrow[l, shift left, "k_3\quad +L"'] \arrow[d, shift left, "+I\quad k_{i2}"] & ILR \arrow[u, shift left, "k_{-i3}"] & \\
 & IIR \arrow[u, shift left, "k_{-i2}"] & &
\end{tikzcd} \tag{6.18}$$

Note that there is only one complex `LR` in reaction scheme (6.18) and that the ternary complex is denoted as `LLR` in order to account for the order of the bound ligand. An inhibitor which follows the same binding scheme may also form the complexes `IR` and `IIR`. There are two markedly different ternary complexes formed with ligand, inhibitor and receptor. From the complex `ILR` the ligand can only dissociate once the inhibitor has been released, whereas the ligand has to dissociate first from the complex `LIR`. Some of the differential equations were getting rather long, so that four lines were cut with the help of the ... (three dots) continuation marker.

```
function dx=kin5F(x,t)
global k1 km1 ki1 kim1 k2 km2 ki2 kim2...
 k3 km3 ki3 kim3 ;

dL=-k1*L*R+km1*LR-k2*L*LR+km2*LLR-...
k3*L*IR+km3*LIR;
dR=-k1*L*R+km1*LR-ki1*I*R+kim1*IR;
dLR=k1*L*R-km1*LR-k2*L*LR+km2*LLR-...
ki3*I*LR+kim3*ILR;
dI=-ki1*I*R+kim1*IR-ki2*I*IR+kim2*IIR-...
ki3*I*LR+kim3*ILR;
dIR=ki1*I*R-kim1*IR-ki2*I*IR+kim2*IIR-...
k3*L*IR+km3*LIR;
dLLR=k2*L*LR-km2*LLR;
dIIR=ki2*I*IR-kim2*IIR;
dILR=ki3*I*LR-kim3*ILR;
dLIR=k3*L*IR-km3*LIR;
```

Dissociation kinetics can only be calculated from these equations when the correct initial concentrations are known. The initial concentrations `L1, R1, LR1` and `LLR1` are calculated in lines `43–47` from binding equilibria with `fsolve` and the function `kin5EQF.m`. The initial concentrations of complexes with inhibitor (lines `48–51`) are, of course, zero before the inhibitor is added.

```
function F = kin5EQF(x)
global KD1 KD2 R0 L0;
R=x(1); L=x(2);
F(1)=R+L*R/KD1+L*L*R/(KD1*KD2)-R0;
F(2)=L+L*R/KD1+2*L*L*R/(KD1*KD2)-L0;
```

The dissociation kinetics was then calculated within a loop (lines `52–66`) for three different inhibitor concentrations. The total bound ligand is calculated in line `65`, as a matrix `CB` of column vectors for the different inhibitor concentrations `k`. The result can be seen immediately if one types `plot(t,CB)` in the Octave or MATLAB command windows. The three vectors for the three concentrations are displayed as a function of time. The more complex plot command in line `71` is used to introduce the symbols, so that the three inhibitor concentrations can be distinguished when Fig. 6.8 is printed in black and white.

```
filename='kin5';
global KD1 KD2 k1 km1 ki1 kim1 k2 km2 ki2 kim2 k3 km3 ki3 kim3 R0 L0;

tmax=100;
N=30;
t=linspace(0, tmax, N);
I00=logspace(3, 5, 3);
%% Main Program
x0(1)=0.5*R0;
x0(2)=0.5*L0;
x = fsolve('kin5EQF',x0);
R1=x(1);
L1=x(2);
LR1=L1*R1/KD1;
LLR1=LR1*L1/KD2;
IR1=0;
IIR1=0;
ILR1=0;
LIR1=0;
for k=1:3
I1=I00(k);
x1=[L1;R1;LR1;I1;IR1;LLR1;IIR1;ILR1;LIR1];
M=lsode('kin5F',x1,t);
L=M(:,1);
R=M(:,2);
LR=M(:,3);
I=M(:,4);
IR=M(:,5);
LLR=M(:,6);
IIR=M(:,7);
ILR=M(:,8);
LIR=M(:,9);
CB(:,k)=LR+2*LLR+ILR+LIR;
end;
CB1=CB(:,1);
CB2=CB(:,2);
CB3=CB(:,3);
%% Plot
plot(t,CB1,'-+',t,CB2,'-x',t,CB3,'-*');
```

For MATLAB, the solver `ode45` becomes very slow when applied to reaction scheme (6.18). The reason lies in multiple fast reversible reactions which underlie the slow obstruction mechanism. Such a problem is commonly called “stiff” and MATLAB has a large selection of solvers which deal with stiff differential equations. The names of these MATLAB solvers end in “s” (for stiff), such as `ode15s` or `ode23s`. The MATLAB version `kin5M.m` of the program `kin5.m`

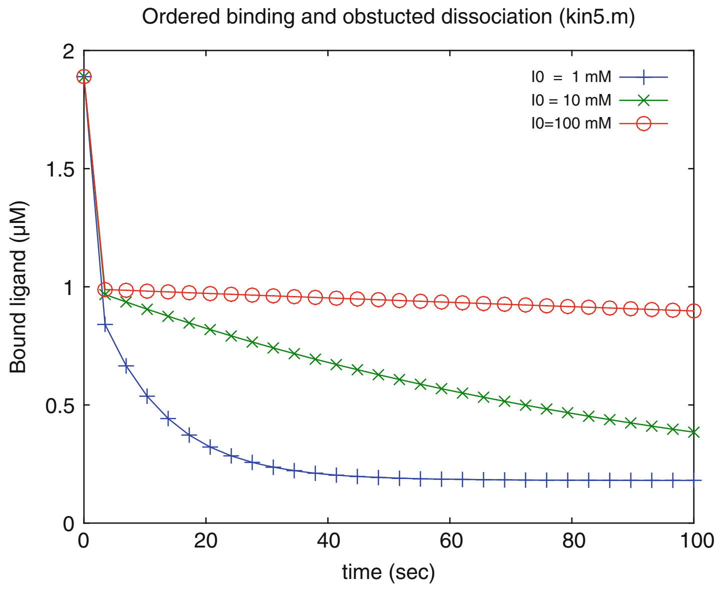


Fig. 6.8 Ordered binding and obstructed dissociation. Reaction scheme (6.18) (sequential binding of ligand and inhibitor) is calculated with $K_D1 = K_D2 = K_D3 = K_I1 = K_I2 = K_I3 = 10$ µM, $k_1 = k_{i1} = k_2 = k_{i2} = 1$ µM^{-1} s^{-1}. The initial concentrations are L0 = 100 µM, R0 = 1 µM. The concentrations of the chasing ligand are 1,000 (+), 10,000 (x) and 100,000 (*) µM

therefore employs the solver `ode23s` instead of `ode45`, which had been used in the previous programs. The Octave function `lsode` used in line `55` is quite capable to handle these types of stiff differential equations.

The resulting dissociation curve (Fig. 6.8) is clearly biphasic. In the first part, the ligand bound to the second site rapidly decreases. In molecular terms, the ligand L dissociates from LLR to LR. All off-rates were assumed to be 10 s^{-1}, which is three orders of magnitude faster than the off-rates assumed for the calculations in Figs. 6.4–6.7. The fast off-rates were chosen because the second part of the dissociation kinetics reveals a unique property of this mechanism: The dissociation kinetics become slower the more inhibitor is added. The inhibitor binds to LR and forms ILR. The sequential mechanism does not allow the dissociation of L from ILR. I have to dissociate first, leading to LR, which again may react rapidly with I. Thereby L is trapped. Such a sequential mechanism had been detected from this type of experiment for the nicotinic acetylcholine receptor from *electrophorus electricus* [5].

How to modify the sample program. The fast phase of the biphasic dissociation in Fig. 6.8 is not resolved. A shorter time range is set with the command `tmax = 1`; in line `36`. The axes are chosen by the plot routine, when the `axis` command in line `75` is defined as a comment with the help of `%`, the percent character.

Figure 6.9 shows that for high inhibitor concentrations the initial phase of the dissociation reaches a maximum velocity. This is due to the first order dissociation

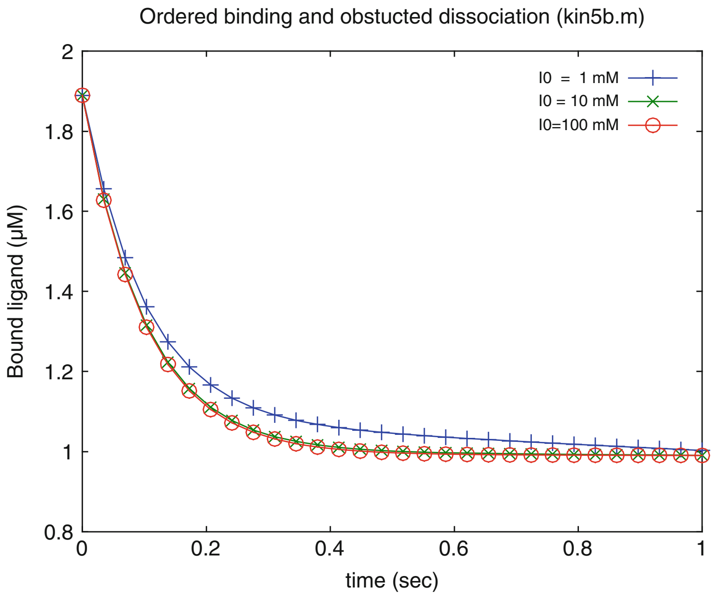


Fig. 6.9 Ordered binding and obstructed dissociation monitored in the sub-second range. Reaction scheme (6.18) is calculated with $K_D1 = K_D2 = K_D3 = K_I1 = K_I2 = K_I3 = 10$ µM, $k_1 = k_{i1} = k_2 = k_{i2} = 1$ μM^{-1} s^{-1}. The initial concentrations are L0 = 100 µM, R0 = 1 µM. The concentrations of the chasing ligand are 1,000 (+), 10,000 (x) and 100,000 (*) µM

of L from LLR. Compare Figs. 6.9 to 6.6 in this respect. Fast kinetics, i.e. reactions which proceed in the sub-second time range, should be monitored with a stopped-flow apparatus. Otherwise they would escape detection.

6.7 Irreversible Inhibition (`kin6.m`)

The final example of this chapter deals with irreversible reactions. Most reactions in biochemistry are reversible, and even covalent bonds typically result from a series of enzyme catalyzed reversible reactions. Nevertheless, proteins contain quite a few reactive groups and reactive chemical compounds may bind irreversibly. The kinetics of irreversible reactions is quite easy to understand. It becomes interesting when irreversible binding leads to unexpected results. This section calculates the influence of irreversible inhibition on dose–response curves of drug screening programs.

Let us first discuss one simple chase experiment with an irreversible inhibitor. Assuming that the inhibitor binds to the ligand binding site, it can only bind to the free receptor R and not to the occupied receptor LR. The mechanism is shown in reaction scheme (6.19). It is the same as reaction scheme (6.15) with $k_{-i1} = 0$.

$$\begin{tikzcd} R \arrow[r, shift left, "+L\quad k_1"] \arrow[d, "+I"', "k_{i1}"] & LR \arrow[l, shift left, "k_{-1}"] \\ IR & \end{tikzcd} \qquad (6.19)$$

Therefore, the function `kin3F` is used for the calculation of reaction scheme (6.19) with `kim1` = `0` in line `22` of the main program `kin6.m`. The program `kin6.m` is very similar to `kin3.m`, in particular since the initial equilibrium for L and R can be calculated from a reversible reaction with `fsolve` and the function `EQ1F.m` in line `32`.

```
22    kim1=0;

27    t=linspace(0, tmax, N);
28    I00=logspace(0, 4, N2);
29    %% Main Program
30    x0(1)=0.5*R0;
31    x0(2)=0.5*L0;
32    x = fsolve('EQ1F',x0);
33    R1=x(1);
34    L1=x(2);
35    LR1=L1*R1/KD1;
36    IR1=0;
37    for k=1:N2
38    I1=I00(k);
39    x1=[L1;R1;LR1;I1;IR1];
40    M=lsode('kin3F',x1,t);
41    LR(:,k)=M(:,3);
42    end;
43    %%Plot
44    plot(t,LR,'-k');
```

Figure 6.10 shows the decrease of the concentration of bound ligand after the addition of different concentrations of inhibitor. The lowest concentration (1 μM) does not significantly reduce the concentration of bound ligand within 10,000 s, which correspond to almost 3 h. The highest concentration (10 mM) completely inhibits the receptor within 20 min (1,200 s). This is to be expected, since irreversible reactions are of the second order and their rate in (6.19) is proportional to the inhibitor concentration. At infinite times, all irreversible inhibitors at any concentration would lead to complete inhibition, of course.

Now, how will the concentration dependence of irreversible inhibition influence dose–response curves in drug screening campaigns? Drug screening procedures typically are high throughput processes which are fully automated: The drugs in

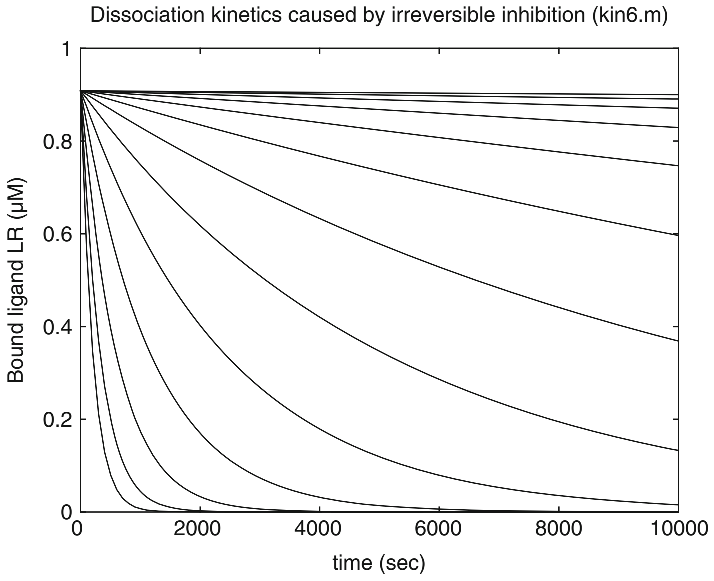


Fig. 6.10 Dissociation kinetics caused by irreversible inhibition. Reaction scheme (6.19) is calculated with KD1 = 10 µM, k_1 = 0.001 μM^{-1} s^{-1} and k_{i1} = 0.00001 μM^{-1} s^{-1}. The initial concentrations are L0 = 100 µM, R0 = 1 µM. Thirteen inhibitor concentrations (one solid line each) are logarithmically distributed ranging from 1 to 10,000 µM

question are incubated for a fixed time (in our example 10 min) with the drug target (typically an enzyme, but let us call it a receptor R in this chapter). Then the assay is started by adding substrate (or ligand L). The assay may run another 10 min and the activity (proportional to LR) is recorded after this time. Such a protocol can be reproduced with a computer program, and `kin6b.m` is a simple example for this. The core of the program `kin6b.m` is contained within two loops in lines `32–45`.

```
25    t=linspace(0,600,10);
26    I00=logspace(0, 4, N2);
27    %% Main Program
28    R1=R0;
29    L1=0;
30    LR1=0;
31    IR1=0;
32    for k=1:N2
33    I1=I00(k);
34    x1=[L1;R1;LR1;I1;IR1];
35    M=lsode('kin3F',x1,t);
36    x2(1,k)=L0;
37    x2(2,k)=M(10,2);
38    x2(3,k)=0;
39    x2(4,k)=M(10,4);
40    x2(5,k)=M(10,5);
41    end;
42    for k=1:N2
43    M=lsode('kin3F',x2(:,k),t);
44    LR(k)=M(10,3);
45    end;
46    %% plot
47    lI0=log10(I00);
48    Rev=0.91./(1+I00./200);
49    plot(lI0,abs(LR),'-ok',lI0,Rev,'-r');
```

The initial concentrations (incubation of receptor R with inhibitor I) are defined in lines `28`–`32`, with the exception of the initial inhibitor concentration `I1` which is set in line `33` within the loop. The loop calculates the irreversible binding of `I` to `R` within the incubation time of 10 min (600 s, as defined in line `25`). The differential equations are solved in line `35` and the resulting concentrations for `R`, `I` and `IR` are used as starting concentrations `R2` = `R`, `I2` = `I`, `IR2` = `IR` for the assay. Since these starting concentrations have been computed from the second time point of the Matrix M, the results of M are directly assigned to the second vector of initial parameters `x2` in lines `36`–`40`. Each of these `k` column vectors correspond to one inhibitor concentration `I00(k)`. The result of the actual binding assay is the concentration of complexes `LR(k)` for all inhibitor concentrations `I00(k)`, as given in line `44`. The corresponding reversible binding (one site, adapted R0 = 0.91, KD = 200 μM) is calculated from (3.1) in line 48. This reversible binding is included in Fig. 6.11 as a reference.

Figure 6.11 shows a dose–response curve for irreversible inhibition which is similar to the dose–response curves in Fig. 5.12, where the effect of multiple allosteric inhibition is calculated. Again, the dose–response curve is not symmetric and thus does not correspond exactly to a logistic curve. If the curve for irreversible inhibition was fitted to a logistic function, the resulting Hill coefficient would larger than one. As noted in Sect. 5.8, Hill coefficients larger than one are typical for high

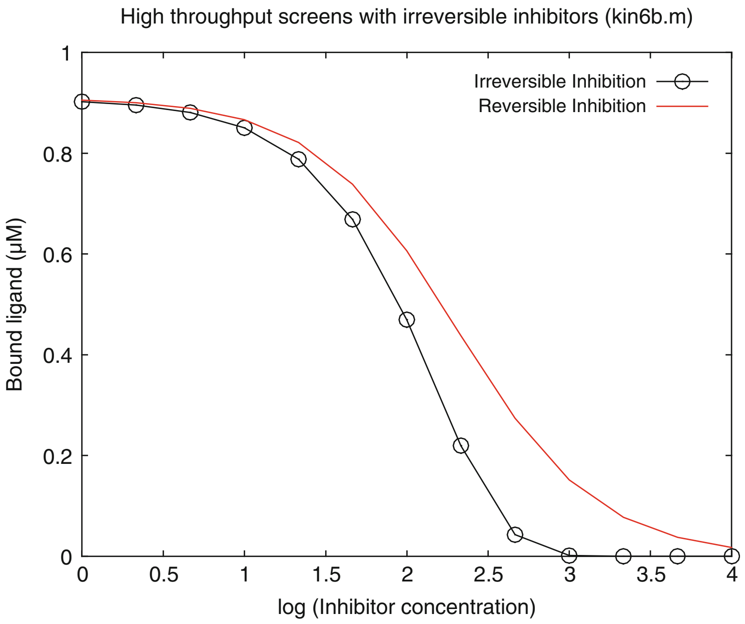


Fig. 6.11 High throughput screens with irreversible inhibitors. Reaction scheme (6.12) is calculated with KD1 = 10 μM, $k_1 = 0.001\ \mu M^{-1}\ s^{-1}$ and $k_{i1} = 0.00001\ \mu M^{-1}\ s^{-1}$. The initial concentrations are L0 = 100 μM, R0 = 1 μM. A typical screening protocol was calculated with inhibitor incubation time of 10 min followed by an assay of 10 min. Activity competition curve for an irreversible inhibitor (o) and reversible inhibition (−) is shown

throughput screening campaigns [6]. Therefore, one reason for larger Hill coefficients may lie in irreversible interactions of reactive compounds to reactive sites on a drug target. This is not unreasonable, since the compound libraries employed in screening campaigns contain quite a few chemically active compounds. Most likely there is more than one reactive group on a protein, so that a combination of multiple allosteric interactions in conjunction with irreversible inhibition may explain any Hill coefficient larger than one.

