Using SIMFIT to plot non-standard characters and create special effects

Figure 1 shows how the effect of plotting a positive 4:4 rational function in semi-logarithmic space can be enhanced by adding Greek letters, subscripts, and superscripts, and by using powers of ten on the X-legend.

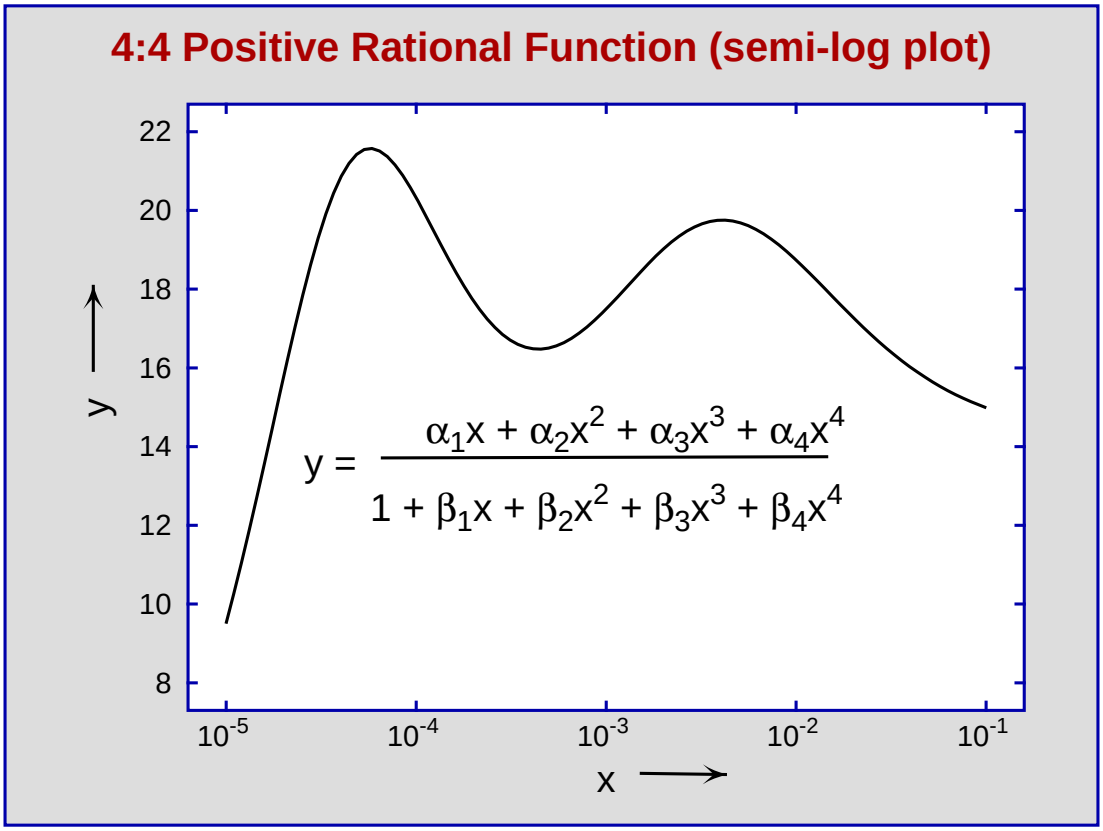


Figure 1: Plotting subscripts, superscripts, and math symbols

This document has details about how to use the SIMFIT package to create such graphs with special characters and extra features as follows:

- Introduction to SIMFIT EPS PostScript files;
- Plotting selected characters as subscripts and superscripts;
- Adding accents to letters in titles, legends, and labels;
- Replacing standard characters by mathematical symbols;
- Editing SIMFIT PostScript files in a text editor;
- Using Postscript specials to add logos etc.; and
- Advanced techniques for LaTeX and PSFrag users.

Contents

1 Introduction to Simfit EPS PostScript files

There are essentially three types of image files as follows.

1. **Bitmaps and compressed bitmaps**

   Raw bitmaps (e.g., `.bmp`) are used to record the characteristics of every pixel in a display or hardcopy, typically a digital photograph. They are limited by the resolution of the captured image and are usually large, also the images break up by pixelation if they are enlarged. They are generally compressed into alternative formats (e.g., `.jpg`, or `.png`) where a certain loss of quality is offset by a great decrease in size. In scientific work they are mainly used for complicated diagrams, e.g., photographs of microscopic sections, where there are no distinct objects such as titles, legends, lines, curves, plotting symbols, etc.

