Petr Kuzmič

DynaFit Scripting

Software Version 4.08

Manual Version 4.08.005

May 11, 2018

BioKin Ltd
www.biokin.com

Preface

DynaFit documentation is divided into several book-length documents. This partitioning was chosen in order to keep each part of the overall documentation reasonably short.

- **Getting Started with DynaFit**
  The *Getting Started with DynaFit* manual describes the user interface, including the menu system and the appropriate formatting of input experimental data. Only a cursory attention is paid to the syntax and semantics of DynaFit scripting. One particular chapter is set aside for step-by-step tutorial solving a real-life research problem involving simultaneous biophysical equilibria.

- **DynaFit Scripting**
  This *DynaFit Scripting* manual contains everything the DynaFit user might need to understand about the DynaFit scripting language under a great majority of experimental circumstances. This release of the scripting manual purposely does not cover the use of certain highly specialized, built-in fitting models, which will be handled in a separate supplemental document.

- **DynaFit Theory and Internals**
  The *Theory and Internals* manual, currently in preparation, will explain the scientific basis and technical aspects of various computational algorithms employed by DynaFit. The book will also provide a detailed explanation of the various types of output produced by DynaFit, focusing in particular on the theory of nonlinear regression analysis.

In addition to book-length volumes, the present book representing the second of three installments, additional DynaFit documentation will consist of multiple shorter manuscripts focusing on various theoretical and practical issues arising in the analysis of bio/chemical equilibria and kinetics.

All parts of the DynaFit documentations are meant to be "living documents", frequently expanded and (presumably) improved by revisions based on user feedback and on new information published in the research literature. For this reason DynaFit users are encouraged to periodically visit the BioKin website http://www.biokin.com for updates.

Acknowledgments. My most sincere thanks go to Dr. Jan Antosiewicz, Department of Biophysics, Institute of Experimental Physics, Warsaw University, Poland, for making many helpful suggestions that significantly improved the quality of this manuscript.

Petr Kuzmič
Watertown, Massachusetts
May 2014, revised March 2018

Copyright Notice

Disclaimer

Contents

Chapter 1
Introduction

DynaFit is driven by plain text files called *scripts*. Each script contains the following information:

- **Task.** We can either simulate artificial data, or we can fit existing experimental data. The script will tell DynaFit what we need to accomplish.
- **Data.** In a particular section of the script, DynaFit is told where in the system (i.e., in which file) to look for experimental data.
- **Model.** Here we tell DynaFit what model or mechanism we have in mind, which presumably underlies the experimental (or simulated) data.
- **Parameters.** DynaFit needs to know the values of model parameters, such as rate constants and starting concentrations of reactants.
- **Output.** When DynaFit runs, it creates a mini "web-site" composed of HTML files, GIF image files, and plain text (ASCII) files. We must tell DynaFit where to put these.
- **Settings.** DynaFit can be optionally fine-tuned for each particular problem, using a large number of control settings located in a special section of the script.

This chapter describes only a few general characteristics of the script files. In later chapters, we will elaborate the necessary details.

1.1 Anatomy of a DynaFit script

The building blocks of every DynaFit are as follows:

Sections and keywords. The text enclosed in square brackets such as [task] or [mechanism] are the *sections* of the DynaFit script. The special reserved words such as progress or fit are DynaFit *keywords*.

Special characters. Each DynaFit script contains certain special characters. For example the "greater-than" character (optionally preceded by any number of dashes) represents the left-to-right reaction arrow, `-->`.

Optional numerical data. The experimental data to be analyzed can be embedded directly in the DynaFit script, or they can be located in separate external files.

Optional comments. A script file *comment* is text that is present for the benefit of the researcher, but it is otherwise ignored by DynaFit. DynaFit scripts can contain three kinds of comments: (a) any text preceding the first [task] section; (b) any text set off by the semicolon through the end of the given line; and (c) any text in the [data] section that does not represent valid numerical values.

Every DynaFit script is compiled out of the "anatomical parts" listed above. The next section of this chapter provides a worked example.

1.2 Introductory example

DynaFit can be used to process four different types of experimental data, as shown in the following table:

Data type	*Independent variable*
Reaction progress	Time
Equilibrium binding	Total (analytic) concentrations
Initial rate enzyme kinetics	Initial concentrations
Arbitrary data	Arbitrary variable (algebraic model)

Regardless of the type of experimental data, the *dependent variable* is almost always some type of physical signal being observed in the experiment, such as absorbance, fluorescence intensity, HPLC peak area, counts-per-second in radiometry, chemical shift, etc.

To get acquainted with DynaFit scripting, this section presents a simple example using reaction progress curves. The example is taken from ref. [2]. Recombinant human 5α-ketosteroid reductase (enzyme concentration 50 pM) was incubated with radioactive testosterone (substrate concentration 31 nM). Samples were withdrawn at different reaction times and analyzed by HPLC. The percentage conversion of substrate to product (dihydrotestosterone) was determined by HPLC with radiometric detection. The results are shown in *Table 1.1*.

It is assumed that the conversion of testosterone to dihydrotestosterone follows the Michaelis-Menten reaction mechanism shown in *Scheme 1.1*. However, we have recently shown that under the usual steady-state conditions, it is not possible to

time, min	Product, %
2	6.4
4	12.4
6	19.5
8	25.5
10	30.7
12	38.0
14	42.6
16	47.4
18	54.4
20	57.4
22	63.4
24	65.7
26	70.6
28	73.7
30	75.9
34	80.9
38	86.0
42	89.2
46	92.4
50	93.9
54	94.9
60	96.5

Table 1.1: Conversion of testosterone (initial concentration 31 nM) to dihydrotestosterone over time, catalyzed by 5α-ketosteroid reductase (50 pM) [2].

extract all three rate constants appearing in *Scheme 1.1*. Instead, we must formulate the theoretical model as the Van Slyke – Cullen mechanism shown in *Scheme 1.2* [1].

$$\mathrm{E + S} \underset{k_2}{\overset{k_1}{\rightleftharpoons}} \mathrm{E.S} \xrightarrow{k_3} \mathrm{E + P}$$

Scheme 1.1

$$\mathrm{E + S} \xrightarrow{k_1^*} \mathrm{E.S} \xrightarrow{k_3} \mathrm{E + P}$$

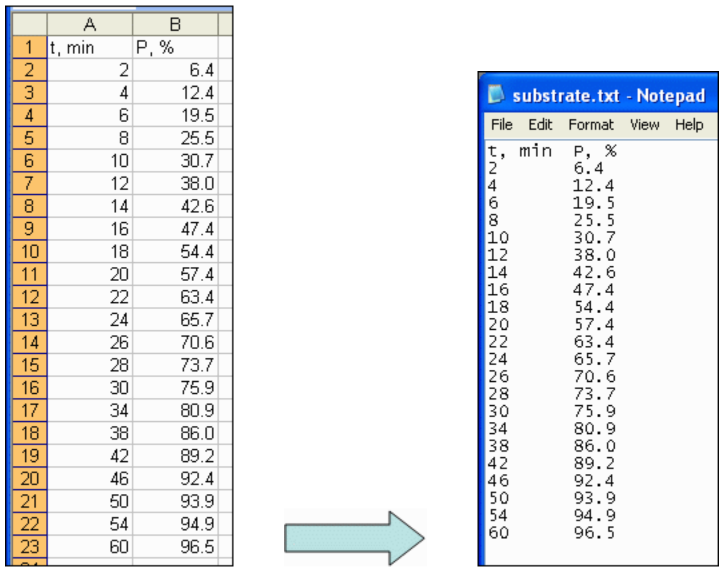
Scheme 1.2

Our task is to determine the rate constants k_1^* and k_3 appearing in *Scheme 1.2* from the experimental data listed in *Table 1.1*. Let us explain, step by step, how this task is accomplished with DynaFit.

Preparation of the input data file

We begin by creating a plain (ASCII) text file to hold the experimental data. Let us assume that the data is originally stored in an Excel spread sheet file. The actual Excel data file for this demonstration, named substrate.xls, is located in the distribution directory ./manual/intro/data.

DynaFit cannot read Excel files, only plain text (ASCII) files. Therefore, we must first copy the numerical data from Excel to a suitable plain-text editor; or save the spread sheet as plain text (Excel menu File ... Save As ... Save As Type ... Text (Tab Delimited)). Either way, we want to end up with a plain text file as shown in *Figure 1.1*.



	A	B
1	t, min	P, %
2	2	6.4
3	4	12.4
4	6	19.5
5	8	25.5
6	10	30.7
7	12	38.0
8	14	42.6
9	16	47.4
10	18	54.4
11	20	57.4
12	22	63.4
13	24	65.7
14	26	70.6
15	28	73.7
16	30	75.9
17	34	80.9
18	38	86.0
19	42	89.2
20	46	92.4
21	50	93.9
22	54	94.9
23	60	96.5

```
t, min  P, %
2       6.4
4       12.4
6       19.5
8       25.5
10      30.7
12      38.0
14      42.6
16      47.4
18      54.4
20      57.4
22      63.4
24      65.7
26      70.6
28      73.7
30      75.9
34      80.9
38      86.0
42      89.2
46      92.4
50      93.9
54      94.9
60      96.5
```

Fig. 1.1: Experimental data: Copying from a spreadsheet program to a plain-text editor.

The resulting ASCII (plain text) data file is named substrate.txt and is again located in the directory ./manual/intro/data.

Preparation of the script file

DynaFit is distributed with 50+ representative example problems. It is very likely that each new user will be able to modify at least one of the distributed example scripts for a particular purpose. However, for the sake of this introductory demonstration let us assume that we set out to prepare a DynaFit script completely from scratch.

Each script must contain a section called `[task]`, so we begin by typing (or inserting) the following text:

```
[task]
   data = progress
   task = fit
```

Next we must tell DynaFit where to find our previously prepared experimental data file. This is accomplished by using the section `[data]`:

```
[data]
   directory  ./manual/intro/data
   file       substrate.txt
```

Note that the leading period in the directory name means "current working directory", that is, the particular directory where the DynaFit program itself is located.

Now let us assume that the HPLC peak areas recorded in our radiometric detector are all systematically "off" by a certain baseline (or offset) value. In other words, what if all substrate conversion values were shifted by one or two percent? This is quite common in quantitative HPLC analysis. to account for this possibility, we now insert the keyword `offset` and give it the value `auto ?`, which means two things. First, the keyword `auto` instructs DynaFit to take as the initial estimate of the baseline offset the first data point. Second, the question mark (`?`) tells DynaFit to treat the baseline offset as one of the adjustable model parameters.

```
[data]
   ...
   offset     auto ?
```

We do have a certain idea about the underlying reaction mechanism for this enzyme reaction, as shown in *Scheme 1.2*. To express this theoretical model in DynaFit notation, we will insert the `[mechanism]` section is shown immediately below. The relationship between this text and *Scheme 1.2* is obvious; the only thing that changed is that we place the names of rate constants after the colon (`:`) separator, instead above or below each reaction arrow:

```
[mechanism]
   E + S ---> ES     :     k1*
   ES ---> E + P     :     k3
```

To fit the available experimental data, DynaFit will perform nonlinear least-squares regression, and therefore we need *initial estimates* of adjustable model parameters. This is frequently the most difficult aspect of using DynaFit, or, for that matter, any other nonlinear regression software package. If we have absolute no idea which starting values to assign to rate constants, we can always start with *unit value* for all of them, and adjust the crude initial estimate interactively if necessary:

```
[constants]           ; units: nM, min
   k1* = 1 ?
   k3  = 100 ?
```

The text set off by the semicolon is a comment (ignored by DynaFit) which reminds us that the concentration units through the script are nM for concentrations and minutes for time. Thus, typing `k1* = 1` means $k_1^* = 1\ \text{nM}^{-1}.\text{min}^{-1}$, or $k_1^* = 1/60 \times 10^9 = 1.67 \times 10^7\ \text{M}^{-1}\text{s}^{-1}$in more conventional units. The units of concentration and time are completely arbitrary. It is however required that they remain consistent throughout the entire script.

Next we must specify the experimental conditions, in this case the initial concentrations of reactants:

```
[concentrations]       ; units: nM
   E  = 0.05
   S  = 31
```

A very important part of the script is the section `[responses]`, which creates a necessary link between the measured experimental signal (absorbance, fluorescence intensity, NMR chemical shift, chromatographic peak area, and the like) and the changing concentrations of directly observable reactants.

In this particular example, one of the reactants is the enzyme *E* but we cannot directly observe it in the experiment; nor can we observe the concentration of the enzyme–substrate complex *ES*. Really the only reactant we are able to monitor in our experimental setup is final reaction product *P*. Therefore, the response section will contain only one species name:

```
[responses]
   P = ...
```

The molar *response coefficient* is defined as follows. It is a number that relates *one concentration unit* (whichever concentration unit was chosen in the given problem) to *one unit of the observed experimental signal*. In this particular example, we have chosen nanomolar concentration units; the experimental signal is percentage of substrate conversion to product. Therefore, the question is how many percentage points of substrate conversion correspond to one nanomole per liter of the reaction product? This is the value of the specific molar response coefficient of product *P*.

To arrive at the correct answer, consider that if 31 nM of substrate *S* were fully converted to the product *P*, this would correspond to 100% conversion. Therefore, 1 nM of product formed corresponds to 100%/31 = 3.22%. Therefore, we complete the `[response]` section as follows:

```
[responses]            ; units: % product/nM substrate
   P = 3.22 ?
```

Note that the response coefficient is considered as an optimized parameter in the fitting model, as indicated by the question mark. We do this because the slope of the calibration curve, to compute percentage substrate conversion from radiometric data, is always subject to finite uncertainty (however small). In other words, what if "100% conversion" by the calibration curve in reality was only 96% conversion? Placing a question mark after the molar response coefficient will allow for this possibility.

Finally, we must tell DynaFit where to store the generated output files, which is done in the `[output]` section of the script file:

```
[output]
   directory  ./manual/intro/output/01
```

Listing 1.1 shows the complete script file to analyze the experimental data shown in Table 1.1. The actual script is located in the distribution directory ./manual/intro.

EXAMPLE SCRIPT

Listing 1.1

```
Script ./manual/intro/01.txt

Fit substrate kinetic data to the Van Slyke-Cullen model.
Data: Moss, M., et al. (1996) Biochemistry 35, 3457-3464
Model: Kuzmic, P. (2009) Anal. Biochem. 394, 287-289
;________________________________________________________
[task]
   data = progress
   task = fit
[mechanism]
   E + S ---> ES     :     k1*
   ES ---> E + P     :     k3
[constants]             ; units: nM, min
   k1* = 1 ?
   k3  = 100 ?
[concentrations]        ; units: nM
   E  = 0.05
   S  = 31
[responses]             ; units: % product/nM substrate
   P = 3.22 ?
[data]
   directory  ./manual/intro/data
   file       substrate.txt
   offset     auto ?
[output]
   directory  ./manual/intro/output/01
[end]
```

This completes the construction of a DynaFit script file that can be used to determine the microscopic rate constants k_1^* and k_3 appearing in *Scheme 1.2*.

1.3 Sections

DynaFit script files are divided into *sections*, with section names being enclosed in square brackets. A brief summary of the purpose of each section, arranged alphabetically, is shown in Table 1.2.

Section	*Usage*
`[concentrations]`	Nonzero values of concentrations applicable to all data sets
`[constants]`	Values of rate constants or equilibrium constants
`[correlations]`	List of model parameters for which to plot correlation diagrams
`[data]`	Location of experimental or simulated data
`[end]`	End of script file
`[mechanism]`	The reaction mechanism
`[model]`	Arbitrary algebraic model
`[output]`	Location of output files to be written to disk
`[parameters]`	Arbitrary model parameters
`[responses]`	Nonzero values of molar responses applicable to all data sets
`[set:...]`	Experimental data proper
`[settings]`	Optional control parameters
`[task]`	The main task to be performed

Table 1.2: Section names recognized in DynaFit scripts

It is not necessary to memorize the section names. The menu **Edit ... Insert ... Section** can be used to reveal all recognized section names and insert the desired choice into the script. This is illustrated in Figure 1.2.

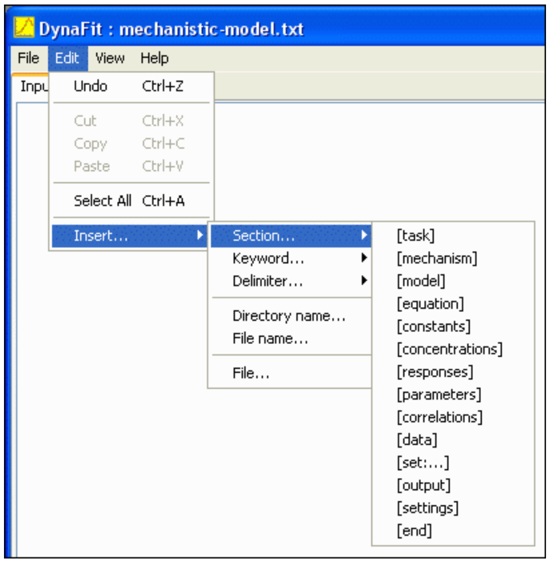


Fig. 1.2 Inserting section names into the script file. Select Edit ... Insert from the main menu to insert an appropriate section name into the text currently displayed in the editable window.

It is not necessary to remember section names and keywords, or even refer to this manual, when constructing DynaFit scripts. The built-in text editor "knows" all valid keywords and sections and will insert these items into the editable area after making an appropriate menu selection.

Most DynaFit script will contain the following sections:

```
[task]
   task = ...
   data = ...
...
[mechanism]
...
[constants]
...
[concentrations]
...
[responses]
...
[data]
...
[output]
   directory ...
[end]
```

The above skeleton script template is available as file ./templates/mechanistic-model.txt. This list of sections applies to DynaFit scripts that are used to automatically derive the mathematical model based on the symbolic notation contained in the `[mechanism]` section.

DynaFit can also be used to fit experimental data to an arbitrary algebraic model, such as for example the Michaelis-Menten equation $v = V_{\max}[S]/(K_M + [S])$, or any other algebraic equation. In that case the DynaFit script will contain the following sections:

```
[task]
   task = ...
   data = ...
...
[parameters]
...
[model]
...
[data]
...
[output]
   directory ...
[end]
```

The above skeleton script template is available as file ./templates/algebraic-model.txt. Under certain circumstances a DynaFit script may also contain the following optional sections:

```
[correlations]
...
[settings]
...
[set:...]
```

Each of the sections enumerated above is addressed in a separate chapter of this book.

1.4 Keywords

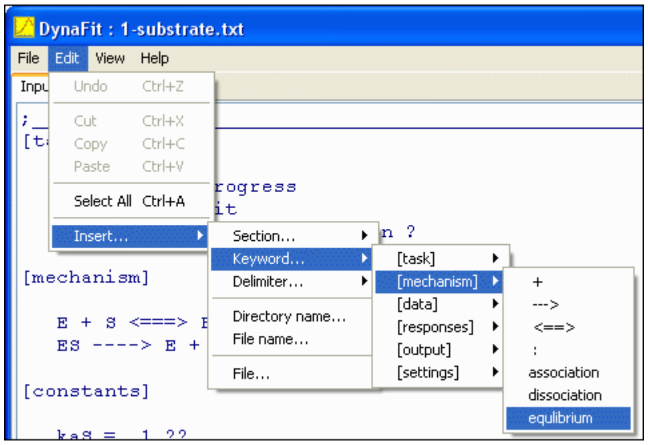
Each particular section of the script can contain various *keywords* recognized by the miniature DynaFit scripting "language". The complete list of keywords, arranged alphabetically, in shown in Table 1.3.

algorithm	extension	plot
approximation	file	poisson
association	fit	power
auto	from	progress
column	generic	proportional
concentration	graph	quadratic
confidence	incubate	rapid-equilibrium
constant	king-altman	rates
data	levenberg-marquardt	response
delay	linear	search
design	logarithmic	set
differential-evolution	mesh	sheet
dilute	model	simulate
directory	mole-fraction	step
dissociation	monte-carlo	task
equilibrate	none	time
equilibria	offset	titration
equilibrium	parameter	to
error	percent	variable
exponential	piecewise-linear	

Table 1.3: Alphabetical list of keywords that can appear in DynaFit script files.

It is not necessary to memorize the approximately 60 keywords recognized by DynaFit. The menu **Edit ... Insert ... Keyword** can be used to reveal all recognized keywords and insert the desired choice into the script. Within the **Edit** menu the keywords are organized by sections, in which they can legitimately appear.

For example, the section [mechanism] can contain only one of three keywords: dissociation, association, or equilibrium. This is illustrated in Figure 1.3.



```
[mechanism]

   E + S <===> E
   ES ----> E +

[constants]
```

Fig. 1.3: Inserting recognized keywords into the Input editable area.

The list of keywords shown in Table 1.3 may appear daunting to first-time users of DynaFit. However, it is important to remember that each particular research project will call for only a small subset of DynaFit keywords to be used in appropriate script files. It is also useful to remember that, especially in the initial phases of each research project utilizing DynaFit, assistance can be obtained by contacting the author and maintainer of DynaFit, Dr. Petr Kuzmič, through the BioKin.com website.

The details of how to appropriately utilize each DynaFit keyword are explained in subsequent chapters of this book.

1.5 Formatting

This section summarizes certain general principles that apply to the formatting of all DynaFit scripts.

Case sensitivity

DynaFit scripts are case-sensitive. For example the section name `[task]` is correctly recognized, whereas `[Task]` (using upper case "T") would not be recognized by the DynaFit scripting engine.

White space

For greater readability, DynaFit scripts can contain any number of consecutive blank lines. For the same purpose, the beginning of each line can be set off from the start of the line by any number of blank characters. Blank characters can also be used to vertically line up certain items, such as rate constant names or molecular species names.

For example, let us compare *Listing 1.2* with the semantically equivalent *Listing 1.3*. Although matters of style can be highly personal, most people would probably agree that the *Listing 1.3* is much easier to understand.

Listing 1.2

```
[task]
data=progress
task=fit
[mechanism]
E + S <=> ES : k1 k-1
ES -> E + P : k2
[constants]
k1=1, k-1=10?, k2=1?
[concentrations]
S=31, E=0.05
[responses]
P=3.21?
[data]
file ./test/data.txt
offset auto?
[end]
```

Listing 1.3

```
[task]
   data = progress
   task = fit

[mechanism]
   E + S <===> ES     :    k(on)      k(off)
   ES ----> E + P     :    k(cat)
```

```
[constants]
   k(on)   =  1
   k(off)  = 10 ?
   k(cat)  =  1 ?

[concentrations]
   S =   31
   E = 0.05

[responses]
   P = 3.21 ?

[data]
   file    ./test/data.txt
   offset  auto ?

[end]
```

A judicious use of white-space is highly recommended when compiling DynaFit scripts. For example, for greater readability, you may place space characters around the equal sign (=) in numerical assignments, as well before each question mark (`?`) representing an optimized model parameter.

Comments

DynaFit scripts can optionally contain two kinds of *comments*, which are pieces of text that are ignored by the DynaFit scripting engine and serve only to benefit the human reader.

Comments before the first `[task]` or after `[end]`

Either before the first `[task]` section or after the `[end]` marker, a DynaFit script can optionally contain free-format text that is ignored by the scripting engine. These comments can often be crucially important for remembering the purpose of a particular script and what was learned by running it. An typical example is given below:

```
This is an attempt to determine the molecular mechanism
of ``slow, tight'' inhibition of HIV protease(lot SX-0015)
by the compound JG-365, using experimental data collected
on March 10, 2013. RESULTS: See notes at end.

[task]
...
[end]
```

```
It looks like this could be a two-step reversible binding,
with initial binding step under rapid equilibrium, but the
Akaike Information Criterion for the next best model
(irreversible!) is very close.  CONCLUSIONS: We need more
data, using more optimally chosen inhibitor concentrations.
```

It is highly recommended to annotate each successive DynaFit script, generated in a data-analytic session where multiple script drafts or versions are involved, by explanatory comments similar to the example above. In this fashion, a collection of scripts can serve as an important part of an electronic laboratory notebook.

Embedded comments

Within the script proper (that is, after the first `[task]` line and before the `[end]` marker) we can embed shorter comments set off by the semicolon character (`;`). Any text starting from the semicolon to the end of the given line will be ignored by the DynaFit scripting engine. This is useful to introduce reminder about concentration units, or mark up other salient feature of the script. An example is given below:

```
[concentrations]      ;   units: micromolar
   Substrate = 10 ?
   Enzyme    = 0.01   ;   fixed parameter !

[data]
   ; Stopped-flow machine in second floor lab (3/10/2013).
   ; The baseline seemed to have been drifting all day!

   directory ./stopped-flow/SFX-100/130310
   file      d01.txt
```

This notation can be used conveniently to "delete" reaction steps from the mechanism we had previously contemplated. In this way we can maintain a record of progressively developing the fitting model, starting from the initial draft and proceeding through multiple refinement steps:

```
Script 'fit-002', a modification of 'fit-001' but with the
isomerization step deleted:
...
[mechanism]
   E + I <==> E.I    :    k(on)       k(off)
 ; E.I <===> (E.I)*  :    k(forward)  k(reverse)   deleted !

[constants]
```

```
   k(on)       = 1 ?, k(off)      = 10 ?
 ; k(forward) = 1 ?, k(reverse) = 0.1 ?            deleted !
```

Abbreviations

Certain DynaFit keywords can be abbreviated to keep the script file shorter and in some cases more readable. Examples are shown in Table 1.4.

Keyword	*Abbreviation*
`association`	`assoc`
`dissociation`	`dissoc`
`equilibrium`	`equil`
`concentration`	`conc`
`response`	`resp`

Table 1.4: Abbreviations that can be used in DynaFit scripts.

A full explanation is offered in subsequent chapter of this book, addressing those particular sections of the script where the particular keywords can legitimately appear.

Line breaks

The DynaFit scripting engine requires that certain pieces of text are placed on separate lines. For example, all section names (such as `[task]` must all appear on their own separate lines. However, occasionally there is a need to improve the readability of the script by stringing together multiple "logical lines" in a single "physical line". This can be accomplished by inserting the vertical bar (`|`) character wherever a line break must appear in the script.

For example, in the section `[data]`, the keyword `concentration` must appear on a separate line following each particular `file` name. The same applies to the keyword `offset`. However, to make the script more readable we can introduce these required "logical" line breaks by a notation similar to the following fragment, in which the vertical bars represent required line breaks:

```
[data]
   directory  ./test/data
   extension  txt
   file       f01 | concentration S = 0.1 | offset auto ?
   file       f02 | concentration S = 0.2 | offset auto ?
   file       f04 | concentration S = 0.4 | offset auto ?
   file       f08 | concentration S = 0.8 | offset auto ?
   file       f16 | concentration S = 1.6 | offset auto ?
```

Comma-separated lists

Comma-separated lists of assignments can appear in DynaFit scripts in one of three situations described below.

Constants, concentrations, and molar responses

The individual rate or equilibrium constants appearing in the `[constants]` section can be listed either on separate lines, or in a list of comma-separated assignments. For example, the following two script fragments (*Listing 1.4* and *Listing 1.5*) are semantically equivalent:

Listing 1.4

```
; Each rate constant on a separate line:
[constants]
   k1 = 1 ?
   k2 = 2 ?
   k3 = 3 ?
   k4 = 4 ?
```

Listing 1.5

```
; All rate constants on a single line:
[constants]
   k1 = 1 ?, k2 = 2 ?, k3 = 3 ?, k4 = 4 ?
```

The same degree of flexibility applies to the contents of the sections `[concentrations]` and `[responses]`.

Concentrations and molar responses in the `[data]` section

Let us assume that the data file f1.txt is associated with the enzyme concentration 0.01 μM, substrate concentration 10 μM, and inhibitor concentration 0.5 μM. In contrast, the file f2.txt is associated with a different set of concentrations, which are two-fold higher for each component. This can be expressed in DynaFit scripts in one of two semantically equivalent ways:

Listing 1.6

```
; local (file-based) concentrations listed separately:
[data]
   file f1.txt
        concentration E = 0.1
        concentration S =  10
        concentration I = 0.5
   file f2.txt
        concentration E = 0.2
        concentration S =  20
        concentration I = 1.0
```

Listing 1.7

```
; local (file-based) concentrations on the same line
[data]
   file f1.txt | conc E = 0.1, S = 10, I = 0.5
   file f2.txt | conc E = 0.2, S = 20, I = 1.0
```

The same degree of flexibility applies to defining local (file-based) molar response coefficients of reactants, using the keyword `response` or, equivalently, the abbreviation `resp`.

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 1.5.

Please note that the number of dashes or equal signs in the reaction arrow notation is not important. For example, the notation `->` has the same meaning as the notation `---->`. Similarly, `<=>` or `<======>` can be used equivalently to represent a reversible reaction.

References

1. Kuzmič, P.: Application of the Van Slyke–Cullen irreversible mechanism in the analysis of enzymatic progress curves. Anal. Biochem. **394**, 287–289 (2009)

Symbol	*Meaning*
`+`	separates reactants in a reaction step
`:`	separating reaction steps from names of rate constants
`*`	linking model parameters by using constant numerical factors
`|`	line break
`,`	lists of values, depending on context
`?`	optimized (adjustable, fitting) parameter
`??`	request confidence interval for an optimized parameter
`;`	comments (anything beyond semicolon (`;`) on any given line is ignored)
`->`	left-to-right reaction arrow
`<-`	right-to-left reaction arrow
`<==>`	double-sided reaction arrow (reversible step)
`[ ]`	names of main sections in the script file
`{}`	in the `[settings]` section of the script file and in the default settings file: names of sections for default settings
`{ , , }`	lists of values in the scan of initial estimates
`..`	optimization ranges

Table 1.5: Special characters and constructs.

2. 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., Patel, I.R.: 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**, 3457–3464 (1996)

Chapter 2
Tasks

The main task to be accomplished by DynaFit is defined in the `[task]` section of the script file. The `[task]` section must always contain at least the keywords `task` and `data`:

```
[task]
   task = ...
   data = ...
```

Optionally, the `[task]` section can also contain one or more of the following keywords:

```
[task]
...
   approximation = ...
   algorithm = ...
   confidence = ...
   model = ...
```

This chapter explains in detail the various values that be legitimately assigned to the above keywords. We also discuss the how the contents of the `[task]` section in certain particular cases interact with the required contents of certain other sections of the script, such as `[data]`.

2.1 Available tasks

The `task` keyword can be assigned one of three possible values:

```
task = fit
task = simulate
task = design
task = plot
```

This section describes the meaning of these alternate `task` choices, and the possible implications of any particular choice for other sections of the script.

2.1.1 Least-squares fit

```
[task]
   task = fit
```

The above input code signifies that DynaFit will perform least-squares fit of experimental data specified in the `[data]` section (see Chapter 7). If the fit is to be performed to a mechanistic model, the script must also contain at least the sections `[mechanism]` and `[constants]`. If the fit is to be performed to an arbitrary algebraic model, the script must also contain at least the sections `[parameters]` and `[model]`.

For a complete working example, see the script files 01.txt and 02.txt located in the directory ./manual/task/task/fit distributed with the program.

EXAMPLE SCRIPT

2.1.2 Simulations

```
[task]
   task = simulate
```

The above input code signifies that DynaFit will perform a simulation. This requires that the `[data]` section must contain the keyword `mesh`:

```
[data]
   mesh ...
```

The `mesh` keyword defines the values of the independent variable (for example, time), at which the value of the dependent variable (for example, fluorescence intensity changing over time) is to be computed. See Section 7.10 on page 103 for details, which explains how to set up the simulation using the keyword `mesh`.

For a complete working example, see the script file 01.txt located in the directory ./manual/task/task/simulate distributed with the program.