How to modify the sample program. The irreversible reaction of Fig. 6.11 can be modified to a reversible reaction simply by changing `kim1` in line `22`. It will become obvious that not only irreversible reactions, but also slow reversible reactions may lead to the type of dose–response curve shown in Fig. 6.11.

References

1. Hindmarsh AC (1983) ODEPACK, a systematized collection of ODE solvers. In: Stepleman RS et al (eds) IMACS transactions on scientific computation, vol 1. Scientific Computing, North-Holland, Amsterdam, pp 55–64
2. Gutfreund H (1995) Kinetics for the life sciences. Receptors, transmitters and catalysts. Cambridge University press, Cambridge

3. Prinz H, Striessnig J (1993) Ligand-induced accelerated dissociation of (+)-*cis*-diltiazem from L-type Ca^{2+} channels is simply explained by competition for individual attachment points. J Biol Chem 268:18580–18585
4. Klebe C, Prinz H, Wittinghofer A, Goody RS (1995) The kinetic mechanism of Ran-nucleotide exchange catalyzed by RCC1. Biochemistry 34:12543–12552
5. Prinz H, Maelicke A (1983) Interaction of cholinergic ligands with the purified acetylcholine receptor protein II. Kinetic studies. J Biol Chem 258:10273–10282
6. NIH Data base: http://www.ncbi.nlm.nih.gov/sites/entrez. Chem BioAssay

Chapter 7
Enzyme Kinetics

Any type of enzyme catalyzed reactions can be calculated from their differential equations, but steady-state equilibrium hides most intrinsic reactions. Mathematically, this problem corresponds to stiff differential equations. Empirically, initial velocities have been used to account for steady-state conditions. Competitive, noncompetitive, uncompetitive, and cooperative inhibition is calculated from initial velocities and steady-state equilibria. Substrate inhibition is calculated as progress curves from differential equations. At the end of this chapter, the reader should be able to calculate all feasible enzyme mechanisms and translate it to traditional interpretations.

7.1 Progress Curves (`enz1.m`)

Enzyme kinetics can be calculated from the differential equations of a reaction scheme, just like any other kinetic experiment. They differ from binding kinetics, because enzymes are biocalalysts which usually are present in small amounts relative to the concentrations of substrates and products. After an initial binding phase, a steady-state condition is established, in which a constant rate of substrate is turned into product. In most textbooks on enzyme kinetics, the steady state is treated as a transient equilibrium, for which analytical solutions can be calculated from Michaelis–Menten approximations (Sect. 3.1.4).

In the beginning of this chapter, we will calculate the full course of an enzymatic reaction with all its concentrations of complexes, substrate, and product. The graph which shows the concentration of product as a function of time is called a "progress curve." The initial part of this curve gives the initial velocity, which commonly is used to calculate Michaelis–Menten enzyme kinetics (3.6). The simplest enzyme mechanism describes the catalytic reaction of one substrate S to one product P. When substrate binds to an enzyme E, it will form a complex ES. The enzyme may then catalyze a reaction, which turns the substrate S into product P. Since this is catalyzed by the enzyme, the product will initially be bound to the enzyme as EP. Eventually it will dissociate to E and P:

$$S + E \underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} ES \underset{k_{-2}}{\overset{k_2}{\rightleftharpoons}} EP \underset{k_{-3}}{\overset{k_3}{\rightleftharpoons}} E + P \tag{7.1}$$

Reaction scheme (7.1) describes the simplest plausible scheme for a single substrate – single product transformation. Note that the third reaction is read from left to right and that the equilibrium dissociation constant for E and P therefore must be $K_D3 = k_3/k_{-3}$. Reaction scheme (7.1) is calculated in the subroutine function `enz1F.m.`

```
dE=-k1*E*S+km1*ES+k3*EP-km3*E*P;
dS=-k1*E*S+km1*ES;
dES=k1*E*S-km1*ES-k2*ES+km2*EP;
dEP=k2*ES-km2*EP-k3*EP+km3*E*P;
dP=k3*EP-km3*E*P;
```

These differential equations are readily solved with the Octave function `lsode` in line `41` of `enz1.m`. For MATLAB, one has to select a solver capable of solving stiff differential equations. These solvers have names ending with “s”. The solver selected in line `41` of `enz1M.m` is `ode23s`. The MATLAB universal solver which had been used for binding kinetics was `ode45`, which is too slow for steady-state kinetics and its stiff differential equations with fast on and off reactions k_1 and k_{-1}.

Figure 7.1 shows a series of progress curves calculated from reaction scheme (7.1) and different rate constants for the initial step k_1. The initial parts of the progress curves give the initial velocities for product increase or substrate decrease. With the exception of the two lowest rate constants, the initial velocities are the same within experimental error. Only initial velocities are taken as experimental data for Michaelis–Menten kinetics, and Fig. 7.1 confirms that these values indeed are safe to measure.

Figure 7.1 in its printed form is a black and white illustration, but when the program `enz1.m` is run in the octave window (or `enz1M.m` in MATLAB), two series of colored graphs are shown. Each value for k1 has the same color, be it for product formation (+) or substrate decrease (o). This is achieved with the variable `fill`, a useful programming feature, described in lines 49 and 50:

```
49    fill=zeros(N,14-N2);
50    plot(t,P,'-+',t,fill,t,S,'-o');
```

`P` and `S` are matrixes of `N` time points and `N2` different `k1` values. `t` is a vector of `N` time points. When `P` is plotted versus `t` without specifying a color, all `N2` different curves will be displayed each with different colors. The plot command has a repertoire of seven different colors for this. When the next set of data (`t,S`, in this case) is called in the plot function, the numbering of the colors is continued, so that after the first 8 colors, color 9, 10, 11..., identical to 2, 3, 4... are displayed. If one wants to have the same set of colors displayed for the second set of data, one has

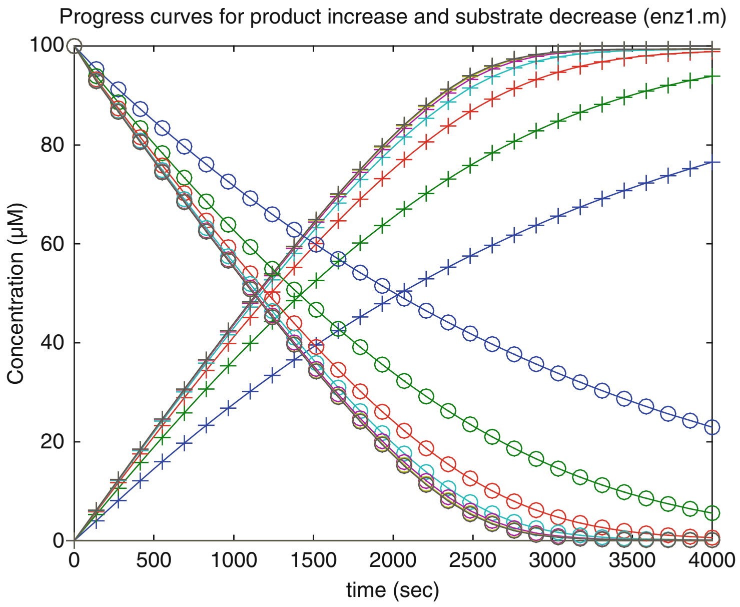


Fig. 7.1 Progress curves for product increase and substance decrease. Product increase (+) and substrate decrase (o) is calculated from reaction scheme (7.1) with various association rate constants. $K_D1 = K_D2 = 100$ µM, $k_2 = 1$ s^{-1}, $k_{-2} = 0.001$ s^{-1}, $k_3 = 0.005$ s^{-1}. The values for k1 are 0.001. 0.0032, 0.01, 0.032, 0.1, 0.32, and 1 μM^{-1} s^{-1}

to include a set of dummy data, which would fill colors 9, 10,. . .14 (a multiple of 7). Then, the next set is displayed beginning with color 1 again. `fill` in line `49` therefore defines a matrix of dummy data. The previous number of plotted curves N2 has to be subtracted from a multiple of 7 (14, in this case). This trick is particularly useful when theoretical curves are fitted to experimental data (Fig. 8.8).

How to modify the sample program. The equilibrium dissociation constants K_D1 and K_D3 in `enz1.m` were set to 100 µM, for substrate and product alike. Therefore, the final equilibrium of substrate and product is determined by K_D2, which is the same as k_{-1}/k_2. When this parameter in line 21 is set to 1, both, substrate and product concentrations will approach 50 µM at infinite times. Enzymes are catalysts, which will catalyze both, the transition from substrate to product and from product to substrate. Reaction scheme (7.1) accounts for it.

7.2 Progress Curves for a Weak Substrate (`enz1b.m`)

Most enzyme kinetics show progress curves similar to Fig. 7.1. We had performed enzyme kinetics with more than ten different protein phosphatases in a drug-screening campaign. As expected, most of the progress curves looked similar to

Fig. 7.1. In some cases, however, the progress curves looked different (Fig. 8.3). The endogenous substrates of protein phosphatases are phosphorylated proteins, which are almost impossible to obtain in a defined state of phosphorylation. p-nitrophenyl phosphate (pNPP) is a common substitute. Its affinity is orders of magnitude lower than the endogenous substrates, but its phenolate product can be monitored at 405 nm. For some phosphatases like CDC25, its affinity is extremely low with a K_m value around 100 mM. Moreover, its maximal reaction rate also was low, so that high enzyme concentrations were required in order to observe the reaction. In this case, the progress curves did not resemble Fig. 7.1 and instead exhibited a pronounced lag phase (Fig. 8.3).

Phosphatase catalyzes a one product, two substrates reaction. We will not use this enzyme as an additional example, but modify the rate constants of scheme (7.1), in order to understand under which conditions such a lag phase in an enzymatic progress curve may appear. One usually would apply a fitting routine such as described in Chap. 8 in order to find suitable sets of rate constants.

Instead, we will vary the parameters (rate constants and concentrations), until a reasonable explanation for this lag phase is found. Such a task is made easier when all relevant parameters are shown explicitly in the final graph, and the command `text (x,y,' label' ,p1,v1)` is used eight times in lines `51–66`. The first two numbers of `text` are x and y coordinates, where the written text should appear. `'label'` is a string of characters which is written at the x, y location. This can be followed by one or more property-value pairs (p, v). When the property value pair `p1 = 'units'` and `v1 = 'normalized'` is used, the x and y coordinates are both normalized to the range between 0 and 1.0. The string `' label'` may be written as a row vector of strings, for example: `['KD1 = ',num2str(KD1),' µM']`. The first and third vector elements are simple strings `' KD1 = '`and `' µM'`. The second element is an interesting function: `num2str(number)` converts any number to a string. Replacing `number` with a parameter, such as `KD1`, allows octave to convert it into a string and to show it in the current plot with the `text` command.

```
53    text(0.3,0.86,[' KD1 = ',num2str(KD1),' µM']...
54    ,'units','normalized');
```

Indeed, one can obtain an initial sigmoid increase of product formation [(+) in Fig. 7.2]. Of course, there may be different reaction schemes or different sets of parameters which give similar curves, but the message is clear: Substrates with an extremely small catalytic rate constant k_2 may need a relatively long time to reach steady state.

Figure 7.2 shows a lag phase correlated to the production of EP (x) from ES (o) in reaction scheme (7.1). Such observations cannot be explained with classical Michaelis–Menten assumptions (see Sect. 3.1.4). This example is presented here because initial velocity enzyme kinetics generally look simple, and their underlying assumptions are valid and allow for easy data analysis. But in some cases one has to go back and calculate the full enzymatic process without these assumptions.

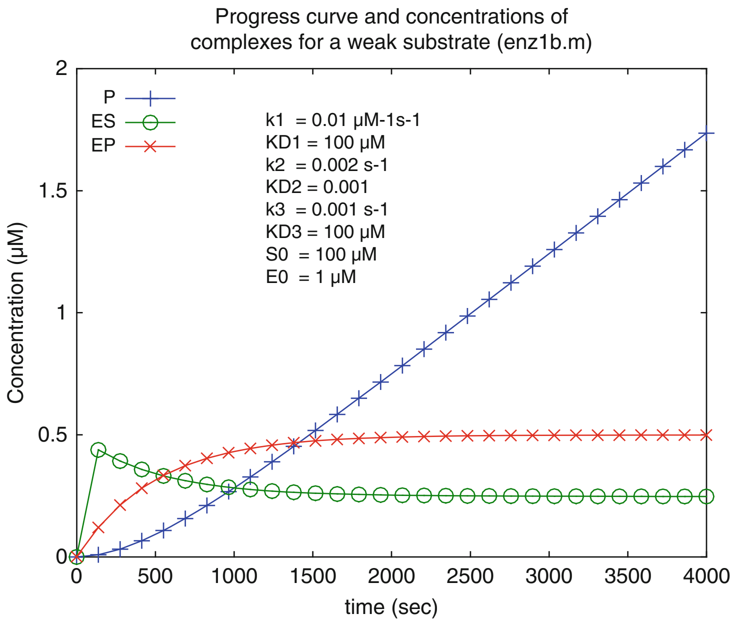


Fig. 7.2 Progress curve and concentrations of complexes for a weak substrate. Reaction scheme (7.1) is calculated with `enz1b.m` and the parameters given in the figure itself

How to modify the sample program. The program `kin1b.m` has been designed to vary all parameters of scheme (7.1). These parameters are entered in lines `18–30`. Do it!

7.3 Michaelis–Menten Enzyme Kinetics (`enz2.m`)

Michaelis–Menten kinetics is a reasonable simplification of reaction scheme (7.1). One assumes a rapid initial equilibrium between free and bound substrate. From bound substrate ES the product is released as a time-limiting (slowest) step. The first-order rate constant of product release is named k_{cat}.

$$S + E \underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} ES \xrightarrow{k_{cat}} E + P \tag{3.4}$$

Scheme (3.4) corresponds to Michaelis–Menten kinetics (section 3.1.4) and the differential equations are part of the subroutine function `enz2F.m`

```
dE=-k1*E*S+km1*ES+kc*ES;
dS=-k1*E*S+km1*ES;
dES=k1*E*S-km1*ES-kc*ES;
dP=kc*ES;
```

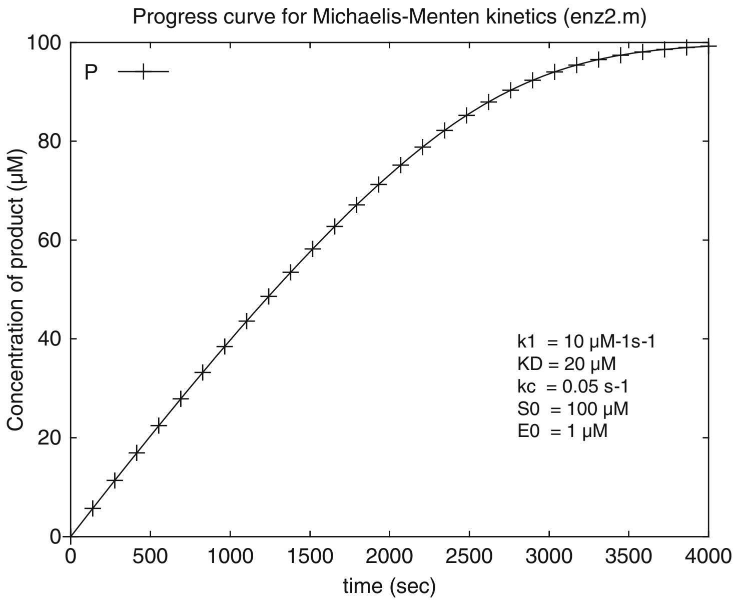


Fig. 7.3 Progress curve for Michaelis–Menten kinetics. Reaction scheme (3.4) is calculated with `enz2.m` and the parameters shown in the figure

Of course, the results depend on the set of parameters. Of these, only K_D1 (k_{-1}/k_1) and k_{cat} are significant, when a fast association rate constant `k1` is chosen in accordance with the Michaelis–Menten assumptions. This is done here with $k_1 = 10^7\ M^{-1}\ s^{-1}$. When this k_1 is changed by a factor of 10, it does not affect the progress curve shown in Fig. 7.3. When these assumptions are not made, the full set of differential equations in `enz2.m` allows the calculation of progress curves with all feasible rate constants of the reaction scheme (3.4).

Figure 7.3 shows a progress curve for an irreversible transition from substrate S to product P [scheme (7.2)]. Therefore, at infinite times the substrate concentration always will be zero and the product concentration will be the same as the initial substrate concentration. The initial part of this curve is linear for a considerable time range. This linear part is the only part which is required to determine the initial velocity. The initial velocity is the experimental value used for a Lineweaver–Burk plot (Fig. 3.6) and the corresponding data evaluation.