2. **Vector graphics**

   Often scientific graphs consist only of lines for axes, curves, plotting symbols, and text for titles and legends with featureless backgrounds, so that storing a bitmap would be wasteful of space. However the main advantage of vector formats (e.g., `.eps`, `.svg`, or `.emf`) is that they are device-independent so they can be displayed or printed at any resolution with no loss of information. They can also be used to generate compressed bitmap files, but it should be noted that scientific graphs can sometimes generate vector hardcopy that is even more bulky than the corresponding bitmap if there are very large numbers of objects being plotted.

3. **Embedded bitmaps**

   Unfortunately vector graphics files can often consist of wrappers containing bitmaps which leads to files with a vector file extension that are actually no better than bitmaps. Some graphics programs export supposed `.eps` files as embedded bitmaps, so losing many of the advantages of true vector files. Also note that `.pdf` files created from vector `.eps` files using GhostScript retain some of the characteristics of actual vector files, but many programs simply distill graphs into `.pdf` files as embedded bitmaps.

Users of the SimFIT package are strongly urged to save all graphs as SimFIT `.eps` files because they have the following advantages.

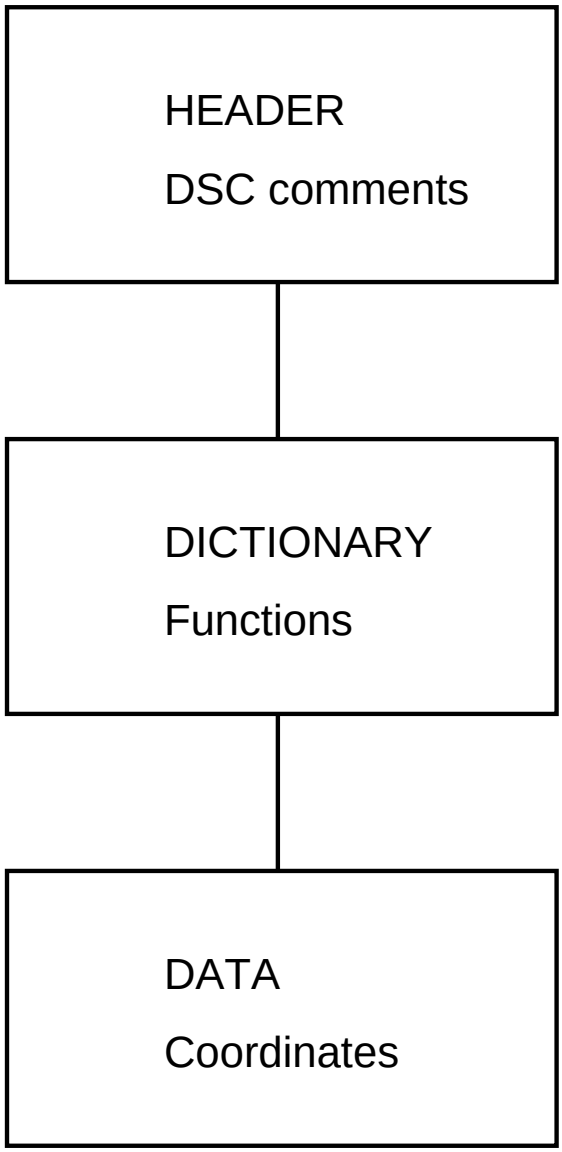
- They are true encapsulated PostScript vector files consisting of a single page with a BoundingBox.
- They are structured in such a way that they can be edited using a text editor to change dimensions, line types, symbols, colours, titles and legends, etc.
- They can be used retrospectively to create alternative types of image files.
- SimFIT provides facilities to edit such files, or make various types of collages.

Although SimFIT can make and edit `.eps` files with no additional software it will be found that, in order to make full use of the PostScript opportunities, it is necessary to download and install the GhostScript package, and also advisable to download and install the GSview package. GhostScript can be used to transform `.eps` files into other graphics formats, while GSview can be used to view and print them.