EXAMPLE SCRIPT

2.1.3 Optimal experiment design

```
[task]
   data = progress
   task = design
```

The above input code signifies that DynaFit will attempt to discover the most optimal starting concentrations of reagents for a planned experiment, where the composition of the reaction mixtures is followed over time (`data = progress`). In the current version of DynaFit only progress curve experiments can be optimized in this way.

The `[data]` section must contain a reference to at least one (but usually more then one) data file associated with a concentration of a particular reagent to be optimized, given the presumed reaction mechanism and the presumed values of microscopic rate constants. The particular starting concentration to be optimized is identified by a double question mark. The allowed concentration range is specified by the notation `?? (LOW .. HIGH)`.

Example 2.1 Optimal experimental design

```
[data]
   mesh      from 0 to 1000 step 10
   file F1 | concentration A = 1 ?? (0.001 .. 100)
   file F2 | concentration A = 1 ?? (0.001 .. 100)
   file F3 | concentration A = 1 ?? (0.001 .. 100)
```

In this example we wish to optimize the concentration of the reactant **A** in a series of three experiments to be analyzed globally. The allowed concentration range is between 0.001 and 100 μM. The time-points are fixed, from zero to 1000 seconds, stepping by 10 seconds.

For relatively complex reaction mechanisms and/or large number of experimental data points, the optimization algorithm may require significantly long execution time to complete, typically up to several hours.

For a complete working example, see the script file 01.txt located in the directory ./manual/task/task/design distributed with the program. EXAMPLE SCRIPT

2.1.4 Plotting of raw experimental data

```
[task]
   task = plot
```

The above input code signifies that DynaFit will generate a simple plot of the raw experimental data identified in the `[data]` section of the script. This feature is very useful in performing exploratory data analysis (EDA), before deciding on a suitable regression model.

2.2 Types of experimental data

DynaFit can be used to handle several distinct types of experimental data:

1. **Reaction progress.** The independent variable is time. The dependent variable is some observable some physical quantity such absorbance, fluorescence, HPLC peak area, and so on.
2. **Complex equilibria.** The independent variable is the total concentration of some particular reagent. There can be any number of total concentrations varied at the same time. The dependent variable again is some observable physical quantity.
3. **Initial reaction rates.** The independent variable is the total concentration of some particular reagent. The dependent variable is the *initial rate of change* in some observable physical quantity.
4. **Arbitrary data.** The dependent and independent variable can be any of any kind. The mathematical model is specified by the usual algebraic notation.
5. **Exponential data.** This is a special case of algebraic model fitting, where the model is represented as a sum of exponential terms. No initial estimates are required for the exponential amplitudes and rate constants.
6. **Piece-wise linear data.** This is another special case of algebraic model fitting, where the model is a "broken line" containing an arbitrary number of segments. This is sometimes useful to getting a preliminary idea for rate changes over the course of a particular reaction.
7. **Linear data.** DynaFit can be used to perform linear regression analysis.

For the first three types of experimental data listed above (progress, equilibria, and rates) the fitting model is represented symbolically by entering the postulated reaction mechanism. In general, the notation to identify a given data type is as follows:

```
[task]
   data = ...
...
```

In the above code snippet, the `data = ...` line can have one of the following values:

```
   data = progress
   data = equilibria
   data = rates
   data = generic
   data = exponential
   data = piecewise-linear
   data = linear
```

2.2.1 Reaction time course

As of version 4.08, released in March 2018, DynaFit recognizes two distinctly different types of reaction time-course data, depending on the type of experiment that was used to generate the given data set:

- Continuous assays
- Discontinuous assays

2.2.1.1 Continuous experiments

A "continuous" assay is an experiment where all reaction components are mixed in a *single* reaction mixture to start the reaction. Subsequently, some type of measuring instrument, such as for example an UV/Vis spectrophotometer or a fluorimeter, is utilized to record potentially very many consecutive readings of the experimental signal originating from the *same* sample.

From the point of view of data analysis, the most important feature of such "continuous" experiments is that the individual readings of the experimental signal are *not statistically independent* but rather they are strongly correlated. For this reason, those statistical data-analytic procedures that strictly rely on the assumption of statistical independence of individual data points cannot be used. This includes for example the computation of formal standard errors or confidence intervals [9, 8].

```
[task]
   data = progress
...
```

The above input code signifies that DynaFit will handle experimental or simulated data where the independent variable is the reaction time in suitably chosen units (seconds, minutes, hours, etc.). The dependent variable is the value of a particular physical quantity such as absorbance or fluorescence recorded at a particular reaction time t, in a *continuous* experiment.

The script must always contain at least the sections `[mechanism]` and `[constants]`. Both of these two script sections must refer only to rate constants, but not to equilibrium constants.

```
[mechanism]
     ...
[constants]
     ...
```

For a complete working example, see the script file 01.txt located in the directory ./manual/task/data/progress distributed with the program. EXAMPLE SCRIPT

2.2.1.2 Discontinuous experiments

A “discontinuous” assay is an experiment where the reaction components are mixed in a *multiple* distinct reaction mixtures, all of which have presumably the same initial composition, to start the reaction. Each sample (reaction mixture) is allowed to undergo time-dependent changes for a distinct amount of time, before the chemical or biological process is stopped, usually by some type of a chemical quenching mechanism. Subsequently the final composition of each distinct reacting mixture is analyzed by some appropriate chemical or physical method.

From the point of view of data analysis, the most important feature of such “discontinuous” experiments is that the individual readings of the experimental signal *are statistically independent*. For this reason, it is safe to utilize all those statistical data-analytic procedures that strictly rely on the assumption of statistical independence of individual data points.

In order to specify that the given experiment is discontinuous, the following notation must be used:

```
[task]
   data = progress discontinuous
...
```

Note that there is no equivalent “continuous” keyword. If the `discontinuous` notation is *not* used, DynaFit will automatically assume that the time-course experiment is continuous in nature.

2.2.2 *Equilibrium data*

```
[task]
   data = equilibria
...
```

The above input code signifies that DynaFit will handle experimental or simulated data where the independent variable is the total or analytic concentration of one or more reactant. The dependent variable is the value of a particular physical quantity, such as (for example) absorbance, fluorescence, NMR chemical shift recorded after the system came to full equilibrium.

The script must always contain at least the sections `[mechanism]` and `[constants]`. Both of these two script sections must refer only to equilibrium constants, but not to rate constants. The script section `[data]` must contain the keyword `variable` followed by variable reactant(s).

```
[mechanism]
     ...
[constants]
     ...
```

```
[data]
   variable ...
```

For a complete working example, see the script file 01.txt located in the directory ./manual/task/data/equilibria distributed with the program. EXAMPLE SCRIPT

2.2.3 Initial rates of enzyme reactions

```
[task]
   data = rates
   approximation = ...
...
```

The above input code signifies that DynaFit will handle experimental or simulated data where the independent variable is the total or analytic concentration of one or more reactant. The dependent variable is the observed initial rate of an enzyme reaction.

The script must always contain at least the sections `[mechanism]` and `[constants]`. These two script sections must refer to at least one rate constant and then also to a mixture of rate and equilibrium constants. The script section `[data]` must contain the keyword `variable` followed by variable reactant(s).

Importantly, the observed rate is expressed as the rate of change of a particular *physical quantity* such as absorbance or fluorescence per a suitably chosen unit of time. Examples include absorbance units per second, fluorescence units per minute, HPLC peak area per hour, or radioactive count per minute. Thus, the *observed* initial rates required by DynaFit is distinctly different from the *absolute* reaction rates, which are always expressed as the rate of change in molar concentrations per unit of time (e.g., micromoles per liter per minute, or moles per liter per second).

```
[mechanism]
    ...
[constants]
    ...
[data]
   variable ...
```

Initial rate kinetics in DynaFit is treated in one of three distinct theoretical frameworks, identified by the keyword `approximation` in the code fragment shown above:

1. Rapid-equilibrium approximation.
2. Steady-state approximation under "classical" experimental conditions (see below).
3. General treatment, invoking no simplifying approximation such as rapid-equilibrium or steady-state.

The DynaFit notation required to invoke these three separate theoretical frameworks is as follows.

2.2.3.1 Rapid-equilibrium approximation

```
[task]
   data = rates
   approximation = rapid-equilibrium
...
```

The above input code signifies that DynaFit will handle experimental or simulated data where the independent variable is the total or analytic concentration of one or more reactant. The dependent variable is the observed initial rate of an enzyme reaction under the rapid-equilibrium approximation [20, pp. 18-505].

Importantly, in contrast to the theoretical treatment found in classical biochemical literature [20], DynaFit allows any number of reactants to engage in "tight binding" interactions. "Non-tight binding" in enzyme kinetics is defined as a particular set of experimental conditions where the total or analytic concentrations of the enzyme is negligibly small compared to all dissociation equilibrium constants. In contrast, "tight binding" is defined as the particular set of experimental conditions where the total or analytic concentration of the enzyme is comparable in magnitude, or even higher than, certain dissociation equilibrium constants, in particular inhibition constants.

The script must always contain at least the sections `[mechanism]` and `[constants]`. The `[mechanism]` section must contain at least one rate constant measuring the rate of formation of the final reaction product form the reactive enzyme–substrate complex. All reversible elementary reactions for the binding and dissociation of substrate(s), inhibitor(s), and/or activator(s) must be expressed as dissociation equilibrium constants.

For a complete working example, see the script file 01.txt located in the directory ./manual/task/data/rates/rapid-equilibrium distributed with the program. EXAMPLE SCRIPT

If should be noted that every enzymatic reaction mechanism cast in the context of initial rates under the rapid-equilibrium approximation can be cast equivalently as a simple equilibrium binding problem (`data = equilibria`). In this case the "k_{cat}" values are folded into the molar response coefficient of the reactive enzyme–substrate complexes. This is shown in the script file 02.txt also located in the directory ./manual/task/data/rates/rapid-equilibrium distributed with the program. Both scripts (01.txt and 02.txt) produce exactly identical results but the second script is probably easier to understand by most DynaFit users. EXAMPLE SCRIPT

2.2.3.2 Classical steady-state approximation (King-Altman)

```
[task]
```

```
   data = rates
   approximation = king-altman
...
```

The above input code signifies that DynaFit will handle experimental or simulated data where the independent variable is the total or analytic concentration of one or more reactant. The dependent variable is the observed initial rate of an enzyme reaction under the “classical” steady-state approximation [20, pp. 506-847].

By “classical” approximation it is meant that the concentration of the enzyme is assumed to be negligibly small when compared to the concentrations of all substrates, inhibitors, activators, and other ligands. The initial rate equation is derived by using the King-Altman method. The derivation is fully automatic according to an algorithm devised by Cornish-Bowden [3].

The script must always contain at least the sections `[mechanism]` and `[constants]`. The `[mechanism]` section must contain only microscopic rate constant but not equilibrium constants. Furthermore, the `[mechanism]` section must contain the keyword `reaction` followed by the overall (un-catalyzed) reaction in the usual bio/chemical notation. If any inhibitors or activators appear in the mechanism, the `[mechanism]` section must also contain the keyword `modifiers` followed by a comma-separated list of reactants.

```
[mechanism]
   reaction ...
   modifiers ...
   ...
[constants]
   ...
```

For a complete working example, see the script file 01.txt located in the directory ./manual/task/data/rates/king-altman distributed with the program. EXAMPLE SCRIPT

2.2.3.3 No approximation

```
[task]
   data = rates
   approximation = none
...
```

The above input code signifies that DynaFit will handle experimental or simulated data where the independent variable is the total or analytic concentration of one or more reactant. The dependent variable is the observed “initial” rate of an enzyme reaction. More precisely, it is the reaction rate that would be observed at a specific time after mixing all reaction components (enzyme, substrate(s), inhibitor(s)). The specific value of this “delay” time is specified in the `[data]` section of the script, as follows:

```
[data]
   delay = ...
```

EXAMPLE SCRIPT

For a complete working example, see the script file 01.txt located in the directory ./manual/task/data/rates/no-approximation distributed with the program.

2.2.4 Generic data

```
[task]
   data = generic
...
```

DynaFit has the ability to handle arbitrary algebraic equations. The above input code signifies that DynaFit will handle experimental or simulated data where the independent variable is specified on the `variable` line in the `[data]` section. The dependent variable is represented by the symbol appearing on the left-hand side of the last algebraic equation in the `[model]` section.

```
[parameters]
    ...
[model]
    ...
```

The script must always contain at least the sections `[parameters]` and `[model]`. The `[parameters]` section must list all the adjustable (or fixed) model parameters. In the `[model]` section we must assign initial (or fixed) numerical values to all model parameters listed in the `[parameters]` section. Additionally, the `[model]` can also contain intermediate algebraic expressions. A symbol for the independent variable must appear as one of the model parameters.

For example, consider the Morrison equation [22] for tight-binding enzyme inhibition. In Eqn (2.1), v is the reaction rate observed at enzyme concentration $[E]_0$ and inhibitor concentration $[I]_0$; V_0 is the corresponding reaction rate observed in the absence of the inhibitor; and K_i^* is the apparent inhibition constant.

$$v = V_0 \frac{[E]_0 - [I]_0 - K_\mathrm{i}^* + \sqrt{\left([E]_0 - [I]_0 - K_\mathrm{i}^*\right)^2 + 4\,[E]_0\,K_\mathrm{i}^*}}{2[E]_0} \tag{2.1}$$

To define a requisite fitting model in DynaFit where the inhibition concentration is varied in the experiment, we can use the following notation:

```
[task]
   data = generic
...
[parameters]
   Io, Eo, Vo, Ki
```

```
[model]
   Eo = 20
   Ki =  4 ?
   Vo = 20 ?
   t  = Eo - Io - Ki
   v  = Vo * (t + sqrt (t*t + 4*Eo*Ki))/(2 * Eo)

[data]
   variable Io
...
```

In the above notation, the independent variable is the inhibitor concentration $[I]_0$, represented by the symbol `Io`. Note that `Io` does appear in the `[parameters]` section, along with the bona-fide model parameters $[E]_0$, V_0, and K_i represented as `Eo`, `Vo`, and `Ki`, respectively. The symbol `t` stands for a temporary variable, $t = [E]_0 - [I]_0 - K_\mathrm{i}^*$, that is used merely to simplify the overall algebraic expression.

The use of question marks next to V_0 and K_i signifies that these two parameter are treated as adjustable in the fitting model. In contrast, the absence of a question mark next to $[E]_0$ signifies that the enzyme concentration is treated as a fixed parameter in the fitting model.

For a complete working example, see the script file 01.txt located in the directory ./manual/task/data/generic distributed with the program. EXAMPLE SCRIPT

2.2.5 Multi-exponential data

```
[task]
   task = fit
   data = exponential
...
```

The above notation signifies that DynaFit will treat the experimental data to be fit as a sum of exponential terms. The simulation mode is not available for this data type, only data-fitting mode. In Eqn (2.2), x is the independent variable; y is the dependent variable; a_0 is the offset (baseline) on the signal axis; a_i $(i = 1, 2, \ldots, n)$ are exponential amplitudes; and b_i $(i = 1, 2, \ldots, n)$ are the corresponding exponential rate parameters.

$$y = a_0 + \sum_{i=1}^{n} a_i \exp(b_i\, x) \tag{2.2}$$

The number of exponential terms, n, can be determined automatically by using the Legendre polynomial method of Martin & Maconochie [13, 14]. Automatic selection of the most appropriate exponential degree is arranged for by default (see `Automatic = y` in the initialization code below), up to a certain maximum number of terms to be considered. By default, the maximum degree is $n = 4$ (see `Degree = 4` in the initialization code listed below).

```
[settings]
{ExponentialFit}
   Degree                          = 4
   Automatic                       = y
```

If we wished to force-fit the data to a certain particular number of exponentials, for example three, we could override the default settings listed above as follows:

```
[settings]
{ExponentialFit}
   Degree                          = 3
   Automatic                       = n
```

EXAMPLE SCRIPT

For a complete working example, see the script file 01.txt located in the directory ./manual/task/data/exponential distributed with the program.

2.2.6 *Piecewise linear data*

```
[task]
   task = fit
   data = piecewise-linear
...
```

The above notation signifies that DynaFit will treat the experimental data to be fit as a disjointed sequence of multiple straight-line segments. The simulation mode is not available for this data type, only data-fitting mode. The only mandatory section of the script file is the `[data]` section. The default settings are shown below:

```
[settings]
{PieceWiseLinearFit}
   Points                          = 0
   Segments                        = 4
   Time                            = 0
```

The meaning of these control settings is as follows:

- If the `Points` parameter above is set to any value other than zero, the program will take the corresponding number of consecutive data points one at a time and will fit each separate segment to a straight line.
- If the `Segments` parameter above is set to any value other than zero, the program will divide the complete kinetic trace into the corresponding number of equal-size segments, measured by the number of data points in each segment (not by the segment duration).
- If the `Time` parameter above is set to any value other than zero, the program will divide the complete kinetic trace into equal-length segments in terms of the duration of each segment.

For a complete working example, see the script file 01.txt located in the directory ./manual/task/data/piecewise-linear distributed with the program. EXAMPLE SCRIPT

2.2.7 *Linear data*

This option is available for processing of experimental data that can legitimately be expected to follow a strictly linear trend. The script file notation is as follows:

```
[task]
   task = fit
   data = linear
...
```

It should be noted that there are very few circumstances arising in chemistry, biology, biochemistry or biophysics (i.e., disciplines where DynaFit is most frequently used) where the the dependence of an observed quantity on model parameters is strictly linear. The linear fit option is provided in DynaFit mostly for compatibility of with other software packages, such as MS Excel, that are frequently used to perform linear regression analyses in the laboratory.

2.3 Confidence intervals

DynaFit uses two different algorithms to estimate confidence intervals for adjustable model parameters. The first (default) algorithm uses the *profile-t* method of Bates & Watts [1, 2]. A pseudo-code for this algorithm is shown in ref. [1, p. 303]. The second algorithm is based on the Monte-Carlo method as described by [21].

2.3.1 *Systematic search:* **Profile-t** *method*

```
[task]
   ...
   confidence = search
```

The above input code signifies that DynaFit will perform a systematic search for the limits of the confidence interval by using the *profile-t* method of Bates & Watts Bates & Watts [1]. The particular model parameters that are to be searched must be identified by the “double question mark” notation, as is shown in the code fragment below.

```
[constants]
   k1 = 123 ??    ; determine full confidence interval
   k2 = 456 ?      ; determine best-fit value only
   k3 = 789        ; treat as a fixed parameter
...
```

In this example, the rate constant k_1 is assigned the initial estimate of $k_1 = 1234$. The double question mark signifies that we wish to obtain not only the best-fit values, but also additionally the confidence interval at a given probability level. The single question mark after $k_2 = 456$ signifies that we wish to obtain the best-fit value using "456" as the initial estimate, but we do not wish to compute the confidence interval. Finally the rate constant $k_3 = 789$ is treated as a fixed parameter in the fitting model, because the numerical value "789" is not followed by either a single or a double question mark.

By default, DynaFit assumes that the desired confidence level is 95%. This default value can be changed by inserting an *initialization* code into the optional `[settings]` section, as shown below:

```
[settings]
{ConfidenceIntervals}
   LevelPercent = 99
```

In this example, the desired confidence level was changed from the default 95% to 99%. This will make all confidence intervals somewhat wider.

EXAMPLE SCRIPT

For a complete working example, see the script file 01.txt located in the directory ./manual/task/confidence/search distributed with the program.

2.3.2 The Monte-Carlo method

```
[task]
   ...
   confidence = monte-carlo
```

The above input code signifies that DynaFit will perform a systematic search for the limits of the confidence interval by using the Monte-Carlo method [21]. In this case there is no need to use the double question mark notation as described in section 2.3.1. The confidence interval will be produced for all adjustable model parameters.

In many cases we might wish to restrict the number of Monte-Carlo *correlation plots* produced by DynaFit. By default, correlation plots are produced for all possible pairs of model parameters, including "nuisance parameters" such as offsets on the signal axis. To reduce the number of correlation plots, we can list in the section `[correlations]` only the model parameters of particular interest. DynaFit will then produce correlation plots for all possible pairs of parameters listed in that section.

```
[correlations]
   k1, k2, k3, k4
```

The example code immediately above lists for four parameters of interest. Therefore, DynaFit will produce $(4 \times 3)/2 = 6$ pairs of correlation plots (k_1/k_2, k_1/k_3, k_1/k_4, k_2/k_3, k_2/k_4, and k_3/k_4).

For a complete working example, see the script file 01.txt located in the directory ./manual/task/confidence/monte-carlo distributed with the program.

EXAMPLE SCRIPT

2.4 Data-fitting algorithms

DynaFit can perform the least-squares fit of experimental data to a given model by utilizing one of three distinct algorithms:

1. The **Hybrid Trust-Region** algorithm of Dennis *et al.* [4, 5, 6].
2. The **Levenberg-Marquardt** algorithm [12] as implemented by Reich [18]
3. The **Differential Evolution** algorithm as implemented by Price *et al.* [17]

Each of these algorithms has its advantages and disadvantages.

2.4.1 The Trust region algorithm

```
[task]
   ...
   algorithm = trust-region
```

Abbreviated version:

```
[task]
   ...
   algorithm = TR
```

The above input code signifies that DynaFit will perform the least-squares fit of experimental data using the hybrid "trust-region" algorithm NL2SOL devised by Dennis *et al.* [4, 5, 6]. This is the default algorithm built into DynaFit, in the sense that if the `algorithm = ...` line is absent in the `[task]` section, DynaFit will use it. The advantage of the trust-region algorithm is that it is relatively fast. Another major advantage is that the algorithm can successfully handle parameter constraints.

A disadvantage is in unfavorable cases the initial estimates of the model parameters must be reasonably close to the true values. In extreme cases, the initial estimates of rate constants or equilibrium constants must be within one order of magnitude, relative to the true "best-fit" values, otherwise the algorithm converges into a false minimum. Unfortunately at the outset of the of the investigation we rarely have the ability to estimate the model parameters with such high precision.

2.4.2 The Levenberg-Marquardt algorithm

```
[task]
   ...
   algorithm = levenberg-marquardt
```

Abbreviated version:

```
[task]
   ...
   algorithm = LM
```

The above input code signifies that DynaFit will perform the least-squares fit of experimental data using the classic Levenberg-Marquardt algorithm as implemented by Reich [18]. The advantage of the Levenberg-Marquardt algorithm is that, as the trust-region algorithm mentioned above, it is relatively fast.

There are also two important disadvantages. The first major disadvantage is that in unfavorable cases the initial estimates of the model parameters must be very close to the true values. The second major disadvantage of the Levenberg-Marquardt algorithm is that it does not deal very well with realistic bounds imposed on model parameters. For example, based on fundamental principles of chemistry and physics, all rate constants, equilibrium constants, or reactant concentrations can only attain positive values. However, in unfavorable cases the Levenberg-Marquardt algorithm has a tendency to produce negative values, which are physically meaningless. To prevent this, DynaFit implements a simple “restart” algorithm originally described by Duggleby [7]. However, the “restart” algorithm often produces extremely slow convergence.

Starting with version 4.08 of DynaFit, the Levenberg-Marquardt algorithm is included in DynaFit mostly for compatibility with older versions of the software package.

In exceptionally difficult and “ill-posed” regression problems, the more advanced hybrid trust-region algorithm NL2SOL and the classic Levenberg-Marquardt algorithm may produce different results. In those situations DynaFit users are advised to check all diagnostic messages, in particular those generated by the L-M algorithm, for any indication that the algorithm may have failed to reach convergence.

2.4.3 The Differential Evolution algorithm

```
[task]
   ...
   algorithm = differential-evolution
```

Abbreviated version:

```
[task]
   ...
   algorithm = DE
```

The above input code signifies that DynaFit will perform the least-squares fit of experimental data using the Differential Evolution (DE) algorithm as implemented by Price *et al.* [17]. The major advantage of the DE algorithm is that it frequently (although not always) converges to the global minimum on the least-squares hypersurface, as opposed to a local (false) minimum. There are also several important disadvantages.

The first major disadvantage is that the DE algorithm can be excruciatingly slow. Even for moderately complex models formulated in terms of differential equations, typical running times can easily reach multiple hours. There have been practically important data-analysis problems [19] where the running time reached up to 20 hours per data set on a multi-core computer with 2.4 GHz clock.

The second major disadvantage of the DE algorithm is that its global convergence behavior is not mathematically guaranteed. Consequently if we obtain a relatively poor fit of the experimental data to any given mechanistic model, we still do not know with perfect certainty whether a better combination of model parameters might exist (the "false minimum" problem).

Finally, the DE algorithm is not very well documented in the literature and therefore its use within DynaFit should be considered merely experimental at this point.

2.5 Multiple tasks and model discrimination

DynaFit has the ability to perform model discrimination analysis using three different statistical criteria:

1. The Bayesian Information Criterion (BIC) [15, 16]
2. The Akaike Information Criterion (AIC) [15, 16]
3. The F-Test for nested models [10, 11]

To arrange for model discrimination analysis using these three statistical criteria, the DynaFit script must contain multiple `[task]` sections. Each `[task]` section must contain the keyword `model =` , followed by an arbitrary model label, followed by the question mark.

For example, let us consider a scenario from enzyme kinetics, where the goal is to discriminate between the "competitive", "noncompetitive", or "mixed" inhibition models. In this case, the DynaFit script would contain the following notation:

```
[task]
   ...
   model = competitive ?
```

```
...
...
[task]
   ...
   model = noncompetitive ?
...
...
[task]
   ...
   model = mixed ?
...
...
```

Only the first task block will contain the `[output]` section, the `[data]` section, and (optionally) any `[settings]`. The different task blocks typically contain a unique notation only for `[mechanism]` and `[constants]`.

In general, all subsequent task blocks will always re-use any sections that would normally appear in the given block, but are missing, from the most immediately preceding task blocks. This applies even to DynaFit scripts that do not involve model discrimination. In the example below, let us assume that the competitive, noncompetitive, and uncompetitive reaction mechanisms involve only one equilibrium constant, K_i. In contrast, let's assume that the mixed reaction mechanism is defined by two separate equilibrium constants, K_i1 and K_i2. Please note how this "sharing" of K_i within the first three mechanistic models is arranged in the code snippet immediately below.

```
[task]
   ...
   model = competitive ?
...
[constants]
  Ki = 1.23 ?  ; ... does not have to be repeated below
...
[task]
   ...
   model = noncompetitive ?
...
[task]
   ...
   model = uncompetitive ?
...
[task]
   ...
   model = mixed ?
...
[constants]
  Ki1 = 1.23 ?
  Ki2= 4.56 ?
...
```

Because the first three mechanisms (competitive, noncompetitive, and uncompetitive) all contain only one equilibrium constant, the initial estimate given in the first (competitive) mechanism will be shared by the following two mechanisms (noncompetitive and uncompetitive). The `[constants]` section does not have to be repeated in the second and third task block. Only after DynaFit encounters the last task block, referring to the mixed reaction mechanism, it is necessary to insert a `[constants]` section with two separate equilibrium constants.

The same section-sharing principle applies to all other sections of any particular DynaFit script.

References

1. Bates, D.M., Watts, D.G.: Nonlinear Regression Analysis and its Applications. Wiley, New York (1988)
2. Brooks, I., Watts, D., Soneson, K., Hensley, P.: Determining confidence intervals for parameters derived from analysis of equilibrium analytical ultracentrifugation data. Meth. Enzymol. **240**, 459–78 (1994)
3. Cornish-Bowden, A.: An automatic method for deriving steady-state rate equations. Biochem. J. **165**, 55–59 (1977)
4. Dennis, J.E., Gay, D.M., Welsch, R.E.: An adaptive nonlinear least-squares algorithm. ACM Trans. Math. Software pp. 348–368 (1981)
5. Dennis, J.E., Gay, D.M., Welsch, R.E.: Algorithm 573: NL2SOL. ACM Trans. Math. Software **7**, 369–383 (1981)
6. Dennis, J.E., Schnabel, R.B.: Numerical Methods for Unconstrained Optimization and Nonlinear Equations. Prentice-Hall, Upper Saddle River, NJ (1983)
7. Duggleby, R.G.: Regression analysis of nonlinear Arrhenius plots: an empirical model and a computer program. Comput. Biol. Med. **14**, 447–455 (1984)
8. Johnson, K.A., Simpson, Z.B., Blom, T.: FitSpace Explorer: An algorithm to evaluate multidimensional parameter space in fitting kinetic data. Anal. Biochem. **387**, 30–41 (2009)
9. Johnson, K.A., Simpson, Z.B., Blom, T.: Global Kinetic Explorer: A new computer program for dynamic simulation and fitting of kinetic data. Anal. Biochem. **387**, 20–29 (2009)
10. Mannervik, B.: Design and analysis of kinetic experiments for discrimination between rival models. In: L. Endrényi (ed.) Kinetic data analysis, pp. 235–270. Plenum Press, New York (1981)
11. Mannervik, B.: Regression analysis, experimental error, and statistical criteria in the design and analysis of experiments for discrimination between rival kinetic models. Methods Enzymol. **87**, 370–390 (1982)
12. Marquardt, D.W.: An algorithm for least-squares estimation of nonlinear parameters. J. Soc. Ind. Appl. Math. **11**, 431–441 (1963)
13. Martin, J.L., Maconochie, D.J.: The direct fitting of a differential equation to experimental data. J. Physiol. **418**, 7 (1989)
14. Martin, J.L., Maconochie, D.J., Knight, D.R.: A novel use of differential equations to fit exponential functions to experimental data. J. Neurosci. Meth. **51**, 135–146 (1994)
15. Myung, J.I., Pitt, M.A.: Model comparison methods. Meth. Enzymol. **383**, 351–366 (2004)
16. Myung, J.I., Tang, Y., Pitt, M.A.: Evaluation and comparison of computational models. Meth. Enzymol. **454**, 287–304 (2009)
17. Price, K.V., Storn, R.M., Lampinen, J.A.: Differential Evolution: A Practical Approach to Global Optimization. Springer Verlag, New York (2005)
18. Reich, J.G.: Curve Fitting and Modelling for Scientists and Engineers. McGraw-Hill, New York (1992)

19. Schwartz, P.A., Kuzmic, P., Solowiej, J., Bergqvist, S., Bolanos, B., Almaden, C., Nagata, A., Ryan, K., Feng, J., Dalvie, D., Kath, J., Xu, M., Wani, R., Murray, B.W.: Covalent egfr inhibitor analysis reveals importance of reversible interactions to potency and mechanisms of drug resistance. Proc. Natl. Acad. Sci. U.S.A. **111**, 173Ű178 (2014)
20. Segel, I.H.: Enzyme Kinetics. Wiley, New York (1975)
21. Straume, M., Johnson, M.L.: Monte Carlo method for determining complete confidence probability distributions of estimated model parameters. Meth. Enzymol. **210**, 117–29 (1992)
22. Williams, J.W., Morrison, J.F.: The kinetics of reversible tight-binding inhibition. Meth. Enzymol. **63**, 437–467 (1979)

Chapter 3
Molecular mechanism

The reaction mechanism for the given chemical or 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     :  kaS   kdS
   ES ---> E + P      :  kr
   E + I <===> EI     :  kaI   kdI
```

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

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

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

```
[mechanism]
   P + P + P <=> T           :   kaP3     kdP3
   T + DNA <==> T.DNA        :   kaTD     kdTD
   T + T + DNA <==> T2.DNA   :   kaT2D    kdT2D
```

Example 3: An oscillatory metabolic cascade

```
[mechanism]
         S1 + E <===> S1.E        :    k1      k2
           S1.E  ---> E + S2      :    k3
         S2 + E <===> S2.E        :    k4      k5
                ---> S1           :    v6
              S2 --->             :    v
```

3.1 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

$$\mathrm{E}+\mathrm{S} \underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} \mathrm{ES}$$

$$\mathrm{ES} \overset{k_2}{\rightarrow} \mathrm{E}+\mathrm{P}$$

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

	$k_{\rightarrow}$	$k_{\leftarrow}$
$\mathrm{E}+\mathrm{S} \rightleftharpoons \mathrm{ES}$	k_1	k_{-1}
$\mathrm{ES} \rightarrow \mathrm{E}+\mathrm{P}$	k_2	

which is represented in DynaFit by the following text:

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

3.1.1 Notational flexibility

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

```
[mechanism]
   E + S ----> E.S      : kaA
   E.S   ----> E + S    : kdA
   E.S   ----> E + P    : kdP
```

or even in a condensed form as

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

where the vertical bar represents a line break.

3.1.2 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.

3.1.3 Equilibrium constants

3.1.3.1 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
```

3.1.3.2 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 `dissoc`. In the following examples, both `Ki` and `Ksi` are dissociation equilibrium constants although the left-to-right convention shows the reaction steps ass association equilibria.

```
E + I <===>  EI      :   Ki      dissoc
ES + I <===> ESI     :   Ksi     dissoc
```

3.1.3.3 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.

