DynaFit
Scripting Manual

ver. 3.28

Petr Kuzmič, Ph.D.

Petr Kuzmič
BioKin, Ltd.
1652 S. Grand Ave. Ste. 337
Pullman, WA 99163
http://www.biokin.com
pkuzmic@biokin.com

Published by BioKin Press
Sixth Edition, July 2005
Typeset in LaTeXby the author.

DYNAFIT SOFTWARE – ACADEMIC LICENSE

This software is made freely available to researchers affiliated with academic institutions (schools, universities, government institutes, and other not-for-profit organizations), provided that any publication resulting from the use of the program contains the bibliographic reference below.

> Kuzmič, P. (1996) "Program DYNAFIT for the analysis of enzyme kinetic data: Application to HIV proteinase." *Anal. Biochem.* **237**, 260–273.

Users affiliated with commercial for-profit establishments must obtain proper SOFTWARE LICENSE by writing to the address above.

Contents

Chapter 1

Introduction

All input data for program DynaFit [1] are simple text files in the ASCII format. The input can be classified into three categories:

1. *Data files* contain experimental data arranged in columns (independent variable vs. dependent variable).
2. *Script files* contain the description of the reaction mechanism, initial estimates of fitting parameters, and the location of experimental data files on the disk.
3. *Initialization files* contains the control settings for numerical algorithms, such as the desired confidence level for estimated parameters.

This manual contains the formal description of DynaFit script files such as the example script listed below.

```
; Fit progress curve to Michaelis-Menten mechanism
;
[task]
   data = progress
   task = fit
[mechanism]
   E + S <==> ES     :     k   ks
   ES --> E + P      :     kr
[constants]
   k = 1 ?, ks = 10 ?, kr = 1 ?
[concentrations]
```

```
   E  = 0.05
   S  = 31
[responses]
   P = 3.226 ?
[progress]
   file        ./data/steroid/i0.txt
[settings]
   <Filter>    | Scale = minutes
[end]
```

1.1 Available tasks

The main task to be accomplished by the program DynaFit is summarized in the `[task]` section of the script file.

The section `[task]` must contain the keywords `task = ..` and `data =..` . The reaction mechanism may be optionally identified by the keyword `mechanism = ...` These keywords must stand on separate lines.

Example 1

```
[task]
   task = simulate
   data = velocities
   mechanism = mixed type
```

Example 2

```
[task]
   task = fit
   data = progress
   mechanism = slow tight inhibition
```

Example 3

```
[task]
   task = compare
   data = equilibria
   mechanism = displacement
```

1.1.1 Task

The keyword `task` must be followed by the equal sign ("`=`") and by one of the following keywords:

`simulate`	Simulate pseudo-experimental data, with or without superimposed random error.
`compare`	Compare the given experimental data and the postulated fitting model, with preliminary diagnostics for the goodness of fit. This option is useful in making initial estimate of fitting parameters.
`fit`	Nonlinear least-squares fit of the postulated theoretical model to the given experimental data.

1.1.2 Data

The keyword `data` must be followed by the equal sign ("`=`") and by one of the following keywords:

`progress`	The measured data (*i.e.*, the dependent variable) are observations of a physical variable such absorbance, fluorescence, or radioactivity over time (*i.e.*, the independent variable).
`velocity`	The data represent measurements of the initial reaction rate. The independent variable is the (initial) concentration of a certain reagent.
`equilibria`	The data are measurements of a certain physical variable (absorbance, fluorescence, radioactivity) observed on a chemical or a biochemical system at equilibrium. The independent variable is the concentration of a certain reagent.

1.1.3 Mechanism

The keyword `mechanism` is optional, unless a model discrimination analysis is required. It is followed by the equal sign and by an arbitrary text identifying the reaction mechanism to the user. The text should be short, at most 32 characters.

Model discrimination analysis

The keyword `mechanism` is required if a given DynaFit script contains multiple tasks, each of which requests the least-squares fit to a different reaction mechanisms. Typically, such model discrimination analysis involves the same set of experimental data. In this case the keyword `mechanism` must be present. Model discrimination analysis is indicated by appending the question mark ("`?`") after the name of each reaction mechanism.

Example

In this example, DynaFit will fit the experimental data indicated in the script file (omitted for brevity and represented by the ellipses) to three different reaction mechanisms. After the analysis is complete, DynaFit will compute a set of statistics deciding which reaction mechanism is most plausible ("model discrimination analysis").

```
[task]
   data = velocity
   task = fit
   mechanism = competitive ?
...
[task]
   data = velocity
   task = fit
   mechanism = un-competitive ?
...
[task]
   data = velocity
   task = fit
   mechanism = mixed type ?
...
```

```
[end]
```

Each DynaFit script may contain in principle an unlimited number of consecutive tasks. The script file sections delineated by the keywords `[task]` or `[end]` must contain other sections, as is described in the following portions of this chapter.

Chapter 2

General Considerations

This chapter of the DynaFit scripting manual describes the general considerations important in the preparation of the script files:

- formatting (case sensitivity, special characters);
- sections;
- keywords;
- ranges and sets of values;
- concentration and time scale;
- optimized parameters.

2.1 Formatting of the script files

2.1.1 Case sensitivity

The distinction between upper and lower case is ignored everywhere in the script files, which is also the case for all other input files (experimental data, initialization files). Unix systems are exceptional in that all *file names* are case sensitive. This is a property of the operating system, not the program.

2.1.2 White space

Position of text on a line is irrelevant because "white space" or indentation is ignored. Blank lines are ignored also. The only instance of white space being required is in the definition of reaction mechanisms. If a species `A` reacts with a species `B`, the plus sign in writing `A + B --> AB` must be surrounded by white space. Without the blank spaces surrounding the plus sign, the program would consider `A+B` to be a single molecular species.

2.1.3 Comments

Script files can contain comments delimited by the semicolon. Any text that follows a semicolon on the given line is ignored by the program. Thus the text

```
   E + S <===> E.S  :  k1  k2  ;  Michaelis complex
 ; E + I <===> E.I  :  k3  k4
   files  f1, f2, f3, f4 ; f5, f6
```

becomes

```
   E + S <===> E.S  :  k1  k2

   files  f1, f2, f3, f4
```

when the script file is read by the program.

2.1.4 Special characters

The following characters have special meaning: `< > [ ] : + * ? | ;`. Detailed explanation of how these special characters are used is given in appropriate sections of this manual. A brief summary is given in Table 2.1.

2.2 Sections

Each script file is divided into several *sections* described separately of this manual. Table 2.2 contains a brief list of section names.

< >	in the [mechanism] section: construction of arrows; in the [settings] section of the script file and in the default settings file: names of sections for default settings
[ ]	names of main sections in the script file
:	in the [mechanism] section: separating reaction steps from names of rate constants
*	in the [mechanism] section: making a reaction step 'slow' compared to other steps (see below)
|	line break
?	optimized (adjustable, fitting) parameter
+	in the [mechanism] section: separates reactants in a reaction step
;	comments (anything beyond semicolon (;) on any given line is ignored)

Table 2.1: Special characters.

Only the first four characters of each section name are used by the program, so [mechanism] can be abbreviated as [mech, [concentrations] can be abbreviated as [conc, etc. Each section name must begin on a separate line. There can be no spaces separating the [character and the section name, but any number of leading blank spaces or tabs are allowed.

2.3 Keywords

Each section of the script uses a collection of *keywords*. The precise meaning of each keyword is explained in relevant parts of this manual. Here we give a complete collection of keywords applicable in the script files (note that the initialization files use a different collection of symbols representing various numerical parameters).

Keywords

```
add
all
association
auto
compare
concentration
constant
```

Section	
`[abstract]`	Summary of a comment section in the script file.
`[concentrations]`	Initial or total concentrations for all datasets in the script.
`[constants]`	Values of rate and equilibrium constants.
`[end]`	End of the input script.
`[equilibria]`	Equilibrium binding data.
`[keywords]`	Keywords in a comments section of the script file.
`[mechanism]`	Reaction mechanism.
`[output]`	Directory for output files.
`[progress]`	Reaction progress curves.
`[responses]`	Molar response coefficients of reaction species.
`[settings]`	Fine-tuning the computational parameters.
`[subtitle]`	Subtitle of a comment section.
`[sweep]`	Simulate several progress curves by 'sweeping' the values of rate constants.
`[task]`	Mode of operation (simulation, comparison, or fit).
`[title]`	Title of a comment secrion.
`[velocities]`	Initial velocity data.

Table 2.2: Script file section names.

```
cubic
delay
dilute
directory
dissociation
dixon
error
extension
equilibrate
equilibrium
file
files
fit
from...to...step
graph
linear
lineweaver-burk
local
```

```
logarithmic
mechanism
mesh
observe
offset
percent
plot
rapid
response
simulate
task
quadratic
variable
vary
velocity
zero
```

2.4 Ranges and sets of values

In several sections of the script file, it is possible to indicate in a condensed fashion a whole range of numerical values. For example in the `[data]` section we can assign a set of concentrations to a set of primary data files simply by listing the concentrations one after each other, *separated by comma*.

Example

```
             files     f1,  f2,  f3,  f4,  f5
vary concentration S = 1.0, 2.0, 3.0, 4.0, 5.0
```

In other sections of the script file we can specify a set of values by using the combination `from x1 to x2 step x3`, which assumes a linear scale for the set of values and creates a sequence $x_1, x_1 + x_3, x_1 + 2x_3, x_1 + 3x_3, \ldots$, or `from...to...step logarithm`, which creates the logarithmic sequence $x_1, x_1 \times x_3, x_1 \times x_3^2, x_1 \times x_3^2 \ldots$.

Example 1

Generate the series of values 1.0, 2.0, 3.0, 4.0, and 5.0 for the concentration of substrate `S`:

```
variable S = from 1.0 to 5.0 step 1.0
```

Example 2

Generate the series of values 1, 2, 4, 8, and 16 for the concentration of substrate `S`:

```
variable S = from 1 to 16 step 2 logarithmic
```

Often we need a logarithmically spaced series of values which however includes zero as the first (lowest) element. For this purpose, it is necessary to add the notation `add zero` to the line of text on which the logarithmic series is defined:

Example 3

Generate the series of values 64, 32, 16, 8, 4, 2, 1, and 0 (!) for the concentration of inhibitor `I`:

```
variable I = from 64 to 1 step 0.5 logarithmic, add zero
```

2.5 Concentration and time scale

It is important to discuss the issue of properly *scaling* all concentrations in such a way that round-off errors are minimized. It is also important to remember that the time unit of all rate constants (for example reciprocal seconds or minutes) must agree with the time unit of the experimental data.

2.5.1 Concentration scale

Optimally all concentrations would take on numerical values that differ from unity at most by three orders of magnitude.

For example, if the typical enzyme concentration in a series of experiments is 10 nM, and the typical concentration of the substrates and inhibitors is between 10 and 100 μM, then we should choose micromolar scale for all concentrations. The reason is that 10^{-6} is between 10^{-8} M for the enzyme and 10^{-4} M for the substrate. In this way both the numerical value

of enzyme concentration (0.01 μM) and the numerical value of the substrate concentration (100 μM) differ from unity at most by two orders of magnitude.

Once a proper scale of concentrations has been determined, it affects the nominal values of two other quantities, namely, the bimolecular association rate constants and the specific molar responses. For example, if all concentrations are expressed in μM, than all bimolecular association rate constants must be expressed in $\mu M^{-1}sec^{-1}$ and all molar responses in signal (e.g. absorbance) change per μM.

Example

In a series of protease assays, the concentration of the enzyme was 1 nM and the concentration of the substrate was 100 μM. The hydrolysis of a chromogenic peptide substrate was followed at spectrophotometrically. At the given wavelength, the difference molar absorption coefficient is -1,300, meaning that a complete cleavage of one mole of the substrate would produce a decrease of absorbance by 1,300 units in a one centimeter cell.

In this case the proper concentration scale is micromolar, which means that the nominal concentration of the enzyme is 0.001 (micromoles per liter), and the nominal concentration of the substrate is 100 (micromoles per liter). Assuming that the bimolecular association rate constant is 10^8 $M^{-1}sec^{-1}$, the nominal value is 100 (liter per micromole per second). The nominal value of the difference absorption coefficient is -0.0013 (absorbance units per micromole per liter per centimeter).

2.5.2 Time scale

The time scale of the experimental data must agree with the time scale of the rate constants. Most published values of rate constants for biochemical reactions are in reciprocal seconds. Therefore it is useful to convert all progress curve data files in such a way that the readings of time are in seconds. DynaFit can convert existing data files automatically, by properly setting the option `Scale` in the `[Filter]` section of the *initialization file*.

Similarly, all initial velocity data should be transformed in such a way that the reaction rates are expressed in concentrations (or other units such as absorbance or fluorescence intensity) per second. If the initial velocity

data were not generated by DynaFit, it might be necessary to convert the data manually. DynaFit does not have the ability to convert the time-scale of initial velocity data from minutes to seconds.

> All experimental data and fitting parameter (rate constants, concentrations, and molar responses) must use *identical units*. It is important to choose concentration units in such a way that the numerical values of concentrations are close to unity.

2.6 Optimized parameters

In several sections of the script file certain parameters (rate constants, initial or total concentrations, molar responses, or offsets on the signal axis) can be designated as optimized or adjustable fitting parameters. An adjustable parameter is indicated simply by appending a question mark (`?`) after the value of the initial estimate. The question mark can either follow immediately after the numerical value, or it can separated by any number of blank spaces.

> *Example*: Optimized rate constants k_s and k_r and initial concentration of substrate S.

```
[mechanism]
   E + S <==> ES      :      k   ks
   ES --> E + P       :      kr
[rate constants]
   k  = 1, ks = 10?, kr = 1 ?
[concentrations]
   S = 2.34 ?
```

Optionally the programs performs a confidence interval estimation for selected rate constants (but not for other kinds of parameters, such as concentrations or molar responses). These rate constants or equilibrium constants are identified by appending to their initial estimate *two* question marks (without any blank spaces between them).

In the example below, both rate constants k_s and k_r are treated as optimized parameter, but dissociation rate constant k_s is given special attention.

The program determines not only its best-fit value and the formal standard error, but also an approximate confidence interval.

Example: Confidence interval for rate constant k_r.

```
[rate constants]
   k  = 1, ks = 10 ?, kr = 1 ??
[concentrations]
   S = 2.34 ?
```

2.7 Multiple tasks

If a DynaFit script file contains multiple `[task]`s referring to the same reaction mechanism, to the same set of kinetic constants, or to the same set of experimental data, the corresponding sections can be omitted in subsequent portions of the script file. This feature is most useful in determining the reaction mechanism ("model discrimination analysis"), or in sequential processing of reaction progress curves followed by the analysis of initial reaction rates.

2.7.1 Model discrimination analysis

Let us assume that the same set of experimental data is to be analyzed by considering several alternate reaction mechanisms. In this case, the only sections of the script file that are different in the each regression analysis are the `[mechanism]` and possibly `[constants]`. It is then sufficient to give the location of the experimental data (section `[progress]` or `[velocity]`) only once, for the first mechanism.

Example : Determining the mechanism of "slow, tight" enzyme inhibition

In this extended example taken from the DynaFit distribution package, the experimental data represent the change in fluorescence upon the hydrolysis of a fluorogenic peptide by the HIV protease [2]. The experimental data are fitted consecutively to four alternate mechanisms of "slow, tight" binding inhibition. The program then decides on the best-fit model by applying certain statistical criteria. The important feature of this

script is that the sections `[responses]`, `[concentrations]` and `[progress]` are shown only for the first mechanisms, because they information is applicable to the remaining reaction mechanisms as well.

```
;-----------------------------------------------------
[task]
   data  = progress
   task  = fit
   model = one-step ?
[mechanism]
   E + S ---> ES     :     kon
   ES ---> E + P     :     kr
   E + I <==> EI     :     kai     kdi
[constants]
   kon = 1.21325, kr = 6.07453
   kai = 10 ?, kdi = 0.01 ?
[responses]
   P = 1.20
[concentrations]
   S = 5.17
[progress]
   directory  ./examples/hiv_protease/slow_tight/data
   extension  txt
   delay      5
   offset     auto ? local
   file i40a | conc. I = 0.04, E = 0.03
   file i40b | conc. I = 0.04, E = 0.03 ?
   file i60a | conc. I = 0.06, E = 0.03 ?
   file i60b | conc. I = 0.06, E = 0.03 ?
;-----------------------------------------------------
[task]
   data   = progress
   task   = fit
   model  = two-step ?
[mechanism]
   E + S ---> ES     :     kon
   ES ---> E + P     :     kr
   E + I <==> EI     :     kai     kdi
   EI <==> EJ        :     kij     kji
```

```
[constants]
   kon = 1.21325, kr  = 6.07453
   kai = 10 ?   , kdi = 0.1 ?
   kij = 0.1 ?  , kji = 0.01 ?
;-------------------------------------------------------
[task]
   data  = progress
   task  = fit
   model = iso-inhib ?
[mechanism]
   E + S <==> ES     :     kon     kds
   ES ---> E + P     :     kr
   I <==> J          :     kij     kji
   E + J <==> EJ     :     kai     kdi
[constants]
   kon = 78.0, kds =  381, kr = 6.04
   kij = 20 ?,  kji = 1000
   kai = 100 ?, kdi = 0.001 ?
;-------------------------------------------------------
[task]
   data  = progress
   task  = fit
   model = iso-enzym ?
[mechanism]
   E + S <==> ES     :     kon     kds
   ES ---> E + P     :     kr
   E <==> F          :     kef     kfe
   F + I <==> FI     :     kai     kdi
[constants]
   kon = 78.0, kds =  381, kr = 6.04
   kai = 100 ?, kdi = 0.002 ?
   kef = 25  ?, kfe = 1000
[end]
```

2.7.2 Varied data types

In some situations, the same reaction mechanism will be used to treat different kinds of experimental data (*e.g.*, reaction progress curves or initial velocities). Again, in this case it is possible to avoid repetition in the script

file of those sections that are shared by the multiple [task]s.

Example: Determination and subsequent fit of initial velocities

In this example taken from the DynaFit distribution package, the sections [responses] and [velocity] are listed only for the first task but not for the second.

```
;-----------------------------------
; Fit each progress curve separately
; to get initial velocities.
;-----------------------------------
[task]
    data  = progress
    task  = fit
    model = compet
[mechanism]
    E + S <===> ES    :  k    ks
    ES ---> E + P     :  kcat
    E + I <===> EI    :  k    ki
[constants]
    k = 100, ks = 4000 ?, kcat = 15 ?
    ki = 10 ?
[responses]
    P = -0.0015
[concentrations]
    E = 0.04
    S = 100 ?
[progress]
    local
    directory      ./examples/pepsin/data
    extension      txt
    offset         auto ?
    error          constant 0.00025 ; spectrometer noise
    delay          5
    files          1,2,3, 6,7,8, 9,10,11, 12,13
    vary conc. I = 0,0,0, 1,1,1, 2,2,2,   0.5,0.5
    files          14,15,16,   17,18,19,   20,21,22
    vary conc. I = 0.1,0.1,0.1,0.2,0.2,0.2,0.05,0.05,0.05
    files          23,24,25
```

```
    vary conc. I = 0.3,0.3,0.3
[velocity]
   variable    I
   file        ./examples/pepsin/data/veloc.txt
;----------------------------------------
; Fit initial velocities determined above.
; Skip [responses] and [velocity] section.
;----------------------------------------
[task]
    data  = velocities
    task  = fit
    model = compet
[mechanism]
    E + S <===> ES    :  Ks    dissoc.
    ES ---> E + P     :  kcat
    E + I <===> EI    :  Ki    dissoc.
[constants]
    Ks = 37.5, kcat = 15 ?
    Ki = 0.1 ?
[concentrations]
    E = 0.04
    S = 100
[progress]
    rapid equilibrium
[end]
```

2.8 Output files

By default, DynaFit generates all output in a directory ./OUTPUT where the dot "./" represents the directory in which the executable file is installed. If this directory does not exist, DynaFit will create it at run time.

Optionally, the program will generate all output files in the directory named in the `[output]` section of the script. The keyword `directory` is the only keyword that can be used; it is followed by the relative path name with or without trailing slash "/" (directory name separator).

Example 1

```
[progress]
```

```
  directory ./examples/cyclophilin/data
  files f1, f2, f3
  ...
[output]
  directory ./examples/cyclophilin/output
```

2.9 Initialization files

DynaFit begins each computational task by reading the list of *initialization parameters* listed in the initialization file. The default location for the initialization file is the directory ./system/dynafit/ and the default initialization file name is settings.ini. A complete description of the initialization file is presented in Chapter 11.

Occasionally the user might wish to *override* certain settings of the master initialization file. This can be accomplished in two different ways, as is shown in the examples below. The first method of overriding the default initialization parameters uses a small external text file containing a subset of initialization parameters (Example 1). The requisite file name is listed in the section `[settings]` of the DynaFit script file.

> *Example 1*
>
> In this example, DynaFit first reads and processes the default initialization file ./system/dynafit/settings.ini. Subsequently it reads the smaller initialization file ./examples/pepsin/fit.ini, which overrides several of the initialization parameters.

```
[settings]
  file ./examples/pepsin/fit.ini
[end]
```

A different method of overriding the DynaFit initialization parameters relies on directly listing selected initialization parameters, described in Chapter 11, in the `[settings]` section of the script file.

> *Example 2*
>
> In this example, DynaFit first reads and processes the default initialization file ./system/dynafit/settings.ini. Subsequently it overrides three of the initialization parameters (`Scale`, `Interrupt`,

and `DependentVar`) by processing the contents of the `[settings]` section of the script file.

```
[settings]
   <Filter>
      Scale = minutes
   <Marquardt>
      Interrupt = 100
   <Plot>
      DependentVar = conversion (%)
[end]
```

Chapter 3

Mechanism

The reaction mechanism for the given biochemical system is specified in the `[mechanism]` section of the script file. Some examples of valid reaction mechanisms translated into DynaFit notation follow.