7.4 Lineweaver–Burk Plots from Progress Curves (`enz3.m`)

Comparing Figs. 7.3 to 7.1 shows that progress curves calculated from the simplified reaction scheme (3.4) give similar results to those calculated from the complex scheme (7.1). Michaelis–Menten kinetics usually is not evaluated by means of progress curves, but from measurements of initial velocities at different substrate concentrations. Under steady-state conditions, this corresponds to an equilibrium-binding curve, as discussed in Sect. 3.1.3. We will therefore calculate reaction scheme (7.1) at different substrate concentrations, determine the initial velocity from this, plot it as a Lineweaver–Burk Plot and determine K_m and k_{cat} from there. All this is done in the program `enz3.m`. The vector S00 of N2 substrate concentrations is defined in line 31:

```
31    S00=logspace(0,3,N2);
```

`N2` gives the number of points. The first parameter in the function `logspace` is the logarithm of the first number, the second parameter the logarithm of the last number. The numbers therefore range from 1 (10^0) to 1,000 (10^3). They are used in the main program within a loop:

```
16    filename='enz3';
17    global k1 km1 k2 km2 k3 km3;

34    for k=1:N2
35    S0=S00(k);
36    x0=[E0;S0;0;0;0];
37    M=lsode('enz1F',x0,t);
38    Px=M(:,5);
39    v0(k)=(Px(3)-Px(2))/(t(3)-t(2));
40    P(:,k)=Px(:);
41    end;
42    %% Plot 1
43    plot(t,P,'-+');
```

Within the loop, each step calculates the differential equations of `enz1F` for an association reaction of `E0` with a different concentration `S0`. The resulting product concentrations as a function of time are the fifth concentrations of the function `enz1F` and therefore are returned as the fifth column of the matrix `M` from `lsode` in line 37. They are assigned to a vector `Px` in line `38`. The rate of product formation `v0` is calculated from $\Delta P/\Delta t$ in line `39`. A matrix `P` of product concentrations for all `N2` different substrate concentrations `k` and all time points `N` is written in line `40` within the loop. The whole matrix can then be plotted with the simple `plot` command in line `43`, and the result is shown in Fig. 7.4.

Figure 7.4 shows a calculated series of progress curves. The initial velocities were calculated from the slopes of third and second points in line `39`. When these values are plotted versus the logarithm of the substrate concentrations in Fig. 7.5, the dose–activity curve follows the expected (sigmoid and symmetric) pattern of a one-site binding curve.

Figure 7.5 shows a dose–activity curve which would be ideally suited for the calculation of K_m and k_{cat}, from a least squares fit. The data points are equally distributed and no bias is introduced by a transformation artifact. For this task, numerical methods for nonlinear data fitting are described in Chap. 8.

The traditional way to analyze simple hyperbolic binding curves is a linear regression in a linearized plot, such as in a double reciprocal plot. For enzyme kinetics, this is the Lineweaver–Burk plot (Sect. 3.1.4), where the reciprocal initial

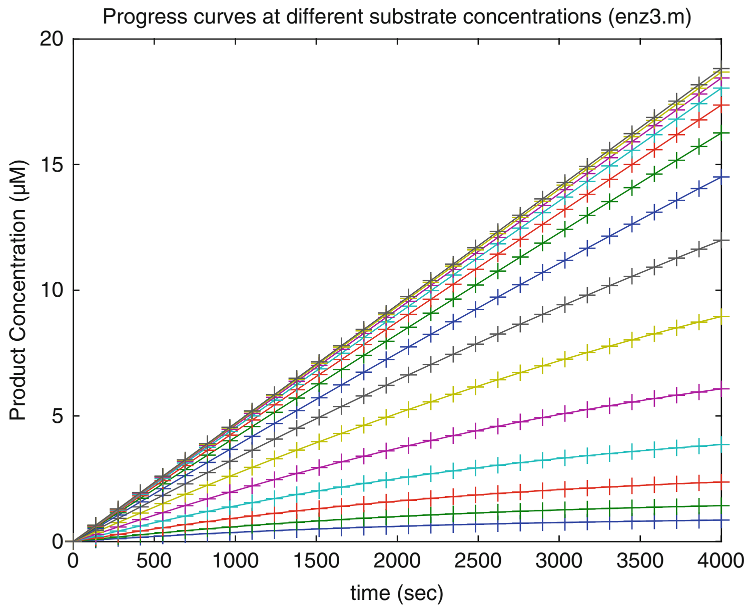


Fig. 7.4 Progress curves at different substrate concentrations. Product increase (+) is calculated from reaction scheme (7.1) and shown as a function of time. $K_D1 = K_D3 = 100\ \mu M$, $k_2 = 1\ s^{-1}$, $k1 = 0.01\ s^{-1}$, $k_{-2} = 0.001\ s^{-1}$, $k_3 = 0.05\ s^{-1}$. The substrate concentrations were taken from a logarithmic distribution between 1 μM and 1 mM

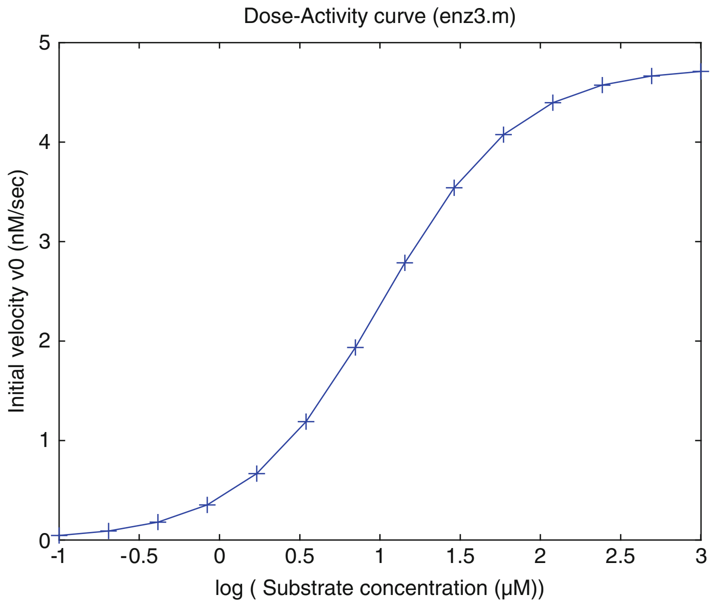


Fig. 7.5 Logarithmic dose–activity curve from initial velocities determined from Fig. 7.4. $K_D1 = K_D3 = 100\ \mu M$, $k_2 = 1\ s^{-1}$, $k1 = 0.01\ s^{-1}$, $k_{-2} = 0.001\ s^{-1}$, $k_3 = 0.05\ s^{-1}$. The initial velocities were calculated from the third and second time point and multiplied by a factor of 1,000

velocity is plotted versus the reciprocal substrate concentration. Anyone with a ruler can draw a straight line through such experimental data and extract the results directly from the plot.

Computers use *simple linear regression* [1], which is an analytical method to fit a straight line through any set of data points. Mathematically, it can be derived by minimizing the residual sum of squares of a fit to this line. Unlike numerical methods of nonlinear regression described in Sect. 8.1, it does not require an initial estimate. The function `linreg.m` takes a pair of row vectors x and y and fits a straight line through the data pairs x1/y1, x2/y2, x3/y3 . . . The result is a row vector a with two components a(1) and a(2), for slope and intersect, so that the resulting linear function is y = a(1)*x + a(2).

```
1      %% Linear regression for data pairs x,y
2      % x and y are row vectors of the same size
3      function a=linreg(x,y);
4      n=max(size(x));
5      a(1)=(n*x*y'-sum(x)*sum(y))...
6      /(n*x*x'-sum(x)*sum(x));
7      a(2)=sum(y)/n-a(1)*sum(x)/n;
```

Note line `4`, which is a function of a function in order to determine the number of points. `size(x)` returns a row vector of the number of rows and columns of `x`. `max(size)` returns the maximal value of size, which is the number of points. Lines `5–7` give the analytical solution for slope a(1) and intersect a(2) [1]. `sum(x)` returns the sum of all elements of `x`. `x* y'` in line `5` is a Scalar product of vectors, which is equal to the sum of $x_i * y_i$. This is the only example in this textbook, where a scalar product of vectors is used instead of element by element multiplication of vectors.

Simple linear regression requires double reciprocal transformation of data. Reciprocals for the Lineweaver–Burk plot are calculated in lines `58` and `59` of `enz1.m`. The resulting vectors `rS` (for reciprocal substrate concentrations) and `rv` (for reciprocal velocities) are then fitted to a straight line with the help of `linreg` (line `60`). From this, K_m, v_{max}, and k_{cat} are calculated in lines `61–63.` The reciprocal data are plotted in line `64`, and the result is shown in Fig. 7.6. The results of the linear regression is the straight line y = a(1)*rS + a(2). When this is included in the plot command of line 64, the line only covers the range of the experimental data. For a Lineweaver–Burk plot, one wants to elongate the line until its intersection with the x-axis. For this, the command `line` in line `69` was used. It plots a straight line from a first data point to a second data point. The coordinates for these data points are defined in line `65` and `66`. Likewise, the y-axis was drawn with the command `line(x1,y1)` in line `70`. A Lineweaver–Burk plot is not particularly suited for showing the whole significant concentration range, since it tends to magnify the low concentrations inappropriately. Therefore, the data for the lowest three substrate concentrations were excluded. The command `axis` in line `71` takes the range from the intersection of the linear regression with the x-axis (the K_m-value) and 110% of the reciprocal value of the fourth substrate concentration for the x-axis. Likewise, the scale for the y-axis was chosen between zero and 110% of the fourth reciprocal initial velocity.

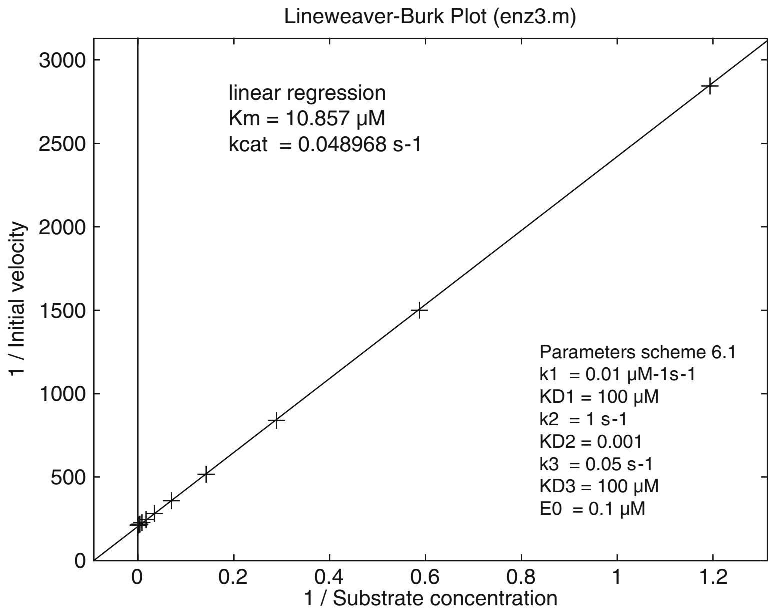


Fig. 7.6 Lineweaver–Burk plot from reaction scheme (7.1). The rate constants from Figs. 7.4 and 7.5 printed in the lower right corner were used to calculate reaction scheme (7.1). The initial slope of the progress curves obtained at different substrate concentrations was plotted versus the substrate concentrations in a double reciprocal manner. From this, K_m and k_{cat} were calculated from a linear regression and printed in the upper left corner

```
58   rS=1./S00;
59   rv=1./v0;
60   a=linreg(rS,rv);
61   KM=a(1)/a(2);
62   VMAX=1/a(2);
63   kcat=VMAX/E0;
64   plot(rS,rv,'+k');
65   x=[-KM,rv(1)];
66   y=[0,rv(1)*a(1)+a(2)];
67   x1=[0,0];
68   y1=[0,1/v0(1)];
69   line(x,y);
70   line(x1,y1);
71   axis([-KM 1.1/S00(4) 0 1.1/v0(4)])
```

Figure 7.6 shows the Lineweaver–Burk plot for the progress curves in Fig. 7.4 and the dose–response curve in Fig. 7.5. The fitted parameters K_m and k_{cat} correspond to reaction scheme (3.4).

How to modify the sample program. The program `enz3.m` has been designed to compare reaction scheme (7.1) with the classical evaluation of Lineweaver–Burk plots. It may be interesting to vary the parameters in lines `18–25`, in order to understand the properties of the Michaelis–Menten approximation. The same program can be used as a template whenever the question arises how any reaction scheme would appear when analyzed as Michaelis–Menten kinetics. In this case, the differential equations of the new reaction scheme would have to be written as a subroutine similar to `enz1F.m`, and the name of this subroutine would have to be called with `lsode` of line `37`.

7.5 Substrate Inhibition (**enz4.m**)

Every enzyme is different and many enzyme mechanisms are discussed in the literature, so that there are many candidates for suitable examples when we want to exploit the power of numerical methods. Some enzymes are inhibited by their own substrates [2, 3], and their initial velocities decrease at high substrate concentrations. The corresponding Lineweaver–Burk plots are not linear any more. There are numerous reaction schemes accounting for such an observation. Some of these have been calculated for acetylcholinesterase [2, 4, 5], an enzyme (E) which hydrolyzes acetylcholine (S) and produces choline (P1) and acetic acid (P2). The acetylated enzyme (EP2) has been identified as an intermediate, and substrate binding to the acetylated enzyme has been suggested as the key step of substrate inhibition leading to the inactive EP2S. This corresponds to reaction scheme (7.2). Note that inhibition is caused by a ternary complex of substrate and product P2. Therefore, scheme (7.2) also had been referred to as "product inhibition" [5].

$$\begin{tikzcd}
S + E \arrow[r, shift left, "k_1"] & ES \arrow[l, shift left, "k_{-1}"] \arrow[r, shift left, "k_2"] & EP2 + P1 \arrow[l, shift left, "k_{-2}"] \arrow[r, shift left, "k_3"] \arrow[d, shift left, "+S\ k_4"] & E + P1 + P2 \arrow[l, shift left, "k_{-3}"] \\
 & & EP2S \arrow[u, shift left, "k_{-4}"] \arrow[ul, "k_5\ -P2"] &
\end{tikzcd} \tag{7.2}$$

The dissociation of P2 from EP2S (the deacetylation in the case of acetylcholinesterase), giving the complex ES is an additional step which has been suggested [4] in order to explain some characteristics of the experimental data. The differential equations for scheme (7.3) are the main part of the function `enz4F.m`:

```
function dx=enz4F(x,t)
global  k1 km1 k2 km2 k3 km3 k4 km4 k5;

dE=-k1*E*S+km1*ES+k3*EP2-km3*E*P2;
dS=-k1*E*S+km1*ES-k4*S*EP2+km4*EP2S;
dES=k1*E*S-km1*ES-k2*ES+km2*EP2*P1+k5*EP2S;
dEP2=k2*ES-km2*EP2*P1-k3*EP2+km3*E*P2...
-k4*S*EP2+km4*EP2S;
dP1=k2*ES-km2*EP2*P1;
dP2=+k3*EP2-km3*E*P2+k5*EP2S;
dEP2S=k4*S*EP2-km4*EP2S-k5*EP2S;
```

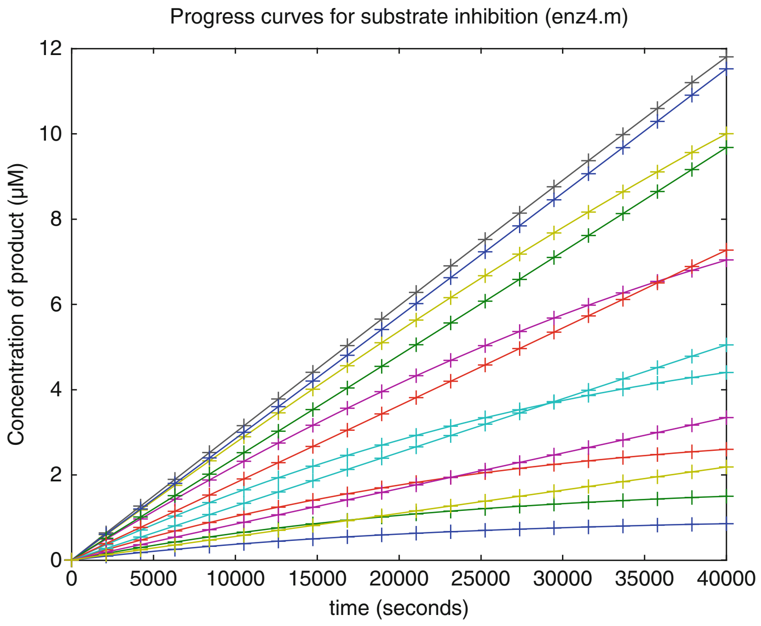


Fig. 7.7 Progress curves for substrate inhibition (7.2). $K_D1 = 100\ \mu M$, $K_D2 = K_D3 = 100$ mM. $k_1 = k_4 = 0.01\ \mu M^{-1}\ s^{-1}$. $k_2 = 1\ s^{-1}$, $k_3 = 0.05\ s^{-1}$, $k_5 = 0.001\ s^{-1}$. The enzyme concentration is 10 nM. The substrate concentration varied in a logarithmic scale from 1 µM to 1 mM

The complex `EP2` is involved in reactions 2, 3, and 4. Line `14` therefore became too long and was extended with the continuation marker `(...)`. The main program `enz4.m` calls the function enz4F in line 47 and displays the results in four different plots. First, the direct progress curves are shown in Fig. 7.7.

Figure 7.7 shows progress curves calculated with the program `enz4.m` and plotted in line `53`. There are 13 curves for 13 different substrate concentrations ranging from 1 µM to 1 mM in a logarithmic scale. The lowest curve is the product formation after the addition if 1 µM substrate. Note its curvature, because the low substrate concentrations sustain the steady-state equilibrium only for a relatively short time. The other curved lines show substrate concentrations of 1.8, 3.2, 5.6, 10, and 18 µM, until a maximum is reached for a substrate concentration of 32 µM. From there on, increasing substrate concentration leads to a decrease in the slope, but the curvature for these high substrate concentrations is significantly decreased. The long linear part reflects stable steady-state equilibrium.