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

All 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
```

3.2 Reaction 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 (please note the presence of blank characters interspersed with non-blank characters):

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

3.3 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:

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

3.4 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 `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:

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

```
   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)
```

3.5 Constant rates in open reaction systems

DynaFit can be used for simulation and fitting of biochemical reactions occurring 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 continuously 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
Rate and equilibrium 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 is explained in section 4.3, nominal values of rate constants depend both on the time scale and on the concentration scale of the experiment.

4.1 Concentration and time scale

It is important to discuss the issue of properly *scaling* all rate constants, equilibrium constants, and 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.

4.1.1 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 4.1 Micromolar concentration scale

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).

In summary, 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.

4.1.2 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.

4.2 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 ?
```

4.3 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.

4.3.1 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 $\times$ 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.

Example 4.2 *Time scale in minutes*

reaction type	order	molecularity	dimension of k
$A \xrightarrow{k}$	0	(constant flux)	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.

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 4.3 Micromolar units for molar response

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 μM^{-1} $\times$ 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, $\varepsilon = 0.01234$ (absorbance units per μM of product).

4.3.2 *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)
$A + B \overset{K}{\rightleftharpoons} C$	concentration^{-1}
$C \overset{K}{\rightleftharpoons} A + B$	concentration
$A + A + A \overset{K}{\rightleftharpoons} A_3$	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.

4.4 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.

4.4.1 *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 $\mathrm{M}^{-1}\mathrm{sec}^{-1}$ 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 $k \approx 10^6$ $\mathrm{M}^{-1}\mathrm{sec}^{-1}$ = 1.0 $\mu\mathrm{M}^{-1}\mathrm{sec}^{-1}$. 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 $k \approx 10^6$ $\mathrm{M}^{-1}\mathrm{sec}^{-1}$ = 60.0 $\mu\mathrm{M}^{-1}\mathrm{min}^{-1}$.

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

4.4.2 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 4.4 Estimating dissociation rate constants

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}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.

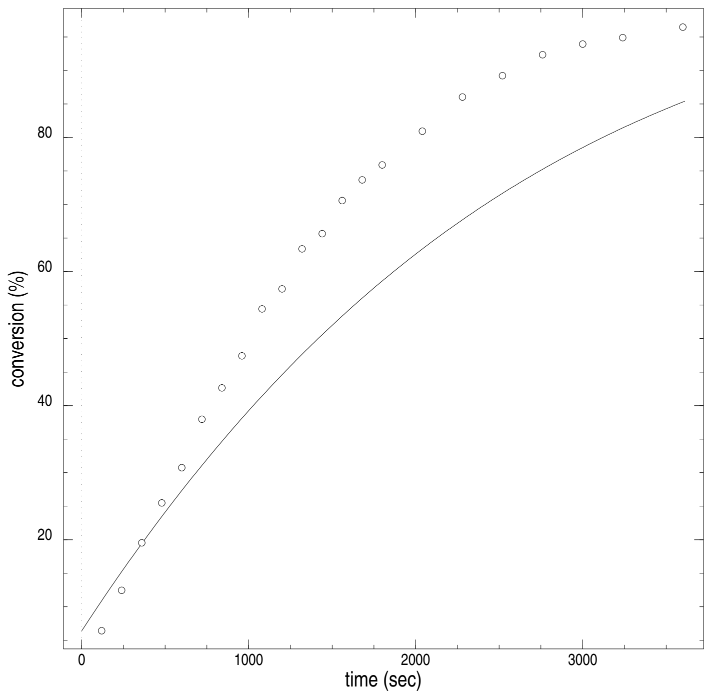


Fig. 4.1: Example of an initial estimate suitable for starting the regression analysis.

4.4.3 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 4.5 Estimating equilibrium constants

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 $K_D \approx 300$ 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.

4.5 Linked rate constants

In many cases the rate constants appearing in the `[mechanism]` section are not fully independent, but are instead mutually dependent in various ways. The three main reasons for rate constant linking are as follows:

1. Ratios of rate constants (i.e., equilibrium constants) are known.
2. Rate constants are linked through statistical factors.
3. The reaction mechanism contains thermodynamic cycles.

These particular circumstances are addressed below in their turn.

4.5.1 Known equilibrium constants

Assume that the given reaction mechanism contains a particular association rate constant k_{on}, and also a dissociation rate constant k_{off}. The corresponding equilibrium dissociation equilibrium constant $K_{\mathrm{d}} = k_{\mathrm{off}}/k_{\mathrm{on}}$ is presumed to be known and therefore it is to be treated as a fixed model parameter.

Under these circumstances we have two equivalent choices in the definition of the the association and dissociation rate constants. Either k_{off} can be expressed as $k_{\mathrm{off}} = K_{\mathrm{d}} \times k_{\mathrm{on}}$, or k_{on} can be expressed as $k_{\mathrm{on}} = k_{\mathrm{off}}/K_{\mathrm{d}}$. In either cases we start from a given numerical value of the equilibrium dissociation constant, for example, $K_{\mathrm{d}} = 5$ in the given concentration units. Then we can express the link between the two dependent rate constants as follows.

Variant A

```
[constants]
   koff = 5 * kon          ;    Kd = 5 = koff/kon
```

Variant B

```
[constants]
   kon = 0.2 * koff        ;    Kd = 5, 1/Kd = 0.2
```

Please note that the linking syntax always includes the *multiplication* sign “*”. Thus in the case of k_{on} we must first compute the numerical value of $1/K_{\mathrm{d}}$ and then use it as a multiplication factor. Also note that the multiplication factor must stand *before* the symbol of the relevant rate constant. In other words the notation `0.2 * koff` is valid whereas the algebraically equivalent notation `koff * 0.2` is not.

4.5.2 Statistical factors

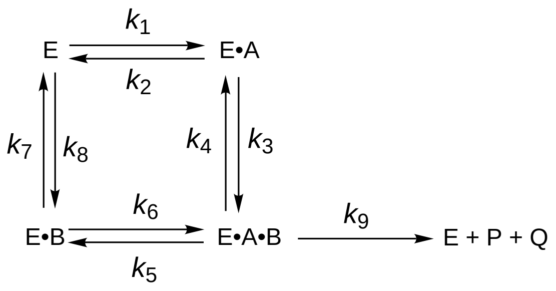
Rate constants or equilibrium constants appearing in the `[mechanism]` section could be linked due to statistical factors expressing the independence of multiple identical and non-interacting binding sites [2]. Thus for example in the case of two ligand molecules binding to two identical and non-interacting binding sites, we must use the following notation, in which the identity and independence of the two binding sites is expressed via the statistical factor “4”.

```
[mechanism]
     L + R <==> R.L     :  Kd1   dissociation
   L + R.L <==> L.R.L   :  Kd2   dissociation
```

```
[constants]
   Kd1 = ...              ;  any numerical value
   Kd2 = 4 * Kd1          ;  statistical factor
```

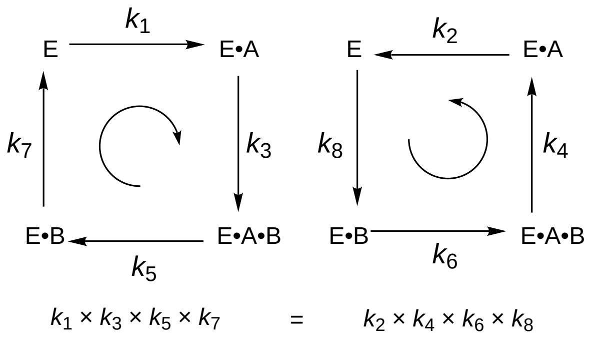
4.5.3 *Thermodynamic cycles*

The issue of thermodynamic cycles [1, p. 271] and the appropriate scripting notation is best explained by way of an example. Let us consider the reaction mechanism shown in *Scheme 4.1*, in which two co-substrates labeled A and B associate with the enzyme E in random order, i.e., through two alternate pathways.



Scheme 4.1

According to fundamental laws of thermodynamics the eight rate constants k_1 through k_8 appearing in the cycle must satisfy the algebraic relationship expressed in *Scheme 4.2*, namely, $k_1 \times k_3 \times k_5 \times k_7 = k_2 \times k_4 \times k_6 \times k_8$.


E
E•A
E•B
E•A•B
k_1
k_3
k_5
k_7
k_2
k_4
k_6
k_8
$k_1 \times k_3 \times k_5 \times k_7 = k_2 \times k_4 \times k_6 \times k_8$

Scheme 4.2

The curved arrows in *Scheme 4.2* illustrate a convenient mnemonic we can use. Namely, tracing the cycle in clockwise direction and multiplying all rate constants we encounter must produce a product that is equal to tracing the rate constants in counter-clockwise direction. The reason is that the overall equilibrium constants must be equal to unity.

The presence of a thermodynamic cycle in *Scheme 4.1* requires that *either* one of the eight rate constants k_1 through k_8 must be expressed in terms of the remaining seven rate constants. The choice of this "dependent" rate constant must be made on a cases-by-base basis, taking into account all all relevant background information. For example, we could choose to express the rate constant k_8 in terms of k_1 through k_7, as

$$k_8 = (k_1\, k_3\, k_5\, k_7)/(k_2\, k_4\, k_6) \qquad .$$

The corresponding DynaFit scripting notation is

```
  k8 = (k1 k3 k5 k7) / (k2 k4 k6)
```

Please note that in this case the multiplication sign "*" is absent. The list of rate constants to the left of the division sign "/" must contain exactly one fewer items compared to the list of rate constants on the right. *Both* lists of rate constants must be enclosed in parentheses. The code fragment below shows the appropriate notation in context.

```
[data]
  data = progress
  ...

[mechanism]
  E + A    <==> E.A        :  k1  k2
  E.A + B  <==> E.A.B      :  k3  k4
  E.A.B    <==> E.B + A    :  k5  k6
  E.B      <==> E + B      :  k7  k8
  E.A.B     --> E + P + Q  :  k9

[constants]
  k1 = ...     ; any numerical values for k1--k7
  k2 = ...
  ...
  k7 = ...
  k8 = (k1 k3 k5 k7) / (k2 k4 k6)      ; cycle!
```

The above notation is particularly important in data fitting as opposed to exploratory simulations. In any given heuristic simulation, we could of course choose the numerical values of all rate constants such that all relevant thermodynamic box rules are perfectly satisfied. In contrast, in data fitting at least some of the rate constants will be free-floating in the kinetic model.

Under those circumstances DynaFit will be adjust all free-floating rate constants to achieve the best possible match to the experimental data, while at the same time making sure that all thermodynamic cycle rules are fully satisfied. In the illustrative example above, at every step the iterative least-squares refinement the value of k_8 is always recomputed from the current estimates of k_1 through k_7.

The current version of DynaFit does not have any ability to automatically discern the presence of thermodynamic cycles in bona-fide *kinetic* models, i.e., in the analysis the reaction progress curves. It is the full responsibility of the investigator to introduce the appropriate linking expressions similar to `k8 = (k1 k3 k5 k7) / (k2 k4 k6)`, as many as necessary.

In contrast, in the analysis of *equilibrium binding* experiments or initial-rate enzyme kinetics under the rapid equilibrium approximation, DynaFit will automatically "discover" thermodynamic cycles if any are present in the `[mechanism]` section, and if necessary it will assure that the numerical values of all equilibrium constants are fully consistent with the fundamental laws or thermodynamics.

References

1. Gilbert, H.F.: Basic concepts in biochemistry, 2nd edn. McGraw-Hill, New York (1999)
2. Kuzmič, P.: DynaFit – A software package for enzymology. Meth. Enzymol. **467**, 247–280 (2009)

Chapter 5
Concentrations

The script file section denoted as `[concentrations]` is optional. However, it must be present file unless the `concentration` keyword is used in the `[data]` section of the script file.

The `[concentrations]` section lists the values of concentrations and (optionally) labels some or all of them as adjustable parameters. As was mentioned before in section 4.3, 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.

Illustrative example

Let us consider a biochemical reaction following the Michaelis-Menten reaction mechanism shown in *Listing 5.1*. The assumed rate constants values were $k_{aS} = 10^6$ $\mathrm{M}^{-1}\mathrm{s}^{-1}$ (`kaS = 1.e+6` in the listing below); $k_{dS} = 20$ s^{-1}, and $k_{dP} = 5$ s^{-1}. The concentration of enzyme was kept constant at $[E] = 10^{-9}$ M (`E = 1.e-9`), while the concentration of substrate $[S]$ was varied between 0.5 $\times 10^{-3}$ M and 8 $\times 10^{-3}$ M (`S = 0.5e-3, 1.0e-3, ..., 8.0e-3`). The formation of one mole per liter of the reaction product `P` corresponds to an increase in the UV/Vis experimental signal by 1500 absorbance units (`P = 1500` in the responses section below).

Listing 5.1

```
; Concentration scale: moles per liter
[task]
   data = progress
   task = simulate
[mechanism]
   E + S <==> ES :  kaS    kdS
   ES --> E + P  :  kdP
[constants]
   kaS = 1.e+6
   kdS = 20
   kdP = 5
[concentrations]
   E = 1.e-9
[responses]
   P = 1500
[data]
   file f1.txt | conc S = 0.5e-3
   file f2.txt | conc S = 1.0e-3
   file f3.txt | conc S = 2.0e-3
   file f4.txt | conc S = 4.0e-3
   file f5.txt | conc S = 8.0e-3
[end]
```

In this example the concentrations vary between nM (10^{-9} M enzyme) to mM (10^{-3} M substrate). Therefore the most natural unit of concentrations is μM (10^{-6} M). This *scaling* determines not only the numerical values of concentrations to be used in the [concentrations] section (see *Listing 5.2* below), but also the values of the bimolecular association rate constants and the molar response coefficient. In the case of the association rate constants, $k_{aS} = 1 \times 10^6$ M^{-1}s^{-1} becomes $k_{aS} = 1$ μM^{-1}s^{-1} (`kaS = 1` in *Listing 5.2*). In the case of the molar response coefficient, $r_P = 1500$ a.u./M becomes $r_P = 0.0015$ a.u./μM (`P = 0.0015` in *Listing 5.2*).

Listing 5.2

```
; Concentration scale: micromoles per liter
[task]
   data = progress
   task = simulate
[mechanism]
   E + S <==> ES :  kaS     kdS
   ES --> E + P  :  kdP
[constants]
   kaS = 1        ;  was 1.e+6
   kdS = 20
   kdP = 5
[concentrations]
```

```
   E = 0.001      ;  was 1.e-9
[responses]
   P = 0.0015     ;  was 1500
[data]
   file f1.txt | conc S =  500 ;  was 0.5e-3
   file f2.txt | conc S = 1000 ;  was 1.0e-3
   file f3.txt | conc S = 2000
   file f4.txt | conc S = 4000
   file f5.txt | conc S = 8000
[end]
```

5.2 Global and local concentrations

Certain concentrations can be made global, that is, applicable to all datasets enumerated in the script files. The values of these global concentrations are listed in the `[concentrations]` section. For example, all five progress curves mentioned in the script file listed above were collected at the same enzyme concentration, $[E] = 0.001\ \mu$M. Therefore the script contains the notation

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

Now let us assume that the enzyme concentration varied from one dataset to another, as did the substrate concentration. In that case the `[concentrations]` section might be omitted completely. Instead, each dataset would be assigned a unique value for both concentrations as is shown in *Listing 5.3*.

Listing 5.3

```
[task]
   data = progress
   task = simulate
[mechanism]
   E + S <==> ES :  kaS     kdS
   ES --> E + P  :  kdP
[constants]
   kaS = 1
   kdS = 20
   kdP = 5
[responses]
   P = 0.0015
[data]
   file f1.txt | conc S =  500, E = 0.001
   file f2.txt | conc S = 1000, E = 0.002
   file f3.txt | conc S = 2000, E = 0.003
   file f4.txt | conc S = 4000, E = 0.004
   file f5.txt | conc S = 8000, E = 0.005
[end]
```

If a concentration value is listed in the `[concentrations]` section (global value) and simultaneously in the `[data]` 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 locally adjustable parameters. This is illustrated in *Listing 5.4*. In this example, all five datasets were obtained with *nominally* identical enzyme concentration, $[E] = 1$ nM. However, because of the inevitable titration error, the *actual* enzyme concentrations will always be slightly different going from one dataset to the next. To achieve satisfactory global fit under these circumstances, it is necessary to treat all *except one* enzyme concentration as adjustable parameters. This is indicated by the presence of question marks after the notation `E = 0.001` for all except one experimental data file.

Listing 5.4

```
[task]
   data = progress
   task = fit
[mechanism]
   E + S <==> ES :  kaS    kdS
   ES --> E + P  :  kdP
[constants]
   kaS = 1
   kdS = 20 ?
   kdP = 5 ?
[responses]
   P = 0.0015 ?
[data]
   file f1.txt | conc S =  500, E = 0.001
   file f2.txt | conc S = 1000, E = 0.001 ?
   file f3.txt | conc S = 2000, E = 0.001 ?
   file f4.txt | conc S = 4000, E = 0.001 ?
   file f5.txt | conc S = 8000, E = 0.001 ?
[end]
```

5.3 Concentrations as optimized parameters

Initial or total concentrations can be treated as adjustable parameters. A given concentration value 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, whereas *local* optimization means that the given adjustable concentration applies only to the given dataset.

These concepts are illustrated in *Listing 5.5*. In that particular example the enzyme concentration was optimized globally, across all five experimental datasets, whereas the substrate concentration was optimized locally.

Listing 5.5

```
[task]
   data = progress
   task = fit
[mechanism]
   E + S <==> ES :  kaS     kdS
   ES --> E + P  :  kdP
[constants]
   kaS = 1
   kdS = 20 ?
   kdP = 5
[concentrations]
   E = 0.001 ? ; best-fit value applies to all 5 datasets
[responses]
   P = 0.0015 ?
[data]
   file f1.txt | conc S =  500
   file f2.txt | conc S = 1000 ?
   file f3.txt | conc S = 2000 ?
   file f4.txt | conc S = 4000 ?
   file f5.txt | conc S = 8000 ?
[end]
```

5.4 Linked concentrations

Two or more concentrations can be *linked* together, meaning that their values are either identical or related through a constant factor. There are two ways to arrange for linking between concentration values:

1. Linking between reaction species names
2. Linking to an arbitrary parameter name

5.4.1 Linking between reaction species names

This type linkage is best explained by way of an example. Let us assume that the *nominal* concentration of reactant **A** appearing in the given reaction mechanism was $[A] = 1.23$ mM. The *actual* concentration of **A** is supposed to be determined from the available experimental data (titration error). Let us also assume that, for some specific reason, the concentration of the reactant **B** is always one fourth of the concentration of **A**. This scenario would be notated in DynaFit as follows:

```
[concentrations]
   A = 1.23 ?
   B = 0.25 * A
```

In general, the notation format is

```
[concentrations]
   SPECIES_X = NUMERICAL_FACTOR * SPECIES_Y
```

This notation must be followed even if the two linked concentrations are supposed to be exactly identical. For example, in the current version of DynaFit it is *not* syntactically valid to write `A = B` if we mean $[A] = [B]$; the correct notation is `A = 1 * B` instead, because the numerical factor (in this case “1”) and the multiplication symbol “*” must always be present.

Illustrative 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 for the *enantiomeric mixture* of both stereoisomers 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, $K_{i(S)} = 10$ nM and $K_{i(R)} = 40$ nM, respectively. If the enzyme is titrated with the enantiomeric mixture, 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 $[(S)I] = [(R)I]$. Such a scenario is described in *Listing 5.6*.

Listing 5.6

```
[mechanism]
   E + S <==> E.S          :   Ks     dissoc
   E.S --> E + P           :   kcat
   E + (S)I <==> E.(S)I    :   KiS    dissoc
   E + (R)I <==> E.(R)I    :   KiR    dissoc

[concentrations]
   E = 0.01
   (R)I = 1 * (S)I         ;  <=== linkage

[data]
   variable (S)I           ;  <=== (R)I is also varied!
   mesh from 0 to 0.1 step 0.01
   ...
   file f1.txt | conc S = 10
   file f2.txt | conc S = 20
   file f3.txt | conc S = 40
   file f4.txt | conc S = 80
```

For a complete working example, see the script file 01.txt located in the directory ./manual/conc/link/species distributed with the program. EXAMPLE SCRIPT

5.4.2 *Linking to an arbitrary parameter*

This relatively complex scenario is best illustrated by way of an example. Let us assume that experimental data files f1 through f4 were obtained at 10 nM *nominal* concentration of the enzyme, $[E]_1 = 10$ nM. However, we wish to determine the *actual* (active site) concentration of the enzyme from the experimental data. Furthermore, we have available data files f5 through f8, in which the *nominal* concentration of the enzyme was twice as high as in the first case, $[E]_2 = 20$ nM. Again, we wish to determine the *actual* enzyme concentration from these four datasets, if possible. To accomplish this task we can use the notation similar to what is shown in *Listing 5.7*.

Listing 5.7

```
[parameters]
   cE1 = 10 ?  ; nM
   cE2 = 20 ?  ; nM

[data]
...

   graph E1

   file f1 | conc E = 1 * cE1
   file f2 | conc E = 1 * cE1
   file f3 | conc E = 1 * cE1
   file f4 | conc E = 1 * cE1

   graph E2

   file f5 | conc E = 1 * cE2
   file f6 | conc E = 1 * cE2
   file f7 | conc E = 1 * cE2
   file f8 | conc E = 1 * cE2
```

In *Listing 5.7* we defined two arbitrary model parameters called `cE1` ("first enzyme concentration") and `cE2` ("second enzyme concentration"). Those parameters are defined in the special `[parameters]` section and, importantly, are both treated as adjustable in the regression model. This is indicated by the presence of the question marks after the numerical values.

Subsequently, in the `[data]` section, we had arranged the eight available data files into two separate groups. In the first group, comprised of experimental data files f1 through f4, the enzyme concentration is set equal to the parameter `cE1`. In the second group, comprised of experimental data files f5 through f8, the enzyme concentration is set equal to the parameter `cE2`. Within both groups of data files, the enzyme concentration will be exactly identical and yet it will also be subject to optimization in the regression analysis.

EXAMPLE SCRIPT

For a complete working example, see the script file 01.txt located in the directory ./manual/conc/link/param distributed with the program.

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 reacting system is always observed by using a specific physical apparatus or instrument. For example, the interacting system might be observed by using one of many experimental techniques enumerated below:

- 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 to emphasize is that *concentrations are never observed directly*. Instead, we always observe a specific physical signal (e.g., absorbance or peak area). Importantly, DynaFit always assumes that the experimentally observed physical signal is related to the concentrations of reactants by a linear relationship.

6.1 Linearity assumption

Specific molar responses are proportionality constants relating concentrations to the observed instrumental response. Molecular species with zero response coefficients need not be listed in the script file. If a species is not mentioned in the `[responses]` section (or after the `response` keyword for local response coefficients) it is assumed that its molar response coefficient is zero.

DynaFit recognizes two fundamentally different types of physical variables that can be observed in any given experiment, namely, extensive physical variables and intensive physical variables.

6.1.1 Extensive physical variables

Experimentally observed value of extensive physical variables are proportional to *concentrations* of molecular species present in the given sample, according to Eqn (6.1), where F is the observable experimental signal; F_0 is the instrument offset ("baseline" signal); n_S is the number of molecular species present in the sample; r_i is the specific molar response coefficient of the ith species; and c_i is the species concentration.

$$F = F_0 + \sum_{i=1}^{n_\mathrm{S}} r_i \, c_i \tag{6.1}$$

Examples of extensive physical variables include fluorescence intensity or NMR peak area.

6.1.2 Intensive physical variables

Experimentally observed value of extensive physical variables are proportional to *mole fractions* of observable molecular species present in the given sample, according to Eqn (6.2). Please note that the summation in the denominator of Eqn (6.2) includes only those molecular species that can be legitimately assigned nonzero specific molar response coefficient.

$$F = F_0 + \frac{\sum_{i=1}^{n_\mathrm{S}} r_i \, c_i}{\sum_{i=1}^{n_\mathrm{S}} \delta_i \, c_i} \tag{6.2}$$

$$\delta_i = \begin{cases} 0 \text{ if } r_i = 0 \\ 1 \text{ if } r_i > 0 \end{cases} \tag{6.3}$$

Examples of intensive physical variables include fluorescence polarization or NMR chemical shift. The following notation specifies in DynaFit scripts that the observable physical quantity is intensive:

```
[responses]
   intensive
   ...
```

The global `[responses]` section of the script must contain the keyword `intensive` standing on a separate line. Intensive and extensive response coefficients cannot be mixed in any given script.

6.1.3 Uniform scaling and concentration units

As was mentioned before in section 4.3, 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.

For example, let us assume that we chose micromolar units throughout the given DynaFit script. Let us further assume that:

- the initial concentration of the reacting species **S** is 3.4×10^{-4} M;
- the bimolecular association rate constant k_{on} has the value 4.5×10^{5} M^{-1}s^{-1};
- the dissociation equilibrium constant K_i has the value 5.6×10^{-9} M;
- the UV/Vis extinction coefficient of the observable species **P** is 6.7×10^{3} M^{-1}cm^{-1}.

In this specific case, after appropriately scaling all relevant quantities to micromolar units, the DynaFit script will contain the following notation:

```
[concentrations]
   S = 340
[constants]
   k(on) = 45
   Ki = 0.0056
[responses]
   0.0067
```

In the specific of the molar response coefficient, if *one mole* per liter of product **P** would give rise to 6.7×10^{3} absorbance units, as is indicated above, then *one micromole* per liter will give rise to an absorbance change one million times lower, i.e., 0.0067 as is shown in the `[responses]` section of the DynaFit code fragment.

6.2 Global response coefficients

Global response coefficients, applicable to all datasets mentioned in the given script file, are listed in the `[responses]` section. The formalism is illustrated in the code snippet below, where A, B, and C are labels for chemical species appearing in the reaction mechanism.

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

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 (extinction coefficient) $\varepsilon = 12,000$ $M^{-1}cm^{-1}$, while the reaction product has practically zero extinction coefficient at the given wavelength. Let us assume that all concentrations in the given script file are expressed in micromolar units. Thus, one μM of substrate corresponds to $12,000 \times 10^{-6} = 0.012$ absorbance units. In this case the DynaFit script will contain the following notation:

```
[mechanism]
   E + S <==> ES  :   kaS   kdS
   ES ---> E + P  :   kdP
[responses]
    S = 0.012
```

In this specific example case it is assumed that the concentration of the substrate is very much larger than the concentration of the enzyme catalyst, so that we can ignore the absorbance due to the Michaelis complex ES.

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  :   kaS   kdS
   ES ---> E + P  :   kdP
```

```
[responses]
   S = +0.0425
   P = -0.0133
[concentrations]
   ...
```

6.3 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 dataset to assign molar response coefficients.

Example 3: 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 dataset (`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 <==> P.DNA.P   :   K2   dissoc
...
[data]
   variable P
   file P-DNA.txt  | response P.DNA   = 1234
   file P2-DNA.txt | response P.DNA.P = 1 * P.DNA
```

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 dataset (zero response coefficient).

6.4 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 4: UV/VIS Spectrophotometry

An enzyme reaction converts the substrate S (molar absorption coefficient ε_S = 1,300 $M^{-1}cm^{-1}$at the given wavelength) to the products P (ε_P = 900 $M^{-1}cm^{-1}$) and Q (ε_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.

```
[task]
   data = progress
   ...
[mechanism]
   E + S <==> ES       :  kaS   kdS
   ES ---> E + P + Q   :  kcat
[responses]
   P = -0.0004 ; response S = 0.0 assumed
[data]
   offset = auto ?
```

In the example above, the keyword `auto` standing next the `offset` in the `[data]` 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.

6.5 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 5: UV/VIS Spectrophotometry

As in the previous example, an enzyme reaction converts the substrate S (molar absorption coefficient ε_S = 1,300 $M^{-1}cm^{-1}$at the given wavelength) to the products P (ε_P = 900 $M^{-1}cm^{-1}$) and Q (ε_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 dataset, are in milliOD per minute. Therefore, we must first multiply by 1000 and then divide by 60 to obtain the correct nominal value of $\Delta\varepsilon$:

```
[task]
   data = rates
   ...
[mechanism]
   E + S <==> ES       :  kaS   kdS
   ES ---> E + P + Q   :  kcat
[responses]
   P = -0.00666 ; = -0.0004 / 60 * 1000
[data]
   file rates.txt  ; second column: milliOD/min
```

6.6 Optimized molar responses

The molar response coefficients can be treated as adjustable parameters. A given response value can be optimized globally or locally. *Global* optimization means that the same best-fit value of an optimized response coefficient applies to all datasets analyzed together, whereas *local* optimization means that the given adjustable response coefficient applies only to the given dataset. An example of a globally optimized response coefficient is shown in *Listing 6.1*.

Listing 6.1

```
[mechanism]
   E + S <==> ES      :      kaS       kdS
   ES ---> E + P      :      kdP
   E + I <==> EI      :      kaI       kdI
   EI <==> EJ         :      kij       kji

[responses]
   P = 3.21 ?         ; <== optimized globally

[data]
   offset -1 ?

   file i000 | conc I =   0
   file i001 | conc I =   1
   file i002 | conc I =   2
   file i004 | conc I =   4
   file i008 | conc I =   8
   file i016 | conc I =  16
   file i032 | conc I =  32
   file i064 | conc I =  64
```

EXAMPLE SCRIPT

For a complete working example, see the script file 01.txt located in the directory ./manual/resp distributed with the program.

6.7 Linked molar responses

Two or more response coefficients can be *linked* together, meaning that their values are either identical or related through a constant factor. There are two ways to arrange for linking between response values:

1. Linking between reaction species names
2. Linking to an arbitrary parameter name

6.7.1 Linking between reaction species names

This type linkage is best explained by way of an example. Let us assume that the *nominal* response coefficient of reactant **A** appearing in the given reaction mechanism was $[A] = 1.23$ arbitrary instrument units per mM. The *actual* response coefficient of **A** is supposed to be determined from the available experimental data. Let us also assume that, for some specific reason, the response coefficient of the reactant **B** is always one fourth of the response of **A**. This scenario would be notated in DynaFit as follows:

```
[responses]
   A = 1.23 ?
   B = 0.25 * A
```

In general, the notation format is

```
[responses]
   SPECIES_X = NUMERICAL_FACTOR * SPECIES_Y
```

This notation must be followed even if the two linked responses are supposed to be exactly identical. For example, in the current version of DynaFit it is *not* syntactically valid to write `A = B` if we mean $\varepsilon_A = \varepsilon_B$; the correct notation is `A = 1 * B` instead, because the numerical factor (in this case “1”) and the multiplication symbol “*” must always be present.