1.1 The sections of Simfit eps files

In order to be able to edit SimFIT `.eps` files it is necessary to appreciate that there are three distinct sections. The first section contains the BoundingBox coordinates that must be present in `.eps` files to specify the size of such one page graphs along with the document structuring comments. The second section is a dictionary containing the functions that can be used in SimFIT `.eps` files to draw lines and plotting symbols. The third section is a list of coordinates and colors where lines, symbols and text strings are to be drawn. This structure is summarized in the next diagram.

SIMFIT .eps file structure



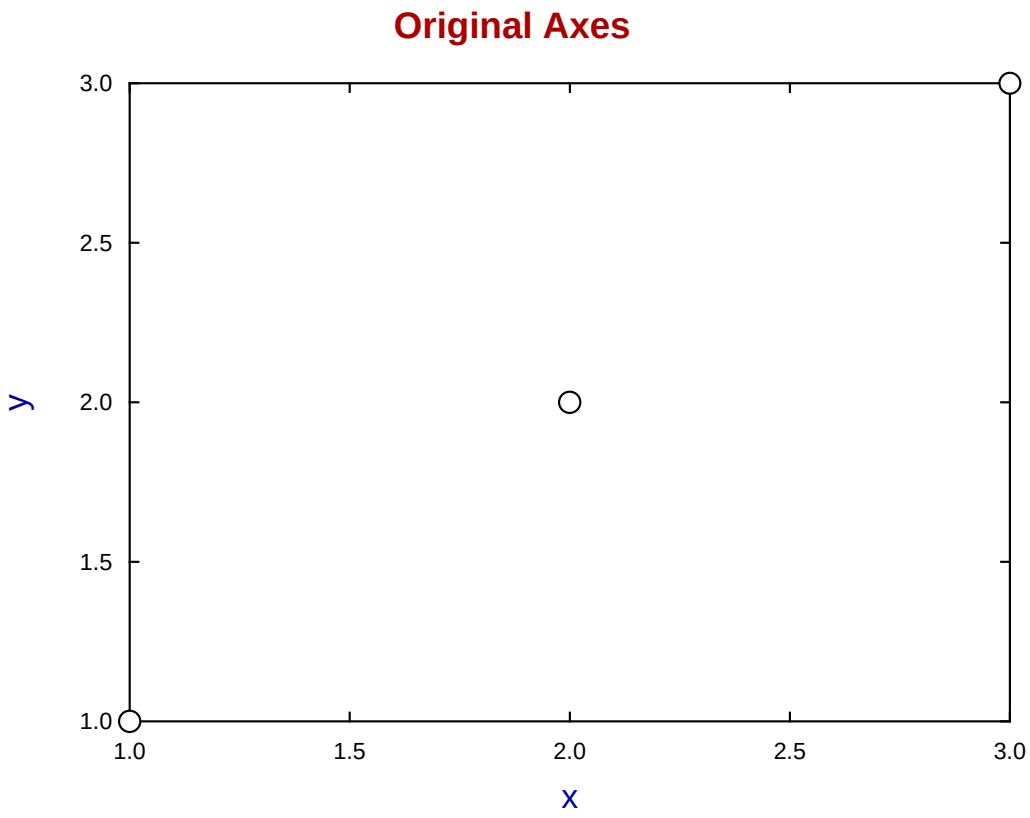
When editing SIMFIT PostScript files retrospectively the following rules must be obeyed until you become familiar with the structure.

1. Never change anything in the HEADER or DICTIONARY sections.
2. You can edit the title, legends, colors, and other display features in the DATA section.
3. Perform editing a step at a time and check the effects using GSview.
4. Make sure you save the file at each stage of editing.

Extensive descriptions about editing the DATA section will be found in the SIMFIT reference manual `w_manual.pdf` available from the SIMFIT website, but a number of simple examples will be given in the succeeding tutorials by way of a more gentle introduction.

1.2 A simple example

As an example consider the following simple default graph which is easily constructed by requesting to plot the coordinates (1,1), (2,2), and (3,3) using program **simplot**



Some of things a user might wish to change retrospectively could be as follows.

- Change the title
- Change the legends
- Change the plotting symbols
- Change the colors

Now SimFIT provides a PostScript editor program **editps** to perform such tasks but, in actual practise, it is easier to edit the PostScript file in a text editor such as **notepad** because of these three special features found in SimFIT Postscript files.