Example 1a: Competitive inhibition of an enzyme

```
[mechanism]
   E + S <===> ES    :  k   ks
   ES ---> E + P     :  kr
   E + I <===> EI    :  k   ki
```

Example 1b: The same mechanisms under rapid-equilibrium approximation

```
[mechanism]
   E + S <===> ES    :  Ks  dissoc
   ES ---> E + P     :  kr
   E + I <===> EI    :  Ki  dissoc
```

Example 2: Two-site binding of a protein trimer to DNA

```
[mechanism]
   P + P + P <=> T          :   k  k1
   T + DNA <==> T.DNA       :   k  k2
   T + T + DNA <==> T2.DNA  :   k  k3
```

Example 3: An oscillatory metabolic cascade

```
[mechanism]
        S1 + E <===> S1.E        :    k      ks1
          S1.E  ---> E + S2      :    kr1
        S2 + E <===> S2.E        :    k      ks2
               ---> S1           :    v1
           S2 --->               :    v2
```

3.1 Molecularity and reaction order

Before we begin to translate our ideas about the reaction mechanism into a DynaFit script file, it is necessary to consider a number of theoretical factors. Especially in writing reaction mechanisms for biochemical systems at *equilibrium*, we must consider the fact that not all equilibrium constants might be independent of each other. This applies in particular to branched pathways, as is explained in section 9.2. In writing reaction mechanism for biochemical systems undergoing changes over time, no such restrictions apply.

3.2 Chemical notation

Writing reaction mechanisms in the script file closely follows the usual chemical notation. The only difference is that rate constants are not placed above and below the arrows, but instead are written on the same line as the reaction step to which they belong. For example, the Michaelis-Menten mechanism

$$\begin{array}{rcl} & \overset{k}{} & \\ \mathrm{E} + \mathrm{S} & \overset{}{\rightleftharpoons} & \mathrm{ES} \\ & \underset{k_s}{} & \\ & \overset{k_r}{} & \\ \mathrm{ES} & \rightarrow & \mathrm{E} + \mathrm{P} \end{array}$$

can be written with each mechanism step on a single line as

				$k_{\rightarrow}$	$k_{\leftarrow}$
$\mathrm{E} + \mathrm{S}$	$\rightleftharpoons$	ES		k	k_s
ES	$\rightarrow$	$\mathrm{E} + \mathrm{P}$		k_r	

which is represented in DynaFit by the following text:

```
[mechanism]
   E + S <==> ES      :   k    ks
      ES  --> E + P   :   kr
```

3.2.1 Notational flexibility

DynaFit allows a significant degree of notational flexibility. The Michaelis-Menten reaction mechanism can be written equivalently as

```
[mechanism]
   E + A ----> E.A     : k+1
   E.A   ----> E + A   : k-1
   E.A   ----> E + P   : k+2
```

or even in a condensed form as

```
[mechanism] | E + A -> EA : k1 | EA <=> E + P : k2 k3
```

where the vertical bar represents a line break.

3.2.2 Formal rules

The plus sign in writing reactions must be surrounded by one or more blank spaces (`E + S`, not `E+S`).

Each elementary step in the reaction mechanism must written on a separate line, unless a particular step denotes a reversible reaction (thus, in fact, it represents two different elementary reactions). In the reversible case, the forward and reverse steps can be written either on separate lines using two single-sided arrows, or on the same line using one double-sided arrow. Thus,

```
E + I <===> EI    :     k1    k2
```

is equivalent to

```
E + I ---> EI     :     k1
EI ---> E + I     :     k2
```

Single-sided arrows can point to either directions. Thus,

```
E + I ---> EI      :      k1
```

is equivalent to

```
EI <--- E + I      :      k1
```

Each elementary step is followed by a colon (`:`) followed by the name of one or two associated rate constants. An irreversible reaction step must be followed always by a *single* rate constant. If the step is reversible, the colon separator is followed by *two* rate constants. The first rate constant always refers to the left-to-right (forward) step, and the second rate constant refers to the right-to-left (reverse) step.

Oligomerization equilibria deserve a special mention here. In a DynaFit script file we are *not* allowed to use numerical stoichiometric coefficients, so that a dimerization equilibrium must be written as

```
A + A <===> A2     :     k1     k2
```

while the alternate notation using stoichiometric coefficients

```
2 A <===> A2       :     k1     k2 ; NOT ALLOWED
```

is not allowed.

3.2.3 Equilibrium constants

Equilibrium constants proper

In the analysis of equilibrium binding data we encounter a special case, where the double sided arrow is followed by a single equilibrium constant, followed by the keyword `equil`. For example, while in the above example `k1` was a label representing an association rate constant for the forward reaction step, here `Ka` is a name of the equilibrium constant for the reaction:

```
E + I <===> EI     :      Ka     equil
```

It is important to remember that the equilibrium constant always refers to the reaction proceeding from left to right. In other words, in the above example `Ka` is the *association* equilibrium constant, with the dimension M^{-1} (liter per mole). If we insisted that an equilibrium be defined as a dissociation constant, with the dimension M (moles per liter), then the reaction step above would have to be written as a dissociation (reading from left to right):

```
EI <===> E + I    :     Kd    equil
```

Dissociation constants

It is possible to override the left-to-right convention and designate certain equilibrium constants specifically as dissociation constants. In this case the name of the equilibrium constant is followed by the keyword `dissociation`, which can be abbreviated as `diss` or `dissoc`. In the following examples, both `Kii` and `Kis` are dissociation equilibrium constants although the left-to-right convention shows the reaction steps ass association equilibria.

```
E + I <===>  EI     :   Kis    dissoc
ES + I <===> ESI    :   Kii    dissoc
```

Association constants

In certain applications (e.g., analytical chemistry) it is common to describe chemical equilibria in terms of association constants, rather then dissociation constants. In this case we can override the conventional left-to-right notation by using the keyword `association`, which can be abbreviated as `assoc`. In the following examples, the dimension of the equilibrium constants `K1` and `K2` is M^{-1} and M^{-2} respectively, because they are treated as association constants.

```
ABC <==> AB + C   :   K1  assoc
AB <==> A + B     :   K2  assoc
```

Total association constant can also be specified in this manner. For example, the total association constant of the complex ABCD below is written as

```
ABCD <==> A + B + C + D  :   K(tot) assoc
```

which has precisely the same meaning as the text below:

```
A + B + C + D <==> ABCD  :   K(tot) assoc
```

3.3 Arrows

Any continuous sequence of characters beginning with the "<" character or ending with the ">" character, appearing in the `[mechanism]` section, is interpreted as an arrow. There are three types of arrows that can appear in the reaction mechanism.

- *Single-sided arrows* represent either an irreversible step or a part (either left-to-right or right-to-left reaction) in a reversible step.
- *Double-sided arrows* represent reversible steps which might participate in rapid equilibria, where rapid equilibrium computation is requested (see section ...).
- *Double-sided arrows with asterisk* represent reversible steps which do *not* participate in rapid equilibria, even if rapid equilibrium computation was requested.

Examples of *single-sided arrows*:

```
->   -->  ----->  ----------->
<-   <--  <----
=>   ==>  =====>  ===========>
<=   <==  <====
```

Examples of *double-sided arrows*:

```
<->   <-->  <----->  <----------->
<=>   <==>  <=====>  <===========>
```

Examples of double-sided arrows for reversible steps that do not participate in rapid equilibria:

```
<-*->  <--*-->  <-----*----->
<=*=>  <==*==>  <=====*=====>
```

Examples of *valid* but unusual and undesirable notation for arrows in biochemical mechanisms:

```
<<===>>    -.-.-.->    <::::>
<   ----   >
```

Examples of *invalid* notation for arrows:

```
- - >    - - - - - >    = = >    ----  >
```

3.4 Species names

Any continuous sequence of characters preceding the colon sign (`:`) on any line in the `[mechanism]` section is interpreted as a name of a reaction species. The names of reaction species must be at most 32 characters long, [1]and must *not* contain the following characters:

```
+   >   <   |   ;   :
```

Examples of recommended names for biochemical species

```
E     E.S     E_S     E-S         E*S
ESI   E.S.I   E*S*I
EAB   E.A.B   E*A*B   E.P.Q.R.I   EPQRI
NADP  Mg      Eu      Ca
```

Examples of valid names for biochemical species which are not recommended:

```
tryptase*inhibitor       ; quite long
x                        ; not expressive enough
```

Examples of invalid names:

[1]It is strongly recommended that names of reacting species be kept shorter than 8 characters.

```
enzyme*substrate*inhibitor*metal_ion   ;   too long
NADP+   Mg(2+)           ; contains '+'
E * S                    ; remove space
```

3.5 Rate and equilibrium constant names

Rate constants appear on each line in the mechanism after the colon sign (:). If a mechanism step is reversible, there must be two rate constants present. If a mechanism step is written with one-sided arrow, either because it is irreversible or because it each step is written individually, only one rate constant must be present.

Names of rate constants may consist of any continuous series of at most 32 characters, including the plus sign (+). A good practice is to keep the names of rate constants short and descriptive. For example, the rate constant for *s*ubstrate*a*ssociation might be called tt ksa, and the rate constant for *s*ubstrate *d*issociation might be named `ksd`. Enzymologists who prefer numerical naming schemes are free to name the rate constants accordingly.

Examples of valid rate constant names:

```
k1    k2    k3      K1   K2   K3
k_1   k_2   k_3     k+1  k-1
k     ki    ks
kas   kds   K(as)     K(ds)  kAS   kDS
kai   kdi   k-ai      k-di
kij   kji   k(i->j)   k(j->i)
```

3.6 Constant rates in open reaction systems

DynaFit can be used for simulation and fitting of biochemical reactions occuring in open systems, where certain species are being continuously supplied at a constant rate, for example via a metabolic pathway. The same or other species may be continously removed at a constant rate, for example due to a deactivation on the surface of the reaction vessel, or via a metabolic pathway.

Constant-rate steps are denoted in DynaFit by an arrow which does not have a species on either the left- or the right-hand side. For example if the

substrate of an enzyme reaction is supplied to the system at a constant rate, v_{in}, and if the product is continuously removed at a constant rate, v_{out}, we may write

```
[mechanism]
    ---> S    :    v(in)
  P ---->     :    v(out)
    etc.
```

Chapter 4

Kinetic Constants

The values of rate constants or equilibrium constants are specified in the `[constants]` section of the script file. While certain sections of the script file are optional, the `[constants]` section must be present always.

The `[constants]` section lists the values of rate and equilibrium constants, and (optionally) labels some or all of them as adjustable parameters. As was mentioned before in section 2.5, nominal values of rate constants depend both on the time scale and on the concentration scale of the experiment.

4.1 Formal rules

There are very few formal rules for writing down values of rate or equilibrium constants in the `[constants]` section of the script file. Any number of rate constants, separated by commas, can be listed on a single line, like this:

```
[constants]
   k1 = 0.1, k2 = 0.2 ?, k3 = 0.3 ??, k4 = 0.4
```

Alternatively the rate constants can be listed on separate lines with or without trailing commas, like this:

```
[constants]
   k1 = 0.1,
   k2 = 0.2 ?
```

```
k3 = 0.3 ??
k4 = 0.4
```

Some example of *incorrect* notation follow.

Incorrect: Missing commas

```
[constants]
   k1 = 0.1   k2 = 0.2 ?   k3 = 0.3 ??   k4 = 0.4
```

Incorrect: Can't assign multiple values

```
[constants]
   k1 = k2 = 0.2 ?
```

4.2 Dimension and unit of scale

Before deciding on the initial estimates for the rate or equilibrium constants, we must consider the dimensions and units. Let us consider in turn the dimension, the unit (scale), and the magnitude of rate constants and of equilibrium constants.

4.2.1 Rate constants

In general the dimension of rate constants strictly follows from the molecularity of the elementary reaction which they describe. Rate constants which describe monomolecular reactions have the dimension [1/time], rate constants which describe bimolecular reactions have the dimension [1/concentration × 1/time], and so on.

Thus in different kinds of rate constants there appear either one or two physical quantities (either time, or time and concentration) for which we must select an appropriate unit. The unit is determined by the experimental data we want to analyze.

The units of time and concentration used for the definition of rate constants must agree with the units of time and concentration used to describe the experimental data.

reaction type	order	molecularity	dimension of k
$A \xrightarrow{k}$	0	(constant influx)	concentration $\times$ time^{-1}
$A \xrightarrow{k} B + \cdots$	1	monomolecular	time^{-1}
$A + B \xrightarrow{k} C + \cdots$	2	bimolecular	concentration^{-1} $\times$ time^{-1}

Table 4.1: Dimension of rate constants.

Example:

An enzyme reaction was followed by monitoring absorbance changes over time. The experimental data are pairs of data values, representing absorbance (dimensionless) *vs.* time in *minutes*. Therefore, unless the time axis for the data is first converted to seconds, the unit of time must be min^{-1} for all first-order rate constants and concentration^{-1} $\times$ min^{-1} for all bimolecular rate constants.

The unit of time for rate constants is determined exclusively by the unit of time used in the experimental data. On the other hand, the concentration unit for rate constants is determined by two important factors, namely, the concentration unit for reactants and the molar instrumental responses.

The unit of concentration for all bimolecular rate constants must be the same as the unit in which concentrations or all reactants are also expressed. However, the molar concentrations of reactants (products, substrates, catalysts) are never measured directly. Instead, the measuring device usually provides values of physical quantities linearly related to concentrations, such as absorbance or optical rotation. The proportionality constant is called the *molar response coefficient*. Thus, the unit of concentration used for all bimolecular rate constants must correspond to the concentration unit obtained when the raw experimental data (in arbitrary instrumental units such as absorbance or fluorescence) are converted to concentrations by using the molar response coefficients.

Example:

An enzyme reaction was followed by monitoring absorbance changes over time. The experimental data are pairs of data values, representing absorbance *vs.* time in *minutes*. Assume that the concentrations throughout the script file are in the micromolar units (μM). Therefore, unless the time axis for the data is first converted to seconds, the unit must be min^{-1} for all first-order rate constants and $\mu\text{M}^{-1} \times \text{min}^{-1}$ for all bimolecular rate constants. One mole-per-liter of the reaction product would an increase of absorbance by 12340 absorbance units. Therefore, the molar response coefficient (see below) must be expressed in micromolar units also, $\epsilon = 0.01234$ (absorbance units per μM of product).

4.2.2 Equilibrium constants

Similar considerations about the *dimension* the *unit*, and the *magnitude* apply for equilibrium constants that appear in the DynaFit script files. The molecularity of forward and backward elementary reactions determine the dimension of each equilibrium constants. Some examples are given in table 4.2.

reaction type	dimension of K
$A \overset{K}{\rightleftharpoons} B$	(none)
$A + B \overset{K}{\rightleftharpoons} C$	$\text{concentration}^{-1}$
$C \overset{K}{\rightleftharpoons} A + B$	concentration
$A + A + A \overset{K}{\rightleftharpoons} A_3$	$\text{concentration}^{-2}$

Table 4.2: Dimension of equilibrium constants.

The scale of each equilibrium constant that appears in the mechanism is strictly dictated by the concentration scale of the experimental data (e.g., mM, μM, or nM). Thus, if the data are in the micromolar scale, all binary dissociation constants must have the same scale, while all binary association

constants have the scale μM^{-1}, a trimerization association constant would have the scale μM^{-2}, and so on.

4.3 Initial estimates

Nonlinear regression analysis requires an intelligent guess of initial estimates, thus data analysis should not (and cannot) be approached without prior knowledge. One must have at least some ideas about the possible values of rate and equilibrium constants that are relevant to the biochemical system at hand.

4.3.1 Association rate constants

In the case of bimolecular association rate constants, we must keep in mind that their values for the association of enzymes with small molecules (e.g., drugs) usually are between 10^5 $M^{-1}sec^{-1}$ and 10^9 $M^{-1}sec^{-1}$. The bimolecular association rate constants for protein-protein interactions are usually somewhat smaller. This background information is applied when we approach the point in writing down the script file below:

```
[mechanism]
   E + S <=> ES  :  k   ks
   ES -> E + P   :  kr
   E + I <=> EI  :  k   kis
  ES + I <=> EIS :  k   kii
[constants]
   k = ...
```

It is recommended to decide on the values for bimolecular rate constants first, keeping in mind that in many experimental situations their exact numerical values cannot be determined. Often one can use estimates for the bimolecular rate constants that are based on the theory of molecular diffusion. For many biochemical mechanisms we may start with the value 10^6 $M^{-1}sec^{-1}$ for all bimolecular rate constants. The fact that all three association rate constants in the above mechanism are supposed to have equal value is represented by the fact that all of them are assigned the same symbol.

Let us assume that in a set of experiments pertaining the mixed-type inhibition mechanism above, all concentrations are on the micromolar scale. In that case all bimolecular association rate constants have to have the

scale $\mu\mathrm{M}^{-1} \times \mathrm{time}^{-1}$. If the units of time used for the description of the experimental data are seconds, then the approximate nominal value of all bimolecular rate constants is

```
[constants]
   k = 1.0    ;  uM(-1)sec(-1)
```

because $k \approx 10^6$ M $^{-1}\mathrm{sec}^{-1} = 1.0$ μM $^{-1}\mathrm{sec}^{-1}$. If however the units of time used for the description of the experimental data were minutes, than the same value of the bimolecular rate constant would be expressed as

```
[constants]
   k = 60.0   ;  uM(-1)min(-1)
```

because $k \approx 10^6$ M $^{-1}\mathrm{sec}^{-1} = 60.0$ μM $^{-1}\mathrm{min}^{-1}$.

> For many biochemical mechanisms it is reasonable to set the initial estimate of all bimolecular association rate constants to 10^6 M $^{-1}\mathrm{sec}^{-1}$.

4.3.2 Dissociation rate constants

Initial values for dissociation rate constants are much more difficult to estimate. Usually we have some notion about the equilibrium constants, though, so from the equilibrium constants and from the association rate constants (set to their diffusion limit) we can deduce the initial estimate for the dissociation rate constant.

Example:

A substrate for an enzyme reaction following the simple Michaelis-Menten mechanism is expected to have the half-saturation point (Michaelis constant) in the millimolar range. The association rate constant is supposed to be diffusion limited (10^6 M $^{-1}\mathrm{sec}^{-1}$). From the reaction velocity observed at saturation, it seems that one mole of the enzyme-substrate complex would produce approximately 0.1 moles of the reaction product per second (turnover

number $k_{cat} \approx 0.1$ sec^{-1}). What is the order of magnitude for the dissociation rate constant? First we need to realize that for the Michaelis-Menten mechanism, $K_m = (k_s + k_r)/k$ and $k_{cat} = k_r$. From this we can estimate $k_s \approx K_m \times k - k_{cat} \approx 1 \times 10^{-3} \times 10^6 - 0.1 \approx 1000$ sec^{-1}.

Very often it is sufficient to come up with crude estimates of rate constants, within several orders of magnitude. Even without the arithmetic shown above we can estimate the dissociation rate constants after several trial simulations. The goal is to have the initial estimate of rate constants produce an qualitative agreement of the simulated data with the experimental data. An agreement at least as good as is shown in Figure 4.1 will probably be sufficient.

4.3.3 Equilibrium constants

Initial values for equilibrium binding constants are somewhat easier to obtain, in comparison with rate constants. In the equilibrium binding experiment we usually monitor a physical property such as fluorescence, or count of radioactivity per unit of time, in dependence on the total concentration of certain biochemical species.

Let us assume that within the range of concentrations that were chosen by the experimenter, the observed physical quantity (absorbance, radioactivity) has changed to a significant degree. Therefore, for the very initial estimate of simple dissociation equilibrium constants we may take the median value of the experimental concentrations.

Example

The equilibrium composition of six different biochemical mixtures containing 50 nM of DNA was measured at different amounts of protein P (c_{P} = 20, 40, 80, 160, 320, and 640 nM). The experimenter necessarily had to make a conscious choice of these concentrations, based on some previous knowledge, or simply by increasing the concentrations until a desired effect was in fact observed (e.g., partial or complete saturation). Assuming that the choice of concentrations was sensible, the dissociation constant(s) probably fall within the same range. Therefore we many first try $K_{\mathrm{D}} \approx 300$ nM, which approximately the median value of the experimental range.

For more complex binding mechanisms including several simultaneous equilibria we usually already have an idea whether or not these different equilibria are described by widely different equilibrium constants. It is however quite reasonable to start the analysis by setting all equilibrium constants to the same value, because DynaFit can often successfully optimize these values within three to six orders of magnitude.