6.7.2 Linking to an arbitrary parameter

This relatively complex scenario is best illustrated by way of an example. Let us assume that experimental data files f1 through f4 were obtained in experiments where the *nominal* molar response of the product P is $\varepsilon_{\mathrm{P},1} = 10$ arbitrary instrument units/μM. However, we wish to determine the *actual* response coefficient from the experimental data. Furthermore, we have available data files f5 through f8, in which the *nominal* response of the product was twice as high as in the first case, $\varepsilon_{\mathrm{P},2} = 20$ arbitrary instrument units/μM. Again, we wish to determine the *actual* response coefficient from these four datasets, if possible. To accomplish this task we can use the notation similar to what is shown in *Listing 6.2*.

Listing 6.2

```
[parameters]
   rP1 = 10 ?  ; a.u./mM
   rP2 = 20 ?  ; a.u./mM

[data]
...

   graph P1

   file f1 | resp P = 1 * rP1
   file f2 | resp P = 1 * rP1
   file f3 | resp P = 1 * rP1
   file f4 | resp P = 1 * rP1

   graph P2

   file f5 | resp P = 1 * rP2
   file f6 | resp P = 1 * rP2
   file f7 | resp P = 1 * rP2
   file f8 | resp P = 1 * rP2
```

In *Listing 6.2* we defined two arbitrary model parameters called `rP1` ("first product response") and `rP2` ("second product response"). Those parameters are defined in the special `[parameters]` section and, importantly, are both treated as adjustable in the regression model. This is indicated by the presence of the question marks after the numerical values.

Subsequently, in the `[data]` section, we had arranged the eight available data files into two separate groups. In the first group, comprised of experimental data files f1 through f4, the product response is set equal to the parameter `rP1`. In the second group, comprised of experimental data files f5 through f8, the product response is set equal to the parameter `rP2`. Within both groups of data files, the product response coefficient will be exactly identical and yet it will also be subject to optimization in the regression analysis.

Chapter 7
Experimental Data

This chapter is focused on the `[data]` section of the DynaFit script files. At the very minimum, the `[data]` section must contain at least one occurrence of the keyword `file` or `set` (see below for a detailed explanation). However, the `[data]` section can potentially contain a diverse list of other keywords, as shown in Table 7.1.

7.1 ASCII text format

All experimental data are represented in DynaFit exclusively in the plain text or ASCII format. One simple test to determine whether or not a data file is in the plain text format is to use a text-editing software to open it.

Examples of text editors include Notepad (Microsoft Windows) or TextEdit (Mac OS X) or gedit (Linux). If a data file is readable by using a plain text or ASCII editor software such as Notepad, it is a text file.

Another good indication that a data file is, in fact, in the appropriate plain text format is the file name extension. For example, if a computer file is named with the extension .txt or .dat, it is very likely to be a plain text file that can be examined using a text editor software.

In converting spreadsheet files, such as Microsoft Excel files, to plain text format it is recommended to follow the following procedure: (1) create a blank plain text file and open it in Notepad; (2) open the Excel file; (3) copy and paste plain text through the system clipboard. Practical experience shows that the *exporting* plain text from software packages such as Microsoft Excel may not always be fully reliable. For example, problems were observed with Excel 2010 running under Windows 7 Ultimate inside a Bootcamp subsystem on a Mac OS X computer. Selecting File ... Save As ... ASCII in Excel 2010 produced extraneous characters.

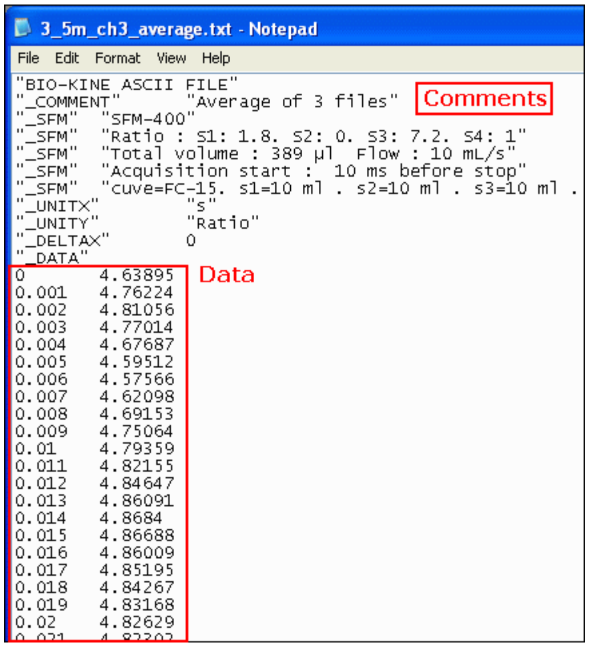
KEYWORD	NOTE
`auto`	Offset is estimated from data
`column`	Used in conjunction with 'sheet'
`concentration`	Local concentrations
`delay`	Mixing delay time
`directory`	Directory holding data files
`equilibrate ..., dilute X`	Concentration jump experiment
`error constant X`	Experimental error
`error constant X percent`	Experimental error
`error data`	Experimental error
`error exponential X Y Z`	Experimental error
`error linear X Y`	Experimental error
`error poisson X`	Experimental error
`error power X Y Z`	Experimental error
`error proportional X`	Experimental error
`error quadratic X Y Z`	Experimental error
`extension`	File name extension
`file`	File name
`graph`	Segregate global data files
`incubate ..., dilute X, time Y`	Concentration jump experiment
`maximum`	Cut-off time value in reaction progress
`mesh from X to Y step Z`	Simulation or interpolation mesh
`mesh logarithmic from X to Y step Z`	Simulation or interpolation mesh
`monitor`	Plot state variables (concentrations)
`offset`	Offset on the signal axis
`parameter`	Special case of algebraic models
`plot logarithmic`	Special plots
`plot mole-fraction`	Special plots
`plot titration`	Special plots
`response`	Local molar responses
`set`	Data set specified inside a script
`sheet`	External file in spreadsheet format
`shift`	Additive constant for progress data
`variable`	Variable concentration

Table 7.1: DynaFit Keywords that can appear in the `[data]` section of the script.

7.1.1 Space-, comma-, and tab-delimited text files

The experimental data must be organized into columns of numbers separated by space, comma, or the tab character. The rarely encountered semicolon-delimited text files (auto-generated by certain scientific instruments) are not readable by DynaFit. An example of an ASCII text file generated by a stopped-flow fluorescence instrument [6] is shown in *Figure 7.1*.

The particular file name generated by the instrument, 3_5m_ch3_average.txt, contains the extension .txt, which suggests that the data file is in plain text format. The file does open in a text editor, and it does contain two columns of numbers separated by the tab character. The first column contains the reaction time in seconds, whereas the second column contains the corresponding fluorescence intensity.



```
"BIO-KINE ASCII FILE"
"_COMMENT"        "Average of 3 files"
"_SFM"   "SFM-400"
"_SFM"   "Ratio : S1: 1.8. S2: 0. S3: 7.2. S4: 1"
"_SFM"   "Total volume : 389 µl  Flow : 10 mL/s"
"_SFM"   "Acquisition start :  10 ms before stop"
"_SFM"   "cuve=FC-15. s1=10 ml . s2=10 ml . s3=10 ml .
"_UNITX"          "s"
"_UNITY"          "Ratio"
"_DELTAX"         0
"_DATA"
0        4.63895
0.001    4.76224
0.002    4.81056
0.003    4.77014
0.004    4.67687
0.005    4.59512
0.006    4.57566
0.007    4.62098
0.008    4.69153
0.009    4.75064
0.01     4.79359
0.011    4.82155
0.012    4.84647
0.013    4.86091
0.014    4.8684
0.015    4.86688
0.016    4.86009
0.017    4.85195
0.018    4.84267
0.019    4.83168
0.02     4.82629
0.021    4.82302
```

Fig. 7.1: Example of an ASCII text file directly readable by DynaFit.

Most data files suitable for analysis by DynaFit are structured similarly to what is shown in *Figure 7.1*.

7.1.2 Comments and annotations

The ASCII data files may optionally contain any number of text lines that serve as comments or annotations. Such comments and annotations are ignored by the software for the purpose of data analysis and serve only as an "electronic notebook" for the benefit of the human reader.

The algorithm used within DynaFit to decide whether or not a given line should be interpreted as a data line or a comment line is as follows. The program ignores any leading white-space characters (blank spaces and tabs) standing at the beginning of the line, and then reads any continuous sequence of non-blank characters (digits, letters, and special characters such as period, plus, or minus). If the given sequence can be interpreted as a valid number, the software will assume that the line represents data as opposed to comments, and vice versa.

For example, the first 11 lines in *Figure 7.1* start with the double quote character (`"`), which cannot legitimately appear in any representation of a numerical value. Therefore the first 11 lines are ignored by DynaFit. The reading of actual data starts on line number 12, because that line starts with the text “0” (zero, a legitimate number).

7.1.3 *Masking individual data points*

The algorithm for separating bona-fide data values from comments and annotations can be used to conveniently exclude a given data point from analysis, while retaining a record of the numerical value that was excluded. This is shown in a code snippet below.

```
initial rates vs. substrate concentration
S,uM   rate

10     0.2404
20     0.3650
30     0.4808
** 40  0.0012    outlier deleted! no enzyme?
50     0.5397
60     0.5348
70     0.5568
80     0.5369
```

In this example, the initial reaction rate corresponding to $[S] = 40\ \mu$M was clearly anomalous, perhaps because the operator omitted to add the enzyme. In such clear-cut cases of gross experimental failures it is legitimate to manually delete the recorded values. The two asterisks standing at the beginning of the “uncommented” line mark the given data point for deletion while maintaining a clear record that something went awry with the experiment.

7.2 External data files

Numerical data can be represented in DynaFit in two different ways, either as external disk files, or as internal blocks of text embedded directly in the DynaFit script. In the first case we need to identify the location of the external file in the computers file system.

7.2.1 *DynaFit startup directory*

The DynaFit *startup directory* is the particular directory or folder, in which the DynaFit executable program itself is located. This can be any directory or folder where the given user has “write privileges” under the operating system constraints. For example, *Figure 7.2* shows that the DynaFit startup directory is c:/Documents and Settings/Petr/My Documents/DynaFit4, because that is where the DynaFit executable file DynaFit.exe is located.

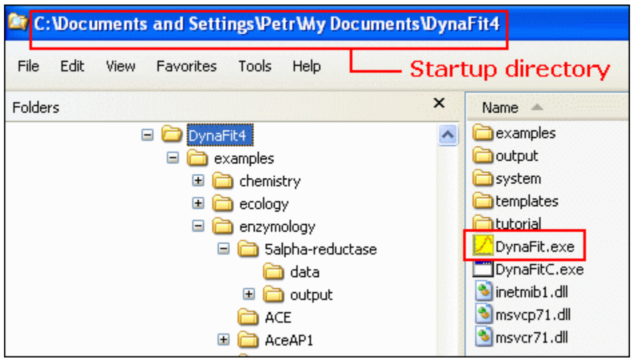


Fig. 7.2: DynaFit startup directory is the directory where the program itself is located.

7.2.2 *Relative and absolute path names*

Within DynaFit scripts the location of data files can be identified in one two ways, either by using absolute path names or by using relative path names. Absolute path names are allowed only for DynaFit users holding the free educational license. Commercial license holders must use relative path names. This constrains the location of DynaFit data files only to the same disk drive, on which the DynaFit binary executable file is located.

Relative path names represent the location of the DynaFit startup directory by the period character (.). Any sub-directory nesting is indicated by forward slashes (/). As an example of a relative path name, the following code snippet identifies a data file located *within* the DynaFit startup directory “.”:

```
[data]
   file ./examples/5alpha-reductase/data/i0.txt
```

More precisely, in this specific instance, DynaFit expects to find a data file named i0.txt located inside the directory data, which is located inside the directory 5alpha-reductase, which is located inside the directory examples, and finally examples is located inside the DynaFit startup directory.

As an example of an absolute path name, the following code snippet identifies a data file named d001.txt which is located on the logical drive X:. This could be a disk drive that is different from the particular disk drive hosting the DynaFit executable program:

```
[data]
   file X:/project/data/2014Apr02/d001.txt
```

Another example of an absolute path name is shown in the code snippet below, where `//DataServer` represents a machine anywhere on the local area network.

```
[data]
   file //DataServer/projectX/2014Apr02/d001.txt
```

7.2.3 Directories, files, and file name extensions

A potentially large group of data files to be analyzed together can be identified conveniently by using the keywords `directory` and `extension`. These keywords must precede the first reference to the actual files (keyword `file`).

For example, let us assume that we wish to subject five separate data files named f01.txt through f05.txt to global statistical analysis[2]. Let us further assume that the data files are located in directory ./inhibition/progress/2014Apr02/data. The analysis can be arranged as is shown in *Listing 7.1*.

Listing 7.1

```
[data]
   file ./assays/2014Apr02/data/f01.txt | concentration I = 0
   file ./assays/2014Apr02/data/f02.txt | concentration I = 1
   file ./assays/2014Apr02/data/f03.txt | concentration I = 2
   file ./assays/2014Apr02/data/f04.txt | concentration I = 4
   file ./assays/2014Apr02/data/f05.txt | concentration I = 8
```

An abbreviated and perhaps more readable and understandable equivalent notation is shown in *Listing 7.2*.

Listing 7.2

```
[data]
   directory ./assays/2014Apr02/data
   extension txt

   file f01 | concentration I = 0
   file f02 | concentration I = 1
   file f03 | concentration I = 2
   file f04 | concentration I = 4
   file f05 | concentration I = 8
```

To generate the full path names for each of the five data files, DynaFit reuses the values of `directory`, as a prefix, and `extension`, as a suffix. The period preceding the file name extension automatically inserted by the program, as is the forward slash separating the directory name and file name.

7.2.4 Revealing file extensions under Windows OS

The Microsoft Windows operating systems ship with a default configuration that purposely hides file extensions of "known file types". The consequence of this is that distinct files that differ only in the "known" file extension will be presented to the user as if they had the same file name. For example, the files MyFile.txt, MyFile.ini, and MyFile.csv will be shown as having presumably an identical file name MyFile, because all three file extensions (.txt, .ini, and .csv) are "known" to the operating system.

This default behavior of the Windows operating systems can cause significant confusion in using DynaFit. It is strongly recommended that all file name extensions are properly revealed. The exact procedure to accomplish this will differ depending on the particular version of MS Windows (WinXP, Win7, Win8, etc.). Under Windows XP, the procedure is as follows:

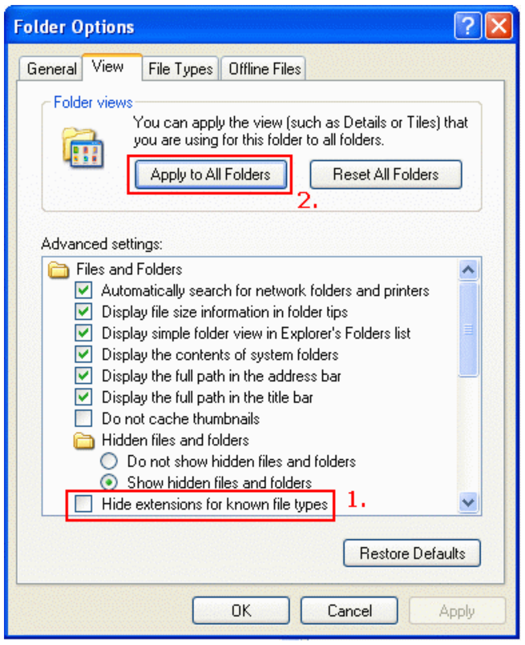
1. Start the **Windows Explorer** program ("Start ... Programs ... Accessories").
2. Select menu **Tools ... Folder Options**.
3. Click on the **View** tab.
4. Uncheck the box **Hide extensions of known file types**
5. Click the **OK** button.
6. Click the **Apply to All Folders** button.

See also *Figure 7.3*. The procedure under Windows 7 and Windows 8 is very similar. If needed please consult a colleague who is well versed in the intricacies of Microsoft Windows operating systems. The goal is to arrive a point where file name extensions such as .txt or .exe are revealed in all directories.

7.3 Independent variables

7.3.1 Concentrations of reactants

In the analysis of complex bio/chemical equilibria or initial enzymatic reaction rates, the `[data]` section must always begin with the keyword `variable`, followed by name of the molecular species that is being treated as the independent variable. The species name must be one of those that appear in the given reaction mechanism, as defined in the `[mechanism]` section. In the illustrative code snippet below the variable molecular species is named **M**.



Folder views

You can apply the view (such as Details or Tiles) that you are using for this folder to all folders.

Apply to All Folders   Reset All Folders

2.

Advanced settings:

Files and Folders
- ☑ Automatically search for network folders and printers
- ☑ Display file size information in folder tips
- ☑ Display simple folder view in Explorer's Folders list
- ☑ Display the contents of system folders
- ☑ Display the full path in the address bar
- ☑ Display the full path in the title bar
- ☐ Do not cache thumbnails

Hidden files and folders
- ○ Do not show hidden files and folders
- ● Show hidden files and folders
- ☐ Hide extensions for known file types 1.

Fig. 7.3 Reveal hidden file extensions for known file types under the MS Windows operating system. Start Windows File Explorer and select "View ... Options" from the main menu, then uncheck the box "Hide extensions of known file types".

```
[task]
   data = equilibria    ;  or 'rates'
...
[data]
   variable M
   file ...
```

In special cases, there can be multiple simultaneously varied reactants. The details are discussed below in section 7.5.1.2.

7.3.2 *Time as implied variable*

In the analysis of the reaction progress, the `variable` keyword is omitted, because DynaFit will automatically assume that the independent variable is the reaction time appearing in the first column of each particular data file.

```
[task]
   data = progress
...
[data]
   file ...
```

7.3.2.1 Mixing delay time

Very often the first recorded time point is artificially shifted on the time axis, due to the mixing delay time.

Let us consider a realistic scenario from a biochemical laboratory.

Assume that the investigator might have initiated an enzyme reaction by adding an aliquot of the enzyme stock solution to a substrate solution. At the very instant the enzyme and substrate solutions are mixed, the enzyme reaction commenced. Now let us assume that after mixing all reactants the reaction vessel (such as a 96-well microplate) was placed in a mixing device (such as a 96-well microplate shaker) to achieve complete homogeneity of the reacting solution. Let us assume that the duration of the mixing period was 60 seconds. Only after the 60 second mixing period elapsed, the microplate was placed into a recording instrument (such as a 96-well plate reader) and the changes in fluorescence intensity were monitored over time.

It is very important to realize that in this hypothetical scenario the plate-reader will have recorded all time values with a 60 second systematic error. In particular, whereas the digital record produced by the plate reader might contain a series of time-point values such as t = 10, 20, 30, 40, ... seconds, in fact the first recorded time point (nominally t = 10 sec) was recorded when the reaction was already proceeding for 70 seconds: 10 seconds in the plate reader, plus 60 seconds previously in the plate shaker. Similarly, where the plate-reader recorded t = 20 seconds, the reaction was already proceeding for 80 seconds, and so on.

One obvious way to account for the mixing delay time would be to manually correct all time values, produced by the recording instrument, and only then submit the corrected data for analysis by DynaFit. Another equivalent method is to utilize the keyword `delay` in the DynaFit script. If we wish to apply an identical mixing delay correction to all data sets analyzed simultaneously, the general pattern is shown in the code snippet below, where `T` stands for the mixing delay time in suitably chosen units.

```
[task]
   data = progress
...
[data]
   delay T

   file ...
   file ...
   file ...
```

Please note that the keyword `delay` appears *before* the first occurrence of the keyword `file`. On the other hand, if we wished to make separate corrections to individual data file, the general pattern is shown in the code snippet below, where `T1`, `T2` and so on stand for the differing mixing delay times in suitably chosen units. In this case the `delay` keyword is placed *after* a particular occurrence of the keyword `file`.

```
[task]
```

```
   data = progress
...
[data]

   file ... | delay T1
   file ... | delay T2
   file ... | delay T3
```

The mixing delay time can optionally be treated as an adjustable model parameter. This is achieved by placing a question mark after the estimated numerical value:

```
[task]
   data = progress
...
[data]
   delay 60 ?
...
```

However, this feature of DynaFit is relatively untested. The user is strongly encouraged to exercise caution and always use common sense to check the plausibility of any "best-fit" values of the mixing delay time. Optimization of the mixing delay time should be attempted only as a last-resort, in those specific cases where there is independent evidence that the mixing delay time might have been recorded incorrectly and where the goodness of fit can be significantly improved.

7.4 Internal data sets

As an alternative to external data files, experimental data can be defined directly within a particular DynaFit script, by using the keyword `set` and the `[set:...]` section. The general syntax is shown in the code snippet below, where `LABEL` is an arbitrary data set label.

```
[data]
   set LABEL
...
[set:LABEL]
...
... numerical data block
...
```

An illustrative example is shown in *Listing 7.3*. In this case, the arbitrarily chosen labels `16U11a` and `16U11b` are used to identify two sets of equilibrium binding data to be analyzed in a global fashion [10].

Listing 7.3

```
[task]
   data  = equilibria
   task  = fit

[mechanism]
   E + D <==> ED      :  Kd1    dissoc
   ED + D <==> EDD    :  Kd2    dissoc

[constants]
   Kd1 = 0.05 ?, Kd2 = 1 ?

[concentrations]
   E = 0.5

[responses]
   intensive
   E   = 0.18 ?, ED  = 0.22 ?, EDD = 0.28 ?

[data]
   variable     D
   plot         logarithmic
   set          16U11a
   set          16U11b

[set:16U11a] ;---------------------------------------
D,uM    anisotropy

0.010   0.18554
0.025   0.18197
0.050   0.18631
...
...
1000.   0.24890
2500.   0.26131
5000.   0.26376

[set:16U11b] ;---------------------------------------
D,uM    anisotropy

 0.10   0.18561
 0.25   0.18933
 0.50   0.19280
...
...
5000.   0.27113
7500.   0.27234
10000   0.27444

[end]
```

For a complete working example, see the script file 01.txtlocated in the directory ./manual/data/intern distributed with the program. Dr. Alex Drohat (University of

EXAMPLE SCRIPT

Maryland) is gratefully acknowledged for providing the data originally published as part of Figure 8 in ref. [10].

7.5 Data organization

The experimental data can be organized for DynaFit processing in two different ways. Each line of input text can represent either a single experimental data point, or it can represent multiple data points in a spreadsheet format, as is described below.

7.5.1 One data point per line

The most typical representation of experimental data in DynaFit is the classic two-column format. The first column represents the independent variable, for example reaction time in suitably chosen units (minutes or seconds). The second column represents the observed physical quantity, for example fluorescence intensity or UV/Vis absorbance. This particular arrangement of experimental, illustrated in the code snippet below, is used in the analysis of the reaction progress.

```
[task]
   data = progress
...
[set:time-vs-absorbance]
t,s   A

  1   0.009
  2   0.023
...   .....
100   0.987
```

In the analysis of initial reaction rates (enzyme kinetics), or in the analysis of binding equilibria (biophysics), the independent variable is the total or analytic concentration of a particular reactant. The numerical value of this particular reactant's concentration is then listed in the first column of the data file.

7.5.1.1 Single variable reactant

In most initial enzymatic rate experiments, as well as in most biophysical binding experiments, only one particular reactant's concentration is varied while all other reactants' concentrations are held fixed. For example, in the study of enzyme inhibition mechanisms, it is a common practice to vary the substrate concentration while keeping the inhibitor concentration fixed at various different levels. In this particular case the experimental data set will contain either two columns or three columns.

In the two-column format, the first column always contains the variable reactant concentration, while the second column contains the experimentally observed value of whatever physical quantity that was being measured. If we also know the uncertainty of each measurement, for example the standard deviation from replicated measurements, then this uncertainty is entered in the optional third column.

Listing 7.4 illustrates how to arrange the experimental data from a series of enzyme kinetics experiments, in which the substrate concentration was varied (`variable S`) while the inhibitor concentration was kept fixed at various levels.

Listing 7.4

```
[task]
   task   = fit
   data   = rates
   approx = rapid-equilibrium

[mechanism]
   E + S <===> ES       :  Ks    dissoc
   ES + S <===> ESS     :  Ks2   dissoc
   ES ---> E + P        :  kcat

   E + I <===> EI       :  Ki    dissoc
   ES + I <===> ESI     :  Kis   dissoc
   ESI ---> E + P       :  kcatp

...

[data]
   variable      S

   set i-0 | concentration I = 0
   set i-1 | concentration I = 22
   set i-2 | concentration I = 44
   set i-4 | concentration I = 88
...

;-------------------------------------------------------

[set:i-0]

I=0 uM   initial rate (mean and std.dev., n = 3)
S,uM   mean   std.dev.

10   0.6466   0.1104
20   0.8069   0.0031
30   0.8529   0.0182
40   0.8351   0.0234
50   0.8635   0.0788
60   0.7571   0.1134
70   0.8282   0.1075
```

```
80   0.7524   0.0406

[set:i-1]

I=22 uM   initial rate (mean and std.dev., n = 3)
S,uM   mean   std.dev.
10   0.4156   0.0098
20   0.5671   0.0005
30   0.6046   0.0532
40   0.6102   0.0843
50   0.5940   0.0467
60   0.5480   0.0380
70   0.5915   0.0080
80   0.5723   0.0431

[set:i-2]

I=44 uM   initial rate (mean and std.dev., n = 3)
S,uM   mean   std.dev.
10   0.3348   0.0172
20   0.4501   0.0120
30   0.4658   0.0401
40   0.4561   0.0532
50   0.5037   0.0724
60   0.4780   0.0192
70   0.4631   0.0008
80   0.4183   0.0204

[set:i-4]

I=88 uM   initial rate (mean and std.dev., n = 3)
S,uM   mean   std.dev.
10   0.1925   0.0011
20   0.3163   0.0047
30   0.3644   0.0153
40   0.3688   0.0212
50   0.3820   0.0072
60   0.3888   0.0113
70   0.3865   0.0104
80   0.4095   0.0137

[end]
```

Each data point in *Listing 7.4* is represented by three numerical values listed in a single line of input text. The first numerical value is the variable substrate concentration, in micromoles per liter. The second value is the observed initial rate, in absorbance units per second, computed as an average from three replicated measurements. The third column is the associated standard deviation from replicates.

7.5.1.2 Multiple variable reactants

In certain specific instances the experiment might involve simultaneous variation of more then one reagent concentrations. DynaFit can accommodate any number of such variable species. The generic notation is illustrated in the code snippet below, where we assume that there were three simultaneously varied reactants, namely, the enzyme (E), the substrate (S), and the inhibitor (I).

```
[data]
   variable    E, S, I
   set         multivar

[set:multivar]

E,uM    S,uM    I,uM   rate

0.1     10      0      1.23
0.2     10      0      2.34
0.1     20      0      3.45
0.1     10      1      1.23
0.2     10      1      2.34
0.1     20      1      3.45
...
...
```

The `variable` line lists the names of simultaneous varied reactants. The species names must match those appearing in the `[mechanism]` section. For N simultaneously varied molecular species, the data will contain either $N+1$ or $N+2$ columns. The first N columns in the data file will contain the concentrations of variable molecular species. The $(N+1)$th column will contain the observed experimental signal. The optional $(N+2)$th column, if any is present, will contain the associated standard error from replicated measurements.

A realistic example taken from ref. [5] is shown in *Listing 7.5*, where **P** for "protein" is the cytochrome P450 enzyme, isoform E1, and **L** for "ligand" stands for cytochrome P450 reductase, or CPR. The enzymatic activity of 1:1 and 1:2 protein–ligand (more precisely, protein–protein) complexes was observed in a kinetic assay. The purpose of the experiment was to determine the stoichiometry of protein–protein binding.

Listing 7.5

```
[task]
   data = equilibria
   task = fit

[mechanism]
   P + L <===> P.L         :     Kd1   dissoc
   L + P.L <===> L.P.L     :     Kd2   dissoc
...

[data]
    variable  P, L
    plot      mole-fraction
    set       job04avg
...

[set:job04avg] ; Job plot - constant (P + L) = 0.4 uM

P, uM   L, uM   rate     std.err

0.010   0.390   0.0697   0.0070
0.020   0.380   0.1919   0.0062
0.030   0.370   0.2639   0.0114
0.045   0.355   0.4728   0.0033
...     ...     ...      ...
...     ...     ...      ...
0.320   0.080   0.4728   0.0064
0.340   0.060   0.3287   0.0171
0.360   0.040   0.2233   0.0392
0.380   0.020   0.0724   0.0058

[end]
```

For a complete working example, see the script file 01.txt located in the directory ./manual/data/multi distributed with the program. See ref. [5] for experimental details. The data set shown here corresponds to Figure 3 on page 10196 of the original journal article. See also *Figure 7.4*.

EXAMPLE SCRIPT

7.5.2 Spreadsheet format

DynaFit can process external data files organized in the spreadsheet format. This option is available only for experiments involving only a single independent variable.

7.5.2.1 Types of independent variables

In the analysis of reaction progress `data = progress` the independent variable is the reaction time. In that case the `[data]` section of the script does not need to contain the `variable` keyword. The general notation for identifying a reaction progress data set is shown in the code snippet below.

```
[task]
   data = progress
```

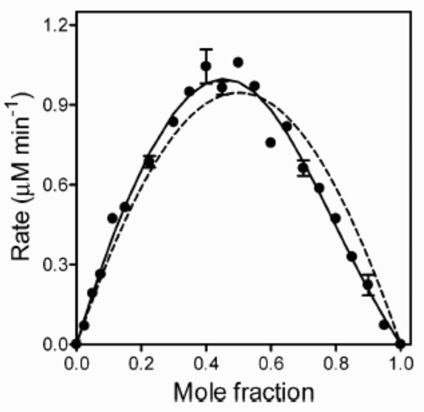


FIGURE 3: Job plot at a total protein concentration of 400 nM. The mole fraction (*x*) is defined by [CYP2E1]/([CYP2E1] + [CPR-K56Q]). Reaction rates were measured while CPR-K56Q and CYP2E1 concentrations were varied such that the total protein concentration remained at 400 nM. These data were fit to two different models using DynaFit (*23*). The dashed curve represents the best least-squares fit to the binary complex model (Scheme 1). The solid curve represents the fit to a model possessing a binary (P450•CPR) and ternary [P450•$(CPR)_2$] complex.

Fig. 7.4: DynaFit analysis produced by using the scrip shown in *Listing 7.5*.

```
...
[data]
   sheet  FILENAME
   column N         ; where N = 2, 3, 4, etc.
```

The above code fragment signifies that the spreadsheet file named next to the `sheet` keyword will contain at least N+1 columns. The first column must contain the reaction time, in suitably chose units. The Nth column (specified by the input `column N`) will contain the experimental data values. Naturally, the lowest possible column number is 2.

In the analysis of biophysical equilibria or initial enzymatic initial rates, the general notation patter in shown in the code snippet below. Please note the appearance of the keyword `variable`.

```
[task]
   data = rates    ; or 'equilibria'
...
[data]
   variable M
   sheet    FILENAME
   column   N         ; where N = 2, 3, 4, etc.
```

As before in the case of progress curves, the above code fragment signifies that the spreadsheet file named next to the `sheet` keyword will contain at least N+1 columns. The first column must contain the total or analytic concentration of the molecular species **M**, in suitably chose concentration units. The Nth column (specified by the input `column N`) will contain the experimental data values.