1. They are ASCII text files and can be edited in any text editor.
2. They are encapsulated PostScript files (`.eps`) and describe just a single page.
3. They have been uniquely designed to make such editing very easy.

As you will need to view the results of editing you will need a PS viewer such as **GSview** which requires an installed copy of **GhostScript**.

1.3 The header section

The following code is contained in the header.

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox:    72   252   520   588
%%Creator: Simfit Version 7.2.8 (simfit.org.uk)
%%Title: colours=72/ISOLatin1Encoding/Accents/special/PSfrag/dict=300
%%CreationDate: Saturday, 15 April 2017
%%EndComments
%
%Start of SIMFIT PostScript file
%
save %save current state before clipping, etc.
    70   250   522   250   522   590    70   590%#clipping
newpath moveto lineto lineto lineto closepath clip newpath
   72.00  252.00 translate    0.07    0.07 scale    0.00 rotate%#portrait
   12.00 setlinewidth 0 setlinecap 1 setlinejoin [] 0 setdash
    2.50 setmiterlimit
%
%prolog(1) to (6) can be used by DVIPS as a header
%****cut the invariant prolog/header out from here
```

It includes the document structuring comments together with some technical instructions, and this section would only be edited by experienced users.

1.4 The dictionary section

This contains definitions for all the plotting functions, colors and fonts required to display the data contained in the data section, and this section would only be edited by very experienced users.

```
/SIMFIT 300 dict def SIMFIT begin
%
% prolog(1): definitions
%
/C{copy}def /D{def}def /E{exch}D /F{findfont}D /GR{grestore}D
/GS{gsave}D /M{moveto}D /N{newpath}D /P{pop}D /R{rmoveto}D
/S{scalefont setfont}D /d{dup}D /i{putinterval}D /p{put}D
%
% prolog(2): construct Greek/maths font
%
...
...
...
/ty-font /Helvetica      D%text right y-mid
/tz-font /Helvetica      D%text left  y-mid
/ti-size 204 D /xl-size 187 D /yl-size 187 D /zl-size 187 D
/tc-size 144 D /td-size 144 D /tl-size 144 D /tr-size 144 D
/ty-size 144 D /tz-size 144 D
/sb-size 0.75 D /sp-size 0.75 D%sub/superscript expansion
/y-down -0.33 sb-size mul D /y-up 0.33 sp-size div D%sub/sup shift
%
foreground thickness setlinewidth
```

1.5 The data section

This is first given in full and then the sections that are most likely to be edited are discussed in detail.

```
/background{c15}D
2 setlinecap
1070 671 5959 671 5959 4215 1070 4215 4 pc%#8
/ty-size ty-size   0.900 mul def
/tl-size tl-size   0.900 mul def
1070 671 1118 671 li%#4
5959 671 5911 671 li%#4
(1.0) 974 671 ty%#()2
(000) fx
1070 1557 1118 1557 li%#4
5959 1557 5911 1557 li%#4
(1.5) 974 1557 ty%#()2
(000) fx
1070 2443 1118 2443 li%#4
5959 2443 5911 2443 li%#4
(2.0) 974 2443 ty%#()2
(000) fx
1070 3329 1118 3329 li%#4
5959 3329 5911 3329 li%#4
(2.5) 974 3329 ty%#()2
(000) fx
1070 4215 1118 4215 li%#4
5959 4215 5911 4215 li%#4
(3.0) 974 4215 ty%#()2
(000) fx
/ty-size ty-size   1.111 mul def
/tl-size tl-size   1.111 mul def
/tc-size tc-size   0.900 mul def
/tl-size tl-size   0.900 mul def
1070 671 1070 719 li%#4
1070 4215 1070 4167 li%#4
(1.0) 1070 462 tc  %#()2
(000) fx
2292 671 2292 719 li%#4
2292 4215 2292 4167 li%#4
(1.5) 2292 462 tc  %#()2
(000) fx
3515 671 3515 719 li%#4
3515 4215 3515 4167 li%#4
(2.0) 3515 462 tc  %#()2
(000) fx
4737 671 4737 719 li%#4
4737 4215 4737 4167 li%#4
(2.5) 4737 462 tc  %#()2
(000) fx
5959 671 5959 719 li%#4
5959 4215 5959 4167 li%#4
(3.0) 5959 462 tc  %#()2
(000) fx
/tc-size tc-size   1.111 mul def
/tl-size tl-size   1.111 mul def
/ti-size ti-size   1.000 mul def
c4
(Original Axes) 3195 4467 ti%#title
(0000000000000) fx
/ti-size ti-size   1.000 mul def
/xl-size xl-size   1.000 mul def
c1
(x) 3515 192 xl%#x legend
```