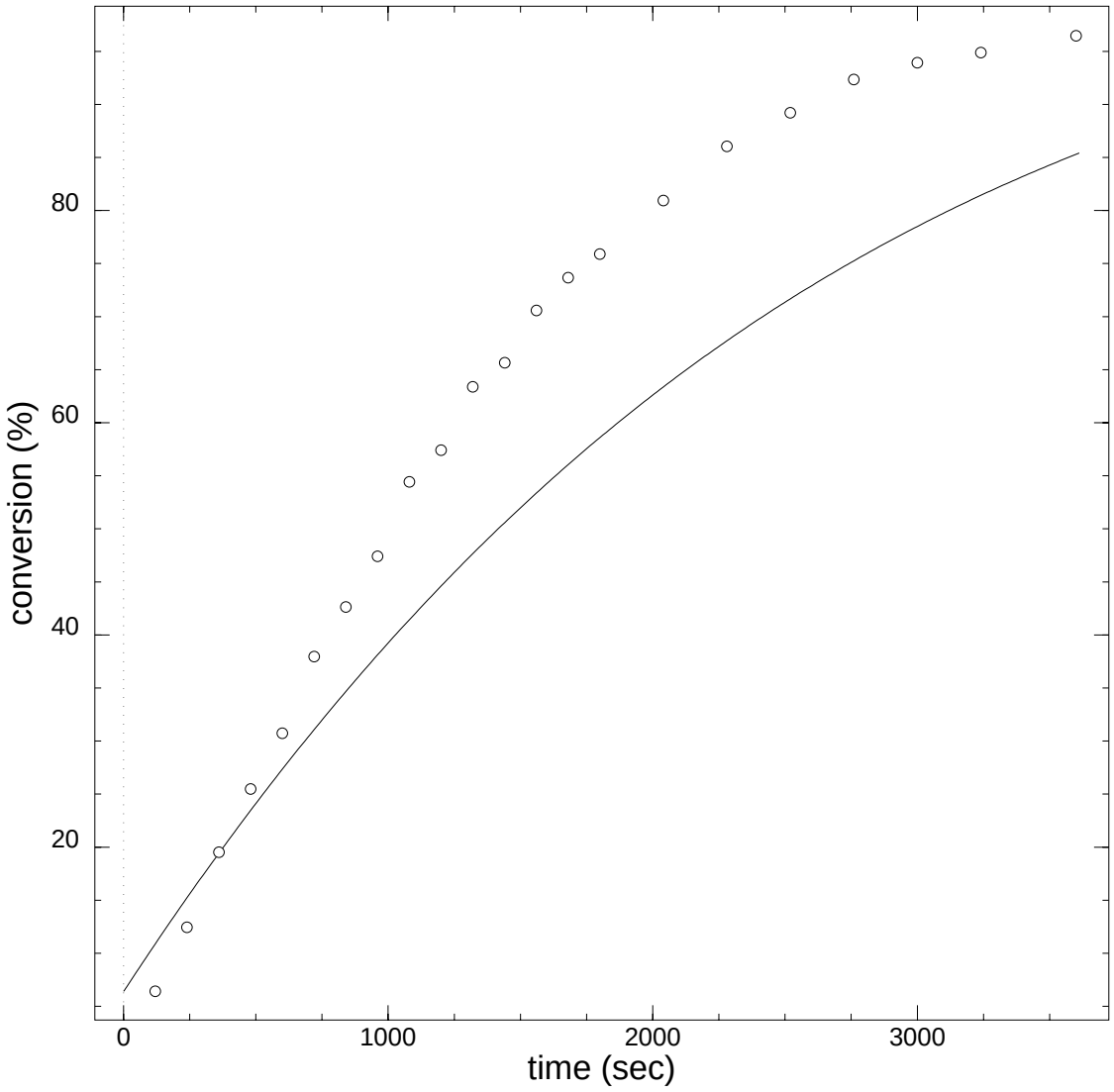

Figure 4.1: Example of an initial estimate suitable for starting the regression analysis.

Chapter 5

Concentrations

The script file section denoted by the keyword `[concentration]` is optional. However, it must be present file unless the `concentration` keyword is used in the `[progress]`, `[velocity]`, or `[equilibria]` section of the script file.

The `[concentration]` section lists the values of concentrations and (optionally) labels some or all of them as adjustable parameters. As was mentioned before in section 2.5, nominal values of concentrations depend on the concentration scale of the experiment.

5.1 Concentration scale

All concentrations mentioned anywhere in the script file must have the same concentrations scale (unit). It is optimal to choose a "natural" concentration scale for the analysis of each experiment, so that the nominal values are as close to unity as possible. This minimizes the truncation and round-off errors in numerical computations.

For example, if all concentrations are in the micromolar range, choose the micromolar unit throughout the script file. If some concentrations are very much different from other concentrations, choose a unit of concentration which is a compromise between the two values.

Example

The concentration of enzyme `E` was 10^{-9} M (kept constant), while the concentration of substrate `S` was varied between 0.5 and 8 $\times 10^{-3}$ M. The most natural unit of concentrations is therefore

μM, which also determines the unit for all bimolecular association rate constants and the molar response coefficient (`k = 1` means $k = 10^6$ M^{-1}sec^{-1}). Let us assume that the formation of one mole per liter of the reaction product `P` would cause an increase in the experimental signal (e.g., absorbance) by 1500 units, which means that one micromole per liter (our chosen unit) produces 0.0015 units of the experimental signal.

```
[mechanism]
   E + S <==> ES : k  ks  |   ES --> E + P :  kr
[constants]
   k = 1, ks = 20, kr = 5
[responses]
   P = 0.0015
[concentrations]
   E = 0.001
[progress]
   files          f1,  f2,   f3,   f4,   f5,   f6
   vary conc. S = 500, 1000, 2000, 4000, 8000, 16000
[end]
```

5.2 Global and local concentrations

Certain concentrations can be made global, that is, applicable to all datasets in the script files. The values of these global concentrations are listed in the `[concentration]` section discussed here. For example, all five progress curves mentioned in the script file listed above share were collected at the same enzyme concentration, $c_{\mathrm{E}} = 0.001$ μM.

```
[concentrations]
   E = 0.001
```

5.3 Local concentrations

The `concentration` keyword (abbreviated as `conc` in other parts of the script file can denote concentrations that pertain to the individual datasets. For example, let us assume that the six progress curves in the script file listed above were collected each at a different concentration of the enzyme. In this case the script file would contain the following text.

```
[mechanism]
   E + S <==> ES  :  k  ks
   ES --> E + P   :  kr
[constants]
   k = 1, ks = 20, kr = 5
[responses]
   P = 0.0015
[progress]
   file  f1 | conc S =   500, E = 0.001
   file  f2 | conc S =  1000, E = 0.002
   file  f3 | conc S =  2000, E = 0.003
   file  f4 | conc S =  4000, E = 0.004
   file  f5 | conc S =  8000, E = 0.005
   file  f6 | conc S = 16000, E = 0.006
[end]
```

If a concentration value is listed in the `[concentration]` section (global value) and simultaneously in the `[progress]`, `[velocity]` or `[equilibria]` section (local value), the local value takes precedence over the global value. The distinction between concentrations considered as global or local parameters becomes very important when concentrations are treated as adjustable parameters.

5.4 Concentrations as optimized parameters

Initial or total concentrations can be treated as adjustable parameters, subject to the optimization limits given by the parameter `ConcError` explained in Chapter 11. A given concentration can be optimized globally or locally. Global optimization means that the same best-fit value of an optimized concentration applies to all datasets analyzed together.

> *Example 1: Global optimization.* Four sets of initial velocity measurements (data file names `f1` through `f5`) were conducted by varying the concentration of a tight-binding inhibitor `I`, while keeping the concentration of the substrate constant in each set. The concentration of the enzyme was identical in all measurements, but it is not known exactly. In fact, one of the purposes of the experiment was to determine it (active site titration). Here

the best-fit concentration of the enzyme applies to the entire superset of experimental data.

```
[mechanism]
   E + S <==> ES : K(s) dissoc
   ES --> E + P  : k(cat)
   E + I <==> EI : K(i) dissoc
[constants]
   K(s) = 100, k(cat) = 10, K(i) = 0.01
[concentrations]
   E = 0.010 ?
[responses]
   P = 1
[velocities]
   variable I ; in the first column of each datafile
   file f1 | conc   S = 10
   file f2 | conc   S = 20
   file f3 | conc   S = 40
   file f4 | conc   S = 80
[end]
```

Example 2: Local optimization. Four sets of progress curve data (data file names `f1` through `f5`) were obtained by varying the concentration of a tight-binding inhibitor `I`, while keeping the concentration of the substrate and enzyme constant. The concentration of the enzyme is not known exactly, and is subjected to least squares fit. Here each of the best-fit concentrations of the enzyme applies to four of the five individual dataset (progress curve).

```
[mechanism]
   E + S <==> ES : k  ks
   ES --> E + P  : kr
   E + I <==> EI : k  ki
[constants]
   k = 100, ks = 10, ki = 0.01, kr = 20
[concentrations]
   S = 10
[responses]
   P = 1
```

```
[progress]
   file f1 | concentration I = 1, E = 0.01
   file f2 | concentration I = 2, E = 0.01 ?
   file f3 | concentration I = 4, E = 0.01 ?
   file f4 | concentration I = 8, E = 0.01 ?
[end]
```

5.5 Linked concentrations

Two or more concentrations can be linked together, meaning that their values are either identical or related through a constant factor.

```
[concentrations]
   A = 1.00 ?
   B = A    ; <=== linkage
```

For example, an enzyme inhibitor might be a 1:1 mixture of two enantiomers with S and R stereochemical configuration, respectively. Let us assume that the dose-response curve was measured by varying the concentration of the inhibitor between zero to 100 μM. Let us also assume that both enantiomers have nonzero inhibitory activity, measured by the inhibition constants $K_{i(S)}$ and $K_{i(R)}$, respectively. In this case the concentration of the S and the R enantiomers are varied simultaneously. This can be indicated in the script file by making the S enantiomer as the varied component, and then linking the concentration of the R enantiomer via the relationship $c_{\mathrm{S}} = c_{\mathrm{R}}$.

```
[mechanism]
   E + A <==> E.A       : Ka    dissoc
   E.A --> E + P        : kcat
   E + (S)I <==> E.(S)I : Ki(S) dissoc
   E + (R)I <==> E.(R)I : Ki(R) dissoc
[constants]
   Ka = 100, kcat = 10
   Ki(S) = 0.01
   Ki(R) = 0.1
[concentrations]
      E = 0.010 ?
   (R)I = (S)I   ;  <=== linkage
```

```
[responses]
   P = 1
[velocities]
   variable (S)I ;  <=== (R)I varied also
   file f1 | concentration S = 10
[end]
```

5.6 Arbitrary linking factor

It is possible to specify a more general relationship between two or more concentrations than simple identity. In particular, pairs of concentrations can be linked through an arbitrary linking factor by using the following notation:

```
[concentrations]
   A = 1.00 ?
   B = 0.5 * A    ; <- linkage
```

This notation would be used for an enantiomerically enriched mixture of two stereochemical isomers, acting as enzyme inhibitors.

The arbitrary linking factor can be considered as optimized parameter, which is indicated by appending a question mark to its numerical value:

```
[concentrations]
   A = 1.00
   B = 0.5 ? * A    ; <- f is optimized
```

In the above example, the concentration of species B is related to the concentration of species A through a constant factor, which is not known but instead determined from suitable experimental data. In this particular case the initial estimate of the best-fit value for the factor f in $c_{\mathrm{B}} = f \times c_{\mathrm{A}}$ is $f = 0.5$.

Chapter 6

Specific molar responses

The program's primary function is to fit experimental data obtained on a (bio)chemical system, either by following the reaction time-course, by measuring the initial reaction velocity, or by measuring the composition at equilibrium. In either case, it is important to realize that the chemical reacting system is always observed by using a certain physical instrument. For example, the chemical system might be observed by using

- fluorescence spectroscopy;
- UV/VIS absorption spectroscopy;
- IR spectroscopy;
- NMR spectroscopy;
- HPLC peak area integration;
- optical densitometry (gel shift assays);
- radiochemical methods;
- conductivity;
- polarimetry;
- mass spectrometry;
- other instrumental methods.

The main point is that concentrations are never measured directly. Instead, we always indirectly observe a certain physical signal (e.g., absorbance or peak area). Importantly, the program always assumes that the measured physical signal is related to the concentrations of reactants by a linear relationship.

> Specific molar responses are proportionality constants relating concentrations to the observed instrumental response.

Specific molar response coefficients can be assigned globally (pertaining to all datasets) or locally (pertaining to individual datasets). Global instrumental responses are gathered in the section of the script file denoted as `[responses]`. Local instrumental responses are listed after each file name either of the following sections: `[progress]` for the analysis of reaction progress curves, `[velocities]` for the analysis of initial reaction velocities, or `[equilibria]` for the analysis of equilibrium composition.

Any number of molar response coefficients can be considered as optimized (adjustable) parameters by appending a question mark after the numerical value. This applies both to the global response coefficients and to the local response coefficients.

Species the molar response coefficients of which are zero need not be listed in the script file. If a species is not mentioned in the `[response]` section (or after the `response` keyword for local response coefficients) it is assumed that its molar response coefficient is zero.

As was mentioned before in section 2.5, nominal values of molar responses depend on the concentration scale of the experiment. The same concentration unit (e.g., mM, μM, or nM) must be used for the following quantities:

- concentrations of reactants;
- specific molar responses;
- bimolecular association rate constants;
- equilibrium constants.

6.1 Global response coefficients

Global response coefficients, applicable to all datasets mentioned in the given script file, are listed in the `[response]` section. The formalism is

```
[response]
   A = 1.23, B = 3.45, C = 5.67
```

or

```
[response]
   A = 1.23
   B = 3.45
   C = 5.67
```

where A, B, and C are labels for chemical species appearing in the reaction mechanism.

Example 1: UV/VIS spectroscopy.

Substrate S is converted to the reaction product P by a catalytic action of an enzyme. The substrate has molar absorptivity 12,000 $\text{M}^{-1} \times \text{cm}^{-1}$, while the reaction product has practically zero absorption coefficient at the given wavelength. Let us assume that all concentrations in the given script file are expressed in micromolar units. Thus, 1 μM of the substrate corresponds to 0.012 absorbance units in 1 cm cell. In this case we will write

```
[mechanism]
   E + S <==> ES  :   k   ks
   ES ---> E + P  :   kr
[response]
    S = 0.012
   ES = 0.012
```

where it is assumed that the binding of the substrate to the enzyme does not change its molar response coefficient. If the concentration of the substrate is very much larger than the concentration of the enzyme catalyst, we can ignore the absorbance due to the Michaelis complex ES and write

```
[mechanism]
   E + S <==> ES  :   k   ks
   ES ---> E + P  :   kr
[response]
    S = 0.012
```

Example 2: Polarimetry.

Michaelis & Menten (1913) followed the changes in optical rotation caused by the hydrolytic action of invertase. In their instrumental setup, one mole per liter of saccharose would cause optical rotation of +42.5 degrees, while one mole per liter of the reaction product mixture would cause optical rotation of -13.3 degrees. Assuming that all concentrations throughout the script file are expressed in millimoles per liter, we will set up the script file (neglecting the optical rotation due to the Michaelis complex) as follows:

```
[mechanism]
   E + S <==> ES    : k   ks
   ES --> E + P     : kr
[responses]
   S = +0.0425
   P = -0.0133
[concentrations]
   ...
```

6.2 Local response coefficients

Often we can collect data files which pertain only to individual chemical species. The most simple case is when the chemical species are first separated by using a physico-chemical separation technique (chromatography, electrophoresis), and subsequently some instrumental signal is measured for each species separately.

Another possibility is to have available a spectroscopic technique which (without separation of chemical components) can provide individual signals for several species present in the reaction mixture (e.g., multi-wavelength UV/VIS spectroscopy).

In both cases we can use the keyword `response` listed after the name of the corresponding datafile to assign molar response coefficients.

Example: Gel shift assay.

A mixture of radioactive DNA, a DNA-binding protein, and two different types of protein-DNA complexes (PDNA and P_2DNA) is separated by electrophoresis. Radioactive areas of the gel plate, each corresponding to a different chemical species, are quantified by using a phosporimetric technique. Each datafile (`P-DNA.TXT` and `P2-DNA.TXT`) then contains pairs of data point, where the independent variable is the total concentration of the protein, and the dependent variable is the experimental signal from the phosphorimeter.

```
[mechanism]
   DNA + P <==> P.DNA     :   K1   dissoc
   P.DNA + P <==> P2.DNA  :   K2   dissoc
...
[equilibria]
   variable P
   file P-DNA.TXT  | response P.DNA  = 1234.00
   file P2-DNA.TXT | response P2.DNA = 5678.00
```

In the above example, it is important that the species for which response coefficients are not listed are assumed to be spectroscopically "invisible" in the given datafile (zero response coefficient).

6.3 Difference response coefficients

In many cases both the substrate and the product will have nonzero molar response coefficients in the given experiment. For example, in the enzymatic hydrolysis of *para*-nitrophenylalanine peptides, the absorbance upon cleavage next to *para*-nitrophenylalanine changes by about 10%. In such cases it is often useful to consider the differential molar response coefficient (i.e., the difference between the response coefficients of the reactants and products) as the only information needed to describe the kinetic assay, while the molar response coefficient of either the reactants or the products can be considered as zero.

Example: UV/VIS Spectrophotometry.

An enzyme reaction converts the substrate S (molar absorption coefficient $\epsilon_S = 1{,}300\ \text{M}^{-1} \times \text{cm}^{-1}$ at the given wavelength) to the products P ($\epsilon_P = 900\ \text{M}^{-1} \times \text{cm}^{-1}$) and Q ($\epsilon_P = 0$). Let us assume that all concentrations throughout the script file are in micromolar units. The conversion of one micromole per liter of the substrate will cause a decrease of absorbance by 0.0004 absorbance units.

```
[mechanism]
   E + S <==> ES      :  k   ks
   ES ---> E + P + Q  :  kr
[responses]
   P = -0.0004 ; response S = 0.0 assumed
[progress]
   offset = auto ?
```

In the example above, the keyword `auto` standing next the `offset` in the `[progress]` section orders the program to construct the simulated progress curve by assuming that it is offset on the signal axis. The magnitude of this offset is given by the first experimental data point. For further explanation of the `offset` keyword see section 7.6.

6.4 Analysis of reaction velocities

In the analysis of (initial) reaction velocities, there are several special considerations with regard to molar response coefficients. Occasionally the initial velocity data might be expressed in different time units (e.g., absorbance units per *minute*) then the rate constants are (reciprocal *seconds*). In such cases, the response coefficient must reflect the disparity in time units.

Example: UV/VIS Spectrophotometry - initial velocities.

As in the previous example, an enzyme reaction converts the substrate S (molar absorption coefficient $\epsilon_S = 1{,}000\ \text{M}^{-1} \times \text{cm}^{-1}$ at the given wavelength) to the products P ($\epsilon_P = 900\ \text{M}^{-1} \times \text{cm}^{-1}$) and Q ($\epsilon_P = 0$). Let us assume that all concentrations

throughout the script file are in micromolar units. The conversion of one micromole per liter of the substrate will cause a decrease of absorbance by 0.0004 absorbance units. However the reaction velocities, listed in the second column of the datafile, are in milli-OD per minute. Therefore, we must first multiply by 1000 and then divide by 60 to obtain the correct nominal value of $\Delta\epsilon$:

```
[mechanism]
   E + S <==> ES      :  k   ks
   ES ---> E + P + Q  :  kr
[responses]
   P = -0.00666 ; = -0.0004 / 60 * 1000
[velocity]
   file   VEL.TXT  ; second column milli-OD/min
```

Chapter 7

Progress Curves

Experimental data are identified in the script file in one of three sections:

- `[progress]` : time course of (bio)chemical reactions
- `[velocities]` : initial reaction velocities
- `[equilibria]` : equilibrium binding data.

In this chapter we discuss the design of the `[progress]` section of the script file. The `[progress]` section in each script file must contain the following keyword:

- `file` : identifies the datafile to be simulated or fitted
- `mesh` : must be present in all progress curve simulations (but not in data fitting); this keyword identifies the mesh of output points (independent variable) for the simulator.

Optionally, the `[progress]` section can contain the following keywords:

- `local` : requests local analysis of the experimental data (each progress curve is analyzed separately);
- `directory` : file directory in which data files are located;
- `extension` : datafile extension (must be the same for all datasets);

- `error` : if the measurement errors are known, the nature of the error function (relating standard error of each measurement to the magnitude of the measured quantity) the can be defined here;
- `offset` : a value to be subtracted from a progress curve on the signal axis (e.g., "baseline" absorbance);
- `delay` : a value to be added to a progress curve on the time axis (mixing delay time);
- `concentration` : concentrations of reactants specific to the given dataset;
- `responses` : molar response coefficients specific to the given dataset;
- `equilibrate / dilute` : for kinetic experiments that are preceded by the equilibration of reactants;

The primary function of the `[progress]` section is to give the location of the experimental or simulated progress curve files. Several preliminary examples are given below.

Example 1

```
[progress]
   offset = 0
   file ./examples/reductase/data/i0.txt
```

Example 2

```
[progress]
   directory    ./examples/conotoxin/data
   extension    txt
   files            058s1, 058s2, 024s,  105s, 115s, 186s
   vary conc.   L = 0.586, 0.586, 0.244, 1.05, 1.15, 1.86
```

Example 3

```
[progress]
    local
    directory      ./examples/pepsin/data
```

```
extension      txt
offset         auto ?
error          constant 0.00025 ; spectrometer noise
delay          5
files             1,  2,  3,  6,  7,  8,  9, 10, 11
vary conc. I =    0,  0,  0,  1,  1,  1,  2,  2,  2
files            14, 15, 16, 17, 18, 19, 23, 24, 25
vary conc. I =  0.1,0.1,0.1,0.2,0.2,0.2,0.3,0.3,0.3
```

7.1 Data files

The location of data files in the computer's file system is given by the keyword `file` followed by the file name.

Platform independent path names

Different computer systems use different ways to specify the location of files. For example, on the Microsoft DOS or Windows systems, the directory separator is the backslash ("\"). Under the Apple Macintosh operating system, the directory separator is the colon character (":"), while in the Unix file naming convention directories are separated by the forward slash ("/").

DynaFit recognizes all platform-dependent file naming conventions and converts them to the appropriate one, depending on the operating system where it is executing. However, it is recommended to use the Unix file naming convention, by follow the rules below.