7.5.2.2 Independent variable column

By default the independent variable (e.g., time in the analysis of reaction progress) is assumed to be located in the first column of the external spreadsheet file. If so, the script does not have to explicitly specify that the first column holds the independent variable. Thus, for example, the code fragment below implies that the independent variable is located in column No. 1 for all three data sets, No. 1–3:

```
[data]
   sheet    FILENAME
   column   2      ; data set #1
   column   3      ; data set #2
   column   4      ; data set #3
   etc.
```

However, it is also possible to specify a the independent variable column separately for each individual data set. This done by using the notation exemplified in the code snippet below, using the colon (`:`) to separate pairs of dependent and dependent variables:

```
[data]
   sheet    FILENAME
   column   1:2      ; data set #1
   column   3:4      ; data set #2
   column   5:6      ; data set #3
   etc.
```

The above notation signifies that the independent variable (“X”) for data sets No. 1, 2, and 3 is located in columns No. 1, 3, and 5, respectively. The corresponding dependent variable (“Y”) is located in columns No. 2, 4, and 6, respectively.

7.5.2.3 Multiple columns

It is possible to merge the contents of multiple columns to compose a data set. The need for merging columns arises when handling independent replicates of a particular experiment. Merging is accomplished by listing multiple comma-separated column numbers after the `column` keyword, as follows:

```
[data]
```

```
   sheet    FILENAME
   column   2,3,4      ; data set #1
   column   5,6,7      ; data set #2
   column   8,9,10     ; data set #3
   etc.
```

In this example, the data set No. 1 will be created by merging the contents of columns No. 2-4. Similarly, data set No. 2 will be created by merging the contents of columns No. 5-7, and so on. It is assumed that the independent variable is in column No. 1.

When multiple columns are merged, it is possible to preserve each individual data point as a separate entity, or alternately it is possible to automatically compute averages and standard deviations from replicates. In the latter case DynaFit will analyze the averages. Averaging of merged columns can be accomplished by inserting the following notation in the DynaFit script:

```
[settings]
{Filter}
   AverageReplicates = yes
```

If replicates are automatically averaged, DynaFit will display the size of the computed error bar (i.e., the standard error) in the graphical and numerical output files.

7.5.2.4 Experimental error column

The size of the error bar associated with each particular data point can also be entered explicitly in the input spreadsheet. This is accomplished by using the keyword `error` followed by the corresponding column number. Consider the following example:

```
[data]
   sheet    FILENAME
   column   1:2 error 3      ; data set #1
   column   4:5 error 6      ; data set #2
   column   7:8 error 9      ; data set #3
   ...
```

In this case the independent variable (e.g. the reaction time) for data set No. 1 is located in column No.1, the corresponding dependent variable (e.g. the observed change in fluorescence intensity over time) are located in column No. 2, and the experimental error associated with each data point is located in column No. 3. Similarly for data sets No. 2 and 3.

The experimental errors are usually used only for the purpose of graphically displaying the uncertainty of each individual data point. However, we can optionally perform *weighted regression* by specifying the the specified experimental error should be used as a weighting factor:

```
[data]
   error    data
   ...
```

7.5.2.5 Tab-, space-, or comma-delimited files

Tab- and space-delimited spreadsheet style text files have to contain the same number of items in each row and column. In other words, the rectangular data table or matrix must be completely filled, with no missing values in any row.

In contrast, comma-separated spreadsheet style text files are allowed to contain missing data points or "uncommented" data points (i.e., entries that cannot be interpreted as a valid number while reading from left to right). This implies that comma-separated data files are allowed to contain an unequal number of items in individual columns.

Given this flexibility of the comma-separated text files, it is highly recommended that DynaFit users do utilize this type of input preferentially. A major advantage is that all major spreadsheet software packages (e.g. Excel, Open Office, Libre Office, etc.) do allow convenient export of the experimental data as CSV ("Comma Separated Values") files.

The recommended procedure for submitting data files for analysis by DynaFit is as follows:

1. Create a spreadsheet file in Excel or a similar program.
2. Save a given sheet as a CSV file.
3. Name the CSV file in the `[data]` section of a DynaFit script.
4. Identify the dependent and independent variables using the `column` keyword.

7.6 Local and global fit

The term *global fit* [2] refers to the particular kind of regression analysis, in which certain model parameters are determined on the basis of multiple data sets combined together and analyzed as one superset of pooled experimental data. In this manual we will use the term *local fit*, or local optimization, to refer to the opposite scenario, in which certain model are optimized such that their best-fit values apply only to one particular data set in the global superset of pooled data. Importantly, a particular organization of the `[data]` section of DynaFit scripts can be used to arrange either for a global fit, or for a local fit, of optimized model parameters. The details are explained in sections 5.2 and 6.2.

7.7 Weighting

The objective function in least-squares minimization is defined in DynaFit as shown in Eqn (7.1), where S is the weighted residual sum of squares; w_i is the weighting factor for the ith data point; n_D is the total number of data points; f_i is the ith data value; and $\hat{f}_i$ is the corresponding theoretical model value.

$$S = \sum_{i=1}^{n_\mathrm{D}} w_i \, \left(f_i - \hat{f}_i\right)^2 \tag{7.1}$$

By default, DynaFit always assigns unit weights to all data points ($w_i = 1$ for $i = 1, 2, \ldots, n_\mathrm{D}$), which effectively results in *unweighted regression*. Unweighted regression is appropriate in those cases where there is a sufficient reason to believe that experimental errors associated with all individual data points are approximately equal in magnitude, or, much more frequently, whenever there is no information available about the possible distribution of experimental errors.

However, in certain cases we do have good information available about the statistical distribution of experimental errors [8, 9]. In those cases it is prudent to perform weighted regression by specifying a particular form of the *error function*, which links the magnitude of the experimental error to the magnitude of the experimental signal.

The weighting coefficients appearing in Eqn (7.1) are normalized such that the sum of all weights adds up to the number of data points, according to Eqn (7.2). This normalization is accomplished according to Eqn (7.3), where e_i is the error function evaluated for the ith data point, according to one of the methods described below.

$$\sum_{i=1}^{n_\mathrm{D}} w_i = n_\mathrm{D} \tag{7.2}$$

$$w_i = n_\mathrm{D} \, \frac{1/e_i^2}{\sum_{i=1}^{n_\mathrm{D}} 1/e_i^2} \tag{7.3}$$

7.7.1 Non-constant variance error functions

The type of error function that will be used to assign experimental errors to each data point is given the keyword `error` followed by one of the associated keywords, such as `linear` or `exponential`, which in turn are followed by numerical values as is described below.

7.7.1.1 Linear error function

```
[data]
   error linear X Y
```

The above code fragment signifies that DynaFit will use as error function Eqn (7.4), where the constants X and Y by specified by the space-delimited encoding `X Y`.

$$e_i = X + Y\, f_i \tag{7.4}$$

A number of enzymatic initial rate datasets with varied substrate concentration appears to conform to this error distribution model.

7.7.1.2 Quadratic error function

```
[data]
   error quadratic X Y Z
```

The above code fragment signifies that DynaFit will use as error function Eqn (7.5), where the constants X through Z by specified by the space-delimited encoding `X Y Z`.

$$e = X + Y\, f + Z\, f^2 \tag{7.5}$$

Mannervik *et al.* found that the error distribution a large experimental data set derived from initial rate enzymatic studies conformed equally well to this polynomial model and to the power function described below.

7.7.1.3 Exponential error function

```
[data]
   error exponential X Y Z
```

The above code fragment signifies that DynaFit will use as error function Eqn (7.6), where the constants X through Z by specified by the space-delimited encoding `X Y Z`.

$$e = X + Y\, \exp(Z\, f) \tag{7.6}$$

The exponential distribution of experimental error is an alternative to the polynomial (quadratic) distribution and also to the exponential distribution.

7.7.1.4 Power error function

```
[data]
   error power X Y Z
```

The above code fragment signifies that DynaFit will use as error function Eqn (7.7), where the constants X through Z by specified by the space-delimited encoding `X Y Z`.

$$e = X + Y\, f^{Z} \tag{7.7}$$

The power error function is one of the earliest experimentally verified models for the distribution of experimental errors in initial rate enzyme kinetic studies. In particular, Mannervik *et al.* [1, 4] found that values of Z ranging from 0.8 to 1.0 were applicable to their particular system, depending on the given experimental conditions. The value of $Z = 1$ corresponds to constant relative error.

7.7.2 Weighting by experimental error from replicates

```
[task]
   task = fit
...
[data]
   error data
   file  ...
...
```

The code fragment above, including the special designation `error data`, signifies that DynaFit will user-supplied error values embedded in the data file. In this case the data file must contain an extra column containing the experimental uncertainty. Typically this will be the standard deviation from replicated measurements.

However, one must exercise a great deal of caution in utilizing this particular feature. It can only be recommended for those special circumstances where the number of replicates is greater than five. This rule of thumb has been well document in the literature [12, 1]. However, typical biochemical or biophysical experiments include at most $n = 3$ replicates (i.e. triplicates). Unfortunately with $n \leq 3$ replication, it can easily happen by accident that a particular replicated data point will be assigned an exceedingly small uncertainty. As a consequence, this seemingly very "precise" data point would unduly sway the results of the regression analysis.

Weighting by experimentally determined standard deviation from replicates should be strictly avoided unless there exist at least **five replicates** *for each individual data point.*

7.7.3 *Constant variance*

As a special case of constant weighting comes into play in DynaFit simulations. Constant (i.e. identical size) experimental error, normally distributed, can be added to simulated data points by using the encoding `error constant`. There are two methods to simulate constant noise, setting either the absolute magnitude or the relative magnitude of the simulated error bars.

```
[task]
   task = simulate
...
[data]
   error constant X
...
```

The above code fragment signifies that DynaFit will simulate all data points with an added pseudo-random noise distributed according to the Normal or Gaussian distribution with zero mean and the standard deviation equal to `X` in absolute value. In contrast, the code fragment below signifies that the standard deviation of the Normally distributed pseudo-random noise will be equal to `X` percent of the largest simulated signal value.

```
[task]
   task = simulate
...
[data]
   error constant X percent
...
```

Simulated data with superimposed pseudo-random noise are very useful in numerical "experiments" focused on identifiability analysis and model discrimination analysis.

7.8 Offset on the signal axis

As was pointed out in Chapter 6, DynaFit constructs the fitting model either according to Eqn (6.1) for extensive physical variables or according to Eqn (6.2) for intensive physical variables. The quantity F_0 appearing in either of these equations is the *instrument baseline*. It is the contribution to the overall observed experimental signal that is not a property of the sample but it is the property of the instrument (a "baseline" signal).

In most cases the baseline signal values needs to be treated as an adjustable model parameter. There are two ways to proceed with the optimization, treating the baseline offset either as a globally optimized model parameter applicable to a global superset of multiple combined data files, or as a locally optimized parameter, specific to a particular data set.

In order to determine the best-fit value of the instrument baseline that is presumably applicable to all data sets analyzed simultaneously, the general DynaFit scripting pattern to be utilized is shown in the code snippet below, where `X` stands for the baseline offset in suitably chosen instrument units.

```
[data]
   offset X ?

   file ...
   file ...
   file ...
```

Please note that the keyword `offset` appears *before* the first occurrence of the keyword `file`. On the other hand, if we wished to determine the best-fit value of baseline offsets that are specific to each individual data file, the general pattern is shown in the code snippet below, where `X1`, `X2` and so on stand for the differing baseline values in suitably chosen units. In this case the `offset` keyword is placed *after* a particular occurrence of the keyword `file`.

```
[data]

   file ... | offset X1 ?
   file ... | offset X2 ?
   file ... | offset X3 ?
```

Very often, but not always, the most suitable initial estimate for the adjustable baseline is the experimental signal recorded as the first time point in each progress curve being analyzed. Under those circumstances we can use the special notation `offset auto` as shown immediately below.

```
[data]

   file ... | offset auto ?
   file ... | offset auto ?
   file ... | offset auto ?
```

7.9 Concentration jump experiments

With DynaFit we can analyze two distinct types of "concentration jump" experiment. In the first kind the reaction mixture is brought to full equilibrium before the final ingredient is added to trigger the kinetic phase of the experiment. In the second type of experiment (the "double jump" experiment) certain components are pre-incubated for a specific amount of time, without necessarily achieving full equilibrium. After the specified amount of time elapses, the final component is added to trigger the kinetic experiment proper.

7.9.1 Equilibration ("single jump")

In certain kinetic experiments it is beneficial (or, at times, even necessary) to first pre-incubate certain interacting components, until full equilibrium is reached. At that stage an additional reagent is introduced to trigger the dynamic phase of the experiment. The general notation for this type of experiment is shown in the code fragment below.

```
[task]
   data = progress
...
[data]
   file ...
      equilibrate ... , dilute ...
      concentration ...
```

For example, in the study of "time dependent" enzyme inhibitors, we often pre-incubate the inhibitor with the enzyme until full equilibrium is reached. Only then the substrate is added to trigger the enzymatic assay. To clarify the general notation listed above, let us assume that during the pre-incubation phase the concentration of the enzyme was $[E]$ = 0.2 μMand the concentration of the inhibitor was $[I]$ = 0.25 μM. To signify this fact, we will use the notation `E = 0.20, I = 0.25` following the `equilibrate` keyword:

```
   equilibrate E = 0.2, I = 0.25, dilute ...
```

Now let us assume that as the substrate solution is added to the pre-incubation mixture, the total volume increases five fold. For example, in a 100 μLplate-reader format, we could have added 80 μLof a substrate stock solution to 20 μLof the incubation mixture. In this hypothetical case the *final* concentrations of the enzyme and the inhibitor will be five fold lower (i.e., 20%) than before substrate addition. This will be represented in the DynaFit script by using "0.2" as the dilution factor:

```
   equilibrate E = 0.2, I = 0.25, dilute 0.2
```

Finally, let us assume that final concentration of substrate in the reaction mixture was 8 μM. This is the numerical value we will place after the keyword `concentration` on a separate line:

```
[data]
   ...
   file F1
```

```
      equilibrate E = 0.2, I = 0.25, dilute 0.2
      concentration S = 8
```

To review and summarize, the above DynaFit notation means that the data file F1.txt originated in an experiment organized as follows:

- the concentrations of the enzyme and the inhibitor *before* dilution (i.e., during the pre-incubation phase) were 0.2 μM and 0.25 μM, respectively;
- the concentration of substrate *after* dilution (i.e., during the kinetic phase proper) was 8 μM; and that
- the preincubated sample was diluted five fold (1/5 = 0.2) upon the addition of the substrate as the last component.

7.9.2 Incubation for a specific time ("double jump")

The notation for the "double jump" experiment is very similar to the previous case of the "single jump" experiment, with two exceptions. First, the keyword `equilibrate` is replaced with `incubate`. Second, the dilution factor is followed by the notation `time X` where **X** is the preincubation time in suitably chosen units (e.g., seconds). The general pattern is illustrated in the code fragment below.

```
[task]
   data = progress
...
[data]
   file ...
      incubate ... , dilute ... , time X
      concentration ...
```

7.10 Simulations

DynaFit can be profitably used not only for least-squares data fitting, but also for heuristic simulations. In that case we have to somehow specify the layout of the data points. This is done by using the keyword `mesh`. Often we wish to simulate not the mathematically pure model curve, but rather we wish to generate quasi-experimental data with superimposed pseudo-random noise. This is accomplished in DynaFit by using the keyword `error`.

7.10.1 Spacing and scaling

```
[task]
```

```
   task = simulate
...
[data]
   mesh from X to Y step Z
```

The above generalized code fragment signifies that DynaFit will simulate the independent variable at values starting from **X**, ending with **Y**, and stepping by an *additive* increment equal to **Z**. The values of **X**, **Y** and **Z** have to be positive numbers. For example the notation below will simulate a reaction progress curve with time-points placed at t = 0, 30, 60, 90, ..., 1740, 1770, 1800 seconds.

```
[task]
   task = simulate
   data = progress
...
[data]
   mesh from 0 to 1800 step 30
...
```

It is also possible to specify a logarithmically spaced mesh of values for the independent variable. The generalized notation is shown below:

```
[task]
   task = simulate
...
[data]
   mesh logarithmic from X to Y step Z
```

The above generalized code fragment signifies that DynaFit will simulate the independent variable at values starting from **X**, ending with **Y**, and stepping by an *multiplicative* increment equal to **Z**. For example the notation below will simulate the equilibrium binding experiment with the concentration of the protein **P** placed at $[P]$ = 0.1, 0.2, 0.4, 0.8, 1.6, 3.2, 6.4, and 12.8 μM. In this case the independent variable is rising by a factor of two (`step 2`) between each two successive values of the protein concentration.

```
[task]
   task = simulate
   data = equilibria
...
[data]
   variable P
   mesh logarithmic from 0.1 to 12.8 step 2
...
```

In certain special cases we might wish to achieve irregular spacing of simulated data point on the X-axis (i.e. the independent variable axis). For example, we might wish to simulate an stopped-flow experiment where the spacing between adjacent time points change abruptly (perhaps by an order of magnitude) at some point in the simulated experiment. In those specific cases we can prepare a named text file, from which the program will read the values of independent variable to be used in the simulation. The special notation is `mesh file`:

```
[task]
   task = simulate
...
[data]
   ...
   mesh file
   file F1
...
```

In the above example the file F1.txt will be utilized both as input for the program (to read the value of the independent variable form the first or only column stored in the file) and also as the program's output. The special notation `mesh file` is available for the simulation all data types, either reaction progress curves, or enzymatic initial rates, or biophysical equilibria.

7.10.2 Experimental error

In DynaFit simulations the standard deviation of the normally distributed pseudo-random noise can be either constant, as is described in section 7.7.3, or it could be made dependent on the simulated values of the idealized (i.e., noise-free) model curve. In the latter case we can use the same notation that is used to specify the presumed distribution of experimental error for the purpose of nonconstant weights in nonlinear regression. A variety of nonconstant error functions are describe in section 7.7.1.

For example, if we wish to simulate an initial rate data set that conforms to the quadratic polynomial distribution of errors [1], we would specify the quadratic polynomial coefficients by using the following notation:

```
[task]
   task = simulate
   data = rates
...
[data]
   variable S
   ...
   mesh logarithmic from 8 to 256 step 2
   error quadratic 0.0003 0.02 0.001
   file F1
...
```

7.11 Plotting

The keywords `graph`, `plot` and `monitor` can be used to generate several kinds of useful graphical output.

7.11.1 Logarithmic plot

```
[data]
   plot logarithmic
```

The code fragment above signifies that DynaFit will produce a plot of the best-fit model function, superimposed on the experimental data, such that the horizontal axis has logarithmic scaling. It is required that neither the experimental data set nor the best-fit model function include a point with X-coordinate equal to zero. Only positive numerical values can be plotted on a logarithmically scaled axis.

7.11.2 Mole fraction plot

In certain equilibrium binding studies it may be advantageous to maintain a constant *total* concentration of interacting components, while their molar ratio is varied. The results of such experiments are conveniently presented such that the abscissa displays the mole fraction of the first varied component. The resulting graph is called a "Job plot" according to its inventor [3]. DynaFit allows automatic construction of mole fraction plots by inserting the line `plot mole-fraction` into the `[data]` section of the input script.

```
[data]
   plot mole-fraction
```

For a complete working example, see the script file 01.txt located in the directory ./manual/data/multi distributed with the program. An excerpt is shown in *Listing 7.6*.

EXAMPLE SCRIPT

Listing 7.6

```
[data]
    variable  P, L
    plot      mole-fraction
    set       job04avg
...

[set:job04avg] ; Job plot - constant (P + L) = 0.4 uM

P, uM   L, uM   rate     std.err
```

```
0.010   0.390   0.0697   0.0070
0.020   0.380   0.1919   0.0062
0.030   0.370   0.2639   0.0114
0.045   0.355   0.4728   0.0033
0.060   0.340   0.5147   0.0144
0.090   0.310   0.6868   0.0216
0.120   0.280   0.8358   0.0004
0.140   0.260   0.9492   0.0031
0.160   0.240   1.0444   0.0645
0.200   0.200   1.0590   0.0071
0.220   0.180   0.9989   0.0412
0.240   0.160   0.7576   0.0196
0.260   0.140   0.8164   0.0021
0.280   0.120   0.6625   0.0288
0.300   0.100   0.5863   0.0086
0.320   0.080   0.4728   0.0064
0.340   0.060   0.3287   0.0171
0.360   0.040   0.2233   0.0392
0.380   0.020   0.0724   0.0058
```

In this experiment the protein **P** and ligand **L** concentrations were varied simultaneously such that the sum of both concentrations remained constant at $[P]+[L]=0.4\ \mu$M. This is shown in the first two columns of the experimental data block in *Listing 7.6*. DynaFit was used to fit the data to a 2:1 stoichiometric model, resulting in a Job Plot shown in *Figure 7.5*.

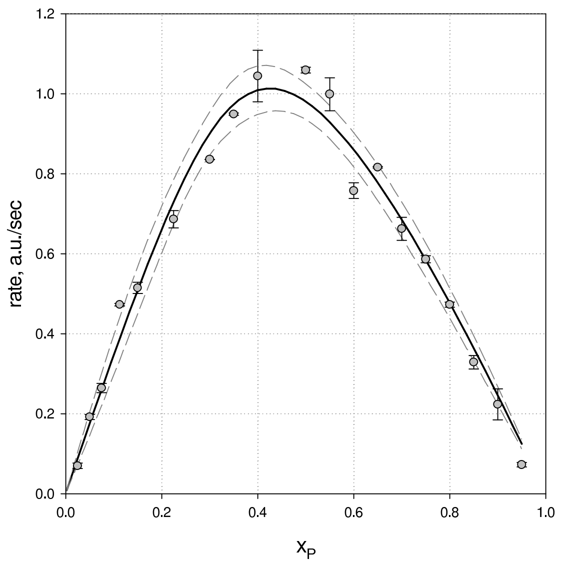


Fig. 7.5 Mole fraction plot generated by DynaFit, based on the experimental data shown in *Listing 7.6*. See also Figure 3 in [5].

7.11.3 Titration plot

In special cases, for example when there is a scarcity of available biological material, an equilibrium binding experiment can be arranged such that aliquots of a ligand stock solution are repeatedly being added to the *same* starting solution of the titrant. Upon each addition the total concentration of the titrant changes, because of the inevitable dilution. This nonstandard situation is handled in DynaFit by application of the keywords `titration`.

```
[data]
   plot titration
```

To arrange for the analysis of binding data from an experiment, in which both interacting components change their concentrations in this particular way, the `variable` line must list the names of two molecular species. The data block itself must contain three columns. The first column will contain the final concentration of the ligand being added; the second column will contain the final concentration of the protein after each addition; and the third column must contain the observed experimental signal. An illustrative example is shown in *Listing 7.7*. For a complete working example, see the script file 01.txt located in the directory ./manual/data/titr distributed with the program.

EXAMPLE SCRIPT

Listing 7.7

```
[data]
   variable L, P
   plot titration

   set 1H.d | resp P = 8.9 ?, P.L = 8.8 ?
   set 1H.e | resp P = 9.0 ?, P.L = 9.1 ?
   set 1H.f | resp P = 8.0 ?, P.L = 8.1 ?

[set:1H.d]

L,mM      P,mM      shift

0.0000    0.1250    8.941
0.0328    0.1249    8.926
0.0655    0.1247    8.899
0.1307    0.1245    8.867
0.2603    0.1240    8.824
0.5164    0.1230    8.800
1.0161    0.1210    8.783
1.9688    0.1172    8.781

[set:1H.e]

L,mM      P,mM      shift

0.0000    0.1250    9.052
```

```
0.0328   0.1249   9.055
0.0655   0.1247   9.060
0.1307   0.1245   9.072
0.2603   0.1240   9.090
0.5164   0.1230   9.098
1.0161   0.1210   9.111
1.9688   0.1172   9.110

[set:1H.f]

L,mM     P,mM     shift

0.0000   0.1250   7.966
0.0328   0.1249   7.977
0.0655   0.1247   8.021
0.1307   0.1245   8.050
0.2603   0.1240   8.090
0.5164   0.1230   8.124
1.0161   0.1210   8.138
1.9688   0.1172   8.141
```

Listing 7.7 displays in the first column the total concentration of a particular ligand (a model peptide representing histone H3); in the second column the total concentration of the RIZ1 tumor suppressor protein; and in the third column the chemical shifts for three different proton nuclei located on the protein molecule. The raw data were generously provided by Dr. Klára Briknarová (University of Montana). Importantly, the data were generated by repeated addition of the peptide stock solution to the same NMR tube containing the protein sample. See also Figure 10.1 in [7].

When DynaFit processes this type of data, identified by the keyword `titration`, it produces a simple Cartesian plot of the observable physical variable plotted against the final ligand concentration, on the horizontal axis. In this type of plot it is *implied that the total concentration of the target being titrated is continuously changing from one data point to the next*. The plot generated by DynaFit for the data shown in *Listing 7.7* is shown in Figure 7.6

7.11.4 Multiple graphs

Occasionally it is useful or even necessary to perform the global fit of very disparate data sets, with widely differing ranges of the experimental data. For example, one might wish to combine observations of changes in proton chemical shifts, ranging typically from 5 to 10 ppm in the observed signal, with observations of changes in nitrogen chemical shifts, ranging typically from 110 to 130 ppm. The general notation suitable for this type of global analysis relies on the keyword `graph`, as is shown in the code fragment below.

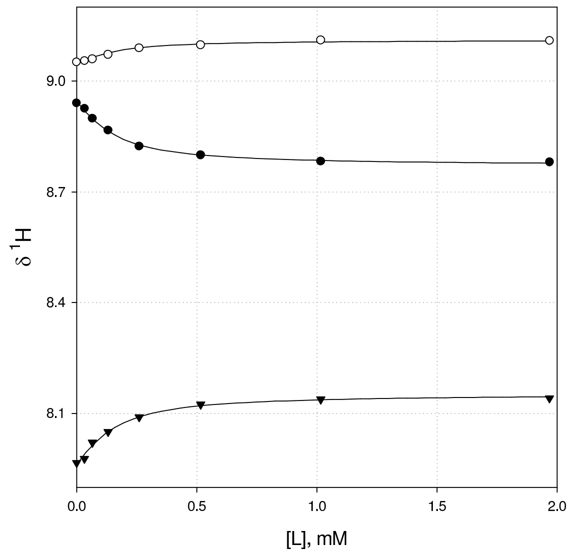


Fig. 7.6 Protein ^{1}H chemical shift titration plot generated by DynaFit, based on the experimental data shown in *Listing 7.7*. Both protein and ligand concentrations were varied, although only the changes in the ligand concentration are shown in the graph.

```
[data]
...
   graph GRAPH-1

   file A
   file B
   file C

   graph GRAPH-2

   file D
   file E
   file F

[end]
```

In the code fragment above, `GRAPH-1` and `GRAPH-2` stand for any arbitrary labels the use can give to different graphs to be produced by DynaFit. Each separate graph will collect within it only those plots that are associated with the data set names immediately following. In this case, `GRAPH-1` will display only plots corresponding to the data files **A** – **C**, whereas `GRAPH-2` will display only data plots corresponding to data file **D** – **F**. Importantly, all six data files **A** – **F** will be analyzed together, in the global fashion [2]. A representative example shown in *Listing 7.8*.

Listing 7.8

```
[data]
   variable L, P
   plot titration

   graph 1H

   set 1H.d | resp P = 8.9 ?, P.L = 8.8 ?
   set 1H.e | resp P = 9.0 ?, P.L = 9.1 ?
   set 1H.f | resp P = 8.0 ?, P.L = 8.1 ?

   graph 15N

   set 15N.a | resp P = 118 ?, P.L = 119 ?
   set 15N.b | resp P = 117 ?, P.L = 116 ?
   set 15N.c | resp P = 122 ?, P.L = 121 ?

; -------------------
; 1H CHEMICAL SHIFTS
; -------------------

[set:1H.d]

L,mM     P,mM     shift

0.0000   0.1250   8.941
0.0328   0.1249   8.926
0.0655   0.1247   8.899
0.1307   0.1245   8.867
0.2603   0.1240   8.824
0.5164   0.1230   8.800
1.0161   0.1210   8.783
1.9688   0.1172   8.781

[set:1H.e]

...

; -------------------
; 15N CHEMICAL SHIFTS
; -------------------

[set:15N.a]

L,mM     P,mM     shift

0.0000   0.1250   118.215
0.0328   0.1249   118.271
0.0655   0.1247   118.361
0.1307   0.1245   118.482
0.2603   0.1240   118.540
0.5164   0.1230   118.682
1.0161   0.1210   118.747
1.9688   0.1172   118.735
```

```
[set:15N.b]

...
```

In *Listing 7.8* the notation `graph 1H` signifies that the three proton shift data sets labeled `1H.d`, `1H.e`, and `1H.f` should be grouped together in one graph, labeled **1H**. In turn the notation `graph 15N` signifies that the three nitrogen chemical shift data sets labeled `15N.a`, `15N.b`, and `15N.c` should be grouped together in a separate graph, labeled **15N**. Very importantly, all six data sets are analyzed together in a global fashion [2]. The reason for this segregation into two graphs is that the proton shift range (8–9 ppm) is very different from the nitrogen shift range (116–122 ppm). If all data sets were plotted in the same graph, all six plots would appear completely flat. The actual display produced by DynaFit is shown in *Figure 7.7*, in which the results of fit are very easily grasped upon visual inspection.

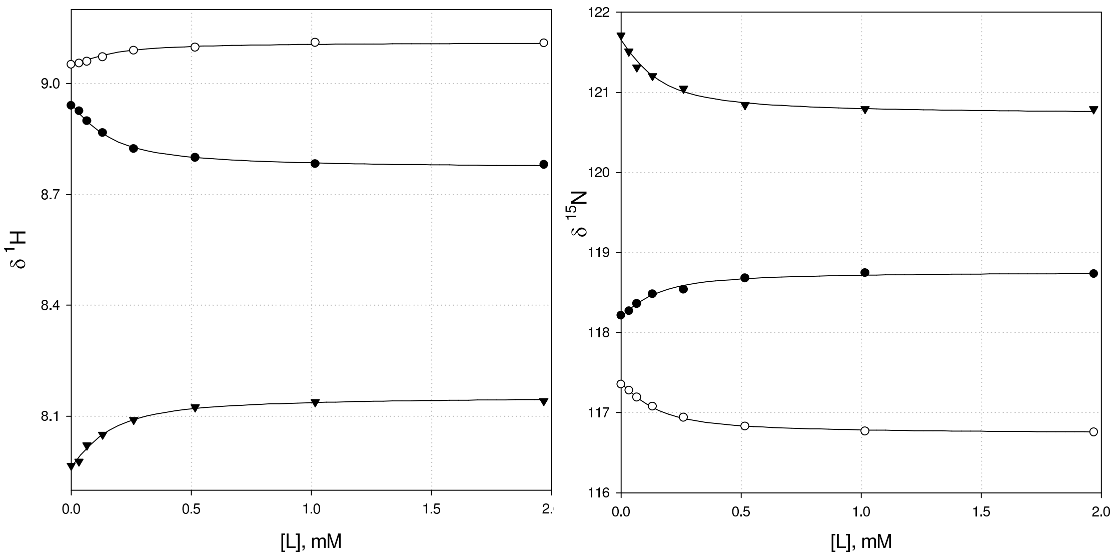


Fig. 7.7: Protein ^{1}N and ^{15}N chemical shift titration plot generated by DynaFit, based on the experimental data shown in *Listing 7.8*. All six data sets are analyzed together [2] and are segregated into two groups of three only for the purpose of plotting.

EXAMPLE SCRIPT

For a complete working example, see the script file 02.txt located in the directory ./manual/data/titr distributed with the program.