Rather than measuring and calculating progress curves, enzyme kinetics usually is evaluated from the initial velocities which reflects the steady state o enzyme catalysis. The initial velocities are calculated in the main program of `enz4.m`, within the loop (`44–51`) for the N2 different substrate concentrations.

```
22    filename='enz4';
23    global k1 km1 k2 km2 k3 km3 k4 km4 k5;

35    N=20;
36    N2=13;
37    S00=logspace(0,3,N2);
38    km1=KD1*k1;
39    km2=k2/KD2;
40    km3=k3/KD3;
41    km4=k4*KD4;
42    t=linspace(0, tmax, N);
43    %% Main Program
44    for k=1:N2
45    S0=S00(k);
46    x0=[E0;S0;0;0;0;0;0];
47    M=lsode('enz4F',x0,t);
48    Px=M(:,5);
49    v0(k)=(Px(3)-Px(2))/(t(3)-t(2));
50    P(:,k)=Px;
51    end;
52    %% Plot1---------------
53    plot(t,P,'-+');

59    % plot 2
60    plot(S00,100*v0/(E0*k2),'-o');

67    %% Plot3---------------
68    rS=1./S00;
69    rv=1./v0;
70    rs1=rS(1:6);
71    rv1(1:6)=rv(1:6);
72    a=linreg(rs1,rv1);
73    x=[-a(2)/a(1),rv(1)];
74    y=[0,rv(1)*a(1)+a(2)];
75    x1=[0,0];
76    y1=[0,1/v0(1)];
77    KM=a(1)/a(2);
78    VMAX=1/a(2);
79    plot(rS,rv,'-o');
80    line(x,y);
81    line(x1,y1);
117   %% Plot4---------------
118   lS00=log10(S00);
119   plot(lS00,100*v0/(E0*k2),'-o');
```

The initial velocity v0 in line 49 is calculated from the difference of the third and second time point. For experimental data, the first time point always should be avoided because there may be experimental artifacts. Even for theoretical data, this point should not be used for the calculation of the initial velocity, because the steady state is not achieved at time $t = 0$. The data are plotted with the second plot command in line 60.

Figure 7.8 shows the initial velocity calculated from the progress curves in Fig. 7.7. The substrate concentrations of Fig. 7.8 only cover the range from 1 to 100 μM, in order to show the significant characteristics in this representation. For the first seven substrate concentrations, it shows an increase expected from a simple

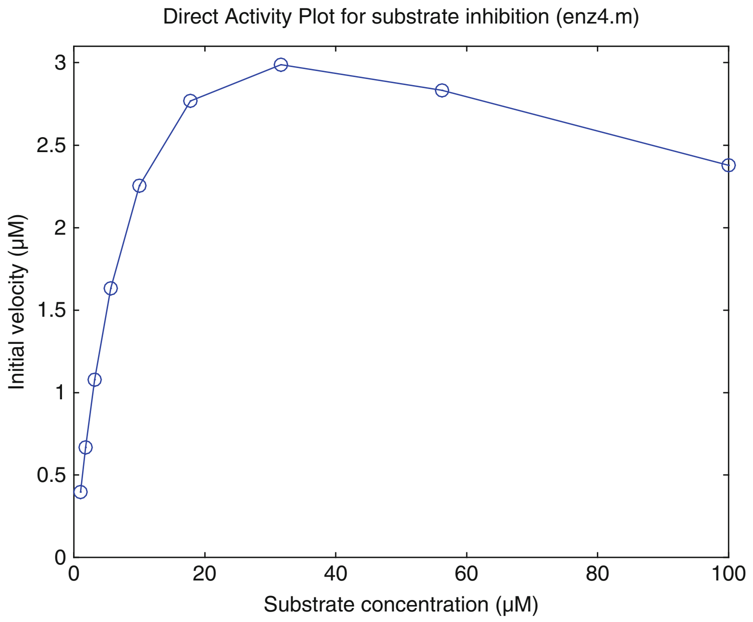


Fig. 7.8 Linear dose–activity plot for substrate inhibition (7.2). $K_D1 = 100$ µM, $K_D2 = K_D3 = 100$ mM. $k_1 = k_4 = 0.01$ µM^{-1} s^{-1}. $k_2 = 1$ s^{-1}, $k_3 = 0.05$ s^{-1}, $k_5 = 0.001$ s^{-1}. The enzyme concentration is 10 nM. The substrate concentration varied in a logarithmic scale from 1 µM to 100 µM. The initial velocity v0 was calculated from the slope between the third and second time point in Fig. 7.7

(hyperbolic) binding curve. For higher substrate concentrations the initial velocity decreases. For real experiments, such a decrease might be masked by experimental error and thus may escape notice. Here, of course, the reaction is calculated over a broader concentration range as shown in Fig. 7.10.

The Lineweaver–Burk plot (Fig. 7.9, plot 3 in line `79` of `enz4.m`) of these data shows a significant curvature for high substrate concentrations. Remember that the low substrate concentrations in this representation are located in the upper right part of the graph. The straight line is fitted to the six lowest concentrations with the function `linreg` as explained in the previous section. The respective program code of `enz4.m` is located between lines `68` and `78`. Line `70` creates a new vector from a range of values of a larger vector. A range is defined with a colon, so that `1:6` is the range of the first six elements, and `rS(1:6)` is the range of vector elements between `rS(1)` and `rS(6)`. `rS` is the vector of reciprocal substrate concentrations, and `rs1` therefore is a new vector consisting of the first six data points of `rS`.

Figure 7.10 (plot 4, line `79` of `enz4.m`) shows a logarithmic dose–activity curve, whereby the activity of the enzyme is defined as the initial velocity of product formation. These logarithmic representations all begin with a sigmoid increase and end with a phase where a final value is gradually approached. They are useful for visualizing a large range of experimental data, and they are particularly suited to compare fitted curves to experiments (Figs. 8.5–8.7).

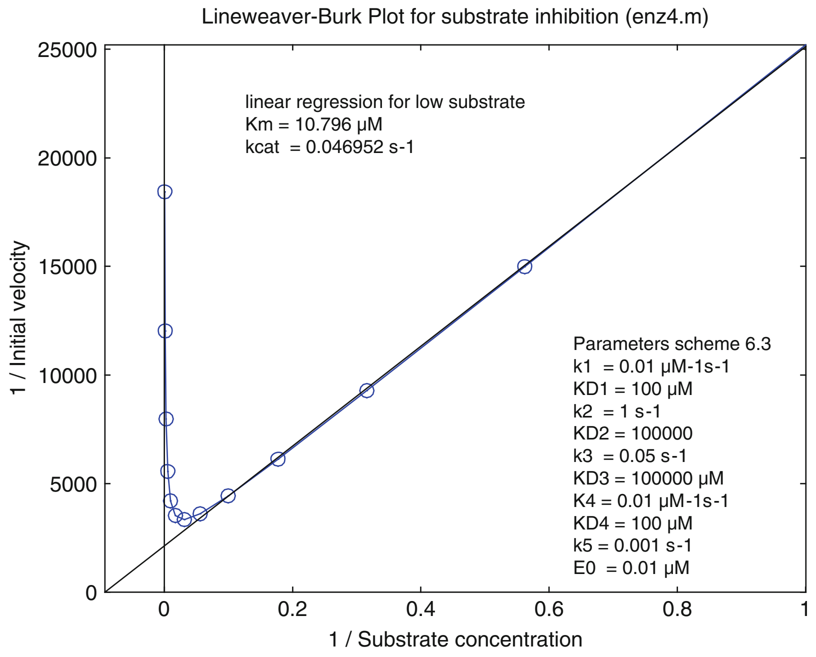


Fig. 7.9 Lineweaver–Burk plot for substrate inhibition (7.2). $K_D1 = 100$ µM, $K_D2 = K_D3 = 100$ mM. $k_1 = k_4 = 0.01$ µM^{-1} s^{-1}. $k_2 = 1$ s^{-1}, $k_3 = 0.05$ s^{-1}, $k_5 = 0.001$ s^{-1}. The enzyme concentration is 10 nM. The substrate concentration is varied in a logarithmic scale from 1 µM to 100 µM. The initial velocity v0 is calculated from the slope between the third and second time point in Fig. 7.7. The parameters indicated in the lower right corner are used to calculate reaction scheme (7.3). The initial slope of the progress curves obtained at different substrate concentrations is plotted versus the substrate concentrations in a double reciprocal scale. From this, K_m and k_{cat} were calculated in `enz4.m` from a linear regression of the six lowest substrate concentrations

7.6 Enzyme Inhibition with Reversible Inhibitors (`Enz5.m`)

Probably most enzymatic assays today are performed in high throughput screening facilities. These assays are performed at very low enzyme and relatively low substrate concentrations, so that the resulting slow reaction times agree with the pipetting processes employed. Substrate inhibition or other variations of the Michaelis–Menten kinetics usually are not observed under these conditions, so that steady-state assumptions are valid. Binding of substrate S to the enzyme E leads to an active enzyme substrate complex ES. The activity of this complex often is measured by a spectroscopic assay which in turn may involve additional components. Adding inhibitors 10–20 min before to the addition of substrate usually establishes equilibria between enzyme and inhibitor. A few minutes after addition of substrate steady-state equilibria between substrate and all complexes is established. Assuming a maximum of two inhibitor binding sites on the enzyme, one may use scheme (7.3) to calculate the coupled equilibria. Note that the reaction

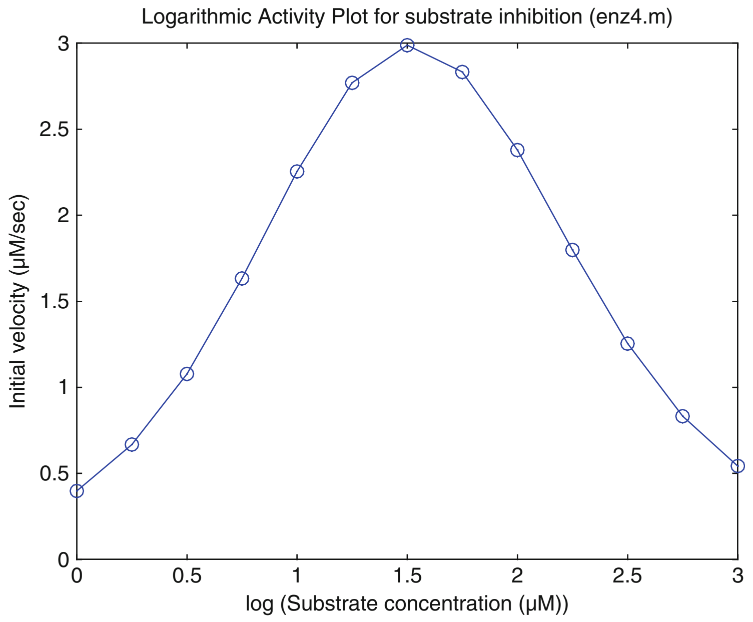


Fig. 7.10 Logarithmic dose–activity plot for substrate inhibition (7.2). $K_D1 = 100\ \mu M$, $K_D2 = K_D3 = 100$ mM. $k_1 = k_4 = 0.01\ \mu M^{-1}\ s^{-1}$. $k_2 = 1\ s^{-1}$, $k_3 = 0.05\ s^{-1}$, $k_5 = 0.001\ s^{-1}$. The enzyme concentration is 10 nM. The substrate concentration is varied in a logarithmic scale from 1 μM to 100 μM. The initial velocity v0 is calculated from the slope between the third and second time point in Fig. 7.7

scheme (7.3) is a sequential scheme like reaction scheme (2.18). It is useful for **equilibrium** studies, where individual values for equilibrium dissociation constants of equivalent intrinsic sites can be calculated from (2.30). For binding **kinetics**, scheme (7.3) should not be used without reason.

$$\begin{tikzcd}
ES \arrow[r, shift left, "+I"] \arrow[r, leftarrow, shift right, "K_I2"'] & ESI \arrow[r, shift left, "+I"] \arrow[r, leftarrow, shift right, "K_I4"'] & ESI_2 \\
E \arrow[u, shift left, "+S"] \arrow[u, leftarrow, shift right, "K_m"'] \arrow[r, shift left, "+I"] \arrow[r, leftarrow, shift right, "K_I1"'] & EI \arrow[u, shift left, "+S"] \arrow[u, leftarrow, shift right, "K_m"'] \arrow[r, shift left, "+I"] \arrow[r, leftarrow, shift right, "K_I3"'] & EI_2 \arrow[u, shift left, "+S"] \arrow[u, leftarrow, shift right, "K_m"']
\end{tikzcd} \tag{7.3}$$

ES is the active complex, the concentration of which is proportional to the enzymatic activity. For all practical cases, its equilibrium dissociation constant is the same as the Michaelis constant K_m. The different K_I are the equilibrium dissociation constants of the inhibitor and the different complexes. The enzymatic activity is proportional to ES, but active ESI or ESI_2 cannot be ruled out. The concentrations of all complexes can be calculated from the concentrations of the free concentrations [S], [E], and [I]. Just as discussed in Chap. 5, this gives three nonlinear equations for E0, S0, and I0 as a sum of free and bound enzyme, substrate and inhibitor, respectively. These equations are solved with `fsolve` in line `41` of the program `Enz5.m`. The function `Enz5F.m` contains the equations

```
function F = Enz5F(x)
global KM KI1 KI2 KI3 KI4 E0 S0 I0;
E=x(1);
S=x(2);
I=x(3);
F(1)=E+E*S/KM+E*I/KI1+E*S*I/(KM*KI2)...
+E*I*I/(KI1*KI3)+E*S*I*I/(KM*KI2*KI4)-E0;
F(2)=S+E*S/KM+E*S*I/(KM*KI2)...
+E*S*I*I/(KM*KI2*KI4)-S0;
F(3)=I+E*I/KI1+E*S*I/(KM*KI2)...
+2*E*I*I/(KI1*KI3)+2*E*S*I*I/(KM*KI2*KI4)-I0;
```

Traditionally, enzyme inhibition curves are presented as series of Lineweaver–Burk plots. For the case of one inhibitor binding site they have been calculated with analytical methods [6–8]. When the corresponding linear regressions intersect on the Y-axis (Fig. 7.12), the inhibitor is competitive, when they intersect on the X-axis (Fig. 7.13), the inhibitor is named noncompetitive, and when the Lineweaver–Burk plots run parallel, the inhibition mechanism was named

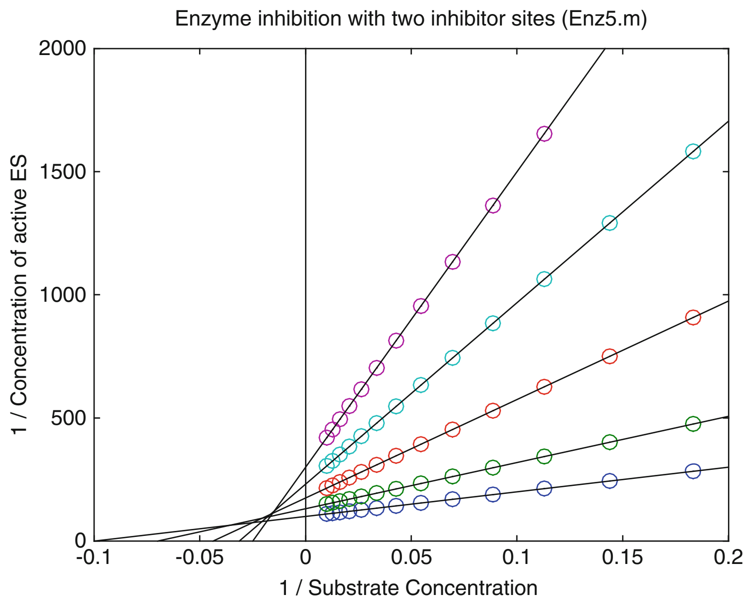


Fig. 7.11 Lineweaver–Burk plot calculated from reaction scheme (7.3). Km was taken as 10 μM, and all KI values with the exception of KI3 were taken as 100 μM. KI3 was taken as 10 μM. E0 was 10 nM. The inhibitor concentrations were 0, 25, 50, 75, and 100 μM

to be uncompetitive. In many cases we found none of these, and sometimes the lines did not even intersect in one point, as shown in Fig. 7.11.

Figure 7.11 shows a calculated set of data which corresponds to some of our experimental findings. For each inhibitor concentration, the reciprocal initial velocities lie on a straight line, but these lines need not intersect when there is more than one inhibitor binding site.

The program `Enz5.m` is useful for the calculation of a many different inhibition patterns in enzyme kinetics. It calculates steady-state equilibria, and therefore corresponds to the usual approach to enzyme kinetics [6]. It allows the calculation of many limiting cases, just by the manipulation of the equilibrium dissociation constants. For example, if one wants to calculate binding only to one site, the equilibrium dissociation constants for the second site (`KI3` and `KI4`) can be set many orders of magnitude higher than the other equilibrium dissociation constants. The simple case of competitive binding at one site is written in `Enz5.m` with Octave code as: `KM = 10;` `KI1 = 10;` `KI2 = 10000000;` `KI3 = 10000000;` `KI4 = 10000000;`

Figure 7.12 indeed shows the characteristic features of competitive binding. When a substrate competes with the inhibitor, the extrapolated value at infinite substrate concentrations must be identical at all inhibitor concentrations. Infinitely high substrate concentrations correspond to $1/S = 0$, and all lines corresponding to all inhibitor concentrations have to intersect at v_{max}, the maximal initial velocity. The intersection with the x-axis equals $-1/Km$, so that the affinity is decreased (The apparent Km is increased) with increasing inhibitor concentration.

The next classical example concerns noncompetitive binding. In this case, the inhibitor may bind to the free enzyme E and to the occupied enzyme ES. When the affinity for both is the same, then the binding of inhibitor cannot change the affinity for the substrate.

Figure 7.13 shows noncompetitive inhibition. The lines intersect at $-1/Km$ on the x-axis. The intersection at the x-axis gives the Km value, so that the affinities for the substrate are the same at all inhibitor concentrations. The substrate cannot compete with the inhibitor, since the ternary complex ESI is stabilized with increasing S at all inhibitor concentrations. Therefore, the lines do not intersect at the same v_{max} value.