1. The forward slash character ("/") is used to separate subdirectories.
2. The leading dot ("./") in directory name specifies the directory where the DynaFit executable file is located.
3. The leading double dot ("../") in directory names specifies a directory located hierarchically higher than the DynaFit executable file.
4. The leading double dot sequences can be chained ("../../../") to indicate progressively higher levels in directory hierarchy.
5. Absolute path names can *not* be used. For example, the notation `file C:\FILES\TEST.TXT` will produce an error.

7.1.1 Data directory

It is possible to specify a data directory by using the keyword `directory`. In this case the file names listed after the keyword `file` are assumed to exist in the specified directory.

Example

In this example we assume that the DynaFit executable file (e.g., DYNAFIT.EXE under the Microsoft Windows operating system) is located in the directory C:\PROGRAMS\DYNAFIT\, and that the data files named F1.TXT, F2.TXT, and F3.TXT are located in the directory C:\DATA\TEST\.

```
[progress]
   directory ../../data/test
   file f1.txt
   concentration I = 1.0
   file f2.txt
   concentration I = 2.0
   file f3.txt
   concentration I = 3.0
```

7.1.2 File extension

The keyword `extension` is used to specify the file name extension such as ".TXT" or ".DAT", assuming that all data files are named using it. In this case the file names proper are listed without the extension, which is automatically appended by the program.

Example

The above example could be equivalently written as

```
[progress]
   directory ../../data/test
   extension txt
   file f1
   concentration I = 1.0
   file f2
```

```
concentration I = 2.0
file f3
concentration I = 3.0
```

7.1.3 Multiple files

Even greater economy of space is accomplished by using the keyword `files` instead of `file`, followed by a comma-separated list of multiple file names (with or without extension such as ".TXT"). This option is applicable if and only if the data files in the list differ in the concentration of one reactant only. The associated concentrations must then be listed as a comma separated list of values, and the keyword `concentration` must be preceded by `vary`.

Example

The example above could be written exactly equivalently as

```
[progress]
   directory ../../data/test
   extension txt
   files f1, f2, f3
   vary concentration I = 1.0, 2.0, 3.0
```

or as

```
[progress]
   directory ../../data/test
   files f1.txt, f2.txt, f3.txt
   vary concentration I = 1.0, 2.0, 3.0
```

The number of files listed and the number of associated concentration values must be identical, otherwise the program will issue an error message.

7.2 Local vs. global analysis

DynaFit can analyze the time-course of chemical reactions using one of two statistical approaches:

1. *Global* analysis [3] involves fitting all progress curves listed in the script file simultaneously to the same set of rate and equilibrium constants.

2. *Local* analysis involves fitting the progress curves separately, one by one, to more-or-less different sets of rate and equilibrium constants.

Global analysis is the default. Local analysis is accomplished by including the keyword `local`, standing on a separate line, in the `[progress]` section of the script file.

Example

In this example, the data files F1, F2, and F3 are analyzed separately. The outcome are more-or-less different sets of rate constants k_1 and k_2, characteristic for each set of experimental data.

```
[mechanism]
   A + B ---> C  :  k1
   A + C ---> D  :  k2
[constants]
   k1 = 1.0 ?, k2 = 2.0 ?
[responses]
   D = 1000
[progress]
   local ; <-- fit progress curves separately
   directory ../../data/test
   extension txt
   files f1, f2, f3
   vary concentration I = 1.0, 2.0, 3.0
```

7.3 Simulation mesh

In the simulation of reaction progress curves, the keyword `mesh` is used to indicate the value of the independent variable (i.e., time) used to construct the simulated curve. Following the keyword `mesh` must stand specification of the desired range using the keywords `from ... to ... step`. The syntax that is used to specify range of values, spaced either logarithmically or linearly is given, in described in section 2.4.

Example 1: Equally spaced time intervals

In this example, the data files F1, F2, and F3 are simulated at the series of time values t = 0, 10, 20, ..., 1000.

```
[progress]
   mesh linear from 0 to 1000 step 10
   file f1.txt
   concentration I = 1.0
```

Example 2: Delay time

In this example, the data files F1, F2, and F3 are simulated at the series of *actual* time values t = 3, 13, 23, ..., 1003. However, the keyword `delay` (see section 7.5) is used to output the simulated data at *nominal* time values t = 0, 10, 20, ..., 1000.

```
[progress]
   mesh linear from 0 to 1000 step 10
   delay 3
   file f2.txt
   concentration I = 2.0
```

Example 3: Logarithmically spaced intervals

In this example, the data file F3 is simulated at the series of time values t = 0, 1, 2, 4, 8, 16, ..., 1024.

```
[progress]
   mesh logarithmic from 1 to 1024 step 2 add zero
   file f3.txt
   concentration I = 3.0
```

7.4 Experimental error

Any measurement is affected by a random experimental error, or instrumental noise. DynaFit assumes that the experimental noise is normally distributed (i.e., follows the Gaussian distribution). The keyword `error` is used to specify the standard deviation of this experimental noise, which could be either constant or dependent on the value of measured signal.

7.4.1 Constant error

If the machine noise is presumed independent on the experimental signal, the magnitude of the standard error is specified by the sequence `error constant` followed by a numerical value. The keyword `constant` may be omitted, as is indicated in the example below.

Example 1: Constants machine error

In this example the data in file FF.TXT are characterized by random noise with the standard deviation 0.0005 instrument units (e.g., absorbance units). The magnitude of the machine noise is presumed independent on the value of the measured signal.

```
[progress]
   mesh linear from 0 to 1000 step 10
   error 0.0005 ; machine noise (e.g., fluorescence)
   file ff.txt
   concentration I = 2.0
```

7.4.2 Constant percentage error

Very often the error distribution is not known with sufficient accuracy. In such cases, it is useful to assign (constant) experimental error to each data point based on the maximum value observed in the data. For example, we might assign to each point a constant error (standard deviation) that is equal to 2.5% of the maximum signal found in the analyzed data. This is accomplished by using the keyword `percent`.

Example 2: Constant percentage error

```
[progress]
   mesh linear from 0 to 1000 step 10
   error percent 2.5
   file ff.txt
   concentration I = 2.0
```

7.4.3 Nonconstant error

In many cases the magnitude of the machine noise depends on the value of the experimental signal actually measured. This is true especially for

absorption spectrophotometers.

Assume for example that a given UV-VIS absorption spectrophotometer has white noise $\sigma = 0.0005$ absorbance units at absorbance $A = 0$, and $\sigma = 0.005$ absorbance units at absorbance $A = 2.5$. If we assume for simplicity that the increase in machine noise is approximately linear, the relationship between the signal A and the noise σ can be described by the function $\sigma = 0.0005 + 0.002 \times A$. In other words, in this case the *error function* is linear with the constant coefficient 0.0005 and the linear coefficient 0.002.

Example 3: Non constant machine error

```
[progress]
   mesh linear from 0 to 1000 step 10
   error linear 0.0005, 0.002
   file ff.txt
   concentration I = 2.0
```

7.5 Mixing delay time

Rationale for a mixing-time constant

The numerical computation of reaction progress curves in DynaFit assumes that at time zero ($t = 0$) the reactants were just brought into contact, which means that molecular complexes (if any) have not yet been fully formed. For example, in an enzyme reaction following the Michaelis-Menten mechanism,

```
[mechanism]
   E + S <===> ES  :  k1  k-1
   ES ---> E + P   :  k2
[concentrations]
   E = 0.001
   S = 100.0
```

the concentrations of chemical species at time zero are exactly equal to the values listed in the `[concentration]` section of the script file. In particular, the concentration of the enzyme substrate complex ES is strictly zero, and the rate with which the product P is formed is also zero.

Many biochemical kinetic problems solved by DynaFit involve the computation not only of the reaction progress, but also the computation of *initial*

reaction velocities. This creates the necessity of introducing the mixing delay time into the computation. As was explained in the preceding paragraph, the reaction velocity at time zero (defined, in this case, as the rate with which the product P is formed) is by definition zero.

Proper unit of time

The mixing delay time is introduced to account for the fact that in real-world experiments, a certain amount of time (albeit a very short amount in specialized instruments) elapses after the reaction components are mixed and before the first experimental data points is recorded. The duration of this time delay is specified by the keyword `delay` followed by a numerical value.

Example

Simulate the reaction progress from time zero to time 300 seconds, with a three-second mixing delay time.

```
[progress]
   mesh from 0 to 300 step 3
   delay 3
   ...
```

> It is important that the unit of time expressing the mixing delay time identical to the unit of time used in the description of experimental data, and in the specification of all rate constants.

The mixing delay time must be the same for all progress curves in the given dataset. It cannot be treated as an optimized, adjustable parameter.

7.6 Offset

Under the term *offset* we mean a constant value of instrumental signal that is added to the simulated reaction progress curve. For example, a typical UV-VIS absorption spectrophotometer shows readings of absorbance that are

proportional to the concentration of certain reactants, *plus* a "baseline" value that may have been introduced by the solvent or by the spectrophotometric cell in which the reaction is taking place. Using the mathematical formalism,

$$A = a_0 + \epsilon \times c$$

where A is the experimental signal (e.g., absorbance), ϵ is the molar response coefficient, c is the concentration of a spectroscopically visible reactant, and a_0 is the baseline offset.

Baseline offset in DynaFit is given by the keyword `offset` followed by a numerical value in proper instrumental units (e.g., absorbance units or relative fluorescence units).

Example

In this example each simulated progress curve has the value 0.01 (e.g., absorbance units) added to each data point.

```
[progress]
   delay  3
   offset 0.01
   file   ff.txt
```

Adjustable offset

In the statistical analysis of experimental data, it is often advantageous to treat the instrument offset as an adjustable parameter. In this case the numerical value of the baseline offset is followed by the question mark.

Example : Adjustable offset

In this example the initial value of the baseline offset (0.01) is optimized in the least-squares regression.

```
[progress]
   delay  3
   offset 0.01 ?
   file   ff.txt
```

Automatic offset

DynaFit can treat the first data point on each experimental reaction progress curve as the baseline offset. The value of the independent variable (e.g., absorbance or fluorescence) is automatically set to zero for the first data point, and the remaining data points are adjusted accordingly. This kind of automatic baseline adjustment is indicated by the keyword `auto`.

Example : Automatic baseline offsets

The experimental data in file FF.TXT are adjusted by setting the first data point to zero on the vertical axis, and adjusting accordingly the remaining data points in the set.

```
[progress]
   delay  3
   offset auto
   file   ff.txt
```

Automatic adjustable offset

It is possible to treat the automatic baseline offset as an adjustable parameter. In this case the keyword `auto` is followed by the question mark.

Example : Automatic baseline offsets

The experimental data in file FF.TXT are adjusted by setting the first data point initially to zero, and subsequently optimizing the baseline adjustment in the least-squares regression.

```
[progress]
   delay  3
   offset auto ?
   file   ff.txt
```

Locally adjustable automatic offset

If multiple files are being analyzed simultaneously ("global analysis"), it is possible to treat the automatic baseline offset as a parameter adjustable separately for each data set. In this case the keyword `auto` is followed by the question mark and the keyword `local`.

Example : Automatic baseline offsets adjusted separately for each dataset

The experimental data in file F1.TXT, F2.TXT, and F3.TXT are adjusted by setting the first data point initially to zero, and subsequently optimizing the baseline adjustment in the least-squares regression. A separate best-fit value of the baseline offset is obtained for each dataset.

```
[progress]
   delay  3
   offset auto ? local
   file   f1.txt
   file   f2.txt
   file   f3.txt
```

Example : Globally adjustable offset

The experimental data in file F1.TXT, F2.TXT, and F3.TXT are adjusted by setting the first data point initially to zero, and subsequently optimizing the displacement of progress curves from the origin. The same best-fit value of the baseline offset is obtained for all dataset.

```
[progress]
   delay  3
   offset auto ?
   file   f1.txt
   file   f2.txt
   file   f3.txt
```

7.7 Local concentrations

Initial concentrations of reactants that are applicable to all progress curves listed in the given script are given in the `[concentration]` section of the script file (Chapter 5). In addition, the `[progress]` curve section can contain concentrations that are specifically linked to particular data. The concentration values related to a particular progress curve file are indicated by the keyword `concentration`, which can be abbreviated as `conc`.

Example 1

The concentration of enzyme E is 0.001 μM in datasets F1, F2, and F3. The concentration of substrate S is 10, 20, and 30 μM, respectively.

```
[concentrations]
   E = 0.001
[progress]
   file f1
   concentration S = 10
   file f2
   concentration S = 20
   file f3
   concentration S = 30
```

The concentration of a certain species can appear both in the `[concentration]` section (globally) and in the `[progress]` section (locally). In this case the local concentration overrides the global concentration.

Example 2

The concentration of enzyme E is 0.001 μM in datasets F1, F2, and F3. The concentration of substrate S is 10 μM in data sets F1 and F3 but 30 μM in data set F3.

```
[concentrations]
   E = 0.001, S = 10
[progress]
   file f1
   file f2
   file f3
   concentration S = 30
```

Locally optimized concentrations

It is often advantageous to treat certain concentrations as locally optimized parameters, to account for the fact that in delivering reaction volumes the experimenters necessarily make random errors (pipetting errors). Some or

all local concentrations can be made adjustable by appending the question mark after their numerical values.

Example

The concentration of enzyme E is held constant at 0.001 μM in datasets F1, F2, and F3. The concentration of substrate S is held constant at 10 μM in data set F1, while the concentrations 20 and 30 μM, respectively, in data sets F2 and F3 are treated as locally optimized parameters.

```
[concentrations]
   E = 0.001
[progress]
   file f1
   concentration S = 10
   file f2
   concentration S = 20 ?
   file f3
   concentration S = 30 ?
```

Multiple local concentrations

The keyword `concentration` or `conc` can be followed by any number of species names and associated concentrations, separated by commas. Some or all concentrations listed in this manner can be treated as locally adjustable parameters.

Example

The concentration of enzyme E is held constant at 0.001 μM in the dataset F1. The same value is treated as a locally adjustable parameter for data sets F2 and F3. The concentration of substrate S is held constant at 10 μM in data set F1, while the concentrations 20 and 30 μM, respectively, in data sets F2 and F3 are treated as locally optimized parameters. Here we used the abbreviated form of the keyword `concentration` and the shorthand "`|`" for line break.

```
[concentrations]
```

```
   ; empty
[progress]
   file f1 | conc E = 0.001  , S = 10
   file f2 | conc E = 0.001 ?, S = 20 ?
   file f3 | conc E = 0.001 ?, S = 30 ?
```

7.8 Local response coefficients

Practically everything that was said about the local concentrations in section 7.7 applies to the locally defined molar response coefficients.

Molar response coefficients of reactants that are applicable to all progress curves listed in the given script are given in the `[response]` section of the script file (Chapter 6). In addition, the `[progress]` curve section can contain response coefficients that are specifically linked to particular data. The response factors related to a particular progress curve file are indicated by the keyword `response`, which can be abbreviated as `resp`.

Example 1

The concentrations of products P and Q has been monitored by a diode-array spectrophotometer, at three different wavelengths corresponding to data files F1, F2, and F3. The specific molar response coefficients for P and Q are different from each other and also different at each selected wavelength. The numerical values of the response coefficients can be listed individually for each data file as follows.

```
[progress]
   file f1
   response P = 100, Q = 1000
   file f2
   response P = 500, Q = 400
   file f3
   response P = 900, Q = 20
```

The response coefficient of a certain species can appear both in the `[response` section (globally) and in the `[progress]` section (locally). In this case the local value overrides the global value.

Example 2

The response coefficient (e.g., molar absorbance at the given wavelength) of product P is 100 in data files F1 and F2 but 900 in data file F3. The specific molar response or product Q varies as in the preceding example.

```
[responses]
   P = 100
[progress]
   file f1
   response Q = 1000
   file f2
   response Q = 400
   file f3
   response P = 900, Q = 20
```

Locally optimized responses

It is sometimes advantageous to treat certain response coefficients as locally optimized parameters. Some or all local response factors can be made adjustable by appending the question mark after their numerical values.

Example 3

This example is identical to Example 1 above, except for the fact that molar response coefficients of the product P are treated as optimized parameters in files F2 and F3.

```
[progress]
   file f1
   response P = 100  , Q = 1000
   file f2
   response P = 500 ?, Q = 400
   file f3
   response P = 900 ?, Q = 20
```

7.9 Concentration jump experiments

In many useful experiments, a certain number of (bio)chemical reactants are first equilibrated to form various molecular complexes. Subsequently the equilibrated mixture is diluted by the addition of the last reactant.

A good example are studies of "slow, tight" binding enzyme inhibitors. An enzyme E is first incubated with a reversible, "slow, tight" binding inhibitor I, until the mixture containing E, I, and the complex EI is at equilibrium. Finally the substrate S is added to initiate the catalytic reaction $S \rightarrow P$, while at the same the enzyme – inhibitor mixture is diluted. From this kind of a study, the rate constants governing the $E \rightleftharpoons I$ binding and dissociation can be determined.

To set up the analysis of such biphasic experiments (an equilibration phase followed by a dynamic phase), the DynaFit script file must contain in the `[progress]` section the keywords `equilibrate` and `dilute`.

Example 1

Enzyme E and inhibitor I were incubated at 100 nM each until equilibrium was reached. The equilibrated mixture was diluted 1:50 by the addition of substrate S. The final concentration was $[S] = 10.0\ \mu$M.

```
[mechanism]
   E + S
   E + I <==> EI   :  Ki   dissoc
[progress]
   file ff
   equilibrate E = 0.1, I = 0.1, dilute 1:50
   concentration S = 10.0

   ; final concentrations:
   ;    [E] = [I] = 0.002, [S] = 10.0
```

A more complex example is taken from the published biochemical literature [4, 5].

Example 2: Thrombin - Dehydrothrombin - Hirudin

Thrombin (178 nM) and hirudin (208 nM) were incubated until equilibrium was achieved. The total volume was 90 μl. The equilibrated mixture was then diluted with 10 μl dehydrothrombin so that the total concentration of the mutated enzyme was 165 nM. In a complementary experiment, dehydrothrombin (183 nM) and hirudin (208 nM) were incubated in 90 μl of buffer, and 10 μl thrombin was added so that the final concentration was 160 nM. Thus the final concentrations of all components were identical in both experiments, only the order of addition was different. The concentration of free thrombin was followed over time by using a kinetic assay. [4]

```
[task]

   data  = progress
   task  = fit
   model = compet

[mechanism]

   E + L <===> EL  :  k1   k2
   F + L <===> FL  :  k3   k4

[constants]

   k1 = 0.001 ?? , k2 = 0.00005 ??
   k3 = 0.001 ?? , k4 = 0.00005 ??

[concentrations]

   ; All concentrations specified
   ; in [progress] section.

[response]

   E = 1

[progress]

   directory ./examples/thrombin/data
```

```
   extension txt
   delay     30

   file d1e
      equilibrate   E=178  , L=208, dilute 90:100
      concentration F=165 ?

   file d2e
      equilibrate   E=178  , L=208, dilute 90:100
      concentration F=165 ?

   file d3e
      equilibrate   E=178  , L=208, dilute 90:100
      concentration F=165 ?

   file d4e
      equilibrate   F=183  , L=208, dilute 90:100
      concentration E=160 ?

   file d5e
      equilibrate   F=183  , L=208, dilute 90:100
      concentration E=160 ?

   file d6e
      equilibrate   F=183  , L=208, dilute 90:100
      concentration E=160 ?

[end]
```

7.10 Keyword ordering

The `[progress]` section of DynaFit script files is one of few places where the ordering of keywords is important. Here are few rules to follow.

- The keyword `file` must precede the associated `concentrations` and `responses`.
- The keyword `file` must also precede the associated pair `equilibrate .. dilute`.

- The keyword `file`, followed by a file, name must stand on a separate line.
- The keywords `directory` and `extension` must precede any `file` names.
- The keywords `mesh`, `error`, and `offset` must precede any `file` names and stand on separate lines.
- The order in which `mesh`, `error`, and `offset` are entered is irrelevant.

Chapter 8

Velocities

The measurement and statistical analysis of initial reaction velocities, in dependence on the initial concentration of reactants, is a useful method to investigate the mechanisms of enzyme reactions [6]. The computation of initial velocities is initiated by the sequence `data = velocities` in the `[task]` section of the script file (see Section 1.1).

The information required by DynaFit to compute initial reaction rates is summarized in the `[velocity]` section of the script file.

The keywords that are found in the `[velocity]` section are listed below in alphabetical order.

```
concentration
directory
dixon
error (constant, linear, quadratic,
       cubic, percent, data)
extension
file
from .. to .. step
graph
linear
lineweaver-burk
logarithmic
mesh
plot
variable
```

Also important are the keywords `rapid equilibrium`, which must be listed in the section `[progress]` for some types of velocity calculations. Also of importance is the keyword `delay` which is located in the same section.

8.1 Molar response coefficients

DynaFit *internally* defines (bio)chemical reaction rates as the change in molar concentrations per unit of time (*e.g.*, seconds, minutes, hours) in which the experimental data are expressed. However, the *observed* reaction rates are always expressed in instrument units (*e.g.*, absorbance, fluorescence, or radioactivity) per unit of time.