7.11.5 Concentration plot: State variables

In the analysis of the reaction progress, it is very often advantageous to examine not only the best-fit model curve overlaid on the experimental data, but also the corresponding plot of underlying concentrations of some or all molecular species. This accomplished by using the keyword `monitor`, followed by a comma-separate list of molecular species we wish to monitor. The general pattern is shown in the code fragment below.

```
[data]
...
   monitor MOLECULAR SPECIES LIST

   file ...
   file ...
   file ...
```

Depending on the physical representation of the experimental data, the keyword `file` appearing in the code snippet above might be replaced with `set` or `column`. A realistic example is shown in *Listing 7.9*. For a complete working example, see the script file 02.txt located in the directory ./manual/data/monitor distributed with the program.

EXAMPLE SCRIPT

Listing 7.9

```
[mechanism]
   E + S <===> ES        :      kaS    kdS
   ES ----> E + P        :      kdP
   E + I <==> EI         :      kaI    kdI
   EI <==> EI*           :      kif    kib
...
[data]
   sheet  ./manual/data/monitor/data/sheet.txt
   offset -1
   monitor E, ES, EI, EI*

   column 5 | conc I =  4
   column 6 | conc I =  8
   column 7 | conc I = 16
   column 8 | conc I = 32
   column 9 | conc I = 64
```

In this particular example DynaFit was used to fit a set of reaction progress curves describing the inhibition of 5*alpha*-ketosteroid reductase by the drug finasteride. The experimental data are from ref. [11]. The inhibition mechanism involves an initial binding of inhibitor followed by a reversible isomerization of the enzyme–inhibitor complex. The overlay of the experimental data and the best-fit model curves is shown in the left hand panel of *Figure 7.8*.

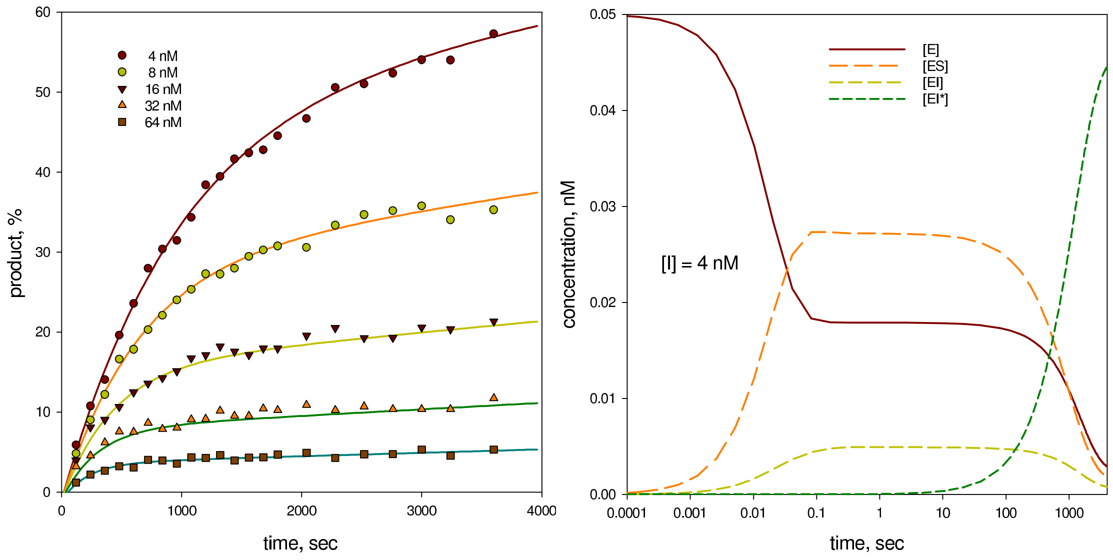


Fig. 7.8: *Left*: Data and best-fit model overlay. *Right*: Concentration plot corresponding to the $[I]$ = 4 nM model curve in the left-hand panel generated by the use of the keyword `monitor`. See *Listing 7.9* and ref. [11].

The notation `monitor E, ES, EI, EI*` signifies that for each reaction progress curve we asked DynaFit to produce a plot of the concentration of molecular species **E** (the free enzyme), **ES** (the Michaelis complex), **EI** (the initial enzyme–inhibitor complex) and **EI*** (the final, isomerized complex). Because there are five reaction progress curves being analyzed, DynaFit generated five sets of concentration plots. One of those plots, corresponding to the lowest inhibitor concentration ($[I]$ = 4 nM) is shown in the right-hand panel of *Figure 7.8*. Additional concentration plots (not shown) were automatically generated by DynaFit for the remaining inhibitor concentrations shown in *Listing 7.9*.

7.11.6 Arbitrary interpolation mesh

This `mesh` keyword is also available in data fitting projects, not only in heuristic simulations. DynaFit always chooses a certain particular interpolation mesh for the model curves superimposed on the fitted experimental data. However, in certain specific instances we might wish to override the default spacing and the extent of the best-fit model curve. In that case we can use the keyword `mesh` to specify the values of independent variable that should be used for the construction of the best-fit model curves. The generalized notation is shown below:

```
[task]
```

```
   task = fit
...
[data]
   mesh from X to Y step Z    ; best-fit model curve
```

A very similar notation can be used for logarithmically spaced model curves (`mesh logarithmic`) generated in data fitting projects.

7.12 Preprocessing

This section describes two methods that can be used to pre-process raw experimental data, which are physically represented as external disk files. Both of these methods arose in the analysis of the reaction progress curves, in particular in the analysis of stopped-flow rapid kinetic data.

7.12.1 Maximum reaction time

The first method (relying on the keyword `maximum`) can be used to edit out "late" portions of kinetic traces *selectively* and separately for individual data files being subjected to global regression analysis.

An introductory example will help illuminate the motivation. Let us assume the existence of two data files named F1.TXT and F2.TXT, respectively. Both data files contain readings of fluorescence recorded over time, from time zero to the maximum time of 10 seconds. Now let us assume that we wish to analyze on the first first *two seconds* from file F1.TXT and the first *five seconds* from file F2.TXT. This can be accomplished as shown in the listing immediately below:

Example 1

```
[task]
   task = fit
   data = progress
[data]
   maximum 2.0  ; delete data points at t > 2.0 sec
   file F1.TXT

   maximum 5.0  ; delete data points at t > 5.0 sec
   file F2.TXT
```

The `maximum` keyword will apply to all data files that follow, until either another `maximum` values is listed in the `[data]` section of the script, or until the special value `maximum off` is found:

Example 2

```
[task]
   task = fit
   data = progress
[data]
   maximum 2.0  ; delete t > 2.0 in files F1, F2, F3
   file F1.TXT
   file F2.TXT
   file F3.TXT

   maximum 5.0  ; delete t > 5.0 in files F4, F5, F6
   file F4.TXT
   file F5.TXT
   file F6.TXT

   maximum off  ; no data deletion in files F7, F8, F9
   file F7.TXT
   file F8.TXT
   file F9.TXT
```

7.12.2 Additive constant

The keyword `shift` can be used to introduce spacing between plots of individual progress curve data sets being analyzed in a global fashion. Consider the following illustrative example. Let us assume that we wish to perform a global fit of seven reaction progress curves, represented as columns number 2 through 8 in the comma-separated spread sheet file N4.csv. The usual coding to accomplish this task is shown in the listing immediately below (Example 1). The corresponding graph is shown in *Figure* 7.9.

Example 1

```
...
[data]
   directory ./test/_temp/data
   plot      logarithmic
   maximum   0.5
   sheet     N4.csv
   column 2 | offset auto ? | conc N =  250 | label 0.25 mM
   column 3 | offset auto ? | conc N =  500 | label 0.5 mM
   column 4 | offset auto ? | conc N = 1000 | label 1 mM
   column 5 | offset auto ? | conc N = 2000 | label 2 mM
   column 6 | offset auto ? | conc N = 4000 | label 4 mM
   column 7 | offset auto ? | conc N = 6000 | label 6 mM
   column 8 | offset auto ? | conc N = 8000 | label 8 mM
...
```

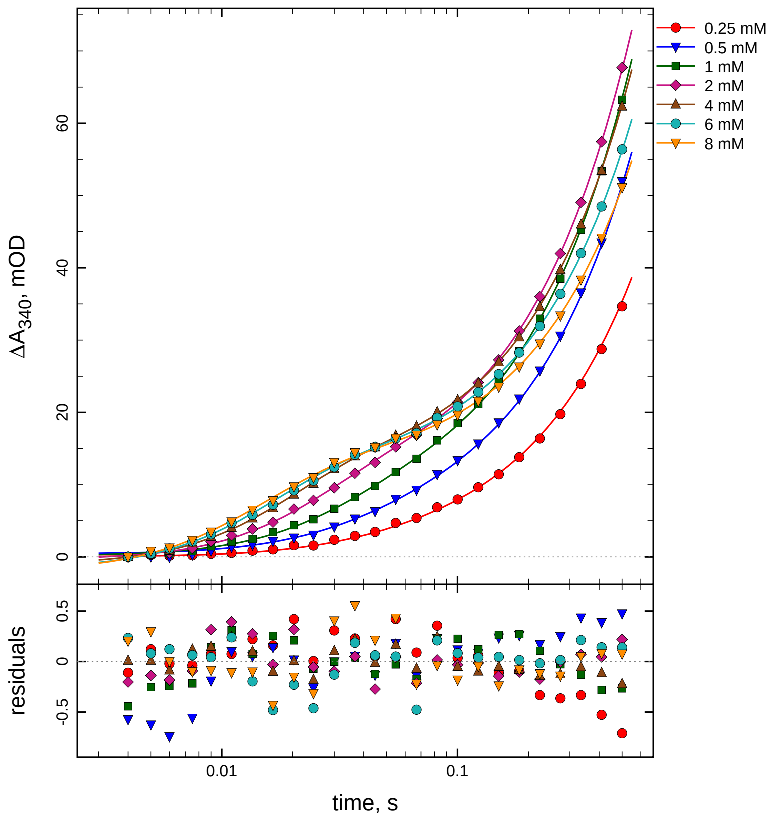


Fig. 7.9: Example of global fit. Note the inconvenient overlap of several kinetic traces, which make the resulting graph largely "unreadable".

The keyword `shift` can be used to add a fixed constant to all experimental signal values, separately for each kinetic trace, such that the overlap of traces in *Figure 7.9* is eliminated. The requisite listing is shown in listing Example 2 below. Here we have added the absorbance value "1" to all data in column no. 3, absorbance value "2" to all data in column no. 4, and so on. The resulting graph is shown in *Figure 7.10*.

Example 2

```
...
[data]
   directory ./test/_temp/data
   plot      logarithmic
   maximum   0.5
   sheet     N4.csv
   shift  0 | column 2 | offset auto ? | conc N =  250 | label 0.25 mM
   shift  1 | column 3 | offset auto ? | conc N =  500 | label 0.5 mM
   shift  2 | column 4 | offset auto ? | conc N = 1000 | label 1 mM
   shift  4 | column 5 | offset auto ? | conc N = 2000 | label 2 mM
   shift 12 | column 6 | offset auto ? | conc N = 4000 | label 4 mM
   shift 20 | column 7 | offset auto ? | conc N = 6000 | label 6 mM
   shift 28 | column 8 | offset auto ? | conc N = 8000 | label 8 mM
...
```

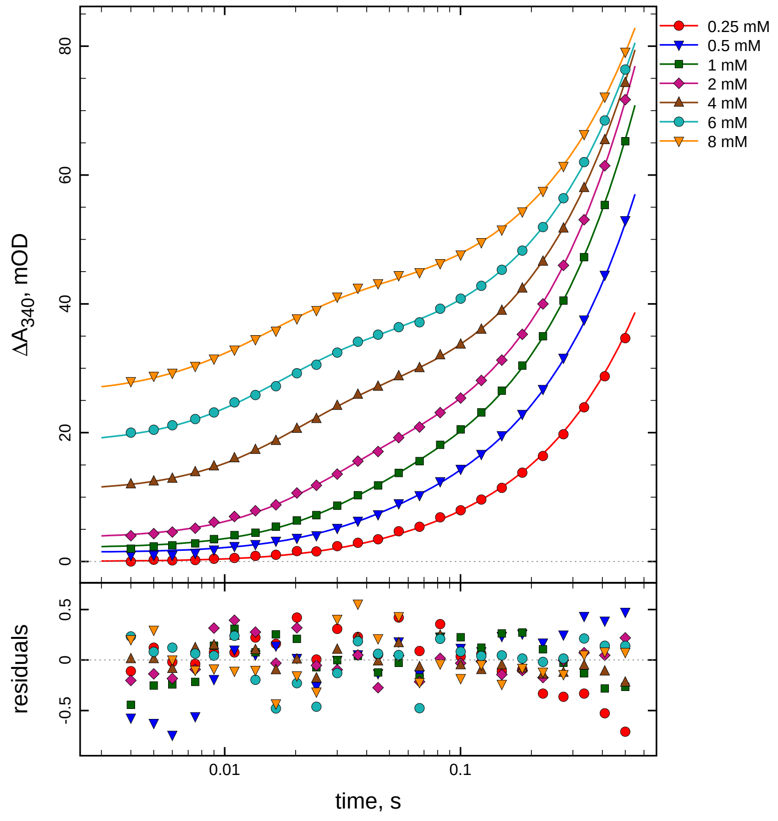


Fig. 7.10: Example of global fit after applying the keyword `shift` selectively to each progress curve (compare with *Figure 7.9*).

The data traces shown in *Figure 7.10* are much easier to follow and understand visually. The results of fit are unaffected by shifting the kinetic traces on the vertical (signal) axis, because the offset on the signal axis (keyword `offset`) is being optimized in the nonlinear least-squares regression.

References

1. Askelof, P., Korsfeldt, M., Mannervik, B.: Error structure of enzyme kinetic experiments: Implications for weighting in regression analysis of experimental data. Eur. J. Biochem. **69**, 61–67 (1976)
2. Beechem, J.M.: Global analysis of biochemical and biophysical data. Meth. Enzymol. **210**, 37–54 (1992)
3. Huang, C.: Determination of binding stoichiometry by the continuous variation method: The Job Plot. Meth. Enzymol. **87**, 509–525 (1982)
4. Jakobson, I., Askelof, P., Warholm, M., Mannervik, B.: A steady-state-kinetic random mechanism for glutathione s-transferase A from rat liver: A model involving kinetically significant enzyme-product complexes in the forward reaction. Eur. J. Biochem. **77**, 253–262 (1977)
5. Jamakhandi, A.P., Kuzmič, P., Sanders, D.E., Miller, G.P.: Global analysis of protein-protein interactions reveals multiple cytochrome P450 2E1Űreductase complexes. Biochemistry **46**, 10,192–10,201 (2007)
6. Kuzmic, P., Lorenz, T., Reinstein, J.: Analysis of residuals from enzyme kinetic and protein folding experiments in the presence of correlated experimental noise. Anal. Biochem. **395**, 1–7 (2009)
7. Kuzmič, P.: DynaFit – A software package for enzymology. Meth. Enzymol. **467**, 247–280 (2009)
8. Mannervik, B.: Design and analysis of kinetic experiments for discrimination between rival models. In: L. Endrényi (ed.) Kinetic data analysis, pp. 235–270. Plenum Press, New York (1981)
9. Mannervik, B.: Regression analysis, experimental error, and statistical criteria in the design and analysis of experiments for discrimination between rival kinetic models. Methods Enzymol. **87**, 370–390 (1982)
10. Morgan, M.T., Maiti, A., Fitzgerald, M.E., Drohat, A.C.: Stoichiometry and affinity for thymine dna glycosylase binding to specific and nonspecific dna. Nucl. Acids Res. **39**, 2319Ű2329 (2011)
11. 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., Patel, I.R.: 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**, 3457–3464 (1996)
12. Storer, A.C., Darlison, M.G., Cornish-Bowden, A.: The nature of experimental error in enzyme kinetic measurements. Biochem. J. **151**, 361–367 (1975)

Chapter 8
Output files and directories

The `[output]` section of each DynaFit script tells the program where to deposit the various types of output files automatically produced by during its execution. This is arranged by the use of the keyword `directory`. The general pattern is shown in the code fragment below.

```
[output]
   directory DIRECTORY_NAME
```

In this code fragment, `DIRECTORY_NAME` is a place holder for the path name of the output directory to be utilized by DynaFit. If the directory does not exist, it will be automatically created. If the directory does exist, the output files located in it during previous program executions will be automatically rewritten. The directory name could be an absolute path or a relative path; please consult section 7.2.1 for the detailed discussion of relative vs. absolute path names.

8.1 Output file types

When DynaFit runs, it automatically generates three types of output files:

- **Plain text**. These tab-delimited text files are written into a special subdirectory named **txt**, which is automatically created in the main output directory. The text files are very useful for importation into software packages that specialize in publication-quality graphics, such as SigmaPlot or GraphPad.
- **GIF images**. The GIF image files are written into a subdirectory **gif**, which is also automatically created. The GIF files serve mainly to display the results within the HTML output files (see below).
- **HTML files**. When DynaFit executes it creates a variety of interlinked HTML files, starting from the file named index.html located in the root of the main

output directory. The remaining HTML files are written into a newly created subdirectory **htm**.

8.2 Initial rate file

DynaFit can be used for convenient automation of a two-stage kinetic analysis procedure. In the first stage, DynaFit will determine the initial reaction rates of enzymatic assays analyzed individually (in "local" mode as opposed to globally [1]). The initial reaction rates are written to a disk file, along with the associated concentration of a variable reactant, such as the substrate or the inhibitor. The location of this newly created data file is specified by the keyword `rate-file`, as is shown in the code fragment below.

```
[output]
   rate-file FILE_NAME
```

The advantage of this method of determining initial rates is that the fitting model, formulated as a system of differential equations, can be as complex as necessary to account for any possible nonlinearities. In the second stage, DynaFit then proceeds to analyze the newly (and fully automatically) created initial rate data file, again using the familiar symbolic notation.

For a complete working example, see the script file 01.txtlocated in the directory ./manual/data/rates distributed with the program. The script file is displayed in full in *Listing 8.1*.

EXAMPLE SCRIPT

Listing 8.1

```
; ------------------------------------------------------------
; Part 1: determine initial rates and write them to the disk.
; ------------------------------------------------------------

[task]
   data  = progress
   task  = fit

[mechanism]
   E + S <===> ES       :     kaS    kdS
   ES ----> E + P       :     kdP
   I -->                :     dummy

[constants]
   kaS = 1, kdS = 2 ?
   kdP = 1 ?
   dummy = 1

[concentrations]
   S = 31
   E = 0.05
```

```
[responses]
   P = 3.21

[data]
   sheet  ./manual/data/rate/data/sheet.txt

   column  2 | offset -1 ? | conc I = 0

[output]
   directory ./manual/data/rate/output/01
   rate-file ./manual/data/rate/data/rates.txt

[settings]
{Filter}
   XMax = 900
   TimeInitialRate = 1

[task] | data = progress | task = fit
[data] | column 3 | offset -1 ? | conc I = 1

[task] | data = progress | task = fit
[data] | column 4 | offset -1 ? | conc I = 2

[task] | data = progress | task = fit
[data] | column 5 | offset -1 ? | conc I = 4

[task] | data = progress | task = fit
[data] | column 6 | offset -1 ? | conc I = 8

[task] | data = progress | task = fit
[data] | column 7 | offset -1 ? | conc I = 32

[task] | data = progress | task = fit
[data] | column 8 | offset -1 ? | conc I = 64

[task] | data = progress | task = fit
[data] | column 9 | offset -1 ? | conc I = 128

; ------------------------------------------------------------
; Part 2: fit the initial rates to the simplest binding model.
; ------------------------------------------------------------

[task]
   data  = equilibria
   task  = fit

[mechanism]
   E + I <==> EI      :     Ki    dissoc

[constants]
   Ki = 0.1 ??

[concentrations]
```

```
   E = 0.05

[responses]
   E = 20 ?

[data]
   variable I
   file ./manual/data/rate/data/rates.txt

[end]
```

The example problem shown in *Listing 8.1* utilizes the same 5α-reductase experimental data [2] that were discussed previously in section 7.11.5. Please note the uses of the notation `rate-file .../data/rates.txt`. This DynaFit encoding is responsible for the automatic creation of the initial rate data set. In the final task (`data = equilibria`) the initial rates are fit to the simple 1:1 enzyme:inhibitor binding model under rapid-equilibrium approximation. The results are summarized graphically in *Figure 8.1*.

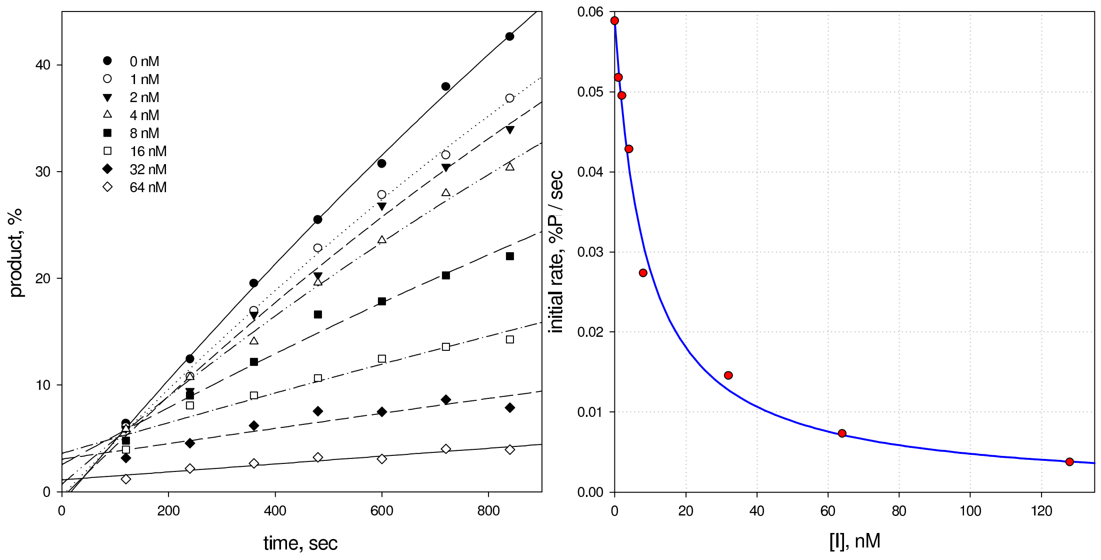


Fig. 8.1: *Left*: "Local" fit of reaction progress curves to determine the initial reaction rates. *Right*: Fit of the automatically created initial rate dataset to the simplest 1:1 binding model. See *Listing 8.1* and ref. [2].

References

1. Beechem, J.M.: Global analysis of biochemical and biophysical data. Meth. Enzymol. **210**, 37–54 (1992)
2. 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., Patel, I.R.: 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**, 3457–3464 (1996)

Chapter 9
Initialization and control settings

This chapter describes the master configuration file that sets up default control parameters for many algorithms implemented in DynaFit. It also describes how to override these default control settings in script files.

9.1 Default initialization file

When DynaFit starts up, it reads the default configuration file named ./system/DynaFit/settings.txt, where "." stands for the DynaFit installation directory. This configuration file contains many parameters for a number of numerical algorithms implemented in DynaFit. The entire default initialization file is shown in *Listing 9.1*.

Listing 9.1

```
{DynaFit}
   RandomizationSeed = 4357 ; | 0 for system time
   DefaultFittingAlgorithm = trust-region ; | marquardt
   PointsParametersRatio = 10

{ODESolver}
   Iterations    = 1000
   AbsoluteError = 1.e-14
   RelativeError = 1.e-8

{TrustRegion}
   IterationsPerParameter    = 100
   FunctionCallsPerParameter = 200
   ConstrainedFit            = y
   RobustFit                 = n
   EqualizeDatasets          = n

{Marquardt}
   IterationsPerParameter = 100
   RestartPerturbation    = 0.1
   Restarts               = 2
   RestartsConfidence     = 1
   RobustFit              = n
   EqualizeDatasets       = n
   FixRedundantParameters = y

{ConfidenceIntervals}
   LevelPercent           = 95
   OnlyConstants          = y
   JointProbability       = n
   MaxSteps               = 30
   SquaresIncreasePercent = 0 ; | 10 for continuous assays

{DifferentialEvolution}
   PopulationSizeFixed            = 0
   PopulationSizeMinimal          = 300
   PopulationSizePerParameter     = 5
   PopulationSizePerOrderOfMag    = 3
   MinimumGenerationsPerParameter = 5
   MaximumGenerationsPerParameter = 100
   MaximumEvolutions              = 6
   MinimumEvolutions              = 4
   RandomSeed                     = 1234
   RootMeanSquareMin              = 0
   RootMeanSquareMax              = 0
   OutputTextFileType             = csv ; | txt

{Constraints}
   Constants                  = 1000000
   Responses                  = 1000000
   Concentrations             = 1000
   AllParametersConstrained   = y
   AllParametersRelativeBound = 1000000

{Filter}
   XMin                = 0
   XMax                = 0
   XShift              = 0
   XFirstMesh          = 0
   YMin                = 0
   YMax                = 0
   PointsPerDataset    = 0
   ExponentialSpacing  = n
   ReadEveryNthPoint   = 0
   SkipFirstNPoints    = 0
   TimeInitialRate     = 1
   PrintInitialRate    = y
   SmoothData          = n
   SmoothingMethod     = savitzky-golay ; | average
   SavitzkyGolayWindow = 10
   SavitzkyGolayDegree = 4
   ExtrapolationMethod = quadratic ; | linear
   SmoothingPasses     = 4
   AverageReplicates   = n
   ZeroBaselineSignal  = n

{PieceWiseLinearFit}
   Points   = 0
   Segments = 4
   Time     = 0
   Overlap  = n

{Output}
   UseDefaultDirectory  = y
   Autocorrelations     = n
   ConfidenceBands      = n
   PredictionBands      = n
   WriteTXT             = n
   WriteEPS             = n
   ColorEPS             = y
   ResidualsEPS         = y
   WriteTeX             = n
   XAxisUnit            =
   XAxisLabel           =
   YAxisLabel           =
   BlackBackground      = y
   IncludeXZero         = n
   IncludeYZero         = n
   InitialRateDigits    = 4
   StartDefaultBrowser  = n
   PlotRatesLogarithmic = n
   PlotStateLogarithmic = n
   SignificantDigits    = 2
   ResidualRange        = 0

{MonteCarlo}
   PerformInitialFit         = y
   Runs                      = 1000
   RandomizationMethod       = simulate  ;  | shuffle | shift
   Distribution              = normal    ;  | cauchy | logistic | uniform
   StandardDeviationSource   = fit       ;  | data | explicit
   StandardDeviation         = 1.2
   SignificantDigits         = 4
   HistogramBuckets          = 20
   TruncateMeanPercent       = 5
   ColorOutput               = y
   RandomizationSeed         = 1267
   ConcentrationErrorPercent = 0
   OriginalEstimates         = n
   ConfidenceLevel           = 100

{EstimateScan}
   ReportSizeMax   = 1000
   RefineEstimates = 10

{ExponentialFit}
   Degree            = 4
   Automatic         = y
   AllowOscillations = n
   TinyAmplitudes    = 0 ; | 0.000001
   RefineMarquardt   = n

{OptimalDesign}
   Algorithm = AS ; | DE | BFGS
   Function  = D  ; | T  | E  | V

{ModelSelection}
   PrioritizeCriterion             = akaike ; | bayesian
   ConfidenceIntervalRangeMax      = 10000
   CoefficientOfVariationMax       = 200
   InformationCriterionDeltaMax    = 10
   InformationCriterionWeightMin   = 0.01
   RelativeSquaresMax              = 1.05
```

It is not recommended that the users modify these default settings by altering the default initialization file, unless there are very compelling reasons to do so and the particular user understands perfectly well any possible adverse consequences. For example, changing the truncation error tolerances of the ODE solvers (keywords `AbsoluteError` and `RelativeError` in section `ODESolver`) should be undertaken only with extreme care and only after first studying thoroughly how does any such change affect the overall numerical precision differential-equation modeling.

9.2 Overriding default initialization

Instead of making any changes in the default initialization file ./system/DynaFit/settings.txt, it is preferable that if necessary the user overrides the default settings by inserting a special section into a particular script file, as is shown in the code fragment below.

```
[settings]
    ...
    ... OVERRIDE DEFAULT SETTINGS HERE
    ...
```

The `[settings]` section, if any is present, applies to the `[task]` block in which it is embedded, and also to all subsequent tasks – unless or until another `[settings]` section is found further down in the given script file. For example in *Listing 9.2*, the contents of the `[settings]` section will apply to all three `[task]` sections contained in the given script file.

Listing 9.2

```
[task] ; #1
...
[settings]
```

```
   ...
   ... These settings will apply to all tasks: #1, #2, #3
   ...
[task] ; #2
...
[task] ; #3
...
[end]
```

In contrast, *Listing 9.3* illustrates how the default DynaFit settings could be modified selectively for individual tasks.

Listing 9.3

```
[task] ; #1
...
[settings]
   ...
   ... These settings will apply to tasks #1, #2
   ...
[task] ; #2
...
[task] ; #3
...
[settings]
   ...
   ... These settings will apply to task #3 only
   ...
[end]
```

9.3 DynaFit control settings

The purpose of this section is to discuss selectively those elements of DynaFit control settings that are likely to be useful to the casual user, i.e., one who is not thoroughly familiar with the intricate details of advanced numerical algorithms. A thorough discussion of these advanced techniques is reserved for the separate Volume 3 of the DynaFit Manual.

9.3.1 Overall settings for DynaFit

The following control settings apply to the DynaFit software package as a whole, regardless of which particular algorithm is utilized for any specific task:

```
{DynaFit}
   RandomizationSeed = 4357 ; | 0 for system time
   DefaultFittingAlgorithm = trust-region ; | marquardt
   PointsParametersRatio = 10
```

9.3.1.1 Random number generator

The pseudo-random number generator employed in DynaFit is essentially the Mersenne Twister algorithm of Matsumoto and Nishimura [25]. The initialization code listed below can be used to set the randomization seed.

```
[settings]
{DynaFit}
   ...
   RandomizationSeed = 4357 ; | 0 for system time
```

If `RandomizationSeed` is set to zero the sequence of pseudo-random numbers will be different every time DynaFit runs. This is useful in certain special cases, for example in performing Monte-Carlo simulations on a multi-processor computer running several images of DynaFit simultaneously, with the goal of eventually merging the results. With nonzero randomization seed all instances of DynaFit running in parallel would produce identical Monte-Carlo results.

9.3.1.2 Default least squarer fitting algorithm

For most data-fitting tasks, DynaFit uses one of two most well known and understood nonlinear least-squares fitting algorithms:

1. The Levenberg-Marquardt method [22] as implemented by Reich [30].
2. The trust-region method (algorithm NL2SOL) of Dennis *et al.* [8, 9, 10].

The initialization file settings.txt contains the following code allowing the user to select which of the two least-squares fitters should be used:

```
[settings]
{DynaFit}
   ...
   DefaultFittingAlgorithm = trust-region ; | marquardt
```

The notation `marquardt` can be replaced by `levenberg-marquardt`. The default algorithm can be overridden in the `[task]` section of the script, as is described in section 2.4. For example, even if the initialization file contains `DefaultFittingAlgorithm = trust-region`, setting `algorithm = LM` in the `[task]` section will force DynaFit to utilize the Levenberg-Marquardt method for the given task.

9.3.1.3 Number of data points vs. optimized parameters

The following control parameter has been newly introduced in DynaFit version 4.08, released in March 2018:

```
[settings]
{DynaFit}
...
   PointsParametersRatio = 10
```

This notation is applicable to *continuous reaction progress* experiments, for example, those arising in continuous enzyme assays. One important data-analytic challenge in the case of continuous assays is that the experimental data points are not statistically independent, and therefore the formal standard errors (*SE*) arising in nonlinear least-squares regression are essentially irrelevant and numerically invalid. The same is true for the *coefficients of variation* defined as

$$CV = 100\,\frac{SE}{\hat{p}} \quad ,$$

where $\hat{p}$ is the best-fit value of the relevant regression parameter. In an attempt to provide at least a somewhat meaningful value of CV for continuous assays, if `PointsParametersRatio` is set to a value grater than unity, DynaFit will compute and report an "empirical" coefficient of variation.