Incidentally, the theoretical lines in `Enz5.m` are calculated slightly different from the theoretical lines in `enz3.m`. In both cases, the function `linreg.m` is used, but whereas in `enz3.m` the lines are plotted separately from the plot with the command `line`, they are now included as theoretical lines in the plot command. This is done in 48–54 of the program `Enz5.m`:

```
48    rS0=1./VS0;
49    rES=1./VES;
50    x=[-1/KM,rS0(1)];
51    for k=1:NI
52    a=linreg(rS0,rES(:,k)');
53    y(:,k)=a(1)*x+a(2);
54    end;
```

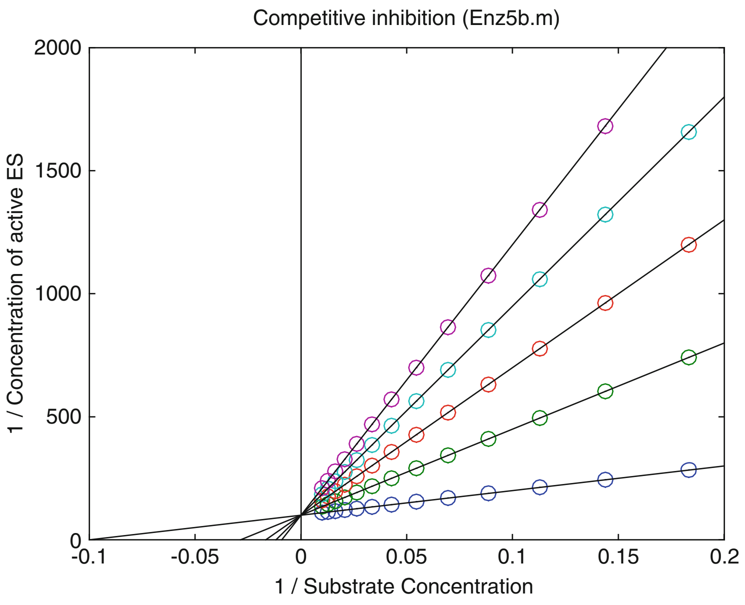


Fig. 7.12 Lineweaver–Burk plot for competitive inhibition (Enz5b.m). Reaction scheme (7.3), Km = 10 μM, K_I1 = 10 μM, K_I2 = K_I3 = K_I4 = 10 M, E0 was 10 nM. The inhibitor concentrations were 0, 25, 50, 75, and 100 μM

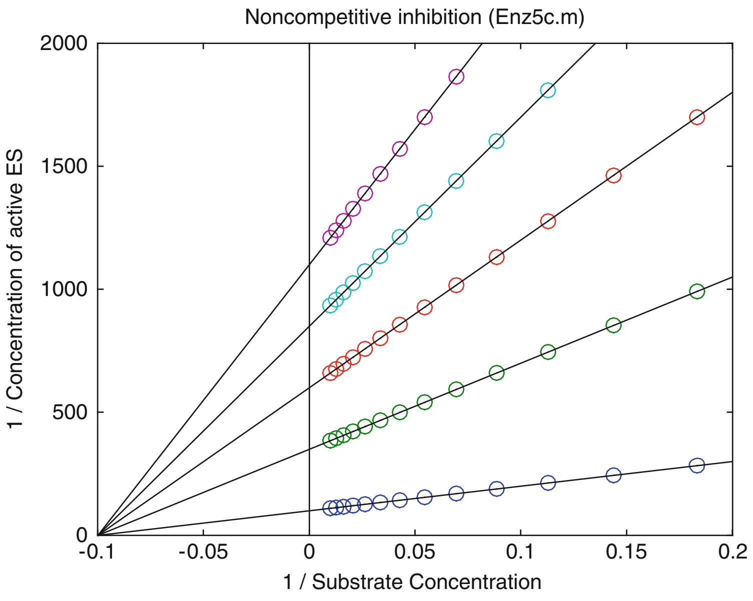


Fig. 7.13 Lineweaver–Burk plot for noncompetitive inhibition (Enz5c.m). Reaction scheme (7.3), Km = 10 μM, K_I1 = K_I2 = 10 μM, K_I3 = K_I4 = 10 M, E0 was 10 nM. The inhibitor concentrations were 0, 25, 50, 75, and 100 μM

`VS0` is the row vector of total substrate concentrations, as before, so that `rS0` becomes the row vector of reciprocal substrate concentrations (line `48`). `VES` is a matrix, a row for inhibitor concentrations of column vectors at different substrate concentrations. `rES` therefore is a corresponding matrix. Since `linreg` can only compute single arrays (vectors), the columns for each inhibitor concentration `k` have to be extracted from that matrix with `rES(:,k)` in line `52.` The function `linreg` requires row vectors as arguments, so that the column vector `rES(:,k)` has to be transposed to a row vector with the transpose operator `(')` in line `52`. The resulting linear function is calculated in line `53`. Note that the range for the independent variable has been extended in line `50` to include values up to the negative `1/KM`.

Uncompetitive inhibition (Fig. 7.14) is observed when the inhibitor only binds to the complex ES formed between the enzyme and the substrate. The inhibitor thus increases the affinity (decreases the Km) and decreases the maximal velocity. The increase in affinity is easy to calculate, but difficult to understand. One would assume that any inhibitor should decrease the apparent affinity of a substrate. Here, however, the inhibitor does not compete with the substrate for the free enzyme, and in fact stabilizes the ternary complex ESI.

Most enzyme kinetics is measured during drug screening campaigns [9]. In this case, the data are not evaluated as Lineweaver–Burk plots, but as dose–response

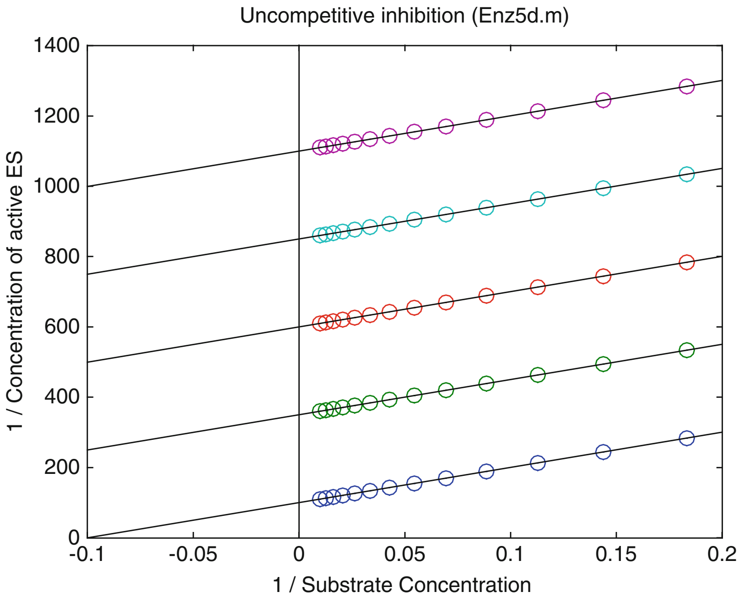


Fig. 7.14 Lineweaver–Burk plot for uncompetitive inhibition (`Enz5c.m`). Reaction scheme (7.3), Km = 10 µM, K_I2 = 10 µM, $K_I1 = K_I3 = K_I4$ = 10 M, E0 was 10 nM. The inhibitor concentrations were 0, 25, 50, 75, and 100 µM

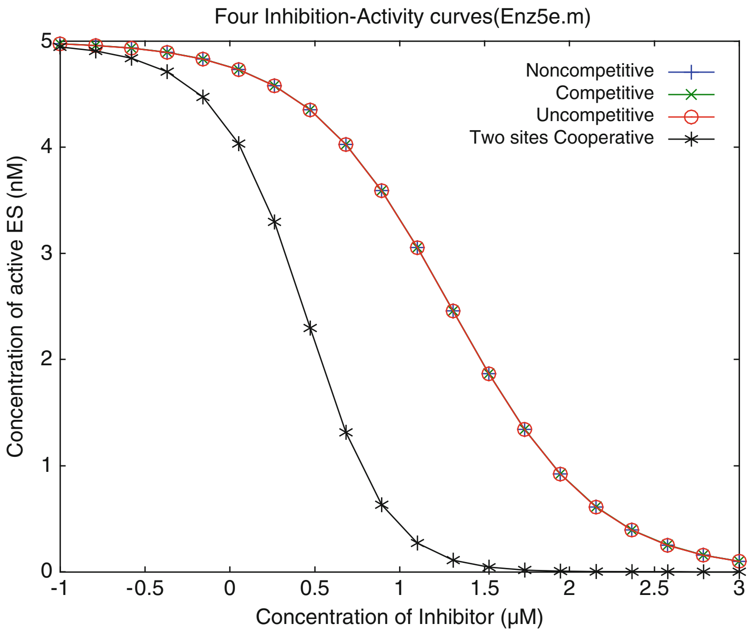


Fig. 7.15 Dose–activity curves for reversible inhibition (7.3). (+) Noncompetitive inhibition KI1 = KI2 = 20 µM. (x) Competitive (KI1 = 10 µM), (o) uncompetitive (KI2 = 10 µM) inhibition. (*) Cooperative inhibition at two sites (KI1 = KI2 = 10 µM, KI3 = Ki4 = 1 µM). E0 = 10 nM, S0 = 100 µM, KM = 10 µM

plots, or dose–activity plots, or dose–inhibition plots. This corresponds to the calculation of active [ES] plotted versus the logarithm of the inhibitor concentration.

Figure 7.15 shows dose–activity curves for competitive inhibition (x), noncompetitive inhibition (+), and uncompetitive inhibition (o) for a single site as shown in Figs. 7.12–7.14. These three dose–activity curves are identical and correspond to a Hill coefficient of 1. Only when a second site becomes available, the shape of the dose–response curve changes, and the slope at its inflection point becomes steeper.

One note for MATLAB users: In the MATLAB environment, the solver `fsolve` in the program `Enz5e.m` could not compute all values of Fig. 7.15, at least in the MATLAB version R2009a. The highest inhibitor concentrations lead to a numerical artifact as had been discussed in Sect. 2.6. Therefore, the MATLAB program `Enz5eM.m` does not cover the whole range of inhibitor concentrations.

Numerical artifacts generally have to be anticipated whenever numerical methods are employed. When computed data show a continuous function with one or two outliers, one can safely assume that these outliers have been generated by numerical artifacts.

References

1. http://en.wikipedia.org/wiki/Simple_linear_regression
2. Reed MC, Lieb A, Nijhout F (2010) The biological significance of substrate inhibition: a mechanism with diverse functions. Bioessays 32:422–429
3. Sekulic N, Konrad M, Lavie A (2007) Structural mechanism for substrate inhibition of the adenosine 5-phosphosulfate kinase domain of human 3-phosphoadenosine 5phosphosulfate synthetase 1 and its ramifications for enzyme regulation. J Biol Chem 282:22112–22121
4. Hofer P, Fringeli UP (1981) Acetylcholinesterase kinetics. Biophys Struct Mech 8:45–59
5. Brockendahl H, Müller T-M, Verfürth H (1968) Zur Kinetik der Produkthemmung. Hoppe Seyler's Z Physiol Chem 349:21–24
6. Cleland WW (1963) The kinetics of enzyme-catalyzed reactions with two or more substrates or products II. Inhibition nomenclature and theory. Biochim Biophys Acta 67:173–185
7. Berg J, Tymoczko J, Stryer L (2002) Biochemistry. W. H. Freeman and Company, New York
8. Dixon M, Webb EC, Thorne CJR, Tipton KF (1979) Enzymes, 3rd edn. Longman, London, p 126
9. Inglese J, Auld DS, Jadhav A, Johnson RL, Simeonov A, Yasgar A, Zheng W, Austin CP (2006) Quantitative high-throughput screening: a titration-based approach that efficiently identifies biological activities in large chemical libraries. Proc Natl Acad Sci USA 103:11473–11478

Chapter 8
Fitting the Data

Fitting models to experimental data is done by nonlinear regression routines, which vary multiple parameters like rate constants, etc. until an optimal fit is achieved. The resulting set of parameters need not be unique. Parameters often are correlated, so that the variation of one parameter can be compensated by variations of other parameters without reducing the quality of the fit. A correlation matrix helps to clarify this point. Experimental procedures, strategies for the reduction of the number of varied parameters, and global fits enhance the reliability of derived rate constants. At the end of this chapter, the reader should be able to import data from a spreadsheet, fit them to any reaction scheme and do a critical assessment of the significance of the fitted values.

8.1 Multi-parameter Fits and Correlation of Parameters (`fit1.m`)

A theory can only be verified on the basis of experimental data. When data and theoretical curves are plotted together, both sets initially do not match, and parameters have to be adjusted until a reasonable fit is achieved. The criterion, by which the quality of such a fit is measured, is the sum of the squared differences between experimental and theoretical points (8.1).

$$\Sigma(\mathrm{dat_i} - \mathrm{theo_i})^2 \tag{8.1}$$

The method to minimize the squares of these differences is commonly called the method of "least squares", so that the Octave routine is called `leasqr` and MATLAB routine is `lsqcurvefit`. The actual function call in Octave and MATLAB is different, but both functions give very similar results. They require a vector **x** of independent variables, a vector **dat** of experimental data for each **x**, and the name of the function which calculates the theoretical values **theo** as a function of **x**. Last but not least the routines require an initial set of parameters (“pin”) which is modified until a good fit is achieved.

Unlike simple linear regression [1], calculated with `linreg` in Sect. 7.4, parameters obtained from multi-parameter nonlinear regression need not be unique. This is illustrated with a drastic example. The program `fit1.m` applies nonlinear regression to a linear function containing an excess of parameters and compares the result with simple linear regression. The problem of correlated parameters immediately becomes obvious, so that one can deduce some general strategies for multi-parameter data fitting.

8.1.1 The Sample Program fit1.m

The linear function (8.2) to be fitted as our drastic example contains an excessive number of parameters.

$$y = (a/b) \cdot x + c - d \tag{8.2}$$

Function (8.2) gives a straight line with a slope of a/d and an interception of c–d at the *y*-axis. The parameters to be fitted (a, b, c and d) are elements of a vector pin. With this vector notation, the function (8.2) is translated to the function `fit1F.m` called from `leasqr` in Octave.

```
function y=fit1F(x,pin);
y=(pin(1)/pin(2))*x + pin(3)- pin(4);
```

For MATLAB, its function `lsqcurvefit` expects a reversed order of arguments, so that the first line of the corresponding MATLAB function is

```
function y=fit1FM(pin,x);
```

Everything else is the same for these functions, whether they are written in Octave and called from `leasqr` or in MATLAB and called from `lsqcurvefit`. The `global` statement (not used in the function `fit1M.m`) can be used to transfer variables which are not varied in the nonlinear regression routine. The sample program `fit1.m` calls the function `leasqr` in lines 22 and 23. Note that `leasqr` expects column vectors for the arguments, so that the row vectors `x` and `dat` have to be transposed (operator ’). Line 22 gives six results of `leasqr`, namely `fcurve`, the resulting fitted theoretical curve, `FP`, the resulting fitted parameters `kvg` and `iter` for information of the fitting process itself (not used here), `corp`, the correlation matrix of parameters, and `covp`, the covariance matrix of parameters.

```
12    filename='fit1';
13    a=10;b=1;c=10;d=1;
14    N=20;
15    x=linspace(0,100,N);
16    y1=(a/b)*x+c-d;
17    r=(1+0.2.*(0.5-rand(1,N)));
18    dat=y1.*r;
19    pa=linreg(x,dat);
20    y=pa(1)*x+pa(2);
21    pin=[1;2;3;4];
22    [fcurve,FP,kvg,iter,corp,covp]=...
23    leasqr(x',dat',pin,'fit1F');
24    % plot ------------------
25    plot(x,dat,'*',x,y,x,fcurve,'k');
```

A set of sample data is generated in lines `16–18`. The function `rand(1,N)` gives a row vector of `N` elements of random numbers between 0 and 1. Therefore, `(0.5-rand(1,N))` is a row vector with random numbers between −0.5 and 0.5, and the result `r` in line `17` is a row vector with a mean value of 1 and a noise with random numbers maximal ±10%. The data set `dat` (line `18`) therefore consists of a noisy linear distribution as shown in Fig. 8.1.

Figure 8.1 shows that the nonlinear regression `leasqr` will find a solution, even if the number of fitted parameters exceeds the number of meaningful parameters.

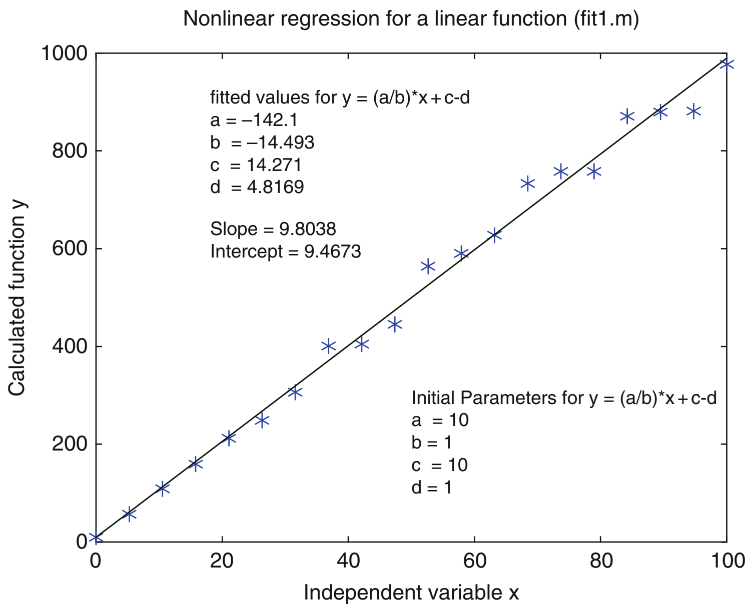


Fig. 8.1 Multi-parameter fits of a linear function. The program `fit1.m` generates data (*) and shows fitted linear functions (−). The results from simple linear regression linreg and from the nonlinear regression leasqr overlap to one solid line

Both, `y` in line `20`, the function fitted from `linreg`, and `fcurve` in line `22`, the function fitted with `leasqr,` give the same results. They are exactly superimposed in Fig. 8.1, so that only one line is visible, although both functions are plotted with the `plot` command in line `25`.