For this reason, the computation of observed reaction velocities requires that at least one molecular species (reactant, product, or catalyst) has nonzero molar response coefficient. In other words the section `[response]` must always contain at least one chemical species that is "visible" by the given instrumental technique. DynaFit then computes the observed reaction rate as

$$\text{observed rate} = \text{molar response} \times \frac{\text{concentration change}}{\text{unit of time}} \quad .$$

Example 1

Compute the reaction rates in mOD-per-minute[1] for the following reaction mechanism. The only spectroscopically visible species is the product P, with $\epsilon_P = 5{,}670$ absorbance units / mol / centimeter at the given wavelength. The rate constants are expressed in reciprocal seconds, and substrate concentrations in the data file MILLIOD-VS-[S].TXT are micromolar.

```
[mechanism]
   E + S <===> ES  :  k1  k-1
   ES ---> E + P   :  k3
[responses]
   P = 0.0945 ; = 1,000 x 5,670 / 1,000,000 / 60
[velocity]
   data MILLIOD-VS-[S].TXT
```

[1] The "milli O.D." unit (from "optical density") is frequently used in optical spectroscopy. It means 0.001 × the absorbance unit.

8.2 Rapid-equilibrium vs. dynamic methods

The initial reaction rates (typically, of enzyme reactions) can be computed in DynaFit with or without the rapid-equilibrium approximation ([6], p. 18; [7], p. 18). The steady-state approximation, also used frequently in the analysis of initial reaction rates, is not available in DynaFit.

8.2.1 Rapid-equilibrium approximation

Under the rapid equilibrium approximation, it is assumed that the non-covalent binding of substrates to the enzyme and the dissociation of newly formed products from the intermediate complexes is infinitely faster than any chemical steps. If this assumption can be used profitably, the DynaFit script file must include the sequence

```
[progress]
   rapid equilibrium
[velocity]
   ...
   equilibrate <list of reactants>
   ...
```

or the sequence

```
[progress]
   rapid equilibrium
[velocity]
   ...
   equilibrate all
   ...
```

Slow steps in the reaction mechanism

Using the rapid equilibrium approximation has implications for the `[mechanism]` section of the script file. If the keyword `rapid equilibrium` is present in the `[progress]` section, *all* reversible steps will be considered effectively infinitely rapid unless the special notation `<==*==>` is used, instead of the usual double sided arrow (`<====>`).

Example 1

In the mechanism below, the reaction steps characterized by rate constants k_1, k_{-1}, k_3, and k_{-3} will be considered as infinitely rapid compared with k_2 and k_{-2}. Transparently to the user, DynaFit will internally represent the first and third reversible steps by using the equilibrium constants $K_1 = k_1/k_{-1}$ and $K_2 = k_3/k_{-3}$, respectively.

```
[mechanism]
   E + S <=====> ES  :  k1  k-1
   ES <==*==> EP     :  k2  k-2
   EP <=====> E + P  :  k3  k-3
[progress]
   rapid equilibrium
[response]
   P = 1.0
```

Kinetic and equilibrium steps

When DynaFit computes initial reaction rates under the rapid-equilibrium approximation, the mechanism must contain at least one "slow" step, that is, a step which does not participate in rapid equilibria. In the preceding example there are two such steps, characterized by rate constants k_2 and k_{-2}.

Example 2

Alternate and equivalent representation of the reaction mechanism in *Example 1* above.

```
[mechanism]
   E + S <=====> ES  :  K(s)   dissoc
   ES ---> EP        :  k2
   EP ---> ES        :  k-2
   EP <=====> E + P  :  K(p)   dissoc
[progress]
   rapid equilibrium
[response]
   P = 1.0
```

There can be as many kinetic ("slow") steps in any given mechanism as is necessary. However, it is important that at least one is present. For

example, an omission of the asterisk in "<==*==>" in *Example 1* would imply that all reversible steps are at equilibrium:

> *Example 3*
>
> This mechanism is notated incorrectly for the computation of initial reaction velocities, because it does not contain any kinetic steps.

```
[mechanism]
   E + S <===> ES  :  K(s)   dissoc
   ES <===> EP     :  Keq    equil   ; ERROR !!
   EP <===> E + P  :  K(p)   dissoc
[progress]
   rapid equilibrium
[response]
   P = 1.0
```

Molar response coefficients

The present version of DynaFit places an important limitation on the nature of (bio)chemical species that appear in "rapid-equilibrium" reaction mechanisms. In particular, a non-zero molar response coefficient can be assigned only to those species that are directly formed in a step *not* participating in rapid equilibria.

> *Example 4*
>
> The initial reaction velocity is computed as the rate of formation for the species P, $v = \epsilon_{\mathrm{P}} \times \mathrm{d}[P]/dt$, where $\mathrm{d}[P]/\mathrm{d}t = k_{\mathrm{cat}} \times [ES]_{\mathrm{eq}}$.

```
[task]
   data = velocity
   task = simulate
[mechanism]
   E + S <===> ES  :  Ks  dissoc
   ES --> E + P    :  kcat
[constants]
   Ks = 1
   kcat = 1
[concentrations]
```

```
   E = 0.001
[responses]
   P = 1
[progress]
   rapid equilibrium
[velocity]
   mesh from 0 to 10 step 1
   variable = S
   file ./output/rapeq_1.txt
[end]
```

If a species with non-zero molar response coefficient is not formed directly in a "slow" step (not a "rapid equilibrium" step), it is still possible to compute the initial reaction velocity using the rapid equilibrium approximation. However, the present version of DynaFit still requires that the molar response coefficient is attached to a species that does *not* participate in rapid equilibria. Therefore, one should choose a chemical species from which the truly observable species is formed in a 1:1 stoichiometric ratio, as is explained on the example below.

Example 5

```
[task]
   data = velocity
   task = simulate
[mechanism]
   E + S <===> ES  :  Ks  dissoc
   ES <=*=> EP     :  kf  kr
   EP <===> E + P  :  Kp  dissoc
[constants]
   Ks = 1, Kp = 1
   kf = 1, kr = 1
[concentrations]
   E = 0.001
[responses]
   EP = 1  ; standing in for 'P' !
[progress]
   rapid equilibrium
[velocity]
   mesh from 0 to 10 step 1
```

```
   variable = S
   file rapeq_2.txt
[end]
```

8.2.2 The dynamic method

An alternate method for computing initial reaction velocities in DynaFit is based on the simulation of pre-steady state dynamics of the (bio)chemical system. In this method, the reaction components are presumed to be mixed at time zero. The chemical composition then changes very rapidly as intermediate molecular complexes are formed by (reversible) molecular association. At a suitable point in time (e.g., one second or ten seconds, depending on the experimental setup) the reaction velocity is computed from a system of differential equations. In this method of computing initial reaction velocities, it is then very important to specify a *nonzero mixing delay time*, by using the keyword `delay` in the `[progress]` section of the script file.

> *Example 6* The initial reaction velocity is computed at time $t =$ 5 seconds as $v = \epsilon_P \times \mathrm{d}[P]/\mathrm{d}t$ (see also Example 4). In this case the elementary rate $\mathrm{d}[P]/dt$ is computed by solving numerically the system of differential equations

$$\dot{c}_E = -k_1\, c_S + k_{-1}\, c_{ES} + k_2\, c_{ES} \tag{8.1}$$

$$\dot{c}_S = -k_1\, c_E\, c_S + k_{-1}\, c_{ES} \tag{8.2}$$

$$\dot{c}_{ES} = +k_1\, c_E\, c_S - k_{-1}\, c_{ES} - k_2\, c_{ES} \tag{8.3}$$

$$\dot{c}_P = +k_2\, c_{ES} \tag{8.4}$$

```
[task]
   data = velocity
   task = simulate
[mechanism]
   E + S <===> ES  :  k1  k-1
   ES --> E + P    :  k2
[constants]
   k1 = 1, k-1 = 1
   kcat = 1
[concentrations]
```

```
   E = 0.001
[responses]
   P = 1
[progress]
   delay = 5
[velocity]
   mesh from 0 to 10 step 1
   variable = S
   file ./output/dynamic.txt
[end]
```

8.3 Location of data files

Initial velocity data files contain, in the first column, the concentration of a certain reactant and, in the second column, the associated initial reaction rate. The location of data files is indicated in the `[velocity]` section of the script file. The formal rules for using the keywords `directory`, `file` or `files`, and `extension`, are the same as was described in section 7.1.

One significant difference between progress curve data files and initial velocity data files is the definition of the independent variable. In the case of progress curves, the independent variable is always time and therefore it does not have to be specified. In the case of initial velocity data, the independent variable is the concentration of a certain reactant. The identity of this variable chemical species must be given by the keyword `variable` before the first `file` is given.

> *Example 1* The directory ./TEST/DATA contains four files named F1.TXT, F2.TXT, F3.TXT and F4.TXT, each of which contains in the first column the concentration of inhibitor I and in the second column the initial reaction rate.

```
[task]
    data  = velocities
    task  = fit
[mechanism]
    E + S <===> ES    :  Ks    dissoc.
    ES ---> E + P     :  kcat
    E + I <===> EI    :  Ki    dissoc.
[constants]
```

```
    Ks = 37.5, kcat = 15 ?
    Ki = 0.1 ?
[responses]
    P = -0.0015
[concentrations]
    E = 0.04
[progress]
    rapid equilibrium
[velocity]
   directory ./test/data
   extension txt
   variable I
   file f1 | concentration S = 20
   file f2 | concentration S = 40
   file f3 | concentration S = 80
   file f4 | concentration S = 160
[end]
```

8.4 Experimental error

Initial reaction velocities are always affected by finite experimental error. The magnitude and type of experimental uncertainty associated with the measurements of initial velocities is expressed in chapter 7.4 discussing reaction progress curves.

For example, the following script assigns a constant error to all initial velocity data points. The standard deviation is equal to 1.5% of the *maximum* data value found across all four data sets (files, f1 through f4).

```
[velocity]
   error percent 1.5
   directory ./test/data
   extension txt
   variable I
   file f1 | concentration S = 20
   file f2 | concentration S = 40
   file f3 | concentration S = 80
   file f4 | concentration S = 160
[end]
```

It is also possible to specify a nonconstant error variance, as is described in section 7.4.3, or a constant error specified in its absolute value (section 7.4.1).

8.4.1 Standard errors of measurements

In addition to defining the standard error of measurements by using a linear, quadratic, or cubic "error function", described in the preceding paragraph, we can also define the standard errors of measurements directly in the data file. This is accomplished by using the keyword combination `error data`, shown in the example below:

```
[velocity]
   error data
   directory ./test/data
   extension txt
   variable I

   ; Text files 'f1.txt' through 'f4.txt'
   ; must contain THREE columns:
   ;
   ;    column 1: concentration [I]
   ;    column 2: initial velocity 'v'
   ;    column 3: standard error of 'v'

   file f1 | concentration S = 20
   file f2 | concentration S = 40
   file f3 | concentration S = 80
   file f4 | concentration S = 160
[end]
```

In this case the program will assume that all data files contain *three* columns of ASCII text. The third column must contain the standard deviation of initial velocities. One example of such data file is shown below:

```
; data file "020710-2SD-8nM"
; active site titration
;
[I] v         error
--- --------  --------
```

```
0   0.7047    0.009512
1   0.4729    0.009351
2   0.3226    0.007252
4   0.1426    0.00873
6   0.06624   0.008714
8   0.0377    0.00869
```

If and when the data files contain only *two columns*, the program will ignore the `error data` instruction and assign equal weights to all data points.

8.5 Diagnostic plots

DynaFit has the ability to produce two classic diagnostic plots frequently used in traditional enzyme kinetic research, namely, the Lineweaver-Burk plot and the Dixon plot.

8.5.1 Lineweaver-Burk plot

In a Lineweaver-Burk plot, both the independent variable (substrate concentration) and the dependent variable (initial reaction velocity) are reciprocally tranformed. The program can be instructed to produce a Lineweaver-Burk diagnostic plot by inserting the line `plot Lineweaver-Burk` into the `[velocity]` section of the script file. The `plot` command must be placed before the first data file is named.

Example 1

In this extended example the program will produce five different Lineweaver-Burk plots, corresponding to the concentration of substrate A = 1, 2, 4, 8, and 16. The reaction mechanism follows a rapid equilibrium random Bi-Bi enzyme system as defined in ref. [6] (p. 643). Note that for the same global data set we produce five separate plots because the system at hand involves *three* varied concentrations (of substrates A, P, and Q). Note that the different graphs required in this example are identified by the keyword `graph`, followed by a short descriptive title (*e.g.*, `A = 4`). A sample output file produced by DynaFit is shown in Figure 8.1.

```
[task]

   task = simulate
   data = velocities

[mechanism]

   E + A <==> EA         :   Ka     dissoc.
   E + B <==> EB         :   Kb     dissoc.
   E + A + B <==> EAB    :   Kab    dissoc.

   EAB ---> E + P + Q    :   kp

   E + P <==> EP         :   Kp     dissoc.
   E + Q <==> EQ         :   Kq     dissoc.
   E + P + Q <==> EPQ    :   Kpq    dissoc.

   EPQ ---> E + A + B    :   k-p

[constants]

   Ka = 1 ?, Kb = 2 ?, Kab = 3 ?
   kp  = 2 ?

   Kp = 5 ?, Kq = 6 ?, Kpq = 7 ?
   k-p = 1 ?

[concentrations]

   E = 0.001

[responses]

   P = 1

[progress]

   rapid equilibrium

[velocity]
```

```
directory  ./examples/segel/Ch9/N/data
extension  txt
mesh       from 1 to 32 step 2 logarithmic
error      linear 0.000001, 0.0001
variable   B
plot       Lineweaver-Burk

graph A = 1

file p0a1 | conc P = 0 , Q = 0 , A = 1
file p1a1 | conc P = 2 , Q = 2 , A = 1
file p2a1 | conc P = 4 , Q = 4 , A = 1
file p3a1 | conc P = 8 , Q = 8 , A = 1
file p4a1 | conc P = 16, Q = 16, A = 1

graph A = 2

file p0a2 | conc P = 0 , Q = 0 , A = 2
file p1a2 | conc P = 2 , Q = 2 , A = 2
file p2a2 | conc P = 4 , Q = 4 , A = 2
file p3a2 | conc P = 8 , Q = 8 , A = 2
file p4a2 | conc P = 16, Q = 16, A = 2

graph A = 4

file p0a3 | conc P = 0 , Q = 0 , A = 4
file p1a3 | conc P = 2 , Q = 2 , A = 4
file p2a3 | conc P = 4 , Q = 4 , A = 4
file p3a3 | conc P = 8 , Q = 8 , A = 4
file p4a3 | conc P = 16, Q = 16 , A = 4

graph A = 8

file p0a4 | conc P = 0 , Q = 0 , A = 8
file p1a4 | conc P = 2 , Q = 2 , A = 8
file p2a4 | conc P = 4 , Q = 4 , A = 8
file p3a4 | conc P = 8 , Q = 8 , A = 8
file p4a4 | conc P = 16, Q = 16, A = 8
```

```
   graph A = 16

   file p0a5 | conc P = 0 , Q = 0 , A = 16
   file p1a5 | conc P = 2 , Q = 2 , A = 16
   file p2a5 | conc P = 4 , Q = 4 , A = 16
   file p3a5 | conc P = 8 , Q = 8 , A = 16
   file p4a5 | conc P = 16, Q = 16, A = 16

[output]

   directory  ./examples/segel/Ch9/N/output

[end]
```

8.5.2 Dixon plot

In a Dixon plot, both the independent variable (concentration of an inhibitor) plotted directly against the reciprocal initial reaction velocity. The program can be instructed to produce a Dixon diagnostic plot by inserting the line `plot Dixon` into the `[velocity]` section of the script file. The `plot` command must be placed before the first data file is named.

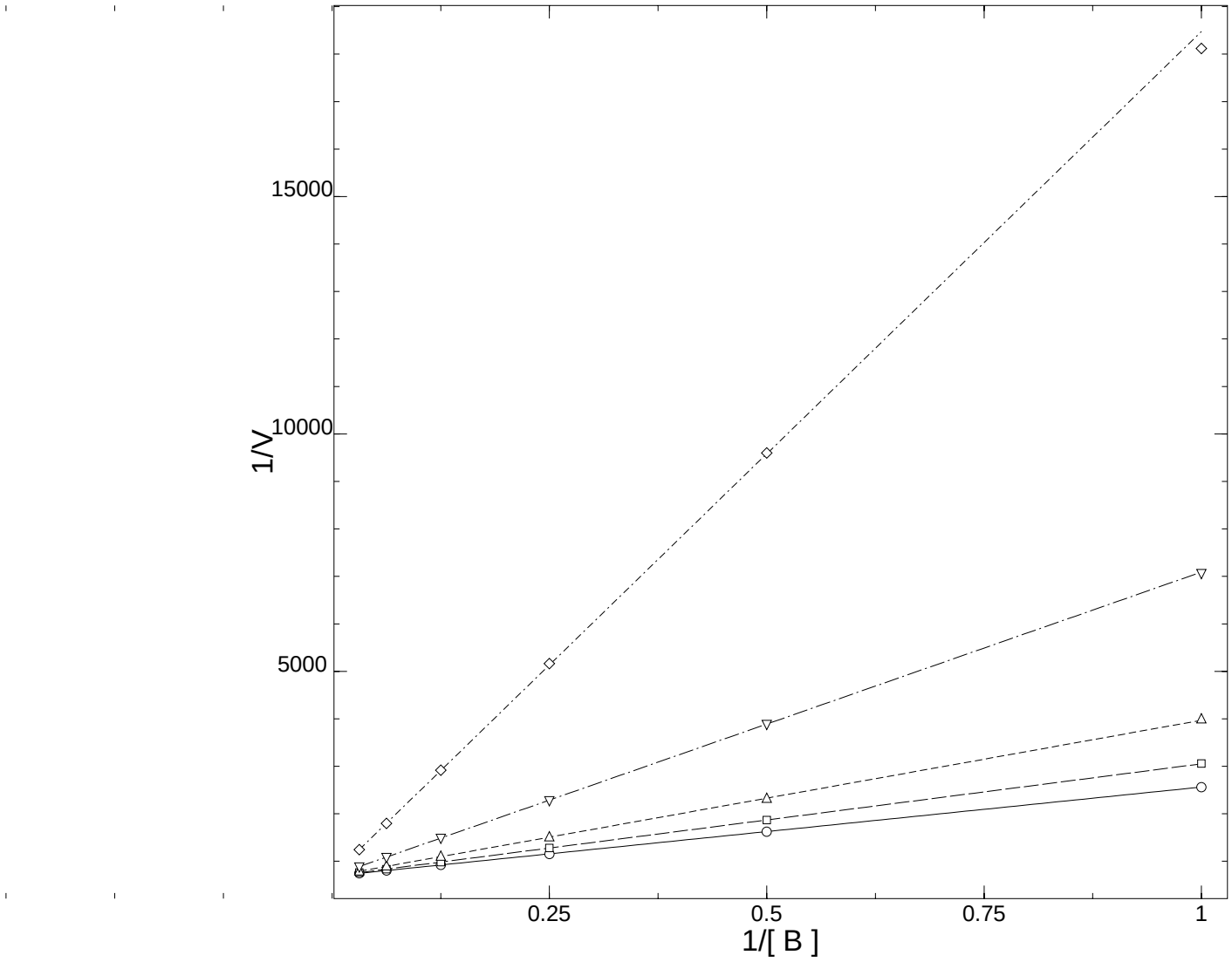

Figure 8.1: Postscript graphics generated by program DynaFit, showing one of five Lineweaver-Burk plots defined in Example 1.

Chapter 9

Equilibria

DynaFit can be used to simulate or fit experimental data obtained in equilibrium binding studies, in which a chemical or biochemical mixture is first allowed to achieve equilibrium composition. Subsequently, the molar responses of one or more chemical species are used to measure their concentrations. The location of experimental data files is indicated in the `[equilibria]` section of the DynaFit script. This chapter describes the necessary syntax.

The computation of initial velocities is initiated by the sequence `data = equilibria` in the `[task]` section of the script file (see Chapter 1.1). The keywords that can be found in the `[equilibria]` segment of the DynaFit script are listed below.

```
concentration
directory
error
extension
file
from .. to .. step
linear
logarithmic
mesh
plot
variable
```

9.1 Location of data files

Equilibrium data files contain, in the first column, the concentration of a certain reactant and, in the second column, the magnitude of a certain physical quantity (e.g., fluorescence) proportional to the concentration of one or more reactants. The formal rules for using the keywords `directory`, `file` or `files`, and `extension`, are the same as was described in section 8.3.

Example 1

The directory ./TEST/DATA contains four files named F1.TXT, F2.TXT, F3.TXT and F4.TXT, each of which contains in the first column the concentration of the ligand and in the second column the fluorescence intensity associated with the protein. It is assumed that the free protein is more fluorescent at the given wavelength than the protein–ligand complex.

```
[task]
    data  = equilibria
    task  = fit
[mechanism]
    protein + ligand <==> complex  :  Keq  dissoc
[constants]
    Keq = 0.01 ?
[responses]
    protein = 100 ? , complex = 10 ?
[equilibria]
   directory ./test/data
   extension txt
   variable ligand
   file f1 | concentration protein = 1
   file f2 | concentration protein = 2
   file f3 | concentration protein = 4
   file f4 | concentration protein = 8
[end]
```

9.2 Restrictions on mechanism

Even more important than our inability to determine certain equilibrium constants is the fact that not all elementary equilibrium constants, which describe individual steps in a reaction mechanism, can even be considered as independent parameters. Some equilibrium constants necessarily must be expressed only as a combination of other equilibrium constants. This has important implications for the way equilibrium binding mechanisms are written in DynaFit script files (see section 3.1).

9.2.1 Branched pathways and equilibrium binding

Consider the following branched reaction mechanism:

			$k_{\rightarrow}$	$k_{\leftarrow}$
$A + B$	$\rightleftharpoons$	AB	k_1	k_2
$AB + C$	$\rightleftharpoons$	ABC	k_3	k_4
$A + C$	$\rightleftharpoons$	AC	k_5	k_6
$AC + B$	$\rightleftharpoons$	ABC	k_7	k_8

It transpires that all eight rate constants in this mechanism can be uniquely determined from the experimental data only if we observe the evolution of the biochemical mixture over time. This statement applies even for the hypothetical case when the measurements are performed virtually with zero experimental error.