Let n_P be the number of adjustable model parameters; n_D the number of experimental data points; and R the numerical value of the control parameter `PointsParametersRatio`. If the ratio R is defined with a value $R > 1$ in the control settings file, *and* if $n_\mathrm{D}/n_\mathrm{P} > R$, the "empirical" coefficient of variation (CV_e) will be reported in the nonlinear regression results as

$$CV_\mathrm{e} = CV\,\sqrt{\frac{n_\mathrm{D}/n_\mathrm{P}-1}{R-1}} = 100\,\frac{SE}{\hat{p}}\,\sqrt{\frac{n_\mathrm{D}/n_\mathrm{P}-1}{R-1}} \quad .$$

In all relevant cases, the best-fit parameter table reported by DynaFit will contain a notation alerting the data analyst that all CV values were "inflated" accordingly. Note that the formal standard error SE in the case of continuous assays will still be reported as the (numerically invalid and therefore irrelevant) *nominal* value.

9.3.2 ODE Solver

The following initialization code controls the settings for the LSODE solver [16] of first-order Ordinary Differential Equations (ODEs), which is the numerical algorithm at the core of the DynaFit numerical engine.

```
{ODESolver}
   Iterations    = 5000
   AbsoluteError = 1.e-15
   RelativeError = 1.e-9
```

These control settings for the LSODE algorithm can be understood by carefully studying the original Fortran-77 code available from the NETLIB repository (http://www.netlib.org). A brief explanation is provided in *Table 9.1*.

Parameter	Default	Explanation
Iterations	5000	Number of corrector iterations in the predictor-corrector numerical algorithm for stiff systems of ODEs [16, 7]
AbsoluteError	10^{-15}	Desired absolute precision of the numerical solution
RelativeError	10^{-9}	Desired relative precision of the numerical solution (9 significant digits)

Table 9.1: Control settings for the ODE solver.

It is highly recommended that the casual user does not modify these settings for the ODE solver. Relaxing the stringent criteria for absolute and relative precision of the numerical solution could result in loss of precision in the best-fit values of model parameters such as rate constants.

The error weights in the original LSODE algorithm (see subroutine EWSET) are set according to Eqn (9.1), where y_i is the ith element of the solution vector; r_i is the requested relative precision for this element; and a_i is the corresponding absolute precision.

$$w_i = r_i|y_i| + a_i \tag{9.1}$$

The existence of the absolute error term a_i in Eqn (9.1) implies that it is very important to properly *scale all concentrations on input, such that their absolute numerical values are as close to unity as possible*. For example in the study of enzyme inhibition, if all inhibitor concentration are in the nanomolar range (absolute value 10^{-9}) then all input concentrations *and* rate or equilibrium constants should be scaled to nanomolar units.

9.3.3 *Trust region least-squares fitter*

The following initialization code controls the settings for the hybrid lest-squares minimization algorithm (NL2SOL version 2.3, released July 2015) originally described by Dennis *et al.* [8, 9, 10].

```
{TrustRegion}
   IterationsPerParameter    = 100
   FunctionCallsPerParameter = 200
   ConstrainedFit            = y
   RobustFit                  = n
   EqualizeDatasets           = n
```

The control parameters `IterationsPerParameter` and `FunctionCallsPerParameter` are self-explanatory. For details, see the documentation for the software package NL2SOL 2.3.

The meaning of the control parameters `RobustFit` and `EqualizeDatasets` is the same as is described in section 9.3.4. The control parameter `ConstrainedFit` (possible values **y** for “Yes” or **n** for “No”) determines whether NL2SOL should use a specialized version with parameter bounds.

9.3.4 Levenberg-Marquardt least-squares fitter

The following initialization code controls the settings for Reich’s implementation [30] of the Levenberg-Marquardt [22] lest-squares minimization algorithm.

```
{Marquardt}
   IterationsPerParameter = 100
   RestartPerturbation    = 0.1
   Restarts               = 2
   RestartsConfidence     = 1
   RobustFit              = n
   EqualizeDatasets       = n
   FixRedundantParameters = y
```

A brief explanation is provided in *Table 9.2*. Please note that in the DynaFit settings file, the Boolean value **n** stands for “no”, whereas the value **y** stands for “yes”.

9.3.4.1 Number of iterations

By default DynaFit will perform at most 100 times as many iterations as there are adjustable model parameters. This is signified by the following initialization code:

```
[settings]
{Marquardt}
   IterationsPerParameter = 100
```

For exceptionally ill-conditioned (i.e., overparametrized) problems one might attempt to reach convergence by increasing the maximum number of iterations even further. However, a much preferable solution is to either modify the fitting model such that all adjustable parameters are well defined by the available experimental data, or to obtain additional experimental data that do support all adjustable model parameters in the postulated fitting model.

Parameter	Default	Explanation
IterationsPerParameter	100	Number of Levenberg-Marquardt iterations per adjustable parameter
Restarts	2	Number of restarts after the presumed least-squares minimum has been reached
RestartsConfidence	2	Number of restarts in each step of a confidence interval search using the *profile-t* method [33])
RestartPerturbation	0.1	Maximum random fractional change in each adjustable model parameter upon restarting
RobustFit	**n**	Whether or not DynaFit should perform robust regression analysis using Huber's Mini-Max method [17] instead of the default ordinary least squares (OLS)
EqualizeDatasets	**n**	Whether or not DynaFit should perform data set equalization.
FixRedundantParameters	**n**	Whether or not DynaFit should automatically remove a completely redundant parameter from the fitting model.

Table 9.2: Control settings for the least-squares fitter.

9.3.4.2 Number of restarts

Authoritative textbooks on nonlinear regression analysis [31, p. 611] recommend periodically restarting the search for least-squares minimum. By default DynaFit will restart twice (`Restarts = 2`) after presumably converging to the least-squares minimum, each time altering all model parameters by a certain fractional perturbation `RestartPerturbation = 0.1`. During confidence interval searches using the profile-t method (see below) DynaFit will restart only once (`RestartsConfidence = 1`) at each step of the confidence interval search.

For particularly well behaved problems it might be possible to turn off restarts by setting `Restarts = 0`, in order to gain a small amount of computational speed. However practical experience shows that if the problem is truly well behaved the savings of time are negligibly small whereas if the problem is ill-conditioned there is a significant risk of landing in a false minimum with restarts turned off.

9.3.4.3 Robust regression analysis

By default DynaFit performs the usual Ordinary Least-Squares (OLS) fit [30]. The following initialization code can be used to accomplish robust regression using Huber's Mini-Max algorithm [17].

```
[settings]
{Marquardt}
   RobustFit = y
```

Huber's algorithm is one particular example of the Iteratively Re-weighted Least Squares method (IRLS). The main advantage of Huber's robust regression method is that any gross outliers in the experimental data are automatically de-emphasized or, in extreme cases, essentially eliminated by setting their least-squares regression weights to nearly zero.

This algorithm is most useful in fully automated or “unattended” processing of very many data sets arising, for example, in high-throughput screening of enzyme inhibitors [20]. In other contexts, especially in model discrimination studies, the casual user should exercise great amount of caution in interpreting the results from robust-regression analysis. In particular, IRLS vs. the usual OLS analysis affects the meaning of the Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC) [26, 27].

9.3.4.4 “Equalization” of data sets

In certain cases we might wish to perform the global fit [3] of multiple data sets that contain vastly disparate number of data points. For example, a stopped-flow kinetic trace might contain 2000 of data points, whereas another kinetic trace included in the global set might contain only 50 data points. Without taking extra steps the kinetic trace with 2000 data points would dominate the regression analysis simply on account of containing 40 times as many data points. The code snippet listed below could be used to “equalize” the regression weights such that all individual data sets included in the global fit have the same weight.

```
[settings]
{Marquardt}
   EqualizeDatasets = y
```

9.3.4.5 Elimination of redundant model parameters

By default, DynaFit will automatically eliminate from the given regression model any adjustable model parameter that are entirely redundant, in the sense that the there is absolutely no information about the possible values of the parameter in the given set of experimental data. Such gross over-parameterization can result from incorrectly including an extraneous step in the reaction mechanism, in which case DynaFit will automatically treat the requisite rate constants as fixed as opposed to adjustable, regardless of the users intentions. This behavior can be turned off by inserting the following code into the DynaFit script.

```
[settings]
{Marquardt}
   FixRedundantParameters = n
```

With this initialization, if or when DynaFit encounters an entirely redundant model parameter it will issue a runtime error and the regression analysis will not proceed any further.

9.3.5 Confidence intervals: Profile-t method

The following block of initialization code controls the behavior of the *profile-t* method for confidence interval estimation described by Bates & Watts [2, pp. 205-216, 302-304] [33, 4]. A brief explanation is provided in *Table 9.3*.

```
{ConfidenceIntervals}
   LevelPercent           = 95
   OnlyConstants          = y
   JointProbability       = n
   MaxSteps               = 30
   SquaresIncreasePercent = 0 ; | 10 for continuous assays
   MaxRefitIterations     = 2
   RefitImprovedLower     = y
   RefitImprovedHigher    = n
```

Parameter	Default	Explanation
LevelPercent	95	Desired probability level (%) for the confidence intervals
OnlyConstants	**y**	Whether to exclude "nuisance" parameters from least-squares optimization while searching for confidence limits of rate constants or equilibrium constants
JointProbability	**n**	Whether to compute the limits of a joint confidence region (**y**) or the limits of the marginal confidence interval (**n**)
MaxSteps	30	Maximum steps to be taken in the *profile-t* search algorithm
SquaresIncreasePercent	0	Desired increase in the residual sum of squares
MaxRefitIterations	2	If greater than zero, data fitting with restart from an improved estimate discovered during a confidence interval search
RefitImprovedLower	**y**	Restart according to `MaxRefitIterations` will occur if the newly discovered optimal value of a fitting parameter is lower than the initial "best-fit" value
RefitImprovedHigher	**n**	Restart according to `MaxRefitIterations` will occur if the newly discovered optimal value of a fitting parameter is higher than the initial "best-fit" value

Table 9.3: Control settings for the *profile-t* confidence interval search algorithm [33].

9.3.5.1 Desired confidence level

By default DynaFit will search for the limits of the non-symmetrical confidence intervals at the 95% probability level:

```
[settings]
{ConfidenceIntervals}
   LevelPercent = 95
```

Other commonly utilized confidence levels are 99% or 90%, depending on the application area. The higher the required confidence level, the wider the confidence intervals. In extreme cases of only very small number of experimental data points being available the confidence interval might be half-opened or even fully opened at 99% or 95% confidence levels. In those specific cases the 90% level interval is probably preferable.

9.3.5.2 Including or excluding "nuisance" parameters

In applied statistics, the term "nuisance parameter" refers to an adjustable parameter in the given regression model that is less important or interesting compared to certain other regression parameters. For example, in fitting enzymatic reaction progress curves to Eqn 6.1, the instrument baseline parameter F_0 must often be treated as an adjustable model parameter in order to achieve a satisfactory goodness of fit. At the same time, we are not really interested at all in the best-fit value of F_0, because it is a property of the instrument, not a property of the biochemical system under investigation.

DynaFit has the ability to turn off the optimization of "nuisance" parameters, such as instrument baselines, initial concentrations of reactants, or molar response coefficients, during the confidence interval search for rate constants and equilibrium constants. This is arranged by the default setting:

```
[settings]
{ConfidenceIntervals}
   OnlyConstants = y
```

It is very important to realize that with these settings the reported confidence interval is merely a *projection* in the multi-dimensional parameter space, whereby all "nuisance" parameters are held fixed at their best-fit values and only rate constants or equilibrium constants are subjected to least-squares optimization. Therefore the reported confidence interval limits are always narrower (i.e., more "optimistic") compared to the scenario where "nuisance" parameters are allowed to float during the confidence interval search (`OnlyConstants = n`).

9.3.5.3 Joint inference regions or marginal confidence intervals

The value of the control parameter `JointProbability` (**y** or **n**) determines which type of confidence interval will be computed by DynaFit. There are two choices, both of which are explained in detail in ref. [33].

```
[settings]
{ConfidenceIntervals}
   JointProbability = n ; | y
```

The default choice is the limits of the *marginal confidence interval* defined by Eqn (9.2), where S_t is the maximum allowed value of the sum of squares at either end of the confidence interval; $S(\hat{\theta})$ is the sum of squares value for the optimal or "best-fit" parameter vector $\hat{\theta}$; N is the number of experimental data points; P is the number of optimized model parameters; and $t^2(N-P;\alpha/2)$ is the "value that isolates an area $\alpha/2$ under the fight tail of the Student's t distribution with $N-P$ degrees of freedom" [33].

$$S_t \leq S(\hat{\theta})\left[1+\frac{t^2(N-P;\alpha/2)}{N-P}\right] \tag{9.2}$$

The marginal confidence interval limits for a particular model parameter are computed under the assumption that the "true" values of the remaining parameters are equal to their best-fit values as determined by the least-squares estimate.

Under the alternate settings (`JointProbability = y`) DynaFit will compute the limits of the *joint confidence region* defined by Eqn (9.3), where $F(P;N-P;\alpha)$ is the "value which isolates an area α under the right tail of Fisher's F distribution with P and $N-P$ degrees of freedom" [33].

$$S_F \leq S(\hat{\theta})\left[1+\frac{P}{N-P}F(P;N-P;\alpha)\right] \tag{9.3}$$

Joint confidence regions are always wider (less "optimistic" but in some sense probably more "realistic") than the corresponding marginal confidence intervals. The choice of the given type of confidence interval, joint or marginal, will depend on the particular problem under investigation.

9.3.5.4 Increase in the residual sum of squares

The profile-t method for confidence interval estimation [2] assumes that each individual data point is statistically independent. However, in many types of experiments this requirement is not satisfied. For example in *continuous* enzyme assays we often collect hundreds or even thousands of measurements while following changes in some physical quantity (fluorescence or absorbance) over time.

One very important point to emphasize is that in similar "continuous" experiments the individual data points are not statistically independent, but rather are strongly statistically correlated. Johnson *et al.* [19, 18] proposed an empirical work-around, by requiring that the confidence interval limits are defined an arbitrary increase in the residual sum of squares. The desired level of increase is defined by the following control parameter:

```
[settings]
{ConfidenceIntervals}
   SquaresIncreasePercent = 0 ; | 10 for continuous assays
```

Johnson *et al.* [19, 18] recommended a 10% or 25% increase as a suitable target value for confidence interval estimation in the analysis of "continuous" experiments. Our experience shows that either 5% or 10% increase is also suitable.

The importance of this particular control parameter cannot be overstated. It is imperative that DynaFit users always utilize at least 5%, but preferably 10%, increase in the sum of squares in the analysis of all "continuous" experimental data.

9.3.5.5 Restart of data fitting (false minimum)

```
{ConfidenceIntervals}
   MaxRefitIterations = 2
   RefitImprovedLower = y
   RefitImprovedHigher = n
```

In certain exceptionally difficult cases, depending on the initial estimate of model parameters, the initial least-squares fit fails to locate the global minimum on the least-squares hyper-surface. In those unfavorable cases, the systematic confidence interval search might a best-fit value of the residual sum of squares that is *lower* ("better") than the SSQ value discovered by the initial fit.

If a lower ("better") value of the residual sum of squares is in fact discovered during the confidence interval search, the control parameter `MaxRefitIterations` determines whether or not an attempt should be made to restart the entire least-squares minimization, including a subsequent confidence interval search, from the improved estimate of model parameters. If `MaxRefitIterations = 0`, no restart will ensue. The default value is "2", based on practical experience showing that the restart might be repeated more than once to successfully locate the true global minimum.

The control parameter `RefitImprovedLower` enables the restart if and when the improved (in terms of the associated SSQ) value of the given parameter of interest is lower than the initially identified "best-fit" value. Conversely, the control parameter `RefitImprovedLower` enables the restart if and when the improved value of the given parameter of interest is higher than the initially identified "best-fit" value.

The choice of the default values shown above (`RefitImprovedLower = y`, `RefitImprovedHigher = n`) is informed by practical experience. In particular, DynaFit is often utilized to determine the best-fit values of *dissociation rate constants* or *dissociation equilibrium constants*. Experience shows that repeated restarts of the least-squares minimization algorithm is not productive when, in fact, the dissociation constant essentially tends to infinity, signifying that the associated reaction step is in fact missing from the postulated reaction mechanism.

9.3.6 The Differential Evolution (DE) algorithm

The Differential Evolution (DE) algorithm [29] is an evolutionary strategy (ES) computational approach aimed at finding *global optima* of various types. Within DynaFit, the DE algorithm is used either for least squares data fitting (i.e., minimizing the residual sum of squares) or for the D-optimal [1] design of experiments (i.e., maximizing the determinant of the Fisher information matrix).

The operations of the DE algorithm are controlled by the following set of control parameters. A brief description of each parameter is shown in *Table 9.4*.

```
{DifferentialEvolution}
   PopulationSizeFixed             = 0
   PopulationSizeMinimal           = 300
   PopulationSizePerParameter      = 5
   PopulationSizePerOrderOfMag     = 3
   MinimumGenerationsPerParameter  = 5
   MaximumGenerationsPerParameter  = 100
   MaximumEvolutions               = 4
   MinimumEvolutions               = 1
   RandomSeed                      = 1234
   RootMeanSquareMin               = 0
   RootMeanSquareMax               = 0
   OutputTextFileType              = csv ; | txt
```

Space constraints do not permit a detailed, in-depth explanation of the basic principles underlying the DE algorithm. The reader is encouraged to consult the original source [29].

Briefly, if p_f is set to any positive value, then DynaFit uses that particular population size and the control parameters p_m, p_p, and p_o are ignored. Otherwise the population size is set to $p_m + N \times p_p + M \times p_o$, where N is the number of optimized model parameters and M is sum total of the orders magnitude spanned by the optimized parameter ranges.

Each evolutionary sequence is terminated either when convergence is reached (a suitable set of convergence criteria will described in a forthcoming publication) or when then the generation count reaches $N \times g_{\max}$. However the evolutionary cycle will proceed at least $N \times g_{\min}$ regardless of convergence criteria.

DynaFit will simulate at least $e_{\min} + 1$ separate evolutions and compare the final results. If the final (presumably, globally optimal) sets of model parameters agree to

Parameter	Default	Explanation
PopulationSizeFixed	0	= p_f, see comments in text
PopulationSizeMinimal	300	= p_m
PopulationSizePerParameter	5	= p_p
PopulationSizePerOrderOfMag	3	= p_o
MinimumGenerationsPerParameter	5	= $g_{\min}$
MaximumGenerationsPerParameter	100	= $g_{\max}$
MaximumEvolutions	4	= $e_{\max}$
MinimumEvolutions	1	= $e_{\min}$
RandomSeed	1234	Seed for built-in random number generator
RootMeanSquareMin	0	Stopping criterion for “best” RMS value ($r_{\min}$)
RootMeanSquareMax	0	Stopping criterion for “worst” RMS value ($r_{\max}$)
OutputTextFileType	csv	Text file format for intermediate results

Table 9.4: Control settings for the Differential Evolution algorithm [29].

within 6 significant digits, no further evolutions will be attempted and the program stops. Otherwise DynaFit will attempt at most $e_{\max}$ separate evolutions, trying to obtain repeatable results.

The DE algorithm should be considered an experimental feature in DynaFit. Practical experience shows that although the changes of reaching a *global* minimum on the least-squares hypersurface are vastly improved when using DE, in comparison with the classic Levenberg-Marquardt algorithm [30], global convergence is *not* guaranteed.

To utilize DE global minimization instead of the default Levenberg-Marquardt algorithm, the DynaFit script must contain the `algorithm` keyword, as follows:

```
[task]
   task = fit
   algorithm = differential-evolution ; or algorithm = DE
   data = ...
```

Lastly, it should be noted that in the current implementation the DE algorithm is excruciatingly slow, requiring typically many hours of computing time on hardware that is considered “top of the line” at the time of writing, in 2014 (for example, a 3.4 GHz 64-bit multicore microprocessor).

9.3.7 *Default parameter constraints*

DynaFit implements a very simple version of constrained least-squares minimization, following a restart algorithm described by Duggleby [11]. The following block of initialization code controls how parameter constraints are handled.

```
{Constraints}
   Constants                   = 1000000
   Responses                   = 1000000
   Concentrations              = 1000
   AllParametersConstrained    = y
   AllParametersRelativeBound = 1000000
```

Parameter	Default	Explanation
Constants	10^6	= B_k, see comments in text
Responses	10^6	= B_r
Concentrations	10^3	= B_c
AllParametersConstrained	**y**	Constrained optimization ? (**y** or **n**)
AllParametersRelativeBound	10^6	= B_p

Table 9.5: Control settings for parameter constraints.

All constraint settings (B_k, B_r, and B_c in *Table 9.5*, or B_x in general) must be positive numbers. Their interpretation depends on whether or not the given constraint is less or greater than unity.

For constraints that are greater than unity, the bounds for the given parameter are set from $p_0 \times 1/B_x$ to $p_0 \times B_x$, where p_0 is the initial estimate of the given model parameter.

For illustration, let us consider the default bounds, $B_k = 10^6$, for all rate constants and equilibrium constants. This means that all optimized rate or equilibrium constants will be allowed to float within 12 orders magnitude. For example, in the specific case of a particular rate constant, for which the initial estimate is $k = 0.123$ s^{-1}, the optimization bounds will range from $k_- = 1.23 \times 10^{-7}$ s^{-1}to $k_+ = 1.23 \times 10^{+5}$ s^{-1}.

For constraints that are smaller than unity, the bounds for the given parameter are set from $p_0 \times (1 - B_x)$ to $p_0 \times (1 + B_x)$, where p_0 is the initial estimate of the given model parameter.

For example, if wished to allow all adjustable concentrations to vary only within a 10% titration error limit, we would include the following initialization code in the particular DynaFit script:

```
[settings]
{Constraints}
   Concentrations = 0.1 ; = 10% titration error
```

In the case of algebraic models (`data = generic`), DynaFit assumes that all arbitrary algebraic model parameters have relative bounds set from 10^{-6} fold to 10^{+6} fold of the initial estimate. To perform unconstrained optimization, set `AllParametersConstrained = n` in the given script file.

9.3.7.1 Arbitrary parameter constraints

DynaFit can impose any arbitrary constraints on adjustable model parameters. The appropriate notation is shown schematically below:

```
   P = 1.23 ? (0.45 .. 6.7)
```

In this code snippet, **P** is an arbitrary model parameter, such as a rate constant or equilibrium constant. The value 1.23 is the initial estimate. The question mark indicates that this parameter indeed should be treated as adjustable in the nonlinear regression. The 0.45 is the lower limit, and the value 6.7 is the upper limit.

9.3.7.2 Important caveat

It should be noted that the simple restart algorithm for "bouncing off" parameter bounds, as implemented in DynaFit and in Duggleby's classic DNRP53 computer program [11], performs very poorly if the bounds are very narrow. The least-squares minimization convergence can be very extremely slow and the minimum may not be found even after very many steps. The Differential Evolution algorithm as implemented in DynaFit is more suitable for handling many narrow parameter constraints, but again the convergence is very slow. Efforts are currently under way to implement a more suitable constrained optimization algorithm based on Quadratic Programming (QP).

9.3.8 Pre-processing of raw experimental data

DynaFit allows certain types of common preprocessing of raw data sets, such as selecting only a portion of the complete raw data trace, smoothing, and filtering. The control parameters are listed below and summarized in *Table 9.6*.

```
{Filter}
   PointsPerDataset    = 0
   ExponentialSpacing  = n
   ReadEveryNthPoint   = 0
   SkipFirstNPoints    = 0
   XMin                = 0
   XMax                = 0
   XShift              = 0
   YMin                = 0
   YMax                = 0
   XFirstMesh          = 0.00001
   TimeInitialRate     = 1
   PrintInitialRate    = y
   SmoothData          = n
   SmoothingMethod     = savitzky-golay ; | average
   SavitzkyGolayWindow = 10
   SavitzkyGolayDegree = 4
   ExtrapolationMethod = quadratic ; | linear
   SmoothingPasses     = 4
   AverageReplicates   = n
   ZeroBaselineSignal  = n
```

9.3.9 Piecewise linear fit

DynaFit can perform a simple piecewise linear fit of the experimental data. To trigger this type of analysis, the DynaFit script must contain the following notation:

```
[task]
   task = fit
   data = piecewise-linear
```

The control settings for the piecewise linear fitting algorithm are shown below.

```
{PieceWiseLinearFit}
   Points   = 0
   Segments = 4
   Time     = 0
   Overlap  = n
```

If `Points` is set to a nonzero value, DynaFit will fit this many data points at a time to the straight-line model. Otherwise if `Segments` is nonzero, DynaFit will divide the raw data trace into this many equal-length segments and fit those individually to the straight-line model. Otherwise if `Time` is nonzero, DynaFit will divide the abscissa into segments that have the same duration and fit those segments to the straight line. If `Overlap` is set to **y**, DynaFit will reuse the last data point in a preceding segment as the starting point in the following segment.

9.3.10 Adjusting output from DynaFit

The user can, to a limited degree, adjust certain aspects of the output produced by DynaFit. The requisite control parameters are listed below.

```
{Output}
   UseDefaultDirectory  = y
   Autocorrelations     = n
   ConfidenceBands      = n
   PredictionBands      = n
   WriteTXT             = n
   WriteEPS             = n
   ColorEPS             = y
   ResidualsEPS         = y
```

Parameter	Default	Explanation
PointsPerDataset	0	If nonzero, the data set will be sampled to contain this many data points.
ExponentialSpacing	**n**	If **y**, the data set will be sampled to create exponential (a.k.a. “logarithmic” spacing) of data points.
ReadEveryNthPoint	0	If nonzero, only every *n*th raw data point will be read and analyzed.
SkipFirstNPoints	0	If nonzero, this many data points will be deleted from the start of raw data set.
XMin	0	If nonzero, experimental data up to (and including) this X-coordinate will be ignored.
XMax	0	If nonzero, experimental data higher than this X-coordinate will be ignored.
XShift	0	If nonzero, this value will be added to the X-coordinate of all data points.
YMin	0	If nonzero, experimental data up to (and including) this Y-coordinate will be ignored.
YMax	0	If nonzero, experimental data higher than this Y-coordinate will be ignored.
XFirstMesh	0	If nonzero, the model interpolation mesh will be plotted starting from this X-value forward.
TimeInitialRate	0	The “initial” reaction rate will be computed at this time coordinate.
PrintInitialRate	**y**	If **y**, DynaFit will report “initial” reaction rates in the output.
SmoothData	**n**	If **y**, DynaFit will perform smoothing of raw data before the analysis proper.
SmoothingMethod	**savitzky-golay**	If **savitzky-golay**, DynaFit will perform Savitzky-Golay smoothing using a modification of a familiar algorithm. [28] If **linear**, DynaFit will perform a simple averaging of neighboring data points.
SavitzkyGolayWindow	10	The width of the Savitzky-Golay smoothing window.
SavitzkyGolayDegree	4	Polynomial degree to be used in the Savitzky-Golay smoothing algorithm.
ExtrapolationMethod	**quadratic**	If **quadratic**, the first and last several data points from Savitzky-Golay smoothing will be extrapolated using a quadratic function. Otherwise DynaFit will perform a linear extrapolation.
SmoothingPasses	4	Number of times the given smoothing algorithm should be applied.
AverageReplicates	**n**	If **y**, any replicated data points present in the raw data set will be automatically averaged before analysis.
ZeroBaselineSignal	**n**	If **y**, the Y-coordinate of the first time-point will be set to zero and the rest of the data set will be adjusted accordingly.

Table 9.6: Control settings for pre-processing of raw experimental data.

```
WriteTeX            = n
XAxisUnit           =
XAxisLabel          =
YAxisLabel           =
BlackBackground      = y
IncludeXZero         = n
IncludeYZero         = n
InitialRateDigits    = 4
StartDefaultBrowser  = n
PlotRatesLogarithmic = n
PlotStateLogarithmic = n
SignificantDigits    = 1
```

Most of the control parameters listed above are self-explanatory.

When `IncludeYZero` is set to **y**, DynaFit plots will always include zero on the Y-axis. Otherwise DynaFit will use an internal “intelligent” algorithm to decide whether the zero point should be included. Similar considerations apply to `IncludeXZero`.

The parameter `InitialRateDigits` sets the number of significant digits in the automatically generated concentration vs. velocity data files.

The parameter `XAxisUnit` is used to add a concentration unit to the horizontal axis of graphs generated in the analysis of initial rate or equilibrium binding data. In those case, DynaFit normally labels the horizontal axis with the reaction species named on the `variable` line in the `[data]` section of the script. The `XAxisUnit` value is the appended to the species name enclosed in square brackets.

The parameter `SignificantDigits` determines how many significant digits of the formal standard error from nonlinear regression should be printed in the output. The best-fit value is then automatically rounded to the same number of decimal places. In special cases when the formal standard error is larger than the best-fit value, the best-fit value is always rounded to two significant digits. Setting `SignificantDigits = 0` will disable any automatic rounding and all numerical values will be printed out with the default precision of six significant digits.

... ...

By default DynaFit does *not* draw confidence bands or prediction bands around best-fit model curves. To arrange for this one must modify the default output settings, as is shown in the initialization code fragment immediately below.

```
[settings]
{Output}
   ConfidenceBands = y ; and/or :
   PredictionBands = y
```

For an example of a model confidence band plotted around the best-fit model curve, see *Figure 7.5* in Chapter 7. As an important caveat, it should be noted than if or when the fitting model is severely over-parameterized the inference bands grow extremely wide and/or are extremely jagged due to inevitable numerical instabilities in the computation of nonlinear regression leverages, h_{ii} (i.e., the diagonal elements of the “hat matrix” [2, p. 27]).

If `ConfidenceBands = y`, DynaFit will plot two envelope curves (upper and lower) around each best-fit model curve, as the *model confidence band*. This is the band the encloses an area where, at the given confidence level, we can reasonably expect *all possible model curves* to lie given the statistical uncertainty in the available experimental data.

Depending on the value of `JointProbability` (**n** or **y**) DynaFit will compute either the “1-α approximate inference interval” [2, p. 59] according to Eqn (9.4), or the “1-α approximate inference band” [2, p. 60] according to Eqn (9.5).

$$f_m(\mathbf{x}_0,\theta)\ \pm\ s\ ||\mathbf{v}_0^{\mathrm{T}}\ \hat{\mathbf{R}}_1^{-1}||\ t(N-P;\alpha/2) \tag{9.4}$$

$$f_m(\mathbf{x},\theta)\ \pm\ s\ ||\mathbf{v}^{\mathrm{T}}\ \hat{\mathbf{R}}_1^{-1}||\ \sqrt{F(P;N-P;\alpha)} \tag{9.5}$$

For explanation of the mathematical symbols utilized in Eqns (9.4) and (9.5), please consult ref. [2, pp. 59-60], from which these equations are copied.

If `ConfidenceBands = y`, DynaFit will plot two envelope curves (upper and lower) around each best-fit model curve, as the *model confidence band*. This is the band the encloses an area where, at the given confidence level, we can reasonably expect *any additional (hypothetical) data points* to lie, given the uncertainty in the currently available experimental data.