Note the parameters returned from `leasqr.` They are shown as a, b, c, and d in Fig. 8.1. Each time the sample program is executed (type `fit1` in the Octave terminal window), a different data set is generated and new parameters are fitted. The resulting parameters `a,b,c,d` differ considerably for each execution of `fit1`. Try it out! The erratic results for `a,b,c,d` are in contrast with the reasonable fluctuations of `pa(1)` (=Slope) and `pa(2)` (=Intercept), returned from `linreg`. Even though the results for `a` and `b` cannot be predicted, its quotient `a/b` is the same as `pa(1)`, the slope returned from `linreg`. Likewise, the difference `c-d` is the same as `pa(2)`, the intercept returned from `linreg`. All this is not surprising. Exactly two parameters (slope and intercept) define a straight line and one cannot obtain more than two parameters from a fit to such a line. But let us pretend that we do not know this, and that we have performed numerous fits with the program `fit1.m,` alas with the same data set. When we then plot the resulting parameter `a` as a function of `b`, and the parameter `c` as a function of `d`, we find that all these data pairs lie on straight lines:

$$\texttt{a} = \texttt{f(b)} = \texttt{b} \cdot \texttt{pa(1)} \tag{8.3}$$

$$\texttt{c} = \texttt{f(d)} = \texttt{d} + \texttt{pa(2)} \tag{8.4}$$

In the general case of multi-parameter fits to unknown functions, the parameters which give the same (optimal) fit will not lie on straight lines, as in (8.3) and (8.4), but will show a certain distribution. In the ideal case, this distribution is narrow, so that a variation of one parameter `x` could not be compensated by the variation of another parameter `y`, and still gives the same quality of a fit. The linear dependence of two parameters can be described with the help of Pearson's correlation coefficient [2–6]. This coefficient can have values between 1 and -1. If it is one, the parameters are correlated, so that the increase in x can be completely compensated by an increase in y. If it is -1, an increase in x can be compensated by a decrease in y. If it is zero, the parameters are not correlated, and an increase in x cannot be compensated by a variation of y. For `a` and `b` of (8.2) and (8.3), an increase in `a` can be compensated by an increase in `b`. Its correlation coefficient is 1, the same as the correlation coefficient of c and d in (8.4). For the ideal case of significant parameters, a variation of x cannot be compensated by a variation of y and the correlation coefficient should be near zero.

For MATLAB, the function `lsqcurvefit` returns no statistical information like `corp` or `covp`. For this, one would have to use the function `nlinfit` from the MATLAB Statistics toolbox. The MATLAB program `fit1M.m` differs from the Octave program `fit1.m` only in lines 22 and 23.

```
22    FP=lsqcurvefit('fit1FM',pin,x,dat);
23    fcurve=fit1FM(FP,x);
```

The order of arguments in lsqcurvefit differs from leasqr, and this is reflected in the order of arguments of fit1FM.m. The MATLAB function returns the vector par of fitted parameters, but not the fitted theoretical curve. This can be computed (line 23) from the same function (fit1FM) which was used for the nonlinear regression in line 22. The MATLAB function lsqcurvefit accepts row or column vectors, but size has to be identical for x and dat.

8.1.2 Strategies to Fit Correlated Parameters

fit1.m gives a drastic example to demonstrate the problem arising from the correlation of parameters. For reaction schemes, one typical example of correlated parameters are forward and back reaction rate constants under steady state conditions. An increase of the forward rate constant of a reversible reaction would be compensated by an increase of the back reaction rate constant, in order to keep the equilibrium constant. Their correlation coefficient can be expected to be 1. For kinetic experiments it generally is a good idea not to vary both rate constants, but to vary one rate constant and one equilibrium constant instead.

There generally are numerous rate constants or intrinsic equilibrium dissociation constants which are important for the understanding of the mechanism, but which cannot be determined experimentally. Reaction scheme (7.1), for example, is a plausible minimal reaction scheme for enzyme kinetics, but it contains six rate constants, which simply cannot be fitted from progress curves. Progress curves are dominated by steady state equilibrium (but see Sect. 7.2) so that quotients of rate constants (i.e. the equilibrium constants) are fitted together with one of the rate constants.

$$S + E \underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} ES \underset{k_{-2}}{\overset{k_2}{\rightleftharpoons}} EP \underset{k_{-3}}{\overset{k_3}{\rightleftharpoons}} E + P \qquad (7.1)$$

This still results in six parameters to be fitted, namely K_D1, k_1, KD_2, k_2, K_D3, and k_3. With the exception of k_3, all rate constants can be assumed to be fast, so that they do not influence the shape of the progress curve. K_D1 is the initial equilibrium which corresponds to the Km value under some conditions. For reversible reactions, one might assume the extreme case that the product has a similar affinity as the substrate, so that $K_D3 = K_D1$. When the reversible reaction (substrate formation from product) is unlikely, one may set K_D3 to a large value such as 1 M. This leaves three parameters to be fitted, namely, K_D1, K_D2 and k_3. Such a strategy does, however, not *prove* the assumptions employed for the fitting routine, and a fit performed under a set of assumptions does not return unambiguous experimental results. Incidentally, scheme (3.4) of Michelis–Menten kinetics follows from such a strategy and gives a commonly accepted simplified version of (7.1).

The next part of a fitting strategy consists of employing as many different experiments as possible. Enzyme kinetics, for example, cannot be measured from one progress curve. A series of curves has to be measured, and that series has to cover a broad concentration range. When a series of experimental curves measured under different conditions is fitted with one set of rate constants, the strategy is named “global fit”. A global fit may generate additional parameters. For example, there may be variations in the initial concentrations used in different experiments, or there may be different background signals for different individual experiments. As a rule, these additional parameters for individual experiments are not correlated with the global ones.

Statistical values such as correlation coefficients and the sum of squares can help to assess the quality of the results, but they never should be taken as standalone criteria. There are at least three questions which have to be asked:

1. Does the fit reflect the properties of the data? If there is a systematic deviation between experimental data and the fitted theoretical curve(s), either the model or the set of parameters is not correct.
2. Can a variation of one parameter be compensated by the other ones? If, for example, the parameter k1 resulting from a multi-parameter fit is changed by a factor of 2 or 10 and kept constant, can a new fit with the other parameters compensate this? A correlation coefficient will help to provide a strategy for the choice of fixed or varied parameters.
3. How does the fit and how do the parameters vary with repetitions of the experiments?

As will be discussed below, individual parameters cannot be determined unambiguously when they are correlated with others. But a successful fit *proves* that a model with its derived set of parameters is adequate to quantify the fitted data.

8.2 Experimental Setup

Data analysis requires reliable data, and bioanalytical studies require the precise handling of small volumes. Each pipetting step introduces an experimental error, so that the first aim of experimental design must be the reduction of pipetting steps. A second concern is the reproducibility. Many samples are prepared from living organisms, so that they may vary from batch to batch. For a large quantitative study one should never change the batch within a study. It is a better idea to pool batches and to work from this pool. The same holds for all stock solutions. Weighing inhibitors, substrates, buffers, etc. typically has a larger experimental error than the subsequent pipetting steps.

Calibration errors can be avoided by dilution series, whereby a given stock solution is diluted 1:1 by a series of dilutions with the same volume. This is shown in Fig. 8.2. If the same pipette is used for all steps, there is no systematic calibration error. Moreover, the concentrations will be distributed in a logarithmic scale, so that experimental data can be equitably distributed over a broad range of concentrations.

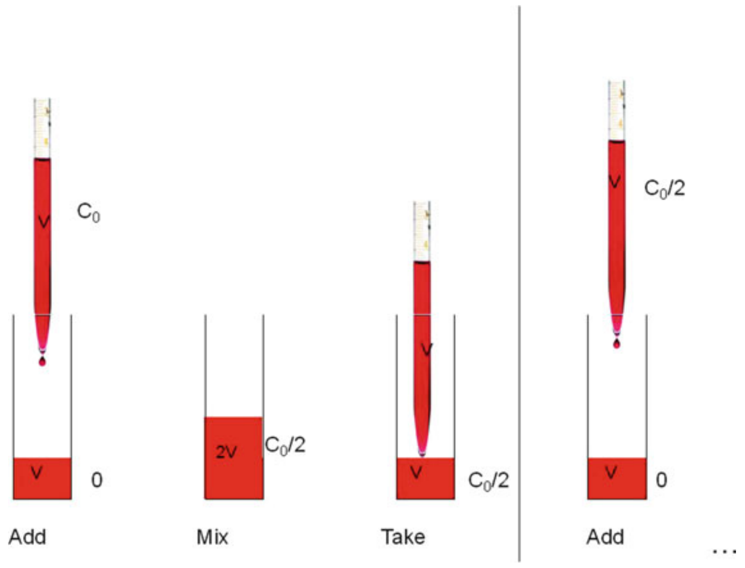


Fig. 8.2 Serial dilution 1:1. First, a volume V is dispersed to a series of tubes or microtiter wells. Then the same volume V of a stock solution is added to the first tube with the same pipette to give double the volume and half of the concentration. Then the same volume V is taken off with the same pipette and added to the second well, and so forth. This procedure results in concentrations of C_0, $C_0/2$, $C_0/4$, $C_0/8$ ... The last volume V, taken from the last tube after the mixing step is discarded

There are many other concerns for experiments intended for global fits. The temperature for kinetic experiments has to be maintained; all controls have to be performed at the same day, possibly before and after the experiment, etc. It only requires common sense, but one should note that global fits may not work for older experimental data which had not been recorded consistently.

8.3 Entering the Experimental Data (fit2.m)

Data input and output is done slightly differently in Octave and MATLAB. MATLAB reads the Microsoft Excel format with the command `xlsread`. In GNU Octave, `dlmread` only reads tab-delimited text files, which may be generated within Excel when a spreadsheet is saved in the *.txt format with the option “tab separated”. The Octave function `dlmread` in line 13 of the sample program `fit2.m` returns a numeric matrix M. The MATLAB function `xlsread` returns a vector [M,C]with 2 elements, one numeric matrix M and one cell array C. The argument `GeniosPro_Sample.asc` is the name of the spreadsheet to be read. Here, it is the output of a microtiter plate reader. It consists of 78 readings of a 16 × 24 microtiter plate performed with a time difference of 27 s. These translate into 384 kinetic experiments with 78 time points each. Additional information like time and temperature becomes obvious when the sample data are opened as a spreadsheet. Screening results from our Genios Pro reader are chosen as a realistic example for experimental data.

Once the data are read in line 13, the size of the matrix M is determined in line 14. For Octave, this size is [149,625], 1,496 rows and 25 columns. In MATLAB, s = [148,024]. Both languages obviously have a different algorithm to identify numerical data. The readings are separated by three lines, so that there are 16 + 3 = 19 lines for each experiment. Comparing M and `GeniosPro_Sample.asc`, one can deduce the equation to calculate the number of time points in line `15`. From this, a three-dimensional matrix `mes(i,k,j)` is calculated from `M` in line `19`. The first dimension is time and the second and third dimensions are coordinates of the microtiter plate. A vector `time` for `N` time points is constructed in line `20`.

```
13    M=dlmread('GeniosPro_Sample.asc');
14    s=size(M);
15    N=(s(1,1)-14)/19;
16    for i=1:N
17    for k=1:16
18    for j=1:24
19    mes(i,k,j)=M(1+(i-1)*19+k,j+1);
20    time(i)=(i-1)*27;
21    end;end;end;
22    save('save.mat');
23    clear;
24    load('save.mat');
25    plot(time,mes(:,:,1))
```

Reading data from a spreadsheet takes a while. When extracted data are used for fitting procedures, it is a good idea to save them in a MATLAB (or Octave) format. This is done in line `22` with the command `save`. The use of the corresponding `load` command is shown in line `24`. It restores all parameters, values, and variable names after the `clear` command in line `23`. The options and data formats for `save` are different in Octave and MATLAB, but the syntax in lines `22` and `24` is identical in both languages. The plot command in line 25 leads to Fig. 8.3, where 16 experiments of the first column of the microtiter plate are shown.

Figure 8.3 shows a screen for CDC25 inhibition with 16 substances from the first column of a 384 well microtiter plate. The program `fit2.m` extracts data from a worksheet and saves them in Octave format.

How to modify the program. The program `fit2b.m` retrieves these data from the file `save.mat`, which had been saved in Octave format, and therefore is much faster. When data are retrieved with the `load` command, all names are retrieved and the variable names assigned in the program are overwritten. The command `load('save.mat');` therefore has to appear in the very beginning (line `12`) of the program `fit2b.m`. If it would appear later, the `filename 'fit2b'` (line `13`) would be overwritten with `'fit2'`, the filename which had been stored. The `plot` command in line `14` plots all data. This is not possible in MATLAB, where three-dimensional matrices cannot be plotted.

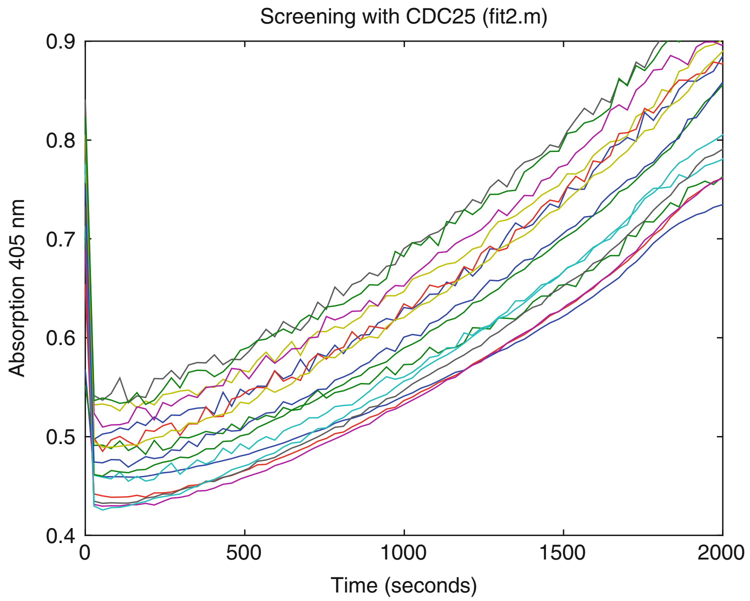


Fig. 8.3 Screening phosphatase activity of CDC25.The substrate pNPP was added to a concentration of 50 mM, before the absorption at 405 nm was recorded. Data were retrieved with the program `fit2.m` from the spreadsheet `GeniosPro_Sample.asc`, stored in Octave format as sample.mat and retrieved with the load command

```
clear;
load('save.mat');
filename='fit2b';
plot(time,mes(:,:,:))
```

The resulting plot is not shown explicitly, since it is very similar to Fig. 8.3. Try to plot different data with the program `fit2b.m`. For example, mes(:,3,:) is the third row of the microtiter plate measured at all time points. Or (2:N,:,8) is the 8th column of all but the first time points.

8.4 Fitting Substrate Inhibition

A fitting procedure can be a lengthy process. Once a solid set of experiments has been translated into a data file and stored in Octave format, a fitting strategy has to be devised. Usually this is accompanied with the development a suitable reaction scheme. For substrate inhibition, reaction scheme (7.2) has been used successfully [7]. It is an example for an enzyme mechanism which cannot be analyzed with Michaelis–Menten approximation. First, a sample data set for (7.2) is generated with the program `fit3.m`. Then, the initial velocities from these sample data are calculated and fitted to a simplified steady state model with the program `fit4.m`. Both covariance and correlation matrix reveal that the fitted parameters are not reliable. The model is further simplified and calculated with `fit4r.m`. The resulting parameters show a reasonable standard deviation, but the simplified model is difficult to relate to the molecular mechanism.

A second fitting strategy `fit5.m` does not change the model but tries to minimize the number of parameters with reasonable assumptions. It has to make use of all experimental information available, so that global fits to progress curves are performed. The statistical information of the fitting process is analyzed from the confidence interval and the correlation matrix. Additional information is introduced to fix parameters which have a high statistical error and high correlation coefficients. This leads to a reasonable set of parameters, which do not only depend on the experimental data, but also on the additional information invested in the fitting process.

$$\begin{tikzcd}
S + E \arrow[r, shift left, "k_1"] & ES \arrow[l, shift left, "k_{-1}"] \arrow[r, shift left, "k_2"] & EP2 + P1 \arrow[l, shift left, "k_{-2}"] \arrow[r, shift left, "k_3"] \arrow[d, shift left, "k_4\ +S"] & E + P1 + P2 \arrow[l, shift left, "k_{-3}"] \\
& & EP2S \arrow[u, shift left, "k_{-4}"] \arrow[ul, "k_5\ -P2"] &
\end{tikzcd} \tag{7.2}$$

8.4.1 *Generating Sample Data (`fit3.m`)*

Sample data are generated in `fit3.m` by calculating a set of theoretical data and adding random noise. The substrate concentration is computed from a series of dilutions by a factor 2 (multiply 0.5) from the initial concentration `S00`, just as shown in line `45`. The differential equations of scheme (7.2) are solved with `enz4F.m` (Sect. 7.4) and called from `lsode` in line `47`.

```
43    %% Generate sample data
44    for k=1:N2
45    S0=S00*0.5^(k-1);
46    x0=[E0;S0;0;0;0;0;0];
47    M=lsode('enz4F',x0,t);
48    P(:,k)=M(:,5);
49    VS0(k)=S0;
50    end;
51    ct=t';
52    Da=P.*(1+0.03.*(0.5-rand(N,N2)));
53    save('SubInhib.mat',...
54    'ct','P','Da','VS0');
55    plot(t,P,'-r',t,Da,'+k');
```

The product concentration P for all times (`:`) and each substrate concentration (`k`) is extracted from the matrix M in line `48`. Less than 3% random noise is added to the product concentration in order to simulate experimental data (`Da`) in line `52`. The vector `t` = time had been defined with `linspace` in line `16` as a row vector. It is transposed in line `51` to a column vector `ct`. This agrees with the matrixes `P` and `Da`, where the rows correspond to time and the columns to the substrate concentration. Note the serial dilution (line `45`), so that the first substrate concentration is the largest one. The `save` command in lines `53` and `54` does not save all parameters, but selects the vectors time (`ct`) and substrate concentrations (`VS0`) together with the matrices `P` and `Da`. The generated data (+) are random variations of the theoretical values (−) as shown in Fig. 8.4.

8.4.2 *Calculating Steady State Equilibria* (`fit4.m`)

Enzyme kinetics can be calculated from initial velocities or progress curves at different substrate concentrations. Let us begin with initial velocities. The data are loaded from the file generated in `fit3.m`. Unlike `store`, the `load` command in line `33` does not require variable names, even when specific variables had been stored (line `54` in `fit3.m`). The initial velocities are calculated from the slope of initial data points in lines `33`–`35`. `ct`, the column vector for time, and `Da`, the corresponding vector of simulated data, had been retrieved with the `load` command in line `32`.

```
32    load('SubInhib.mat');
33    for k=1:N2
34    slope(k)=(Da(3,k)-Da(2,k))/(ct(3)-ct(2));
35    end;
36    lKD1=log(KD1);lKD4=log(KD4);lkcat=log(kcat);
37    %% Main Program --------
38    pin=[lKD1;lKD4;lkcat];
39    vs=VS0';sl=slope';
40    [fcurve,FP,kvg,iter,corp,covp]=...
41    leasqr(vs,sl,pin,'fit4rfit');
42    squares=(sl-fcurve).**2;
43    sos=sum(squares);
44    StdDev=sqrt(diag(covp));
45    delta=exp(FP+StdDev)-exp(FP);
```

The logarithms in line `36` are calculated because equilibrium dissociation constants and rate constants will be varied in a logarithmic scale (`pin` in line `38`). The routine `leasqr` (lines `40` and `41`) calls the function `fit4fit` which in turn calculates binding equilibria from reaction scheme (8.5) with the function `Enz4EQF.m`.