If the mixture is first allowed to come to equilibrium and then a measurement is performed on it, there is no possibility of determining the values of all individual equilibrium constants $K_{12} = k_1/k_2$, $K_{34} = k_3/k_4$, $K_{56} = k_5/k_6$, and $K_{78} = k_7/k_8$. The branched pathway in the reaction mechanism must be somehow eliminated, because one of the equilibrium constants is expressed in terms of the remaining three.

			$k_{\rightarrow}^{\leftarrow}$
$A + B$	$\rightleftharpoons$	AB	K_{12}
$AB + C$	$\rightleftharpoons$	ABC	K_{34}
$A + C$	$\rightleftharpoons$	AC	K_{56}
$AC + B$	$\rightleftharpoons$	ABC	K_{78}

The only way to eliminate a branched pathway from the reaction mechanism, without arbitrarily deleting those reaction steps that we know for certain must be present, is to formulate the reaction mechanism in terms of the overall formation of complex species, even if it means using elementary reactions of higher order (molecularity). In this example, we will consider the overall formation constant of the ternary complex ABC.

			$k_{\rightleftharpoons}$
$A + B$	$\rightleftharpoons$	AB	K_{12}
$AB + C$	$\rightleftharpoons$	ABC	K_{34}
$A + B + C$	$\rightleftharpoons$	ABC	K_{abc}

The above reaction mechanism is the only one that should be used for the biochemical mixture consisting of the molecular species, A, B, C, AB, AC, and ABC, if we can measure only the composition of the mixture at equilibrium (as opposed to monitoring the formation or dissociation of the complexes over time). This follows from the general principle that at equilibrium the chemical system has lost the "memory" of how it arrived at the equilibrated state.

9.3 Example problem

This example problem is taken from the published biochemical literature [8] and is a part of the DynaFit distribution.

Human recombinant cyclophilin was incubated with a fluorescent Cyclosporin-A analog, identified as the fluorescent `probe*` in the reaction mechanism below. The dissociation constant for the probe molecule ($K_1 = 5.3$ nM) was measured in dependently. In a series of experiments, 200 nM of the probe molecule and 500 nM of a newly synthesizes Cyclosporin-A analog were mixed with varying amounts of cyclophilin (zero to 650 nM). Upon binding to cyclophilin, the fluorescent probe increases its specific molar response from approximately 0.2 fluorescence units per micromole per liter to about 0.5 fluorescence units per micromole per liter. Fluorescence intensity was recorded after 60 minutes of incubation and entered into the following text file, containing in the first column the protein concentration in micromoles per liter. The datafile was saved on the disk under the name ./EXAMPLES/CYCLOPHILIN/DATA/CSA.TXT.

```
[P],uM    fluorescence
```

```
0.00      0.420
0.00      0.476
0.01      0.588
0.01      0.634
0.02      0.727
0.02      0.802
0.03      0.951
0.04      1.119
0.06      1.025
0.08      1.136
0.10      1.116
0.12      1.545
0.17      1.758
0.20      2.028
0.22      2.204
0.26      2.166
0.27      2.529
0.30      2.808
0.34      2.928
0.38      2.871
0.42      2.907
0.48      2.727
0.52      2.810
0.56      3.042
0.60      2.770
0.65      3.272
```

The goal of this experiment [8] was to determine the dissociation equilibrium constant of the "dark" (non-fluorescent) protein ligand complex formed between cyclophilin and the newly synthesized Cyclosporin-A analog. To this end, we have prepared a DynaFit script file as follows.

```
[task]
   task = fit
   data = equilibria
[mechanism]
   protein + probe* <==> complex*  :  K1  dissoc
   protein + ligand <==> complex   :  K2  dissoc
[constants]
   K1 = 0.0053, K2 = 0.02 ?
```

```
[concentrations]
   probe* = 0.2, ligand = 0.5
[responses]
   probe* = 2 ?, complex* = 10 ?
[equilibria]
   variable protein
   file ./examples/cyclophilin/data/csa.txt
[end]
```

When the script was loaded and executed in DynaFit, the program generated the numerical solution in the LaTeX file ./OUTPUT/INDEX.TEX. A section of this file was pasted into this document as Table 9.1. Thus, the dissociation constant of the cyclophilin-ligand complex is 44 ± 14 nM.

Set	Par.	No.	Initial	Fit	Error	%Error
	K_2	*1*	0.02	0.044	0.014	33
	$r_{\text{probe}*}$	*2*	2	2.5	0.42	17
	$r_{\text{complex}*}$	*3*	10	16	0.51	3.2

Table 9.1: Fluorescence displacement assay for Cyclosporin-A analogs binding to recombinant human cyclophilin. Parameters and formal standard errors

DynaFit also generated a graphical output file in the Encapsulated Postscript (EPS) format, named ./OUTPUT/EPS/FIT_0101.EPS, which was directly imported into this document as Figure 9.1.

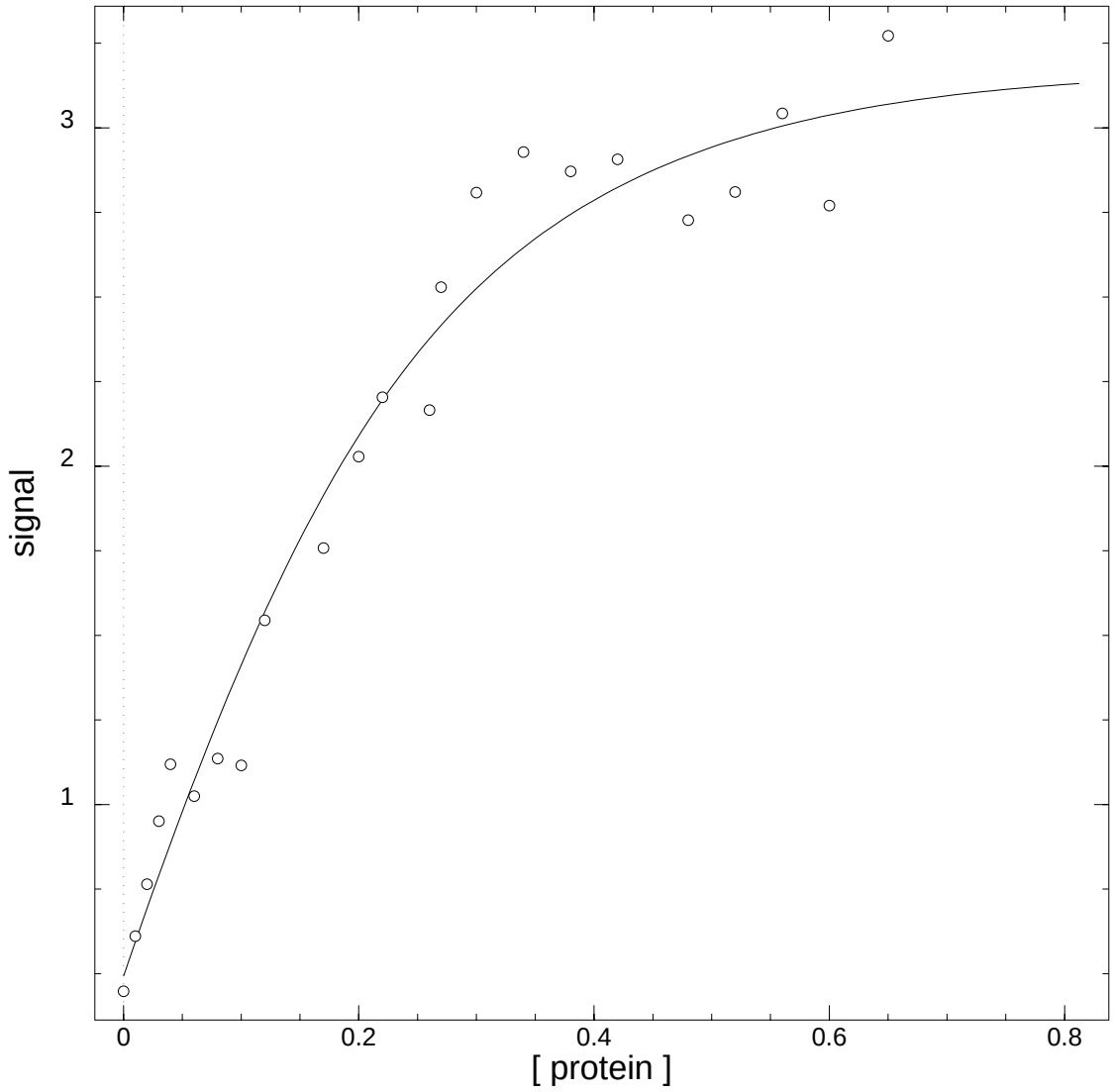

Figure 9.1: Fluorescence displacement assay for Cyclosporin-A analogs binding to recombinant human cyclophilin.

Chapter 10

'Sweeping' rate constant values

A special section of the script file was designed for the purpose of simulating a whole set of progress curves at different values of one or more rate constants. Such exploratory simulations often reveal the properties of the biochemical system which certainly are not apparent to the uninitiated observer (see the script file **oscill.txt** in the directory **script/simulate**).

Example

```
[mechanism]
  S1 + E <===> S1.E     :    k      ks1
    S1.E  ---> E + S2   :    kr1
  S2 + E <===> S2.E     :    k      ks2
          ---> S1       :    v1
       S2 --->          :    v2
[constants]
    k = 10
    ks1 = 10, kr1 = 10
    ks2 = 0.05
    v1 = 1, v2 = 1
[sweep]
    ks2 = 0.01, 0.03, 0.09, 0.27, 0.81, 2.43
```

Here the program is ordered to simulate six different progress curves in one run and display them in a single graph. Each progress curve corresponds

to one of the values of the rate constant k_{s2} liste in the `[sweep]` section of the script file.

10.1 Formal rules

The following are formal rules which the `[sweep]` section must follow in order to simulate and plot a certain number of reaction progress curves at once.

1. Up to 16 progress curves can be simulated.

2. Any number of rate constants or initial concentrations can be varied simultaneously.

3. Each set of rate constant values can be specified either by enumeration, or by the `from .. to .. step` construct, optionally with the keyword `logarithmic`.

Example 1

```
[sweep]
   k1 = 1,2,3,4,5

[sweep]
   k1 = from 1 to 5 step 1

[sweep]
   k1 = 1,2,4,8,16

[sweep]
   k1 = from 1 to 16 step 2 logarithmic
```

In *Example 1* the program will simulate five progress curves at five different values of rate constant k_1 (only one of the four entries above would appear in the given script file).

Example 2

```
[sweep]
   k1 = 1,2,3,4,5
   k2 = 10,20,30,40,50

[sweep]
   k1 = from 1 to 5 step 1
   k2 = from 10 to 50 step 10
```

In *Example 2* the program will simulate five progress curves at five different values of rate constants k_1 and k_2. The first progress curve is simulated with $k_1 = 1$, $k_2 = 10$, and the fifth progress curve with $k_1 = 5$, $k_2 = 50$.

10.2 Limitations

In the present version of the program it is not possible to simultaneously vary concentrations and rate constants to simulate a family of progress curves. Also, it is currently not possible to simulate an entire family of dose-response curves (either initial velocities or equilibrium binding data) by listing the values of rate or equilibrium constants in the `[sweep]` section of the script file.

Chapter 11

Initialization file

DynaFit is initialized at program startup and before each individual `[task]` (see Section 1.1) is attempted. The initialization parameters are stored either in the script file (see Section 2.9) or in a special initialization file described in this chapter.

11.1 Initialization file

The default initialization file settings.ini must be present in the directory ./system/dynafit/ at the start of program execution. This file is read and processed every time DynaFit is started, and every time a new task (for example, least-squares fit of initial velocities) is selected from the main menu.

The initialization file is divided into several sections dealing with different topics, such as input of experimental data, output of results, and default settings for various numerical algorithms. Sections are separated by section names enclosed in angle brackets ("`<...>`"). Each section of the default initialization file is explained below.

11.2 Control parameters

This part of the DynaFit scripting manual describes the sections of the initialization file, and the entries contained in each section.

11.2.1 <ODE Solver>

This section contains the default settings for algorithm LSODE [9], used in DynaFit to compute the time course of biochemical reactions. The distribution copy of file dynafit/settings.ini contains the following settings, which are explained below.

```
<ODE Solver>

   Iterations        = 500
   AbsTolerance      = 1.e-16
   RelTolerance      = 1.e-6
   AbsAccurateTol    = 1.e-16
   RelAccurateTol    = 1.e-6
   NonNegative       = No
   SmartWeighting    = Yes
```

Iterations = Maximum number of LSODE iterations

This parameter represents the number of integration steps that the algorithm LSODE [9] is allowed between two successive output points on the reaction progress curve. For example, if the progress curve consists of equally spaced data points, with one-second interval between two adjacent points, LSODE is allowed at most 500 internal integration steps to traverse each of the one-second intervals.

Users of DynaFit normally should not change the value of this parameter. Only if LSODE fails during the numerical simulation of a particularly difficult kinetic problem, the user might attempt to increase this parameter to a larger value (e.g, 5000). Practical experience shows that LSODE can fail on kinetic mechanisms which include zeroth-order steps, for example, a removal of the enzyme at a constant rate due to zeroth-order deactivation.

AbsTolerance = Absolute tolerances for local truncation error

The absolute tolerance ϵ_{abs} represent the absolute precision with which the algorithm LSODE [9] computes concentrations in simulating the time course of biochemical reactions. The precise meaning of the absolute error tolerance depends on the concentration scale used in the particular kinetic problem.

For example, if all concentrations are in the micromolar scale, the absolute error tolerance in moles per liter is $10^{-6} \times 10^{-15} = 0.000001$ fM.

The value of `AbsTolerance` normally should not be changed (see comments on `RelTolerance`below). In some cases it might be desirable to request pure relative error tolerances, by setting `AbsTolerance` to zero. Setting the absolute tolerances should be avoided in all kinetic mechanisms where a certain concentration can possibly become zero (for example, because a reacting species is continuously removed from the systems).

`RelTolerance` = Relative tolerances for local truncation error

The relative tolerance ϵ_{rel} represents the relative precision with which LSODE [9] computes concentrations. The default value (0.000001) means that all concentrations are computed with six-digit accuracy.

This value should not normally be changed. Only if LSODE fails during the numerical integration, the user might with to request a five-digit accuracy (0.00001). However it should be realized that the local truncation errors accumulate. LSODE uses several hundred internally generated output points to traverse a typical reaction progress curve. In each internal integrations step the local truncation errors accumulate, which can make the global truncation error significantly large.

`AbsAccurateTol` = Enhanced absolute tolerances for local truncation error

The "enhanced" absolute tolerance ϵ^*_{abs} represent the absolute precision with which the algorithm LSODE [9] computes concentrations in simulating the time course of biochemical reactions *in the final step of a least-squares regression analysis*, when the standard error of parameters is being computed.

This value should be either equal to ϵ^*_{abs} or, if necessary, approximately an order of magnitude smaller. The user should experiment with the settings for ϵ^*_{abs} until for the given class of problems the formal standard error of adjustable parameters does not change upon making ϵ^*_{abs} smaller.

`RelAccurateTol` = Enhanced relative tolerances for local truncation error

The "enhanced" absolute tolerance ϵ^*_{rel} represent the absolute precision with which the algorithm LSODE [9] computes concentrations in simulating the

time course of biochemical reactions *in the final step of a least-squares regression analysis*, when the standard error of parameters is being computed.

This value should be either equal to $\epsilon^*_{\rm rel}$ or, if necessary, approximately an order of magnitude smaller. The user should experiment with the settings for $\epsilon^*_{\rm rel}$ until for the given class of problems the formal standard error of adjustable parameters does not change upon making $\epsilon^*_{\rm rel}$ smaller.

`NonNegative` = Non-negativity flag for ODE solution

For certain class of problem in chemical kinetics, it is necessary to take special step in order to maintain non-negativity of solution, that is, of the reactant concentrations. This arises in systems with constant outflow of materials, such as in certain metabolic or compartmental systems. If the solution of the given ODE systems ever becomes negative, the user may consider setting the `NonNegative` flag to `no`.

`SmartWeighting` = Flag for automatic error tolerances

The allowable values for this control parameter are `yes` and `no`. If the flag is set to `no`, the program uses error control parameters as is specified in the preceding section of this paragraph.

If `SmartWeighting` is set to `yes`, the program determines suitable error control parameters according to the following algorithm. First, the largest concentration ($c_{\rm max}$, in the chosen units) is found among the chemical species present at time zero. Second, the error control parameters are set according to the values listed in Table 11.1.

$c_{\rm max}$	$\epsilon_{\rm abs}$	$\epsilon_{\rm rel}$	$\epsilon^*_{\rm abs}$	$\epsilon^*_{\rm rel}$
< 10	10^{-17}	10^{-7}	10^{-20}	10^{-8}
< 100	10^{-16}	10^{-6}	10^{-17}	10^{-7}
> 100	10^{-16}	10^{-6}	10^{-16}	10^{-6}

Table 11.1: Automatic setting of relative and absolute error tolerance in numerical integration of ODE system using algorithm LSODE.

11.2.2 <Equilibrium Solver>

The composition of chemical and biochemical mixtures at equilibrium is computed by using a modification of the numerical algorithm EQUIL described in [10]. The default settings for the equilibrium solver are listed below.

```
<Equilibrium Solver>

   Iterations        = 500
   AbsTolerance      = 1.e-15
   RelTolerance      = 1.e-6
```

Iterations = Maximum number of iterations for the equilibrium solver

The computation of chemical and biochemical equilibria is an iterative procedure [10], based on the multidimensional Newton-Raphson method for solving a system of nonlinear algebraic equations (em i.e., the mass balance equations for the given system). The parameter `Iterations` sets the maximum number of iterations that the solver is allowed to execute. The default value should be increased approximately ten-fold if the equilibrium solver fails on a particularly difficult problem.

AbsTolerance = Absolute tolerances for local truncation error

RelTolerance = Relative tolerances for local truncation error

The two parameters named above control the absolute and relative error, respectively, allowed in the iterative computation of chemical equilibria using a variant of the algorithm EQUIL [10]. The iteration is terminated when the relative difference between two successive estimates of all species concentrations is smaller than `RelTolerance` and, simultaneously, when the absolute difference between two successive estimates of all species concentrations is smaller than `AbsTolerance`.

The user should not change these values unless the equilibrium solver reports a failure to converge, in which case the error tolerances should be increased (*e.g.*, by an order of magnitude). It is important to emphasize that relaxing the error tolerances increases the numerical error with which the results are computed, and therefore it should be done with caution.

11.2.3 <Marquardt>

Nonlinear least-squares regression of experimental data [11] is accomplished in DynaFit by using the Levenberg-Marquardt (LM) algorithm [12]. For full explanation of the control parameters described here, the reader is directed to these references. The default values of parameters controlling the LM algorithm are listed below.

```
<Marquardt>

   Iterations        = 50     ; per parameter
   Subiterations     = 10
   Interrupt         = 10
   Restarts          = 2
                              ; Marquardt parameter:
   InitLambda        = 1
   UpLambda          = 4
   DownLambda        = 2
   ReinitLambda      = 0
                              ; Line-search parameters:
   StepLine          = 0
   StrideLine        = 1.0
                              ; Output options:
   IterPrint         = Yes
   SubiterPrint      = Yes
   FilePrint         = No
   ShowProgress      = Yes
                              ; Stopping criteria:
   StopLambda        = 0
   StopParam         = 0.00001
   StopSquares       = 0.000001
```

`Iterations` = Number of iterations

This is the maximum number of iterations *per optimized parameter* that will be taken by the LM least-squares fitting routine. For particularly difficult problems, it might be appropriate to increase the value of `Iterations` to approximately 100. Problems that require larger number of iterations are

probably ill-posed and require decreasing the number of adjustable parameters.

`Subiterations` = Number of sub-iterations

The LM algorithm [12] contains a major iteration loop and a minor iteration loop. The minor iteration loop, controlled by the parameter `Subiterations`, is trying to navigate between the steepest-descend (or gradient) method and a quadratic approximation (or Gauss-Newton) method of minimization. The default value should rarely be changed. For a particularly difficult problem the user might increase the number of sub-iterations to at most 100.

`Interrupt` = Number of iterations before switching to interactive mode

The LM least-squares fitting algorithm will proceed automatically for `Interrupt` iterations, after which it will stop and ask the user whether to proceed with the minimization or accept the current values of adjustable parameters.

`Restarts` = Number of times the LM algorithm should restart

According to the recommendation of Seber and Wild [11], the LM algorithm should be restarted after an apparent minimum has been found on the least-squares surface. For difficult problems with shallow minima on the least-squares surface, restarting the algorithm after it apparently converged proves useful. In extreme cases the user might increase the number of restarts to approximately five.

`InitLambda` = Initial value of the Marquardt compromise parameter

The precise meaning of the Marquardt compromise parameter λ is explained in the original report [12], where the proposed value was 0.1. However, we found that in many chemical kinetic problems, convergence of the algorithm can be improved by starting from a larger value. If, for the given problem, the LM algorithm appears to take steps that are too large, the `InitLambda` might be increased to a value of approximately 10.

`UpLambda` = Increase in the Marquardt compromise parameter

After an *unsuccessful* step in the LM minimization algorithm, the Marquardt compromise parameter λ is increased by a factor `UpLambda`. This makes the search algorithm more similar to the steepest descend method than to the Gauss-Newton quadratic approximation method [11]. Marquardt [12] originally proposed the value of 10.0 for the fractional increase. However, we found that a smaller value increases the robustness of the search albeit at the cost of somewhat slower convergence in the case of easy minimization problems.