```
(0) fx
/xl-size xl-size   1.000 mul def
/yl-size yl-size   1.000 mul def
(y) 501 2443 yl%#y legend
(0) fx
/yl-size yl-size   1.000 mul def
0 setlinecap
c0
1070 671 59 ce%#2
3514 2443 59 ce%#2
5959 4215 58 ce%#2
```

1.6 Editing the title and legends

Searching for the key word title locates the next section.

```
c4
(Original Axes) 3195 4467 ti%#title
(0000000000000) fx
/ti-size ti-size   1.000 mul def
/xl-size xl-size   1.000 mul def
c1
(x) 3515 192 xl%#x legend
(0) fx
/xl-size xl-size   1.000 mul def
/yl-size yl-size   1.000 mul def
(y) 501 2443 yl%#y legend
(0) fx
```

Interchanging the colors c1(blue) and c4(red) and altering the title and can then be done leading to the replacement section shown next.

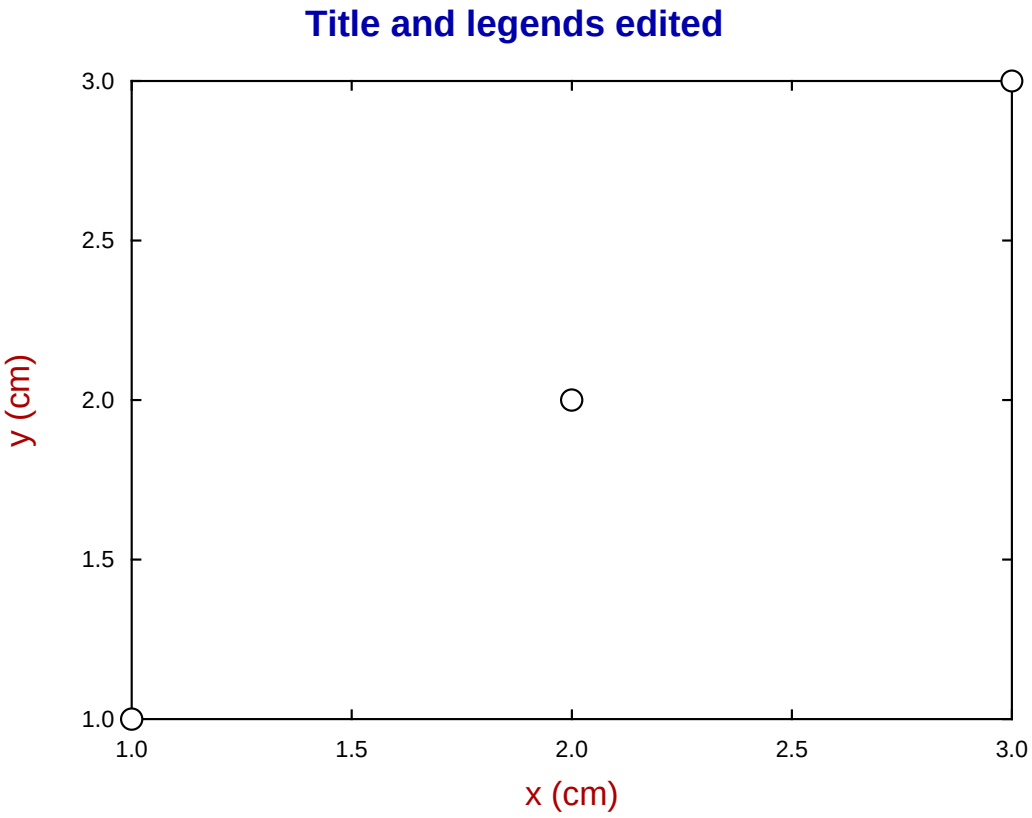
```
c1
(Title and legends edited) 3195 4467 ti%#title
(000000000000000000000000) fx
/ti-size ti-size   1.000 mul def
/xl-size xl-size   1.000 mul def
c4
(x \(mm\)) 3515 192 xl%#x legend
(00000000) fx