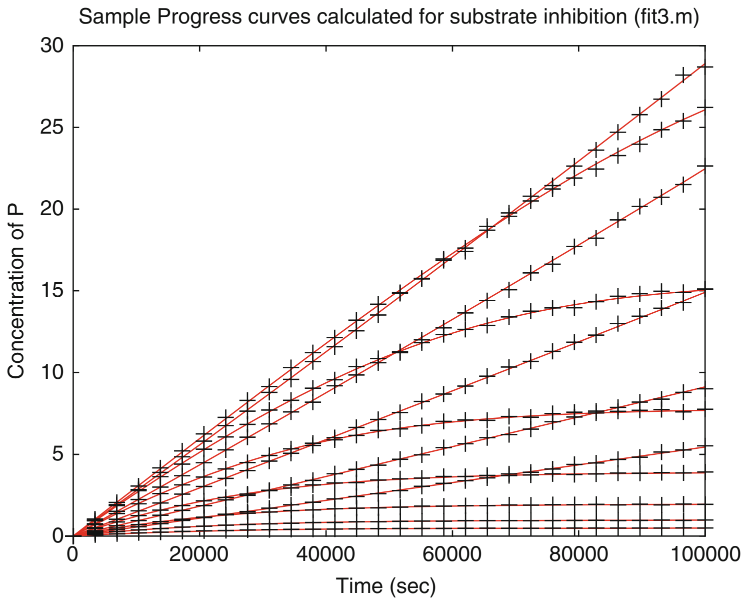


Fig. 8.4 Sample progress curves calculated for substrate inhibition. (−) theoretical progress curve for product concentration P1 in scheme (7.2) with $K_D1 = 100\ \mu M$, $K_D2 = K_D3 = 100$ mM. $k_1 = k_4 = 0.01\ \mu M^{-1}\ s^{-1}$. $k_2 = 1\ s^{-1}$, $k_3 = 0.05\ s^{-1}$, $k_5 = 0.001\ s^{-1}$. The enzyme concentration is 10 nM. The substrate concentration varied in a logarithmic scale from 1 μM to 1 mM. (+) simulated experimental data as variations of P1

$$\begin{tikzcd} & P1 & E+P2 & \\ E+2S \arrow[r, shift left, ""] \arrow[r, phantom, shift right=2, "K_D1"'] & ES+S \arrow[u] \arrow[l, shift left] \arrow[r, shift left] \arrow[r, phantom, shift right=2, "K_D2"'] & EP2+S \arrow[u] \arrow[l, shift left] \arrow[r, shift left] \arrow[r, phantom, shift right=2, "K_D4"'] & EP2S \arrow[l, shift left] \end{tikzcd} \quad (8.5)$$

A steady state equilibrium is established if the reversible reactions in (8.5) are much faster than the dissociation of products P1 and P2. Reaction scheme (8.5) is a simplification of scheme (7.2). It is part of this exercise for simplification strategies. The equations describing the steady state equilibrium are contained in lines 5–8 of the function `Enz4EQF.m`.

```
1      function F = Enz4EQF(x)
2      global KD1 KD2 KD4 E0 S0;
3      E=x(1);
4      S=x(2);
5      F(1)=E+E*S/KD1+E*S/(KD1*KD2)...
6      +E*S*S/(KD1*KD2*KD4)-E0;
7      F(2)=S+E*S/KD1+E*S/(KD1*KD2)...
8      +2*E*S*S/(KD1*KD2*KD4)-S0;
```

The fitting procedure itself is based on the function `leasqr` (lines `40` and `41` of the main program `fit4.m`). It requires the function `fit4fit.m` to calculate the theoretical initial velocities. Initial velocities are equal to $k_{cat} \cdot [ES]$. The substrate–enzyme complexes [ES] are calculated from the steady state equilibria at different substrate concentrations. Therefore, `fit4fit.m` solves the set of equations `Enz4EQF` in line `17`, calculates `VES`, the vector of [ES] in line `20`, and the initial velocities in line `22`. The sum of squares (`sos`) is calculated independent of the fitting algorithm in lines `42` and`43` of the main program `fit4.m`.

```
1      function mes=fit4fit(vs0,pin);
2      global KD1 KD2 KD4 E0 S0 MIN MAX N2 kcat;
3      ep=exp(pin);
4      for I=1:size(pin)
5      if (ep(I)<MIN(I)) pin(I)=log(MIN(I));
6      elseif (ep(I)>MAX(I)) pin(I)=log(MAX(I));
7      end;
8      end;
9      KD1=exp(pin(1));
10     KD2=exp(pin(2));
11     KD4=exp(pin(3));
12     kcat=exp(pin(4));
13     for k=1:N2
14     S0=vs0(k);
15     x0(1)=0.5*E0;
16     x0(2)=0.5*S0;
17     x = fsolve('Enz4EQF',x0);
18     E=x(1);
19     S=x(2);
20     VES(k)=E*S/KD1;
21     end;
22     mes=kcat*VES'; % Column vector
```

Any curve fitting procedure basically is a "try and error" method. The initial parameters (`pin`) are used to calculate an initial theoretical curve, from which the sum of squares (8.1) is calculated. Then the parameters are varied, the next sum of squares is obtained and compared with the first one, and so forth. Sometimes this variation of parameters leads to "local minima", which are combinations of parameters which do not give the optimal fit. Sometimes this leads to unreasonable values. Local minima can be avoided by changing `pin`, the vector of initial parameters. Unreasonable values can be avoided by setting limits to the fitted parameters. The MATLAB procedure `lsqcurvefit` allows selection of upper and lower limits. The Octave function `leasqr` does not have this extra, so that it is included in lines `3–8` of `fit4fit.m`.

`MIN` and `MAX` are the vectors of lower and upper limits of `pin`. They are defined in the main program and are transferred to `fit4fit.m` via the `global` statement in line `2`. Since `pin` itself contains the logarithms of parameters, they have to be re-calculated with the exponent function `exp` in line `3`, before they can be compared to the limits. Comparison between values is done with the `if` statement. This statement asks whether a condition is fulfilled. If the condition is fulfilled, a command or a series of commands is executed. If the condition is not fulfilled, a second `if` statement (`elseif`) may ask if a second condition is fulfilled and allows conditional execution of a second command or series of commands, until the `end` statement is encountered. The `if` statement of `fit4fit.m` in lines `5–7` has the following structure:

`if` (condition1) commands1; `elseif` (condition2) commands2; `end`;

Condition 1, `ep(I)` $<$ `MIN(I)` is true when the exponent of the fitted logarithm of a parameter is smaller than `MIN`. In this case, the fitted parameter will be set equal to the logarithm of `MIN` with the command `pin(I)` $=$ `log(MIN(I))`. Condition 2 is true when the exponent of the fitted parameter is larger than `MAX`, and consequently it then will be forced to be equal to `log(MAX)`. For the actual calculations, the varied parameters `pin` have to be translated into meaningful rate or equilibrium constants. This is done within `fit4fit.m` in lines `9–13`. Calculating the exponent from the logarithm has the additional advantage that the constants never reach negative values. Negative concentrations or rate constants simply do not make sense.

Note that the function `leasqr` in lines `40` and`41` of the main program `fit4.m` expects column vectors as arguments. The transpose operator ’ in line `22` of `fit4fit.m` is used for this purpose. Also note that the resulting parameters from `pin` in lines `9–12` also are defined in `global`. This may seem unnecessary, but sometimes one likes to repeat the fit and keep one parameter fixed. This can easily be achieved without re-writing the whole program, simply by inserting the percent symbol `%` in front of a line. This turns the line into a comment. For example, line `9` written as `%KD1` $=$ `exp(pin(1));` will not be executed. Because `KD1` is included in the `global` statement, this variable can be used inside the function `fit4fit.m`. Its value then is not varied and kept at the initial value defined in line `24` of `fit4.m`. The results of `fit4.m` are shown in Fig. 8.5.

For MATLAB users, the programs `fit4M.m` and `fit4fitM.m` are slightly different. The MATLAB routine `lsqcurvefit` allows the selection of `MIN (lb)` and `MAX(ub)` values directly, so that the `if` and `elseif` statements in lines `3–8` are not required. The function `lsqcurvefit` is rather flexible, but this flexibility requires options to be set. This is done in line `42` of the program `fit4M.m` with the function `optimset`.

```
42  options=optimset('TolFun',1e-10);
43  FP=lsqcurvefit('fit4fitM',pin,vs,sl,lb,ub,options);
44  fcurve=fit4fitM(FP,vs);
```

In general, MATLAB users may have to spend some time trying to find the best options for the function `lsqcurvefit`. This is similar to finding the optimal solver for differential equations. GNU Octave usually runs quite smoothly with its default parameters both for `leasqr` and for `lsode`.

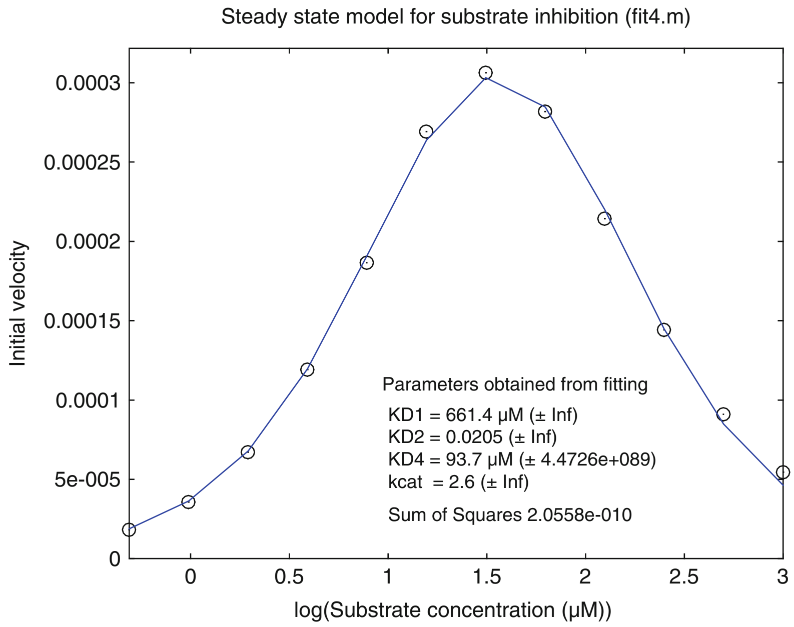


Fig. 8.5 Multi-parameter fit of initial velocities fitted to a steady state approximation of substrate inhibition (8.5). The "experimental" data from `fit3.m` [scheme (7.2)] were fitted to the steady state approximation with the program `fit4.m`. The fitted values are shown in the plot. The calculated standard deviations are excessively high, so that the resulting parameters are not reliable

Figure 8.5 shows a good fit of theoretical curves (straight line) to "experimental" initial velocities (o). However, the standard deviations are excessively high, so that the resulting parameters are not reliable. The standard deviations of the fitted parameters are calculated in line `44` of the main program `fit4.m` from the diagonal elements of the covariance matrix `covp` (result from `leasqr` in lines `40` and`41`). These standard deviations are the standard deviations of the logarithmic scale used for the fitting process. They are translated to `delta` values of the linear scale in line `45`.

The Octave function `leasqr` also returns a correlation matrix `corp` (lines `40` and `41`). The matrix is displayed by typing `corp` in the Octave terminal window after running the program. A correlation matrix contains as many rows and columns as there are parameters. The respective correlation coefficients are shown at the respective intersections of rows and columns. The correlation of parameter 2 (KD2) and 4 (kcat), for example, is the matrix element `corp(2,4)`, which is identical to the matrix element `c orp(4,2)`. Correlation coefficients can vary between −1 and 1. The absolute value gives the amount of correlation ranging from 0 (not correlated) to 1 (strongly correlated). Positive correlation coefficients of two parameters indicate that an increase of parameter 1 can be compensated with an increase of parameter 2, whereas negative values indicate that an increase of

Table 8.1 Matrix of correlation coefficients from the fit of Fig. 8.5 calculated with the program `fit4.m`

	KD1	KD2	KD4	kcat
KD1	1	−1	1	1
KD2	−1	1	−1	−1
KD4	1	−1	1	1
kcat	1	−1	1	1

parameter 1 can be compensated by a decrease of parameter 2. Each parameter is correlated with itself, so that the diagonal elements of any correlation matrix are 1. Every time a correlation coefficient with an absolute value near 1 appears outside the diagonal, the values are heavily correlated and cannot be regarded as independent results. The correlation matrix resulting from `fit4.m` is shown in Table 8.1.

Indeed, the correlation matrix in Table 8.1 shows heavy correlation of parameters, which explains why the standard deviations of the parameters are so high. Here, we follow the strategy of model simplification. We therefore will not try to keep one parameter fixed, but instead will further simplify the model. Reaction scheme (8.6) is the simplest conceivable model to calculate substrate inhibition under steady state conditions.

$$E+2S \underset{K_D1}{\rightleftharpoons} ES+S \underset{K_D4}{\rightleftharpoons} ES_2 \tag{8.6}$$

The simplified scheme (8.6) is calculated with the program `fit4r.m`, which calls the fitting routine `fit4rfit.m`, which in turn solves the set of equations `Enz4EQRF.m`. The result is shown in Fig. 8.6. These programs are very similar to the originals (`fit4.m`, `fit4fit.m` and `Enz4EQF.m`) and need not be discussed.

```
function F = Enz4EQRF(x)
global KD1 KD4 E0 S0;
E=x(1);
S=x(2);
F(1)=E+E*S/KD1+E*S*S/(KD1*KD4)-E0;
F(2)=S+E*S/KD1+2*E*S*S/(KD1*KD4)-S0;
function mes=fit4rfit(vs0,pin);
for k=1:N2
S0=vs0(k);
x0(1)=0.5*E0;
x0(2)=0.5*S0;
x = fsolve('Enz4EQRF',x0);
E=x(1);
S=x(2);
VES(k)=E*S/KD1;
end;
mes=kcat*VES';
```

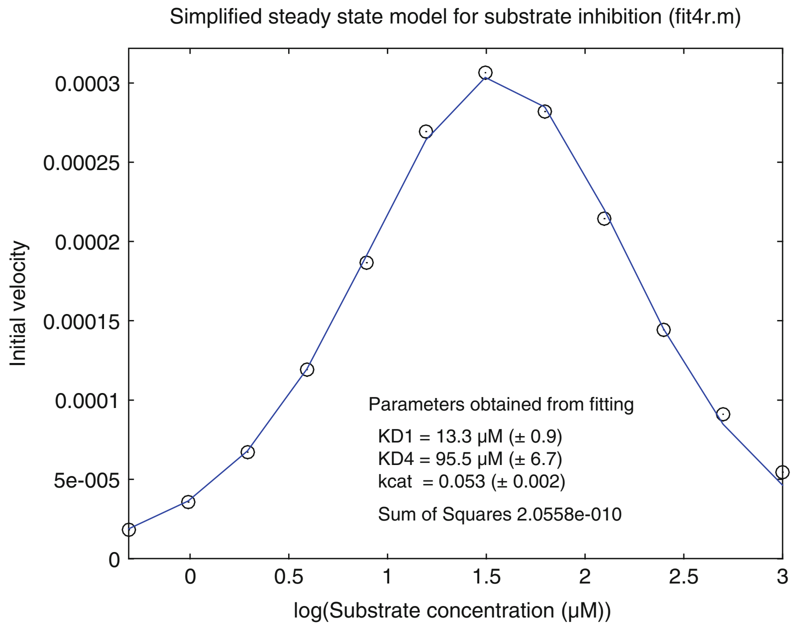


Fig. 8.6 Multi-parameter fit of initial velocities for substrate inhibition. The "Experimental" data from `fit3.m` were fitted to the steady state approximation (8.6) by means of `fit4r.m`. The fitted parameters and their standard deviations are printed within the plot. The correlation coefficients are listed in Table 8.2

Table 8.2 matrix of correlation coefficients from the fit of Fig. 8.6 calculated with the program `fit4r.m`

	KD1	KD4	kcat
KD1	1.0	−0.85	0.94
KD4	−0.85	1.0	−0.94
kcat	0.94	−0.94	1.0

The correlation matrix `corp` for the fit of Fig. 8.6 is shown in Table 8.2. The fit is excellent, and gives the same sum of squares as for reaction scheme (8.5). The errors are smaller, so that individual parameters can be determined with much better precision. The parameters in Table 8.2 still are correlated, but not as heavy as in Table 8.1. Reaction scheme (8.6) cannot be simplified any more.

The numeric results of a fit to reaction scheme (8.6) could not have been inferred from the calculated reaction of scheme (7.3). Or to summarize it more ruthlessly: The simplified reaction scheme (8.6) may be appropriate for fitting initial velocities with a minimum of parameters, but does not reflect the underlying mechanism.

8.4.3 Global Fit of Progress Curves (`fit5.m`)

A global fit of progress curves is the most rigorous method to analyze enzyme kinetics. It takes into account all available experimental information. For the global fit calculated with `fit5.m`, these are the progress curves of the matrix `Da` generated by `fit3.m` and stored in `SubInhib.mat`. The octave function `leasqr` allows entering sets of different experiments only when they are arranged into one single column vector. The matrix (spreadsheet) `Da` (Fig. 8.4) is arranged in `N2` = 12 rows, corresponding to the 12 concentrations, and `N` = 30 columns, corresponding to 30 time points. This matrix is turned into a single column vector of $12 \times 30 = 360$ rows with the commands in lines `46–50`. The name of the linearized data set is `lDa`. The function `leasqr` requires a vector of the same size for the independent variable (the time). Such a vector is created in line `46` with the `zeros` command. It creates a column vector `lt` of `N* N2` rows containing zeros. This vector `lt` is just a dummy.

```
46    lt=zeros(N*N2,1);
47    for k=1:N2
48    for j=1:N
49    lDa(j+(k-1)*N,1)=Da(j,k);
50    end;end;
```

The real time vector is ct, which is handed over to the function `fit5fit` by means of the `global` statement in lines `23–24` of `fit5.m`, just as lines `2–3` of `fit5fit.m`, listed below. Of course, the theoretical fit has to have the same structure as the experimental data in the main program, and therefore the fitted values `mes` returned from the function `fit5fit` consist of same single column vector format, lines `27–30`. `mes` is calculated from the matrix of product concentrations `P(:,k)` at different substrate concentrations `k` (line `29`).