`DownLambda` = Decrease in the Marquardt compromise parameter

After a *successful* step in LM minimization algorithm, the Marquardt compromise parameter λ is decreased by a factor `DownLambda`. This makes the search algorithm more similar to the Gauss-Newton quadratic approximation method [11] than to the steepest descend method. Marquardt [12] originally proposed the value of 10.0 for the fractional decrease. However, we found that a smaller value increases the robustness of the search albeit at the cost of somewhat slower convergence in the case of easy minimization problems.

`ReinitLambda` = Reinitialization flag for the Marquardt compromise parameter

This flag with possible values `yes` and `no` determines whether the Marquardt compromise parameter λ will be reset to its initial value `InitLambda` after each successful iteration. The flag should almost always be set to `no`, except for extremely ill-conditioned minimization problems where the user might set it to `yes` as a very last resort strategy.

`StepLine` = Number of steps in a line-search algorithm

Some authors [11] recommend that the LM minimization algorithm [12] be augmented in each iteration with a line search, along the current direction in the parameter space. The parameter `StepLine` gives the maximum number of steps to be taken in this line-search (default value zero). Enabling line-search in the LM algorithm can cause a significant slowdown, but for many ill-conditioned problems it is useful or even necessary.

`StrideLine` = Initial step size in the line-search algorithm

This parameter control the initial step size in the line-search algorithm, which is an implementation of the *golden section* search as described in [13] (p. 397). The initial step size parameter `StrideLine` is the fraction (by default, 100%) of the current ML step size that is used to bracket the minimum on the least squares surface. The value `StrideLine = 1.0` has proven useful in many chemical kinetic problems. For particularly ill-conditioned systems, the user might decrease this parameter to approximately 0.3.

`IterPrint` = Printing of major iterations in the LM algorithm

If this flag is set to `yes`, the program will display on the screen or write into a log file (see below) the adjustable parameters in every iteration of the LM minimization algorithm. This might be useful in analyzing the progress of minimization in particularly difficult cases.

`SubiterPrint` = Printing of minor iterations in the LM algorithm

If this flag is set to `yes`, the program will display on the screen or write into a log file (see below) the adjustable parameters in every sub-iteration of the LM minimization algorithm. This might be useful in analyzing the progress of minimization in particularly difficult cases.

`FilePrint` = Logging the progress of the LM minimization in a disk file

If this flag is set to `yes`, the program will write the values of all adjustable parameters into an output file in each iteration and / or subiteration of the LM minimization algorithm.

`ShowProgress` = Displaying the progress of the LM minimization on the screen

If this flag is set to `yes`, the program will display the values of all adjustable parameters on the screen in each iteration and / or subiteration of the LM minimization algorithm.

`StopLambda` = Stopping criterion for the Marquardt compromise parameter

Duggleby [14] has designed a least-squares fitting program based on the LM algorithm, in which the minimization is continued until the Marquardt compromise parameter λ decreases below a certain limiting value. When λ is sufficiently small, typically about 0.00001, the LM algorithm behaves as the quadratic Gauss-Newton method, which indicates that it is near a true minimum. The parameter `StopLambda` sets this critical value of λ. When `StopLambda` is initialized to zero, the stopping criterion based on λ is not used.

`StopParam` = Stopping criterion for adjustable parameters

The control parameter `StopParam` specifies the largest allowable relative change in the value of adjustable parameters. If in the given iteration the relative changes in the value of *all* adjustable parameters is smaller than `StopParam`, the iterations in the LM algorithm may be terminated. However, certain classes of adjustable parameters (rate constants, initial concentrations, molar responses, and baseline offsets) have their own characteristic termination criteria explained in Section 11.2.5.

`StopSquares` = Stopping criterion for the reduced sum of squares

The control parameter `StopSquares` specifies the largest allowable relative change in the sum of squared deviations. If in the given iteration the relative changes in the sum of squares is smaller than `StopSquares`, the iterations in the LM algorithm may be terminated.

> The Levenberg-Marquardt algorithm [12] is allowed to terminate the iterations only if *all* of the termination criteria are satisfied simultaneously.

11.2.4 `<Confidence Intervals>`

The set of initialization parameters listed below controls a specialized search algorithm for the determination of approximate *confidence intervals*, sometimes referred to as *inference intervals*. The reader is directed to publica-

tions [15, 16] in which the significance of statistical confidence intervals is explained, along with basic strategies for systematic search methods. DynaFit uses the "profile-t" method described by Bates and Watts ([17], p. 205, 297).

```
<Confidence intervals>

   Level             = 99.0
   Interrupt         = 501
   OnlyConstants     = yes
   FTestLevel        = 99.0
```

`Level` = Percentage confidence level

This parameter sets the desired confidence level (in percentage points) of the confidence intervals for adjustable model parameters. Typical values found in the literature are 68%, 90%, or 95%. For most rigorous investigations, the user might select 99% confidence level on model parameters.

`Interrupt` = Interrupt least-squares fit after every n iteractions

See also section sect:Marquardt-Interrupt. The LM least-squares fitting algorithm will proceed automatically for `Interrupt` iterations, after which it will stop and ask the user whether to proceed with the minimization or accept the current values of adjustable parameters.

`OnlyConstants` = Exclude other types of parameters from minimization

In many cases it is advangeous to perform confidence interval search while treating all model parameters other than rate constants (or equilibrium constants) as fixed parameters. This speeds up the confidence interval search considerably.

`FTestLevel` = Confidence level for model discrimination

This parameter represents the percentage confidence level (typically, 95% or 99%) to be used for model discrimination analysis using the method

described by Mannervik [18]. In particular, `FTestLevel` determines the probability level at which the Fisher's F-statistic should be computed for nested models. For example, assume that the number of data points is n, and the the number of adjustable parameters is p_1 in the first model and p_2 in the second model to be compared. Then, setting `FTestLevel = 95.0`, the critical value of Fisher's F-statistic is computed as $F_{0.025}(p_2 - p_1, n - p_2)$. For further details, see ref. [18].

11.2.5 <Constraints>

DynaFit is treating all adjustable parameters in the least-squares regression as intrinsically *constrained* by certain bounds. This section of the initialization file sets those bounds for the four different kinds of adjustable parameters. It also sets the stopping criteria for each kind of parameter that is used in the Levenberg-Marquardt minimization algorithm.

```
<Constraints>
                              ; Rate constants
   RateErrAbsRel     = 0
   RateError         = 1000000.0
   RateStopAbsRel    = 0
   RateStop          = 0.0001
                              ; Concentrations
   ConcErrAbsRel     = 0
   ConcError         = 0.10   ; Titration error (10%)
   ConcStopAbsRel    = 0
   ConcStop          = 0.01
                              ; Responses
   RespErrAbsRel     = 0
   RespError         = 1000.0
   RespStopAbsRel    = 0
   RespStop          = 0.01
                              ; Offsets
   OffsErrAbsRel     = 1
   OffsError         = 1000.0
   OffsStopAbsRel    = 1
   OffsStop          = 1
```

`RateErrAbsRel` = **Flag for absolute or relative bounds**

`RateError` = **Optimization bounds for rate constants**

When `RateErrAbsRel` is set to zero (`0`), the program will interpret the value of `RateError` as a relative bound. In this case, the upper bound on optimized rate constants is computed as `RateError` times the initial estimated value. The lower bound on all optimized rate constants is computed as the initial value divided by `RateError`.

On the other hand, when `RateErrAbsRel` is set to one (`1`), the program will interpret the value of `RateError` as an absolute bound. In this case, the upper bound on optimized rate constants is computed as `RateError` plus the initial estimated value. The lower bound on all optimized rate constants is computed as the initial value minus `RateError`.

`RateStopAbsRel` = **Flag for absolute or relative stopping criterion**

`RateStop` = **Stopping criterion for rate constants**

When `RateStopAbsRel` is set to zero (`0`), the program will interpret the value of `RateStop` as a relative stopping criterion. In this case, the Levenberg-Marquardt iterations are terminated when the relative difference between two successive estimates of all optimized rate constants decrease below `RateStop`.

On the other hand, when `RateStopAbsRel` is set to one (`1`), the program will interpret the value of `RateStop` as an absolute stopping criterion. In this case, the Levenberg-Marquardt iterations are terminated when the absolute difference between two successive estimates of all optimized rate constants decrease below `RateStop`.

`ConcErrAbsRel` = **Flag for absolute or relative bounds**

`ConcError` = **Optimization bounds for concentrations**

When `ConcErrAbsRel` is set to zero (`0`), the program will interpret the value of `ConcError` as a relative bound. In this case, the upper bound on optimized concentrations is computed as `ConcError` times the initial estimated value. The lower bound on all optimized concentrations is computed as the initial value divided by `ConcError`.

On the other hand, when `ConcErrAbsRel` is set to one (`1`), the program

will interpret the value of `ConcError` as an absolute bound. In this case, the upper bound on optimized concentrations is computed as `ConcError` plus the initial estimated value. The lower bound on all optimized concentrations is computed as the initial value minus `ConcError`.

`ConcStopAbsRel` = **Flag for absolute or relative stopping criterion**

`ConcStop` = **Stopping criterion for concentrations**

When `ConcStopAbsRel` is set to zero (`0`), the program will interpret the value of `ConcStop` as a relative stopping criterion. In this case, the Levenberg-Marquardt iterations are terminated when the relative difference between two successive estimates of all optimized concentrations decrease below `ConcStop`.

On the other hand, when `ConcStopAbsRel` is set to one (`1`), the program will interpret the value of `ConcStop` as an absolute stopping criterion. In this case, the Levenberg-Marquardt iterations are terminated when the absolute difference between two successive estimates of all optimized concentrations decrease below `ConcStop`.

`RespErrAbsRel` = **Flag for absolute or relative bounds**

`RespError` = **Optimization bounds for molar responses**

When `RespErrAbsRel` is set to zero (`0`), the program will interpret the value of `RespError` as a relative bound. In this case, the upper bound on optimized molar responses is computed as `RespError` times the initial estimated value. The lower bound on all optimized molar responses is computed as the initial value divided by `RespError`.

On the other hand, when `RespErrAbsRel` is set to one (`1`), the program will interpret the value of `RespError` as an absolute bound. In this case, the upper bound on optimized molar responses is computed as `RespError` plus the initial estimated value. The lower bound on all optimized molar responses is computed as the initial value minus `RespError`.

`RespStopAbsRel` = **Flag for absolute or relative stopping criterion**

`RespStop` = **Stopping criterion for molar responses**

When `RespStopAbsRel` is set to zero (`0`), the program will interpret the value of `RespStop` as a relative stopping criterion. In this case, the Levenberg-

Marquardt iterations are terminated when the relative difference between two successive estimates of all optimized molar responses decrease below `RespStop`.

On the other hand, when `RespStopAbsRel` is set to one (`1`), the program will interpret the value of `RespStop` as an absolute stopping criterion. In this case, the Levenberg-Marquardt iterations are terminated when the absolute difference between two successive estimates of all optimized molar responses decrease below `RespStop`.

`OffsErrAbsRel` = **Flag for absolute or relative bounds**

`OffsError` = **Optimization bounds for baseline offsets**

When `OffsErrAbsRel` is set to zero (`0`), the program will interpret the value of `OffsError` as a relative bound. In this case, the upper bound on optimized baseline offsets is computed as `OffsError` times the initial estimated value. The lower bound on all optimized baseline offsets is computed as the initial value divided by `OffsError`.

On the other hand, when `OffsErrAbsRel` is set to one (`1`), the program will interpret the value of `OffsError` as an absolute bound. In this case, the upper bound on optimized baseline offsets is computed as `OffsError` plus the initial estimated value. The lower bound on all optimized baseline offsets is computed as the initial value minus `OffsError`.

`OffsStopAbsRel` = **Flag for absolute or relative stopping criterion**

`OffsStop` = **Stopping criterion for baseline offsets**

When `OffsStopAbsRel` is set to zero (`0`), the program will interpret the value of `OffsStop` as a relative stopping criterion. In this case, the Levenberg-Marquardt iterations are terminated when the relative difference between two successive estimates of all optimized baseline offsets decrease below `OffsStop`.

On the other hand, when `OffsStopAbsRel` is set to one (`1`), the program will interpret the value of `OffsStop` as an absolute stopping criterion. In this case, the Levenberg-Marquardt iterations are terminated when the absolute difference between two successive estimates of all optimized baseline offsets decrease below `OffsStop`.

11.2.6 <Simulate>

This section of the DynaFit script file collects control parameters that are used in the simulation of progress curves and initial velocities. The least-squares fit of relevant data is also influenced by these parameters because each regression analysis consists of multiple simulations of the theoretical model using gradually improved values of adjustable parameters.

```
<Simulate>

   Sensitivity       = No
   MeshDefault       = 301
   Equalize          = 1
   FiniteDifference  = 0.0001
   Interpolate       = No
   Increment         = 0.1
   WaitBatch         = 1.0       ; second
```

`Sensitivity` = Flag for the simulation of parametric sensitivities

Parametric sensitivities are partial derivatives of the response function (observed or independent variable) with respect to the adjustable parameters. When the `Sensitivity` is set to `yes`, DynaFit in the simulation mode (as opposed to the least-squares fitting mode) will simulate not only the theoretical model but also the partial derivatives with respect to all those model parameters that would be optimized in the least-squares regression.

Example

In this example, the program will produce not only the simulated progress curves but also the first derivatives of the progress curves with respect to the rate constant k_1 and with respect to the molar response coefficient r_S.

```
[mechanism]
   E + S <===> ES  :  k1   k2
   ES ---> E + P   :  k3
[responses]
   S = 1 ?
```

```
[constants]
   k1 = 10 ?, k2 = 3
...
```

`MeshDefault` = Number of points on a simulated curve

This parameter sets the number of points on each simulated curve used for plotting the best fit model. For example, if a set of experimental data subjected to the nonlinear least-squares regression has only 10 points, the best-fit curve plotted only with 10 points would not appear sufficiently smooth. A reasonable value for this parameter is `MeshDefault = 101` or `301`.

`Equalize` = Flag for equal weighting of data sets

This parameter controls the behavior of the least-squares fitting algorithm in those cases where the individual data files contain different number of data points. When `Equalize` is set to nonzero value (e.g., `Equalize = 1`) the program will adjust the weights in the least-squares regression in such a way that all data sets will have equal influence.

For example, due to a change in instrument settings, two related progress curves might have been collected with different spacing of points on the abscissa (one second and five seconds, respectively). If both reactions were monitored for five minutes, one progress curve would contain 300 data points and the other 60 data points. Let us assume that the two progress curves are taken into a *global* regression analysis. With identical weighting, the 300-point data set would have five a times greater influence on the regression model compared to the 60-point data set. Setting `Equalize = 1` will remedy the situation and assure that two data sets contribute equally.

`FiniteDifference` = Finite-difference coefficient

This parameter sets the coefficient δ in the computation of *forward finite-difference* derivatives. In particular, derivatives of the observed response function y with respect to the adjustable parameter p_i are compute in DynaFit by using the equation 11.1.

$$\mathrm{d}y(\mathbf{p})/\mathrm{d}p_i = \frac{y(p_1, p_2, \ldots, (1+\delta) \times p_i, \ldots, p_n) - y(\mathbf{p})}{\delta \times p_i} \qquad (11.1)$$

A suitable value of δ is 0.0001. Values much smaller than 0.0001 are

problematic because the response function y itself is computed with limited accuracy, for example, by solving *numerically* a system of differential equations.

`Interpolate` = Flag for interpolation of progress curves

In certain numerically difficult problems involving systems of ordinary differential equations (ODEs), the data points might be too widely spaced for the ODE integrator. In this case it might be necessary to allow the integrator to traverse the widely separated output points by using a mesh of interpolation points. This feature should be used sparingly, only if the ODE integrator reports failure to converge, because the interpolation slows down the program significantly. Under normal circumstances the parameter `Interpolate` should be set to the default value `no`.

`Increment` = Time increment in the interpolation of progress curves

If `Interpolate` (see above) is set to `yes`, the value of `Increment` gives the spacing of interpolation points used by the ODE integrator. The time unit used here is the same as that used in the description of the experimental data. For example, let us assume that the experimental data set at hand involves five data points spaced by 1000 seconds. If the ODE integrator fails because the system is numerically too stiff [19] for the wide spacing of output points, we might try to set `Increment = 1.0` or even `Increment = 0.01`.

`WaitBatch` = Time delay in batch simulations

Normally DynaFit requires user input at many points along the data analysis, for example, by requesting the user's confirmation whether a given regression analysis should continue after a certain number of iterations. However, DynaFit can also performs certain tasks in a batch mode, where no user input is required. The parameter `WaitBatch` sets the time in seconds that is used by the program to display images or error messages in the batch mode.

11.2.7 <Filter>

This section of the DynaFit initialization file controls the pre-processing of progress curve data files.

```
<Filter>

   Points            = 1500      ; per dataset
   TMin              = 0.0
   TMax              = 100000.0
   Scale             = seconds
   SetTZero          = No
   SetSigZero        = No
   Smoothing         = 0
```

Points = Maximum points in a data set

If an individual progress curve contains more than Points points, DynaFit will apply a symmetric filter to reduce the number of data items to the number set by this control parameter. By "symmetric" filtering we mean that the program might delete every third data point, or ever other data point, or two thirds of points by *leaving* every third point.

This feature is useful when processing primary data produced by those computer-interfaced spectrophotometers or fluorimeters that generate very closely spaced data (*e.g.*, with the interval 0.1 seconds for a 10-minute experiment).

TMin = Minimum reaction time

Imperfect mixing, temperature equilibration, or other experimental factors that may case irregularities in the initial portion of reaction progress curves. In these special cases, the user might wish to delete automatically the first several points from each data set. The parameter TMin should be set to a non zero value that specifies the cut-off point below which the experimental data excluded from analysis.

`TMax` = Maximum reaction time

In many cases the raw experimental data are collected over a period of time that is excessively long. One way to solve this problem is to manually edit the data files before subjecting them to the statistical analysis by DynaFit. Alternately, the user might set the value of `TMax` to a value that is lower then the default ($t_{\max} = 10000$). Data points that have abscissas larger than `TMax` will be excluded from analysis.

`Scale` = The unit (scale) of time

The allowable values of `Scale` are `microseconds`, `microseconds`, `seconds`, `minutes`, `hours`, or `days`. Upon reading experimental data, DynaFit always convert all time values to the S.I. units (*i.e.*, seconds) by using the proper scaling factor as defined by the keyword `Scale`.

`SetTZero` = Flag for resetting time values to zero

In some cases it is profitable to subtract from the time-coordinate (independent variable) of each data point the value associated with the first data point on the given progress curve. If the `SetTZero` flag is set to `yes`, the program will perform this kind of automatic transformation.

`SetSigZero` = Flag for resetting signal values to zero

In some cases it is profitable to subtract from the signal value (dependent variable) of each data point the value associated with the first data point on the given progress curve. If the `SetSigZero` flag is set to `yes`, the program will automatically perform the subtraction upon reading the raw data from the disk.

`Smoothing` = Number of smoothing passes

Smoothing of time-domain data ([13], p. 650) can be applied to pre-process raw data from particularly noisy spectrometers. If such smoothing is necessary, and if the input data are equally spaced, DynaFit will use a simple *moving window average* method of digital filtering. Only the most immediate neighbors of each data point are used, according to the formula

$$\tilde{y}_i = (y_{i-1} + y_i + y_{i+1})/3 \quad , \quad i = 1, 2, \ldots, N \quad , \tag{11.2}$$

where N is the number of data points in the given progress curve.

The control parameter `Smoothing` sets the number of passes that are applied to each progress curve data set upon reading. The default value is `Smoothing = 0`, which means no smoothing is applied. If the progress curve data are extremely noisy, a useful value is `Smoothing = 3`.

11.2.8 <Output>

The parameters collected in this section of the DynaFit initialization file control the appearance of the output results, both on the screen and in various kinds of output files (graphic files and text files).

```
<Output>

   CreateDirectories  = Yes
   WriteTextFiles     = Yes
   WriteHTMLfiles     = Yes
   WriteLATEXfiles    = Yes
   WriteGIFfiles      = Yes
   WriteTABfiles      = Yes
   WritePSfiles       = Yes
   WriteStoich        = No

   ; The following two lines are Macintosh-specific

   TextFileCreator       = R*ch
   PostscriptFileCreator = gsVR

   ; Best-fit parameters:

   VarianceInflation  = No
   Covariance-Correl  = Yes
   CollinearityIndex  = No
   Eigenvectors       = Yes
   RedundancyGrade    = Yes
```

```
; Graphical analysis of residuals:

Variance-Signal    = No
CumulativeDistrib  = No
NormalPlot         = No
SerialCorrelation  = No
AutoCorrelation    = No
PowerSpectrum      = No
LocalResid         = No

; Numerical analysis of residuals:

RunsOfSigns        = No
Rayner-Best        = No
Kolmogorov-Smirnov = No
Durbin-Watson      = No
Tukey              = No

; Model discrimination analysis:

WriteFTest         = No
```

`CreateDirectories` = **Create output directories?**

If this flag is set to `yes`, DynaFit will attempt to create new output directories as indicated in the `[output]` section of the output file. If the flag is set to `no`, the program will insist that the output directories exist in advance, otherwise it will issue a run-time error and terminate execution.

`WriteTextFiles` = **Create output files in simple text format?**

When this flag is set to `yes`, the program will create output files in simple text format, collecting numerical either the results of numerical simulations or the results of nonlinear least-squares regression.

`WriteHTMLfiles` = **Create output files in HTML format?**

When this flag is set to `yes`, the program will write the computational results on the disk as a collection of HTML files connected by a set of hyperlinks.