/xl-size xl-size   1.000 mul def
/yl-size yl-size   1.000 mul def
(y \(mm\)) 501 2443 yl%#y legend
(00000000) fx
```

Two comments are needed in order to understand the results of this editing.

1. When a text string such as a title or legend is edited it is necessary to make sure that the character key string underneath the text is padded or contracted if required to make sure the key has at least as many characters (in this case the 0 characters denoting a normal font) as the text string.

2. Since character strings in PostScript are enclosed in round brackets it is necessary to prefix any brackets introduced inside the string by the backslash escape character.

The edited file is shown next.



1.7 Editing line and symbol types

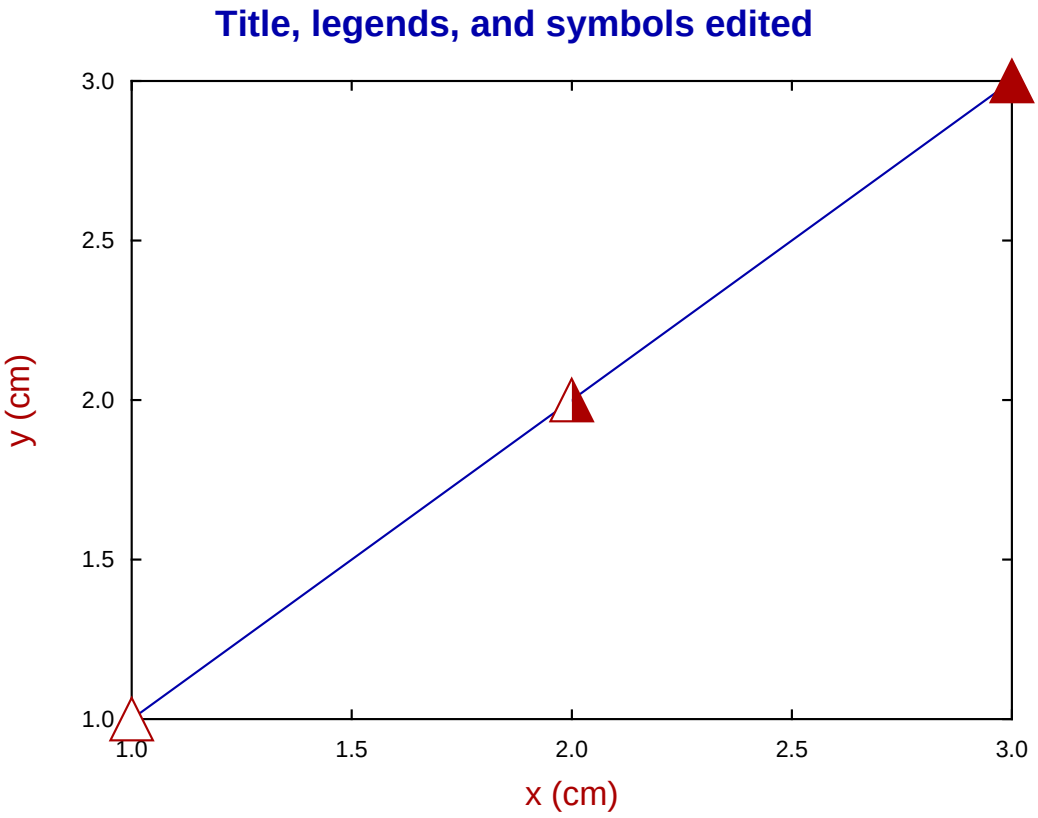
It is frequently required to change the size, colors, and types of lines and plotting symbols and the code in the data section defining the black (c0) empty circles (ce) is as follows.