The particular width of the data prediction band again depends the setting of `JointProbability` (**n** or **y**). For example, with `JointProbability = n` the data prediction band is computed according to Eqn (9.6), whereas with `JointProbability = y` the width of the data prediction band is given by Eqn (9.7). In Eqns (9.6) and (9.7), h_{ii} is the nonlinear regression *leverage* of the ith data point and w_i is the corresponding statistical weight.

$$f_p(\mathbf{x}_0,\theta)\ \pm\ s\,t(N-P;\alpha/2)\ \sqrt{\frac{1+h_{ii}}{w_i}} \tag{9.6}$$

$$f_p(\mathbf{x},\theta)\ \pm\ s\ \sqrt{F(P;N-P;\alpha)}\ \sqrt{\frac{1+h_{ii}}{w_i}} \tag{9.7}$$

It could be shown that the data prediction bands are by definition always wider than model confidence bands.

9.3.11 Confidence intervals: Monte-Carlo method

DynaFit can be used to investigate confidence intervals for nonlinear model parameters using the Monte-Carlo method. [32] This is arranged by including the following code at the start of the DynaFit script:

```
[task]
   data = ...
```

```
task = fit
confidence = monte-carlo
```

The default values of relevant control parameter are listed below and are briefly summarized in *Table 9.7*.

```
{MonteCarlo}
   PerformInitialFit          = y
   Runs                       = 1000
   RandomizationMethod        = simulate
   Distribution               = normal
   StandardDeviationSource    = fit
   StandardDeviation          = 1.2
   SignificantDigits          = 4
   HistogramBuckets           = 20
   TruncateMeanPercent        = 5
   ColorOutput                = y
   RandomizationSeed          = 1267
   ConcentrationErrorPercent  = 0
   OriginalEstimates          = n
   ConfidenceLevel            = 95
```

Parameter	Default	Explanation
PerformInitialFit	**y**	See comments in the main text.
Runs	1000	The number of synthetic data sets to analyze.
RandomizationMethod	**simulate**	Otherwise **shuffle** or **shift**. See comments in text.
Distribution	**normal**	Statistical distribution of pseudo-random errors.
StandardDeviationSource	**fit**	Source for the magnitude of pseudo-random errors.
StandardDeviation	1.2	The magnitude of pseudo-random noise.
SignificantDigits	4	Number of significant digits in the simulated data.
HistogramBuckets	20	Number of histogram buckets to use for the summary of results.
TruncateMeanPercent	5	How to compute truncated means for the summary of results.
ColorOutput	**y**	If **n**, DynaFit will create gray-scale images summarizing the results.
RandomizationSeed	1267	Seed for the random number generator.
ConcentrationErrorPercent	0	If nonzero, DynaFit will randomly introduce “titration error” of this magnitude.
OriginalEstimates	**n**	If **y**, DynaFit will start each minimization form the original estimates of model parameters. Otherwise from the best-fit estimates, after the initial least-squares minimization.
ConfidenceLevel	95%	If nonzero, DynaFit compute “truncated” confidence bounds by excluding (in this case) 5% of the extreme values found in the Monte-Carlo interval search.

Table 9.7: Control settings for Monte-Carlo investigations of confidence intervals.

A number of the control parameters listed in *Table 9.7* are self-explanatory. The remaining parameters are discussed below.

9.3.11.1 Performing the initial least-squares-fit

If `PerformInitialFit` is set to **y**, DynaFit will perform an initial least-squares fit starting from the initial estimates given by the user, and then proceed to investigate the Monte-Carlo confidence intervals of the best-fit parameter estimates. This is by far the most common method of utilizing Monte-Carlo confidence intervals in DynaFit.

Otherwise, if `PerformInitialFit` is set to **n**, DynaFit will consider the initial estimates of model parameters to be "best-fit" solution already and will immediately proceed to investigate their Monte-Carlo confidence intervals in the usual manner. This feature is useful under certain special circumstances. In that case the DynaFit script will include the following initialization code.

```
[settings]
{MonteCarlo}
   PerformInitialFit = n
```

9.3.11.2 Randomization methods

The `RandomizationMethod` parameter can legitimately attain one of three values, which are explained below.

- **simulate**: DynaFit will use a *random number generator* to compute a pseudo-random deviation from the best-fit model at each value of the independent variable. The particular statistical distribution to be used, and the magnitude of the deviate (e.g. the standard deviation) is explained below.
- **shuffle**: DynaFit will utilize the *residuals of fit* from the initial least-squares regression. Each value of the independent variable, in each data set, will be assigned a randomly selected residual.
- **shift**: DynaFit will again utilize the *residuals of fit* from the initial least-squares regression, but this time the residuals will not be fully randomized. Instead, the sequence of residuals will be shifted (and then wrapped around) along the independent variable axis, starting from a randomly selected data point.

The **shuffle** method attempts to circumvent any particular distributional assumption (e.g., Normal), which may or may not be applicable. The **shift** method additionally attempts to remove the assumption that the experimental errors are serially uncorrelated. Neither of these simplifying assumptions (Normal distribution, absence of serial correlation) is very realistic when applied to real-life data set. For example, we have shown [21] that the random experimental errors in stopped-flow protein folding experiments are very strongly serially correlated.

Thus the **shuffle** and **shift** randomization methods are likely to provide more realistic results then conventional Monte-Carlo simulations. However, it is very important to note that both method rely on the assumption that the random experimental errors are have *constant variance*, independent on the value of the experimental signal. That assumption may or may not hold, depending on the case.

9.3.11.3 Statistical distributions

The `Distribution` parameter can legitimately attain one of four values, which are explained below.

- **normal**: DynaFit will simulate pseudo-random noise with zero mean and the appropriate standard deviation (see below) drawn from the normal or Gaussian distribution.
- **cauchy**: DynaFit will simulate pseudo-random noise with zero location and the appropriate scale drawn from the Cauchy distribution.
- **logistic**: DynaFit will simulate pseudo-random noise with zero location and the appropriate scale drawn from the logistic distribution.
- **uniform**: DynaFit will simulate pseudo-random noise with zero location and the appropriate range drawn from the uniform distribution.

It should be noted that the uniform distribution practically never occurs in scientific practice and is included as an option only for exploratory purposes. In invoking the Cauchy and logistic distributions, the appropriate scale is formally represented in the control settings file as `StandardDeviation` (see below). The same applies to uniform distribution range.

The choice of the given distribution (in particular either normal or Cauchy) will depend on the distributional properties of the experimental noise actually observed in the given system under investigation. Normal distribution of noise has often been confirmed in a variety of experimental settings. However, the Cauchy distribution should not be ignored because it provides the ability to simulate moderately “spiky” data, with a realistic representation of outliers.

9.3.11.4 Magnitude of pseudo-random noise

The `StandardDeviationSource` parameter can legitimately attain one of three values, which are explained below.

- **fit**: DynaFit will use as the standard deviation (assuming normal distribution) the *standard deviation of fit* from the initial least squares regression, multiplied by the particular values of the `StandardDeviation` parameter.
- **data**: DynaFit will use as the standard deviation (assuming normal distribution) the standard error of measurement supplied explicitly with each individual data point.

- **explicit**: DynaFit will use as the standard deviation (in the case normal distribution) or scale (in the case of Cauchy, logistic, or uniform distributions) the value specified explicitly by the `StandardDeviation` parameter.

Please note that with `StandardDeviationSource = data`, the program expects the input file to always contain *three columns* representing the independent variable, the experimentally observed dependent variable, and the associated standard error of measurement. Of course the standard error can be computed as having any particular relationship to the experimental signal. In this fashion the Monte-Carlo simulations in DynaFit can be performed with virtually any arbitrary distributional assumptions.

9.3.11.5 Simulated titration (volume delivery) errors

In Monte-Carlo investigations of confidence intervals described in the literature (for example, see ref. [32]) it is assumed that the concentration of reactants are known with perfect accuracy and only the observed experimental signal is affected by pseudo-random errors. However, in the study of reaction progress curves, the *shape* of each progress curve can subtly change due to slight variations in the initial concentrations. To account for this fact, DynaFit allows us to perform Monte-Carlo investigations of confidence intervals by introducing pseudo-random errors not only into the simulated experimental signal, but also into the initial concentrations of reactants.

For example, to perform a Monte-Carlo investigation of confidence intervals corresponding to a given progress curve experiment, under the assumption that all initial concentrations are affected by ten percent titration error (i.e., random error in volume delivery), we could use the following initialization code:

```
[settings]
{MonteCarlo}
   ConcentrationErrorPercent = 10
```

Practical experience suggests that the confidence intervals for rate constants obtained in this fashion are much more realistic than if the titration error is presumed to be absent.

9.3.12 Systematic scan of initial estimates

DynaFit has the ability to evaluate a very large number of initial estimates and select for actual least-squares optimization only those that appear most promising, in terms of the agreement between the theoretical model (simulated with each set of initial estimates) and the experimental data. This is arranged by the special “curly bracket, comma delimited” notation best explained by way of an example.

Let us assume that the given fitting model for a global set of reaction progress curves contains five rate constants labeled k_1 through k_5. Let us also assume that there is very little information available regarding the possible values of these constants, other than to assume their numerical values potentially could all span six orders of magnitude, from 10^{-3} to 10^3. Finally, let us assume that the problem under investigation is somewhat sensitive to the initial estimates, such that we would like to examine all possible estimates stepping by only one order of magnitude (i.e., factor of 10).

In this particular situation DynaFit could be used to rank all possible $7^5 = 16,807$ initial estimates, in terms of the residual sum of squares they produce, by specifying the following input code:

```
[constants]
   k1 = {0.001, 0.01, 0.1, 1, 10, 100, 1000} ?
   k2 = {0.001, 0.01, 0.1, 1, 10, 100, 1000} ?
   k3 = {0.001, 0.01, 0.1, 1, 10, 100, 1000} ?
   k4 = {0.001, 0.01, 0.1, 1, 10, 100, 1000} ?
   k5 = {0.001, 0.01, 0.1, 1, 10, 100, 1000} ?
```

Indeed this notation leads to 16,807 initial parameter estimates to evaluate, because there are *seven* different starring values (0.001, 0.01, ... 100, 1000) to examine for *five* different rate constants, which leads to $7 \times 7 \times 7 \times 7 \times 7 = 7^5 = 16,807$ different combinations of starting values. With the above notation in the input script file, DynaFit will perform the following sequence of steps:

1. Evaluate the sum of squared deviations for all 16,807 initial parameter estimates.
2. Rank the results, from lowest to highest residual sum of squares.
3. Report a certain number of the best initial estimates in the final report.
4. Perform the full least-squares fit starting from a certain number of the best initial estimates.
5. Rank the results of fit, from lowest to highest residual sum of squares.
6. Report the best-fit parameter estimates for all parameter combinations that were subjected to full least-squares optimization.
7. Identify the overall best-fitting combination in terms of the final (fully optimized) residual sum of squares.

The operation of this algorithm are controlled by the following initialization code:

```
{EstimateScan}
   EstimatesMax    = 10000
   ReportSizeMax   = 1000
   RefineEstimates = 10
```

The value of `EstimatesMax` determines how many combinations of individual parameters' estimates DynaFit should attempt to handle. It is important to realize that, as the number of rate constants increases in a relative modest fashion, the total number of *combinations* of initial estimates grows very rapidly.

Let us assume that we wish to examine *seven* different estimates for each rate constant. For example, this might involve a certain "middle" value of a particular rate constant (such as "1") surrounded by three lower values, stepping by an order of magnitude (0.001, 0.01, 0.1), and three higher values, again stepping by an order of magnitude (10, 100, 1000). How many combinations of initial estimates are involved, depending on the number of rate constants (n) we wish to treat in this way? The answer is illustrated in the table below.

n	estimates
1	$7^1 = 7$
2	$7^2 = 49$
3	$7^3 = 343$
4	$7^4 = 2{,}401$
5	$7^5 = 16{,}807$
6	$7^6 = 117{,}649$

Practical experience shows that the maximum number of rate constants or other model parameters that can be systematically searched in this way is approximately four, with approximately seven estimates per parameter.

The value of `ReportSizeMax` determines how many of the best-ranked initial estimates, among those that were actually examined, should be included in the final report. The default value (1,000) is probably the largest realistic number. For approximately 10,000 initial estimates included in the final report the file size grows exceedingly large and the report become unwieldy.

The value of `RefineEstimates` determines how many of the best-ranked initial estimates should be subjected to full least-squares optimization. The default value (10) is almost certainly too low in almost all except perhaps the easiest problems examined to date. A much more realistic value is on the order of 100 or even 1,000 initial estimates subjected to full optimization, depending on the particular problem.

9.3.13 Fast exponential fit

DynaFit provides a specialized algorithm for exponential or multi-exponential fitting. The exponential models are inherently nonlinear and therefore usually require the user to specify initial estimates or both the exponential rate constants and the associated amplitudes. However, the algorithm embedded in DynaFit does not require such estimates. For further details see refs. [23, 24].

It is important to note that the tradeoff for fast execution and the lack of initial estimates means that the algorithm described in [23, 24] occasionally provides on a very crude estimate of the optimal solution. Therefore, it is often necessary to refine this initial estimate by using an more sophisticated method.

```
{ExponentialFit}
   Degree            = 4
   Automatic         = y
   AllowOscillations = n
   TinyAmplitudes    = 0 ; | 0.000001
   RefineEstimate    = y
```

Parameter	Default	Explanation
Degree		Maximum degree of the exponential function. The algorithm will optionally select any appropriate degree at most equal to this.
Automatic	**y**	Whether or not to enable automatic selection of the exponential degree.
AllowOscillations	**n**	Whether or not the multi-exponential function is allowed to contain oscillatory terms (irrational rate constants and amplitudes leading essentially to sine / cosine functions).
TinyAmplitudes	**0**	Maximum acceptable amplitude of any oscillatory terms.
RefineEstimate	**y**	Whether or not the initial estimate should refined by a conventional (iterative) least-squares data fitting algorithm.

Table 9.8: Control settings for automatic multi-exponential fit.

It should also be noted that the algorithm works correctly only if the data points are spaced on the time axis by exactly identical increments. If this requirement is not satisfied, the algorithm can still be used by it is then necessary to always perform a refinement of the initial estimate by using the usual least-squares fitter.

9.3.14 Optimal design of experiments

DynaFit has a capability to provide advice on the optimal design of experiments. The options are controlled by the following settings. See *Table* 9.9 for a brief explanation and the text below for details.

```
{OptimalDesign}
   Algorithm = AS ; | DE | BFGS
   Function  = D  ; | T  | E  | V
```

Parameter	Default	Explanation
Algorithm	**AS**	Optimization algorithm. See text for details.
Function	**D**	Optimization criterion. See text for details.

Table 9.9: Control settings for optimal experiment design.

Optimization algorithms

DynaFit offers a choice among three constrained optimization algorithms:

1. `AS`: A heavily modified variant of the **Active Set** algorithm initially developed by Hager and Zhang [13, 14, 15].
2. `DE`: An adaptation of the **Differential Evolution** algorithm due to Price *et al.* [29]
3. `BFGS`: A variant of a classic algorithm is generally known as L-BFGS-B, or limited memory bound-constrained Broyden-ŰFletcher-ŰGoldfarb-ŰShanno algorithm [6, 34].

The `AS` algorithm is probably the preferred method. It converges very rapidly in almost all cases tested, but it has been observed to occasionally diverge in certain particularly challenging design problems.

The `DE` algorithm seems to always guarantee the correct solution, but the convergence can be excruciatingly slow. It is therefore recommended for follow-up computations, to verify results obtained by `AS` in especially important cases.

The `BFGS` algorithm is experimental and not yet thoroughly tested. It seem to converge exceptionally rapidly in certain "easy" design problems. Overall the `AS` algorithm probably strikes the balance between sufficient speed of computation and guaranteed convergence to a truly optimal design.

Optimization criteria

DynaFit offers a choice among four optimization criteria for the purpose of optimal experimental design:

1. `D`: *Determinant* of the Fisher information matrix (*D*-optimal experimental design [12, 1]).
2. `T`: *Trace* of the Fisher information matrix.
3. `E`: *Eigenvalues* of the Fisher information matrix. More precisely, the algorithm will optimize the experimental design such that the *smallest* eigenvalue is maximized.
4. `V`: *Variance* of the optimized model parameters.

It should be noted that as of this writing the optimal design module in DynaFit has not yet been thoroughly tested. Until sufficient experience accumulates with the various options and algorithms, these settings should be treated as experimental and the results should be interpreted with caution.

9.3.15 Model selection

DynaFit has a capability to provide advice on selecting the most plausible fitting model among multiple candidate models. The options are controlled by the following settings. See *Table 9.10* for a brief explanation and the text below for details.

```
{ModelSelection}
   PrioritizeCriterion               = akaike ; | bayesian
   ConfidenceIntervalRangeMax        = 10000
   CoefficientOfVariationMax         = 100
   CoefficientOfVariationSearchMax   = 200
   InformationCriterionDeltaMax      = 5
   InformationCriterionWeightMin     = 0.01
   RelativeSquaresMax                = 1.05
   OnlyConstants                     = n
```

Parameter	Default	Explanation
PrioritizeCriterion	**akaike**	Priority information criterion (AIC or BIC).
ConfidenceIntervalRangeMax	**10000**	Maximum acceptable confidence interval range
CoefficientOfVariationMax	**100**	Maximum acceptable coefficient of variation (CV)
CoefficientOfVariationSearchMax	**200**	Maximum acceptable CV during profile-*t* CI search
InformationCriterionDeltaMax	**5**	Maximum acceptable Δ_{AIC} or Δ_{BIC}
InformationCriterionWeightMin	**0.01**	Minimum acceptable w_{AIC} or w_{BIC}
RelativeSquaresMax	**1.05**	Maximum acceptable relative sum of squares
OnlyConstants	**n**	Ignore other globally optimized parameters

Table 9.10: Control settings for model selection.

9.3.15.1 AIC vs. BIC priority

The second-order (i.e. small-sample corrected) Akeike information criterion (AIC) is defined by Eqn (9.8), whereas the corresponding Bayesian information criterion (BIC) is defined by Eqn (9.9), where *SSQ* is the sum of squared deviations between the experimental data and the theoretical model; n_{D} is the number of experimental data points; and n_{P} is the number of optimized model parameters.

$$AIC = n_\mathrm{D} \ln \frac{SSQ}{n_\mathrm{D}} + 2\,(n_\mathrm{P}+1) + \frac{2\,(n_\mathrm{D}+1)(n_\mathrm{P}+1)}{n_\mathrm{D}-n_\mathrm{P}-2} \tag{9.8}$$

$$BIC = n_\mathrm{D} \ln \frac{SSQ}{n_\mathrm{D}} + \ln n_\mathrm{D}\,(n_\mathrm{P}+1) + \frac{2\,(n_\mathrm{D}+1)(n_\mathrm{P}+1)}{n_\mathrm{D}-n_\mathrm{P}-2} \tag{9.9}$$

The ranking of candidate fitting models according to *AIC* or *BIC* is usually identical, but in some exceptional cases it can be different. The control parameter `PrioritizeCriterion` can have two values, either `akaike` or `bayesian`. Depending on the particular value, the given information criterion is afforded a priority in ranking candidate theoretical models by their relative plausibility. For further details, see refs. [26, 27, 5].

9.3.15.2 Maximum acceptable confidence interval range

The model selection algorithm proceeds in two stages. In the first stage, the adjustable *parameters* appearing in each candidate fitting model are examined, to see how well the parameters are defined by the data. The best way to make this assessment is to examine the *confidence intervals* of each fitting parameter. As a reminder, we can request the computation of confidence intervals by using the “double question mark” notation, according to the example code fragment below:

```
[constants]
   k_1 = 1.234 ?? ; "??" = compute confidence interval
```

The control parameter `ConfidenceIntervalRangeMax` is used to test the ratio of the upper limit of each confidence interval divided by the interval’s lower limit. If the upper/lower limit ratio is higher than the value defined by `ConfidenceIntervalRangeMax`, the particular fitting parameter is considered as not sufficiently well defined by the available experimental data, and thus the candidate model is declared as implausible.

9.3.15.3 Maximum acceptable coefficient of variation

In certain special cases, we might want to attempt a model-selection analysis even without having computed full confidence intervals for adjustable model parameter. Under those circumstances DynaFit will attempt to assess the validity of adjustable model parameters solely on the basis of the *coefficient of variation* for model parameters. A coefficient of variation (in percentage points) is defined as the standard error of the given model parameter, divided by the parameter’s best-fit value, times one hundred. As a reminder, we can request the computation of standard errors by using the “single question mark” notation, according to the example code fragment below:

```
[constants]
   k_1 = 1.234 ? ; "?" = compute standard error only
```

The control parameter `CoefficientOfVariationMax` is used to assess the plausibility of any given candidate model as follows. If the coefficient of variation for *any* globally optimized model parameter is higher than the value defined by `CoefficientOfVariationMax`, the particular fitting parameter is considered as not sufficiently well defined by the available experimental data, and thus the candidate model is declared as implausible.

For continuous experiments of reaction-progress type (`data = progress`, as opposed to `data = progress discontinuous` or any other data type), the coefficient of variation being tested is actually the "empirical" value rather than *CV* as such, as is explained in 9.3.1.3.

9.3.15.4 Maximum acceptable CV during profile-*t* CI search

In extremely unfavorable circumstances the fitting model might be essentially undefined by the available experimental data ("model redundancy"). Practical experience shows that for certain especially ill-conditioned models, the full systematic confidence interval search might produce *spurious bounds* for some model parameters. The existence of such spurious parameter bounds could distort the model discrimination analysis.

This particular control setting was introduced in order to handle such unfavorable situations, according to the following rule. If the confidence interval appears closed for a given parameter while, at the same time, the value of CoefficientOfVariationSearchMax is greater than specified limit, then the particular parameter will be deemed insufficiently well defined in the context of model discrimination analysis.

To disable this feature, the initialization parameter CoefficientOfVariationSearchMax can be optionally assigned zero value.

9.3.15.5 Maximum acceptable ΔAIC or ΔBIC

All candidate models that survive the acceptance vs. exclusion test described in sections 9.3.15.2 or 9.3.15.3 are subsequently scrutinized in the second stage of the model selection algorithm, as follows.

According to a model selection strategy explained in detail by Burnham & Anderson [5], a set of n candidate theoretical models should first be arranged in order of increasing AIC values associated with each model. The "best" model (associated with the lowest value of AIC, i.e., $\text{AIC}_1 = \text{AIC}_{\text{min}}$) is assigned $\Delta\text{AIC}_1 = 0$ and the remaining models ($i = 2, 3, \ldots n$) are assigned $\Delta\text{AIC}_i = \text{AIC}_i - \text{AIC}_1$. According to a rule of thumb presented by these authors [5, p. 70]:

The larger $\Delta\mathrm{AIC}_i$ is, the less plausible it is that the fitted model [...] is the [...] best model, given the data [...]. Some rough rules of thumb are available and are particularly useful for nested models:

$\Delta\mathrm{AIC}_i$	Level of Empirical Support of Model i
0–2	Substantial
4–7	Considerably less
> 10	Essentially none

In other words, *all* models characterized by $\Delta\mathrm{AIC}_i$ less than 10 are still somewhat plausible, whereas any given model characterized by $\Delta\mathrm{AIC}_i > 10$ can probably be safely excluded from consideration.

The control parameter `InformationCriterionDeltaMax` allows DynaFit users to set this cut-off value of ΔAIC. Any model that is associated with a greater value of ΔAIC will be deemed implausible by DynaFit.

9.3.15.6 Mimimum acceptable Akaike or Bayesian weight

Burnham & Anderson [5, p. 75] introduced a quantity denoted as "Akaike weight" defined in Eqn (9.10), where $w_i^{(\mathrm{AIC})}$ is the Akaike weight of the ith model under consideration; m is the number of candidate models; and $\Delta\mathrm{AIC}_i$ is the differential information theoretic criterion defined in the immediately preceding section.

$$w_i^{(\mathrm{AIC})} = \frac{\exp\left(-\frac{1}{2}\Delta\mathrm{AIC}_i\right)}{\sum_j^m \exp\left(-\frac{1}{2}\Delta\mathrm{AIC}_i\right)} \tag{9.10}$$

According to these authors, the Akaike weight can be viewed as a *model probability*, that is, a number between zero and unity that measures the statistical probability of model i being the "true" model. The value zero means that the i model si completely impossible, whereas a unit value for $w_i^{(\mathrm{AIC})}$ means that the given model is perfectly certain and therefore unambiguously preferred over any remaining candidate models.

The "Bayesian weight", $w^{(\mathrm{BIC})}$, is defined in same way as the Akaike weight, except for the fact that AIC is replaced by BIC in Eqn (9.10).

The control parameter `InformationCriterionWeightMin` allows DynaFit users to set this cut-off value of $w_i^{(\mathrm{AIC})}$ or $w_i^{(\mathrm{BIC})}$. Any model that is associated with a smaller value of $w_i^{(\mathrm{AIC})}$ or $w_i^{(\mathrm{BIC})}$ will be deemed implausible by DynaFit. Practical experience shows that in order for any given fitting model to be considered at least marginally plausible, its Akaike or Bayesian weight must be at least 0.01 (one percent probability of the model being "true").

9.3.15.7 Maximum acceptable relative sum of squares

Very importantly, the ΔAIC and $w^{(\mathrm{AIC})}$ model acceptance tests described in sections 9.3.15.5 and 9.3.15.6, respectively, cannot be applied for continuous experiments of reaction-progress type (`data = progress`, as opposed to `data = progress discontinuous` or any other data type). The reason is that the individual time-points recorded in such experiments are not statistically independent, and therefore the usual statistical tests and procedures do not apply to them.

In the case of continuous reaction progress experiments, DynaFit uses a heuristic rule of thumb similar to the procedure described by Johnson *et al.* [19, 18] for assessing the uncertainty of model parameters. In the case of continuous assays, the information theoretic criteria AIC or BIC are indeed used for the *ranking* of candidate models, but not for the *acceptance or exclusion* of any given model.

Instead, after the models are first ranked by AIC or BIC, the model selection algorithm in DynaFit then evaluates the ratio of the residual sum of squares associated with each particular model divided by the residual sum of squares associated with the "best" model. All candidate models for which this ratio is greater than the particular numerical value set by `RelativeSquaresMax` are excluded as implausible.

9.3.15.8 Utilize only rate constants for model selection

The control parameter `OnlyConstants` can be used to instruct the model-selection algorithm to consider only microscopic rate constants or equilibrium constants in the process of selecting the optimal regression model. This is useful when one or more candidate models contain globally optimized initial concentrations and/or globally optimized molar response factors.

References

1. Atkinson, A., Donev, A.: Optimum Experimental Designs. Oxford University Press, Oxford (1992)
2. Bates, D.M., Watts, D.G.: Nonlinear Regression Analysis and its Applications. Wiley, New York (1988)
3. Beechem, J.M.: Global analysis of biochemical and biophysical data. Meth. Enzymol. **210**, 37–54 (1992)
4. Brooks, I., Watts, D., Soneson, K., Hensley, P.: Determining confidence intervals for parameters derived from analysis of equilibrium analytical ultracentrifugation data. Meth. Enzymol. **240**, 459–78 (1994)
5. Burnham, K.B., Anderson, D.R.: Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach, 2nd edn. Springer-Verlag, New York (2002)
6. Byrd, R.H., Lu, P., Nocedal, J., Zhu, C.: A limited memory algorithm for bound constrained optimization. SIAM J. Sci. Comput. **16**, 1190–1208 (1995)
7. Byrne, G.D., Hindmarsh, A.C.: Stiff ODE solvers: a review of current and coming attractions. J. Comput. Physics **70**, 1–62 (1987)

8. Dennis, J.E., Gay, D.M., Welsch, R.E.: An adaptive nonlinear least-squares algorithm. ACM Trans. Math. Software pp. 348–368 (1981)
9. Dennis, J.E., Gay, D.M., Welsch, R.E.: Algorithm 573: NL2SOL. ACM Trans. Math. Software **7**, 369–383 (1981)
10. Dennis, J.E., Schnabel, R.B.: Numerical Methods for Unconstrained Optimization and Nonlinear Equations. Prentice-Hall, Upper Saddle River, NJ (1983)
11. Duggleby, R.G.: Regression analysis of nonlinear Arrhenius plots: an empirical model and a computer program. Comput. Biol. Med. **14**, 447–455 (1984)
12. Fedorov, V.: Theory of Optimal Experiments. Adademic Press, New York (1972)
13. Hager, W.W., Zhang, H.: A new conjugate gradient method with guaranteed descent and an efficient line search. SIAM J. Optim. **16**, 170–192 (2005)
14. Hager, W.W., Zhang, H.: Algorithm 851: CG_DESCENT, A conjugate gradient method with guaranteed descent. ACM Trans. Math. Softw. **32**, 113–137 (2006)
15. Hager, W.W., Zhang, H.: A new active set algorithm for box constrained optimization. SIAM J. Optim. **17**, 526–557 (2006)
16. Hindmarsh, A.C.: ODEPACK: a systematized collection of ODE solvers. In: R. Stapleman, M. Carver, R. Peskin, W. Ames, R. Vichnevetsky (eds.) Scientific Computing, pp. 55–64. North Holland, Amsterdam (1983)
17. Huber, P.: Robust Statistics. John Wiley, New York (1981)
18. Johnson, K.A., Simpson, Z.B., Blom, T.: FitSpace Explorer: An algorithm to evaluate multidimensional parameter space in fitting kinetic data. Anal. Biochem. **387**, 30–41 (2009)
19. Johnson, K.A., Simpson, Z.B., Blom, T.: Global Kinetic Explorer: A new computer program for dynamic simulation and fitting of kinetic data. Anal. Biochem. **387**, 20–29 (2009)
20. Kuzmic, P., Hill, C., Janc, J.W.: Practical robust fit of enzyme inhibition data. Meth. Enzymol. **383**, 366–381 (2004)
21. Kuzmic, P., Lorenz, T., Reinstein, J.: Analysis of residuals from enzyme kinetic and protein folding experiments in the presence of correlated experimental noise. Anal. Biochem. **395**, 1–7 (2009)
22. Marquardt, D.W.: An algorithm for least-squares estimation of nonlinear parameters. J. Soc. Ind. Appl. Math. **11**, 431–441 (1963)
23. Martin, J.L., Maconochie, D.J.: The direct fitting of a differential equation to experimental data. J. Physiol. **418**, 7 (1989)
24. Martin, J.L., Maconochie, D.J., Knight, D.R.: A novel use of differential equations to fit exponential functions to experimental data. J. Neurosci. Meth. **51**, 135–146 (1994)
25. Matsumoto, M., Nishimura, T.: Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. ACM Trans. Model. Comput. Simul. **8**, 3–30 (1998)
26. Myung, J.I., Pitt, M.A.: Model comparison methods. Meth. Enzymol. **383**, 351–366 (2004)
27. Myung, J.I., Tang, Y., Pitt, M.A.: Evaluation and comparison of computational models. Meth. Enzymol. **454**, 287–304 (2009)
28. Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical Recipes in C. Cambridge University Press, Cambridge (1992)
29. Price, K.V., Storn, R.M., Lampinen, J.A.: Differential Evolution: A Practical Approach to Global Optimization. Springer Verlag, New York (2005)
30. Reich, J.G.: Curve Fitting and Modelling for Scientists and Engineers. McGraw-Hill, New York (1992)
31. Seber, G.A.F., Wild, C.J.: Nonlinear Regression. Wiley, New York (1989)
32. Straume, M., Johnson, M.L.: Monte Carlo method for determining complete confidence probability distributions of estimated model parameters. Meth. Enzymol. **210**, 117–29 (1992)
33. Watts, D.G.: Parameter estimates from nonlinear models. Methods Enzymol. **240**, 23–36 (1994)
34. Zhu, C., Byrd, R.H., Lu, P., Nocedal, J.: Algorithm 778: L-BFGS-B, FORTRAN routines for large scale bound constrained optimization. ACM Trans. Math. Softw. **23**, 550–560 (1997)