Reaction scheme (7.2) uses nine different rate constants, but the experiments do not allow the determination of all of them. Enzyme kinetics is measured under steady state conditions, so that the association rate constants k_1 and k_4 cannot be fitted. They were set arbitrarily to the relatively high value of 0.1 μM^{-1} s^{-1}. Likewise, enzyme kinetics with excess of substrate do not allow the measurement of the back reaction, so that the equilibrium dissociation constants K_D2 and K_D3 could not be fitted and were set arbitrarily to a high value (10 mM). This leaves five parameters to be fitted, namely K_D1, K_D4, k_2, k_3 and k_4. The logarithms of their initial estimates are the elements of the vector `pin`, and the exponent function must be used to extract the parameters from `pin` in lines `10–15` of `fit5fit.m`. The rate constants for the back reactions are calculated from the association rate constants and the equilibrium dissociation constants in lines `17–21`, so that they can be passed to the differential equations (`enz4F.m`) with the `global` statement.

```
1      function mes=fit5fit(lt,pin);
2      global k1 k2 k3 k4 k5 km1 km2 km3 km4 km5...
3      VS0 E0 KD1 KD2 KD3 KD4 MIN MAX N N2 ct;
4      ep=exp(pin);
5      for I=1:size(pin)
6      if (ep(I)<MIN(I)) pin(I)=log(MIN(I));
7      elseif (ep(I)>MAX(I)) pin(I)=log(MAX(I));
8      end;
9      end;
10     KD1=exp(pin(1));
11     KD4=exp(pin(2));
13     k2=exp(pin(3));
14     k3=exp(pin(4));
15     k5=exp(pin(5));
17     km1=KD1*k1;
18     km2=k2/KD2;
19     km3=k3/KD3;
20     km4=k4*KD4;
21     for k=1:N2
22     S0=VS0(k);
23     x0=[E0;S0;0;0;0;0;0];
24     M=lsode('enz4F',x0,ct);
25     P(:,k)=M(:,5);
26     end;
27     for k=1:N2
28     for j=1:N
29     mes(j+(k-1)*N,1)=P(j,k);
30     end;end;
```

One important feature is missing in our example of theoretically calculated data: There are no real experimental errors. Typically, one would expect background variations between the different kinetic experiments. They have to be included as additional parameters. Since they only concern one subset of the experimental data, these fluctuations are not correlated with the other parameters in the fitting procedure. It is better to consider the variation between experiments as additional fitting parameters than to ignore it and get meaningless results.

Once the data have been computed in the global fitting routine as a linear vector `fcurve` (line `51`), they have to be re-arranged in lines `59`–`62`, so that `theo(j,k)` is the theoretical value for each data point `Da(j,k)`, to be used in line `65` for the `plot` command.

```
22     filename='fit5';
23     global k1 k2 k3 k4 k5 ...
24     VS0 E0 KD1 KD2 KD3 KD4 MIN MAX N N2 ct;
25     load('SubInhib.mat');
```

```
51    [fcurve,FP,kvg,iter,corp,covp]=...
52    leasqr(lt,lDa,pin,'fit5fit');
53    save('-ascii',[filename,'_corp.txt'],'corp');
54    squares=(lDa-fcurve).**2;
55    sos=sum(squares);
56    StdDev=sqrt(diag(covp));
57    low=exp(FP-2*StdDev);% 95% confidence interval
58    high=exp(FP+2*StdDev);% 95% confidence interval
59    for k=1:N2
60    for j=1:N
61    theo(j,k)=fcurve(j+(k-1)*N,1);
62    end;end;
63    %% Plot
64    fill=zeros(N,14-N2);
65    plot(ct,Da,'+',ct,fill,ct,theo,'-');
```

The statistical information of the covariance matrix `covp` is used to calculate the 95% confidence interval for the fitted parameters. This is done in lines `56`–`58`. Remember, that for a Gauss distribution 95% of the events are expected in an area of ± two times the standard deviation around the mean value. In lines `57` and `58`, `low` and `high` therefore denote the lower and upper limits of the confidence interval. The matrix of correlation coefficients `corp`, computed with `leasqr`, is saved in `ascii` format in line `53` of `fit5.m`. It can be retrieved with any editor, or one simply can type `corp` in the Octave window. The matrix only contains the numbers of Table 8.3, not the names of the parameters. The parameter names correspond to the order of parameters in the vector pin (line `43`).

The number of data displayed in Fig. 8.7 becomes rather large, so that their length is reduced by means of formatted output. This is specified for the `num2str` commands in lines `72`–`81`. For example, the command `num2str(KD1,'% 4.1f')` used in line `72` turns the number `KD1` into a string. The output format `'%4.1f'` uses four parameters, the percent sign (`%`) to specify that a conversion to formatted output is expected, a letter (`f`, in this case) to specify conversion to a fixed point notation, a number (`4`) to specify the minimum field width and a number (`1`) to specify the decimal precision. This gives the resulting string `51.9` shown in Fig. 8.7. Other useful output conversions are `%e` or `%E` to specify exponential

Table 8.3 Matrix of correlation coefficients from the fit of Fig. 8.7 calculated with the program `fit5.m`

	KD1	KD4	k2	k3	k5
KD1	1.0	1.0	1.0	−1.0	−0.02
KD4	1.0	1.0	1.0	−1.0	−0.04
k2	1.0	1.0	1.0	−1.0	−0.02
k3	−1.0	−1.0	−1.0	1.0	0.03
k5	−0.02	−0.04	−0.02	0.03	1.0

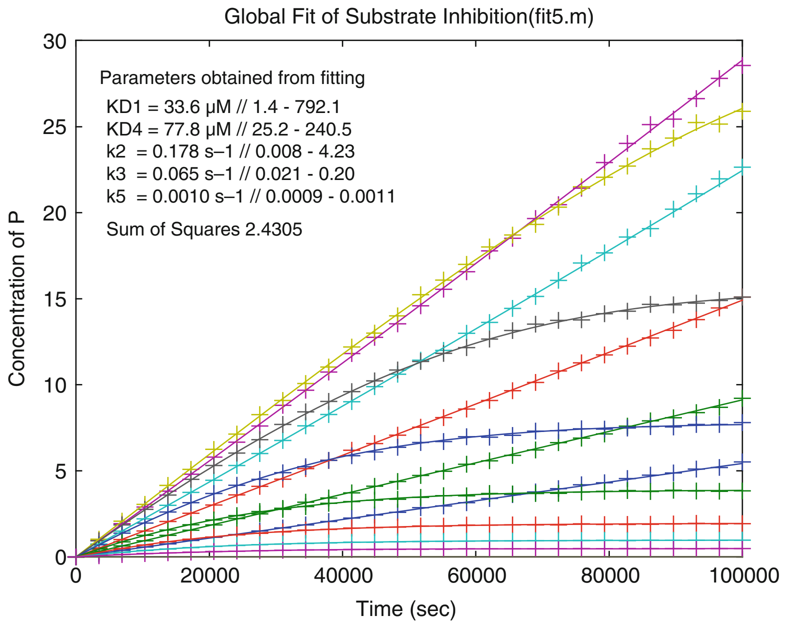


Fig. 8.7 Global fit of substrate inhibition. The "Experimental" data from `fit3.m` ($K_D1 = K_D4 = 100$ μM, $K_D2 = K_D3 = 100$ mM. $k_1 = k_4 = 0.01$ μM^{-1} s^{-1}. $k_2 = 1$ s^{-1}, $k_3 = 0.05$ s^{-1}, $k_5 = 0.001$ s^{-1}) were fitted to reaction scheme (7.2). (–) calculated progress curve (+) simulated experimental data from `fit3.m`. The resulting parameters are shown inside the plot together with a 95% confidence interval. The correlation coefficients are listed in Table 8.3

notation and `%g` or `%G` for either normal (fixed point) or exponential notation, depending on the magnitude.

Running the program `fit5.m` may take several minutes, depending on the computer. Figure 8.7 shows a perfect global fit of the progress curves of substrate inhibition. The confidence intervals, however, were rather large. Note that "experimental" data were generated with random noise in `fit3.m`, so that each data set is different. Only k_5 could be determined with a reasonable accuracy. Likewise, the only rate constant which is not correlated to the other ones is k_5. Indeed it had been reported [8] that this step has a profound influence on the characteristics of substrate inhibition.

Rather than simplifying reaction schemes as in Sect. 8.4.2, we will use a different strategy. We will stick to scheme (7.2), but decrease the number of parameters. At this stage, one typically would take plausible estimates from the literature for individual rate constants or affinities. Here we take the linear regression of the initial part of the Lineweaver–Burk plot (enz4.m, Fig. 7.9) and assume that our K_D1 corresponds to the K_M estimated there. The global fit fit5b.m therefore keeps K_D1 fixed to 10.8 μM. The result is shown in Fig. 8.8.

The quality of the fit shown in Fig. 8.8 is excellent. The 95% confidence intervals of the parameters are rather low, so that one may tend to believe in the results.

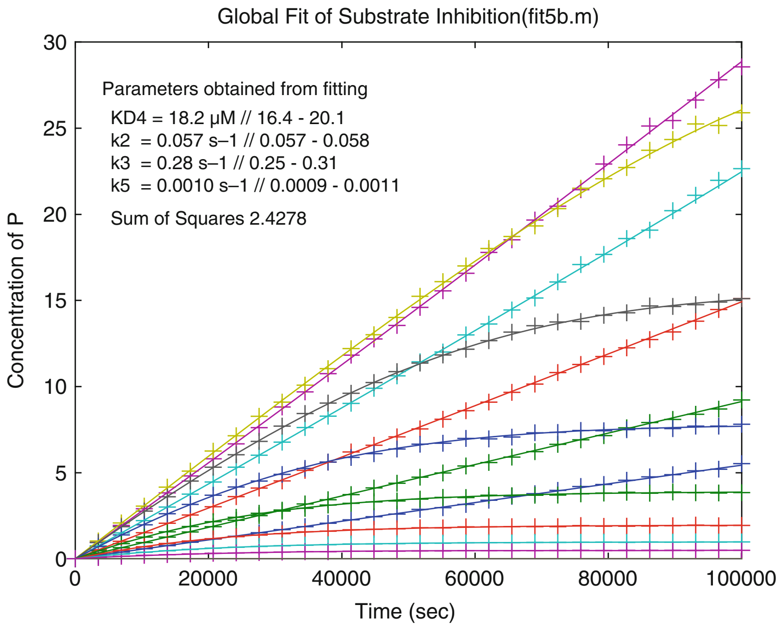


Fig. 8.8 Global fit of substrate inhibition. The "Experimental" data from `fit3.m` ($K_D1 = K_D4 = 100$ µM, $K_D2 = K_D3 = 100$ mM. $k_1 = k_4 = 0.01$ μM^{-1} s^{-1}. $k_2 = 1$ s^{-1}, $k_3 = 0.05$ s^{-1}, $k_5 = 0.001$ s^{-1}) were fitted to reaction scheme (7.2) with KD1 = 10.8 µM taken from the approximation of Fig. 7.9. (–) calculated progress curve (+) simulated experimental data from `fit3.m`. The resulting parameters are shown inside the plot together with a 95% confidence interval. The correlation coefficients are listed in

However, the fitted values do not correspond to the original ones. The "experimental" data were generated with $K_D4 = 100$ µM, but were fitted with 18.2 µM; k_2 was fitted with 0.057 s^{-1} instead of the expected 1 s^{-1}; k_3 was fitted to 0.28 s^{-1} instead of 0.05 s^{-1}. Only k_5 was determined correctly as 0.001 s^{-1}.

The reason for this discrepancy lies, of course, in the assumption that K_D1 corresponds to the Michaelis constant K_M of 10.8 µM determined in enz4.m. It is obvious from the intrinsic parameters listed in Fig. 7.9 that this assumption is wrong. Since all these parameters are correlated, as shown in Table 8.3, one wrong number leads to a domino effect, giving wrong values for all correlated parameters. Only k_5 is not correlated and therefore was not affected by the wrong choice of K_D1.

The final results in all sample programs are plotted with the plot command, which is based on gnuplot, a powerful tool on its own [9]. For scientists who are not familiar with it may be easier to save their results in ascii format (see, for example, lines `39` and `40` of `EQ1.m`) and produce publication quality plots with a known software (Table 8.4).

Global fits are the most rigorous type of quantitative data analysis in life sciences. A successful fit always proves that the underlying molecular model is sufficient to explain all fitted experimental observations. The parameters from a multi-parameter

Table 8.4 Matrix of correlation coefficients from the fit of Fig. 8.8 calculated with the program `fit5b.m`

	KD4	k2	k3	k5
KD4	1.0	0.96	−1	−0.58
k2	0.96	1.0	−0.97	−0.43
k3	−1	−0.97	1.0	0.54
k5	−0.58	−0.43	0.54	1.0

fit need not be unique. When parameters are correlated, there may be infinite combinations of parameters which fit the data equally well. Reducing the number of parameters increases their reliability, but one invalid assumption can corrupt the lot.

References

1. http://en.wikipedia.org/wiki/Simple_linear_regression
2. http://en.wikipedia.org/wiki/Correlation_and_dependence
3. Croxton FE, Cowden DJ, Klein S (1968) Applied general statistics. Pitman, London, p 625
4. Dietrich CF (1991) Uncertainty, calibration, and probability: the statistics of scientific and industrial measurement. Inst of Physics Pub, London, p 331
5. Aitken AC (1936) Statistical mathematics. Oliver and Boyd, Edinburgh, p 95
6. Rodgers JL, Nicewander WA (1988) Thirteen ways to look at the correlation coefficient. Am Stat 42:59–66
7. Hofer P, Fringeli UP (1981) Acetylcholinesterase kinetics. Biophys Struct Mech 8:45–59
8. Krupka RM, Laidler KJ (1961) Molecular mechanisms for hydrolytic enzyme action. J Am Chem Soc 83:1445–1460
9. Janert PK (2009) Gnuplot in action. Understanding data with graphs. Manning Publications Co, Greenwich, CT

Chapter 9
Appendix: Installing GNU Octave for Mac OS and Linux

9.1 Installation Instructions for Mac OS X

Octave for Mac OS X can be downloaded from the GNU webpage http://www.gnu.org/software/octave/download.html. Select Octave.app for Mac OS X and with regard to your processor choose the ocatve-3.2.X-i386.dmg (for Macs newer than 2006 based on Intel processors) or octave-3.2.X-ppc.dmg (for older Macs before 2006 based on Power PC processors). Open the downloaded dmg-package with a double click and drag the icon “Octave” from the open window to the icon “Applications”. That will install Octave. The sample programs in this book require Gnuplot and additional packages.

Gnuplot is installed from the folder “Extras” in the package octave.app.dmg, which had been opened before, by double clicking the gnuplot package. This will open the gnuplot installation package. There, the icon “Gnuplot” has to be dragged to the icon “Applications”.

Ocatave can be expanded with different additional packages. These require the “XCode Apple’s developers tools” for installing. These Apple tools can be installed from the Mac OS X installation DVD listed under “optional installs” or downloaded from Apple’s web site after registration. One has to double click the package “Xcode.mpkg” and follow the installation instructions using the default values. Afterwards, the “X11” package from the “Optional Installs.mpkg” in the same folder has to be installed. Double click the icon and follow the installation instructions, choose “Custom Install”, select the package “X11” found under “Applications” and continue the installation, just follow the instructions.

After installing “XCode” and “X11” your system is ready to install the required Octave packages “miscellaneous”, “io”, “struct" and "optim”. They are found on the octave website (http://octave.sourceforge.net/packages.php). Mac OS X saves the files in the folder “Downloads” in your home directory. Having done this you can run octave now from the applications. A white window, and the terminal console will open. Now type in

```
pkg install Download/miscellaneous-1.0.10.tar.gz
```

The package in this example is called “miscellaneous” in version 1.0.10 (latest version in time of print). If you are downloading a newer version change the command line according to your package version. Install the other packages according to your download path and their version names:

```
pkg install Downloads/io-1.0.13.tar.gz
pkg install Downloads/struct-1.0.8.tar.g
pkg install Downloads/optim-1.0.15.tar.gz
```

You can ignore warnings but be careful for error messages. There should be no error messages if you have installed all required packages and followed the above procedure. Verify that all packages are installed by typing `pkg list`, and the output should be similar to

```
octave-3.2.3:7> pkg list
Package Name   | Version | Installation directory
---------------+--------+---------------------
           io *|  1.0.13 |
/Volumes/HD/Users/username/octave/io-1.0.13
miscellaneous *|  1.0.10 |
/Volumes/HD/Users/username/octave/miscellaneous-1.0.10
        optim *|  1.0.15 |
/Volumes/HD/Users/username/octave/optim-1.0.15
       struct *|   1.0.8 |
/Volumes/HD/Users/username/octave/struct-1.0.8
```

Congratulations! Octave can now be used with Mac OS for all sample programs of this book.

9.2 Installation Instructions for Linux

The operating system GNU/Linux comes in many distributions. Here, the installation for “Ubuntu Linux” and all other distributions based on the debian package archive is shown. Simply start your Linux terminal and run

```
sudo apt-get install octave3.2 octave-miscellaneous
octave-io octave-optim
```

This will install all necessary packages for running all examples in this book. After installing you will find Octave in the Gnome menu. Go to “Applications”, then “Programming” and click on “GNU Octave”. If you are using a terminal you just have to type “`octave`”. Now you can test if all necessary packages have been installed by typing “`pkg list`” in the octave window. The output shows all additional packages for octave with their version numbers similar to the following:

```
octave-3.2.4:1> pkg list
Package Name   | Version | Installation directory
---------------+---------+----------------------
           io *|  1.0.12 |
/usr/share/octave/packages/3.2/io-1.0.12
miscellaneous *|   1.0.9 |
/usr/share/octave/packages/3.2/miscellaneous-1.0.9
        optim *|  1.0.12 |
/usr/share/octave/packages/3.2/optim-1.0.12
```

Index