Optionally, the graphical output is embedded into the HTML files in the GIF format (see below).

`WriteLATEXfiles` = **Create output files in LaTeX format?**

When this flag is set to `yes`, the program will write the computational results on the disk as a collection of LaTeX files, connected via the LaTeX command `input`. Optionally, the graphical output is embedded into the LaTeX files in the Postscript format (see below).

`WriteGIFfiles` = **Create graphical output files in GIF format?**

When this control parameter is set to `yes`, the program will create all graphs displayed on the screen also as GIF graphical files. These files can be viewed either separately, using a GIF file viewer such as any Web browser, or they can be viewed as part of the HTML files also created by DynaFit.

`WritePSfiles` = **Create graphical output files Postscript format?**

When this flag is set to `yes`, DynaFit will create each graph as a Postscript graphical file on the disk. The graphical Postscript files can be imported into various word processing application programs, or viewed and printed by using specialized Postscript file viewers such as GhostScript and GhostView (http://www.cs.wisc.edu/~ghost/).

`WriteTABfiles` = **Create output files in tab-delimited format?**

When this flag is set to `yes`, the program will write on the disk all numerical data that are needed for the creation of graphs using third-party graphing software, such as SigmaPlot or GnuPlot. The tab-delimited output files contain the experimental data and the best fit interpolated curves suitable for the construction of publication-quality graphs.

`TextFileCreator` = **Text file creator string**

`PostscriptFileCreator` = **Postscript file creator string**

These two control parameters are specific for the Macintosh operating system. Users of Microsoft Windows and other versions of DynaFit may safely

ignore them. The creator strings determine which application program (WordPerfect, Microsoft Word, BBEdit) will be associated with the output files created by DynaFit.

creator string	application
`R*ch`	BBEdit
`ttxt`	SimpleText
`WPC2`	WordPerfect
`MSWD`	Microsoft Word
`MPS`	MPW
`ALFA`	Alpha
`gsVR`	GhostScript

Table 11.2: File creator strings for the Apple Macintosh operating system.

`WriteStoich` = Write stoichiometric and formula matrices?

The computation of multiple simultaneous equilibria in DynaFit is accomplished by using so-called formula matrices and stoichiometric matrices [20, 21]. These intermediate results in the equilibrium computations will be printed in the output files if the `WriteStoich` flat is set to `yes`.

Printing out the matrices is useful for verification of the reaction mechanism. In particular, the user should check whether the kinetic compiler in DynaFit correctly identified *component* and *complex* species.

`VarianceInflation` = Print variance inflation factors?

If this flag is set to `yes`, the program will print the *variance inflation factors* [22] for all adjustable parameters in the least-squares regression.

`Covariance-Correl` = Print variance-covariance matrix?

If this flag is set to `yes`, the program will print the *variance-covariance matrix* for all adjustable parameters in the least-squares regression.

`CollinearityIndex` = **Print collinearity index?**

If this flag is set to `yes`, the program will print the *collinearity indices* [22] for all adjustable parameters in the least-squares regression.

`Eigenvectors` = **Print eigenvalues and eigenvectors?**

If this flag is set to `yes`, the program will print the results of spectral decomposition (eigenvalues and eigenvectors) [22] of the Fisher information matrix.

`RedundancyGrade` = **Print redundancy grade?**

If this flag is set to `yes`, the program will print the *collinearity indices* [22] for all adjustable parameters in the least-squares regression.

`Variance-Signal` = **Plot best-fit signal value vs. squared deviations?**

This plot (produced when the flag `Variance-Signal` is set to `yes`) is useful for checking the assumption of constant variance. The squared residuals are plotted against the (sorted) values of dependent variables. This feature is useful when the number of data points in each data set is at least one hundred, such as in the analysis of progress curves.

`CumulativeDistrib` = **Plot empirical cumulative distribution of residuals?**

When the flag `CumulativeDistrib` is set to `yes`), the program will produce the *empirical cumulative distribution* (ECD) of residuals. For a detailed explanation of the empirical cumulative distributions see reference [23]. This feature is useful when the number of data points in each data set is at least one hundred, such as in the analysis of progress curves.

`NormalPlot` = **Produce cumulative normal plot?**

When this flag is set to `yes`, the program will produce cumulative normal plot of residuals as a measure of randomness. For a detailed explanation of cumulative normal plots see reference [23]. This feature is useful when the

number of data points in each data set is at least one hundred, such as in the analysis of progress curves.

`SerialCorrelation` = **Produce serial correlation plot of residuals?**

When this flag is set to `yes`, the program will produce the *serial correlation plot* of residuals [22] as a measure of randomness. This feature is useful only in the analysis of progress curves, provided that each data set contains a sufficient number of data points (larger than one hundred).

`AutoCorrelation` = **Produce autocorrelation plot of residuals?**

When this flag is set to `yes`, the program will produce the *autocorrelation plot* of residuals [22] as a measure of randomness. This feature is useful only in the analysis of progress curves, provided that each data set contains a sufficient number of data points (larger than one hundred).

`PowerSpectrum` = **Produce power spectrum using Fast Fourier Transform?**

When this flag is set to `yes`, the program will produce the *power spectrum plot* of residuals [13] as a measure of randomness. This feature is useful only in the analysis of progress curves, provided that each data set contains a sufficient number of data points (larger than one hundred).

`LocalResid` = **Combine residuals from multiple data sets?**

When this flag is set to `yes`, the program will perform graphical analysis of residuals (e.g., normal plot, serial Correlation plot, etc.) separately for each data set in a global super set of data files. Otherwise, all residuals from multiple data sets are combined into a single pooled super set before residual analysis.

`RunsOfSigns` = **Compute the runs-of-signs test for residuals?**

When this flag is set to `yes`, the program will compute the probability that the observed runs of identical signs in residuals could occur by random chance. The method is explained in references [24] and [22].

`Rayner-Best` = **Compute the Rayner-Best statistics for residuals?**

When this flag is set to `yes`, the program will produce the *Rayner-Best statistics* [25] as a measure of randomness (normal distribution) in the residuals. For a detailed explanation of the Rayner-Best statistics see reference [25]. This feature is useful only in the analysis of progress curves, provided that each data set contains a sufficient number of data points (larger than one hundred).

`Kolmogorov-Smirnov` = **Compute the Kolmogorov-Smirnov statistics for residuals?**

When this flag is set to `yes`, the program will produce the *Kolmogorov-Smirnov statistics* [22] as a measure of randomness (normal distribution) in the residuals. This feature is useful only in the analysis of progress curves, provided that each data set contains a sufficient number of data points (larger than one hundred).

`Durbin-Watson` = **Compute the Durbin-Watson statistics for residuals?**

When this flag is set to `yes`, the program will produce the *Durbin-Watson statistics* [22] as a measure of randomness (normal distribution) in the residuals. This feature is useful only in the analysis of progress curves, provided that each data set contains a sufficient number of data points (larger than one hundred).

`Tukey` = **Compute the Tukey statistics for residuals?**

When this flag is set to `yes`, the program will produce the *Tukey statistics* [22] as a measure of randomness (normal distribution) in the residuals. This feature is useful only in the analysis of progress curves, provided that each data set contains a sufficient number of data points (larger than one hundred).

`WriteFTest` = **Perform model discrimination analysis using Fisher's F-statistic?**

When this flag is set to `yes`, the program will produce a model discrimination report according to the method described by Mannervik [18], which

uses Fisher's F-statistic for nested models. Otherwise, only the Akaike Information Criterion (AIC) method is used for model discrimination [26].

11.2.9 <Plot>

This set of initialization parameters controls the appearance of graphs either on the screen or in the ASCII output files.

```
<Plot>

   IndependentVar    = time (sec)
   DependentVar      = signal
   HighResolution    = Yes
   WaitLocal         = No
   ClickGraphs       = No
   WaitTime          = 3          ; seconds
   XPixels           = 576        ; 640
   YPixels           = 432        ; 480
   ScreenColumns     = 72
   ScreenRows        = 22
   FileColumns       = 72
   FileRows          = 36
   ShowXResiduals    = Yes
   ShowYResiduals    = Yes
   ResidRange        = 8.0        ; standardized residuals
   TMinVelocity      = 0.001
```

IndependentVar = Label for independent variable

The text to the right of the equal sign in `IndependentVar = ...` will be displayed in the output graphs as the label on the horizontal axis (independent variable).

DependentVar = Label for dependent variable

The text to the right of the equal sign in `DependentVar = ...` will be displayed in the output graphs as the label on the vertical axis (dependent variable).

`HighResolution` = High-resolution graphic output?

When this switch is set to `no`, the program will produce on-screen graphics in a simple character mode, using ASCII characters to represent all elements of each graph. This might be useful as an emergency measure under the Windows-98 operating system. Unfortunately, the Windows version of DynaFit has been known to fail on some computers under Windows-98, depending on the manufacturer of the graphics card. Should your system fail to display graphics properly under Windows-98, set `HighResolution = no`.

`WaitLocal` = User input during local analysis of progress curves?

If `WaitLocal` is set to `no`, the "local" statistical analysis of a series of progress curves proceeds automatically without waiting for user input upon displaying each output graph. This feature is useful in automatic determination of initial reaction velocities for a large number of data files.

`ClickGraphs` = Wait for user input upon displaying graphics?

If `ClickGraphs` is set to `yes`, the program will wait for user input when output graphs are displayed on the screen, showing for example the results of fit. When DynaFit displays graphics, the program is waiting for the user to perform a certain action that depends on the type of computer being used. On the Macintosh computer, the operator should click with the mouse anywhere in the graph area to dismiss the graphics and return to the computations. On a DOS/Windows computer, the operator should press the Enter key on the keyboard.

`WaitTime` = Time for display of output graphs

If `ClickGraphs` is set to `no`, the program will display each output graph for the number of seconds defined by this parameter.

`XPixels` = Number of pixels in the horizontal direction

This parameter determines the horizontal size (dimension) of high-resolution graphs displayed on the screen.

`YPixels` = **Number of pixels in the vertical direction**

This parameter determines the vertical size (dimension) of high-resolution graphs displayed on the screen.

`ScreenColumns` = **X-Dimension of the ASCII plot (screen)**

This parameter sets the number of columns on the character output screen that are used for the construction of simple graphs. For example, `ScreenColumns = 72` means that each graph on a console output screen will have 72 columns.

`ScreenRows` = **Y-Dimension of the ASCII plot (screen)**

This parameter sets the number of rows on the character output screen that are used for the construction of simple graphs. For example, `FileRows = 36` means that each graph a console output screen will have 36 rows.

`FileColumns` = **X-Dimension of the ASCII plot (file)**

This parameter sets the number of columns in the ASCII files that are used for the construction of simple graphs. For example, `FileColumns = 72` means that each graph in the ASCII output files will have 72 columns.

`FileRows` = **Y-Dimension of the ASCII plot (file)**

This parameter sets the number of rows in the ASCII files that are used for the construction of simple graphs. For example, `FileRows = 36` means that each graph in the ASCII output files will have 36 rows.

`ShowXResiduals` = **Plot residuals against independent variable?**

If this flag is set to `yes`, DynaFit will plot the residuals of fit against the independent variable.

`ShowYResiduals` = **Plot residuals against dependent variable?**

If this flag is set to `yes`, DynaFit will plot residuals not only against the independent variable (*e.g.*, time in the case of reaction progress curves) but also against the dependent variable *e.g.*, absorbance). These plots are often

more informative than the more conventional plots of residuals against the independent variable.

`ResidRange` = Range of Studentized residuals

If `ResidRange` is set to zero, the program will plot Studentized residuals in a scale that is automatically determined from the data. The resulting plot is constructed with the vertical axis range that accommodates all residuals in the given data set. On the other hand, a nonzero value of `ResidRange` causes the plotting of Studentized residuals within plus or minus the range defined by this parameter.

`TMinVelocity` = Minimum time for plotting velocities

DynaFit always plots not only the simulated reaction progress curves but also their first derivatives with respect to time. Often these derivative plots contain a very sharp spike at the beginning, which causes the resulting graph not to be very informative.

The value of `TMinVelocity` orders the program to begin plotting the derivative curve at reaction time equal to this parameter. Simulated or fitted output points that were computed at reaction times lower than `TMinVelocity` are still plotted in the progress curve, but are ignored in the first derivative plot.

11.2.10 `<Velocity>`

This section of the DynaFit initialization file defines control parameters that are used in the computation or recording of initial reaction velocities.

```
<Velocity>

   AutoWrite         = No
   WriteFittedConc   = No
   AverageInput      = Yes
   AlwaysPositive    = Yes
   ExcludeOutliers   = Yes
   MinimumPoints     = 4
   StandardDeviation = 4.
```

`AutoWrite` = Flag for recording initial velocities on disk

If this flag is set to `yes` during the least-squares fit of reaction progress curves, the program will compose a special two-column, tab delimited output file in the ASCII format. The first column will contain the (initial) concentration of the reactant that is named after the `variable` keyword, in the `[velocity]` section of the DynaFit script. The second column will contain the initial reaction velocity in instrument units (*e.g.*, absorbance or fluorescence per second.

> *Example 1* In this example, DynaFit first performs the least-squares fit of seven progress curve files, named 1.txt, 2.txt, etc., located in the directory ./examples/pepsin/data/. The initial velocities are then written into a newly created file named ./examples/pepsin/data/veloc.txt. This file will contain in the first column the concentration of the inhibitor I (`variable I`), and the initial velocity in the second column.

```
;----------------------------
; Determine initial velocities.
;----------------------------

[task]
    data  = progress
    task  = fit

[mechanism]
    E + S <===> ES    :  k    ks
    ES ---> E + P     :  kcat
    E + I <===> EI    :  k    ki

[constants]
    k = 100, ks = 4000 ?, kcat = 15 ?
    ki = 10 ?

[responses]
    P = -0.0015

[concentrations]
    E = 0.04
```

```
    S = 100 ?

[progress]
    local
    directory      ./examples/pepsin/data
    extension      txt
    offset         auto ?
    error          constant 0.00025
    delay          5

    files          1,2,3,6,7,8,9
    vary conc. I = 0,1,2,3,4,5,6

[velocity]
   variable    I
   file        ./examples/pepsin/data/veloc.txt

[output]
   directory   ./examples/pepsin/output

[settings]
   <Velocity>
      AutoWrite  = yes  ; <== CREATE DATA FILE !

;----------------------------------------
; Fit initial velocities determined above.
;----------------------------------------

[task]
    data  = velocities
    task  = fit

[mechanism]
    E + S <===> ES    :  Ks    dissoc.
    ES ---> E + P     :  kcat
    E + I <===> EI    :  Ki    dissoc.

[constants]
    Ks = 37.5, kcat = 15 ?
    Ki = 0.1 ?
```

```
[responses]
    P = -0.0015

[concentrations]
    E = 0.04
    S = 100

[progress]
    rapid equilibrium

[velocity]
   variable I
   file     ./examples/pepsin/data/veloc.txt

[end]
```

`WriteFittedConc` = Flag for writing optimized concentrations

If this flag is set to `yes`, DynaFit will create the initial velocity data file (see the description of `AutoWrite`) by storing in the newly created data file not the nominal values of concentrations, but the best fit values of concentrations.

Thus, if in Example 1 the inhibitor concentration was considered as an adjustable parameter instead as a constant, DynaFit would write into the first column of file ./examples/pepsin/data/veloc.txt not the nominal concentrations of inhibitor I (0, 1, 2, ..., 6) but instead their best-fit optimized values.

`AverageInput` = Flag for averaging of replicates

If this flag is set to `yes`, the program will examine the input data file for the presence of replicated data points, where the initial velocity was measured at identical concentrations of all reactants. If such replicated data points are found, DynaFit will compute the average and standard deviation for each replicated group. The raw input data are ignored in the subsequent least-squares regression, which instead uses the computed averages. Optionally, the program will also utilize the computed values of standard deviations for weighting in the least-squares regression.

`AlwaysPositive` = Flag for positive velocities

Often the analysis of reaction progress curves involves data sets where the experimental signal, such as absorbance or fluorescence, *declines* over time. In this case the initial reaction velocity, defined as the change in signal per unit of time has *negative* values. However, it is conventional to report the initial reaction velocities (*e.g.*, "mOD/min" or 0.001 × absorbance units per minute) as positive values.

If the flag `AlwaysPositive` is set to `yes`, the program will record the initial reaction velocity (instrument units / time) always as a positive value, in accordance with the convention.

`ExcludeOutliers` = Flag for the exclusion of outliers

If this flag is set to `yes`, the program will attempt to exclude outlying initial velocity data points by using the *jackknife technique* described below. The data exclusion protocol uses the values of `MinimumPoints` and `StandardDeviation`.

`MinimumPoints` = Smallest number of replicates

The automatic outlier exclusion protocol is applied only to those measurements that have been replicated more than `MinimumPoints` times.

`StandardDeviation` = Criterion for the exclusion of outliers

If `ExcludeOutliers` is set to `yes`, the value defined by the `StandardDeviation` is used to decide on the exclusion of outlying data points (initial velocities) using For each group of replicates, the program first computes the average and standard deviation. Subsequently the average and standard deviation is re-computed while excluding the replicated data points one by one. The smallest standard deviation and the largest standard deviation are recorded. If the largest standard deviation exceeds the smallest standard deviation more than `StandardDeviation` times, the corresponding *influential point* is excluded from the analysis.

Bibliography

[1] Kuzmič, P. (1996) Program DYNAFIT for the analysis of enzyme kinetic data: Application to HIV proteinase. *Anal. Biochem.* **237**, 260–273.

[2] Peranteau, A. G., Kuzmič, P., Angell, Y., García-Echeverría, C., and Rich, D. H. (1995) Increase in fluorescence upon the hydrolysis of tyrosine peptides - application to proteinase assays. *Anal. Biochem.* **227**, 242–245.

[3] Beechem, J. M. (1992) Global analysis of biochemical and biophysical data. *Meth. Enzymol.* **210**, 37–54.

[4] Wedemeyer, W. J., Ashton, R. W., and Scheraga, H. A. (1997) Kinetics of competitive binding with application to thrombin complexes. *Anal. Biochem.* **248**, 130–140.

[5] Kuzmič, P. (1999) General numerical treatment of competitive binding kinetics: Application to thrombin-dehydrothrombin-hirudin. *Anal. Biochem.* **267**, 17–23.

[6] Segel, I. H. (1975) *Enzyme Kinetics*. Wiley, New York.

[7] Cornish-Bowden, A. (1979) *Fundamentals of Enzyme Kinetics*. Butterworths, London.

[8] Moss, M. L., Kuzmič, P., Stuart, J. D., Tian, G., Peranteau, A. G., Frye, S. V., Kadwell, S. H., Kost, T. A., Overton, L. K., and Patel, I. R. (1996) Inhibition of human steroid 5α reductases type i and ii by 6-aza-steroids. structural determinants of one-step vs. two-step mechanism. *Biochemistry* **35**, 3457–64.

[9] Hindmarsh, A. C. (1983) ODEPACK: a systematized collection of ODE solvers. In *Scientific Computing,* Stepleman, R. S. et al., editors, 55–64. North Holland, Amsterdam.

[10] I, T.-P. and Nancollas, G. H. (1972) EQUIL – a general computational method for the calculation of solution equilibria. *Anal. Chem.* **44**, 1940–1950.

[11] Seber, G. A. F. and Wild, C. J. (1989) *Nonlinear Regression.* Wiley, New York.

[12] Marquardt, D. W. (1963) An algorithm for least-squares estimation of nonlinear parameters. *J. Soc. Ind. Appl. Math.* **11**, 431–441.

[13] Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (1992) *Numerical Recipes in C.* Cambridge University Press, Cambridge.

[14] Duggleby, R. G. (1984) Regression analysis of nonlinear Arrhenius plots: an empirical model and a computer program. *Comput. Biol. Med.* **14**, 447–455.

[15] Johnson, M. L. (1983) Evaluation and propagation of confidence intervals in nonlinear, asymmetrical variance spaces. analysis of ligand-binding data. *Biophys. J.* **44**, 101–106.

[16] Watts, D. G. (1994) Parameter estimation from nonlinear models. *Meth. Enzymol.* **240**, 23–36.

[17] Bates, D. M. and Watts, D. G. (1988) *Nonlinear Regression Analysis and its Applications.* John Wiley & Sons, New York.

[18] Mannervik, B. (1982) Regression analysis, experimental error, and statistical criteria in the design and analysis of experiments for discrimination between rival kinetic models. *Meth. Enzymol.* **87**, 370–390.

[19] Byrne, G. D. and Hindmarsh, A. C. (1987) Stiff ODE solvers: a review of current and coming attractions. *J. Comput. Physics* **70**, 1–62.

[20] Smith, W. R. and Missen, R. W. (1982) *Chemical Reaction Equilibrium Analysis.* John Wiley, New York.

[21] Royer, C. A., Smith, W. R., and Beechem, J. M. (1991) Analysis of binding in macromolecular complexes: a generalized numerical approach. *Anal. Biochem.* **191**, 287–294.

[22] Rawlings, J. O. (1988) *Applied Regression Analysis – A Research Tool*. Wadsworth, Belmont.

[23] D'Agostino, R. B. (1986) Graphical analysis. In *Goodness-of-Fit Techniques,* D'Agostino, R. B. and Stephens, M. A., editors, 7–62. Marcel Dekker, New York.

[24] Reich, J. G. (1992) *Curve Fitting and Modelling for Scientists and Engineers*. McGraw-Hill, New York.

[25] Rayner, J. C. W. and Best, D. J. (1989) *Smooth Tests of Goodness of Fit*. Oxford University Press, New York.

[26] Burnham, K. B. and Anderson, D. R. (2002) *Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach*. Springer-Verlag, New York, 2nd edition.

List of Figures

List of Tables

Index