```
c0
1070 671 59 ce%#2
3514 2443 59 ce%#2
5959 4215 58 ce%#2
```

For example, using the command c1 and li to add a blue line, altering c0 to c4 to change color to red, then changing empty circles (ce) into empty triangle (te), half-filled triangle (th), and filled triangle (tf), together with doubling the size of the symbols from 59 to 118, as in this code

```
c1
1070 671 5959 4215 li
c4
1070 671 118 te%#2
3514 2443 118 th%#2
5959 4215 118 tf%#2
```

creates the next graph.



Clearly users will require much more information to be able to edit all the features of SIMFIT PostScript files, and the necessary details along with numerous worked examples follows.

2 Plotting non-standard characters

The SIMFIT assumption is that users will want professional quality hardcopy and not be content with standard Windows graphics. In other words, if a permanent record is required for a displayed graph, then a *.eps file will be created for immediate use or retrospective conversion into a *.png or similar file for printing or incorporating into documents. Since the Windows display and PostScript files use different fonts, then the following details should be noted, otherwise the PostScript file will not have the same edited text strings as the Windows display.

The basic problem and the solution to it

As long as the characters to be plotted are from the standard 7-bit ASCII standard set, i.e. characters 32 to 126 then all that is required is to use the Simple editing control to edit text strings for titles, legends, labels, etc. European accented characters from the ISOLatin1 8-bit set that are entered from the keyboard or font table are also plotted correctly due to a built in transformation mechanism. However, to plot characters with arbitrary accents like hats and bars, superscripts, or subscripts, or to plot maths symbols or characters from the Greek alphabet, then the more advanced techniques have to be used. This is to ensure that what is displayed in the Windows bitmap will be the same as what is displayed in the PostScript output.

2.1 7-bit ASCII characters 33 to 126

These, together with the space character (32), are the standard non-accented characters allowed in the simple editing control.

```
! " # $ % & ' ( ) * + , - . /
0 1 2 3 4 5 6 7 8 9
: ; < = > ? @
A B C D E F G H I j K L M N O P Q R S T U V W X Y Z
[ \ ] ^ _ ‘
a b c d e f g h i j k l m n o p q r s t u v w x y z
{ | } ~
```

2.2 The basic character plotting technique

When SIMFIT displays a character string in a plot, then associated with every character is a key indicating if the the character is to be displayed in any special way. For instance in this string key pair

```
Area in cm2
00000000002
```

The 0 indicates a normal character while the 2 indicates a superscript, so the string will be displayed as

$$\text{Area in cm}^2$$

where the 2 is now shown as a superscript.

2.3 Advanced editing

Fortunately the user does not need to know any of these details as, when Advanced editing is selected, users can choose to either

- Edit the individual characters;
- Edit the individual keys, or;

❍ Select a character together with the associated key from a displayed table.

It is assumed that when users want a non-standard character they will first use the simple editing control to input a string, then select the advanced editing control to add details.

As another example, consider the numerator of the formula displayed in figure 1. This would first be entered by simple editing as the character string

```
a1x + a2x2 + a3x3 + a4x4
```

then the advanced control would be used to replace the characters by Greek equivalents, say from the Symbol or maths tables, and the keys would be edited for superscripts and subscripts as shown below

```
a1x + a2x2 + a3x3 + a4x4
31000031x200031x200031x2
```

followed by similar editing of the denominator to give

```
1 + b1x + b2x2 + b3x3 + b4x4
000031000031x200031x200031x2
```

Finally, adding a horizontal line as a an arow/line type of object would result in the display of a properly formatted rational function like this

$$\frac{\alpha_1 x + \alpha_2 x^2 + \alpha_3 x^3 + \alpha_4 x^4}{1 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 x^4}.$$

The rest of this section is for those who are familiar with the PostScript language and wish to use the powerful PostScript techniques supported by SimFIT to create special effects.

2.4 Octal codes

For users who just want to use the simple editing control without resorting to the advanced editing control to plot accented characters not on their keyboard, it is possible to enter octal codes directly. For instance `\361` will produce ñ because the character/key pair will automatically have an 8 placed under the backslash to indicate that the ISOLatin1 character set should be used instead of the Standard font. Note that users of the advanced editing mode will sometimes see a question mark ? or a blank instead of a 0 to represent a normal font, which happens when strings have been shortened or lengthened by the editing process.

The rule is that every character must be paired with a unique identifying key, and the PostScript file will only plot if this condition is satisfied. However, there is an important exception to this rule. When octal codes are used it is vital to realise that the back slash followed by three digits represents a single character, i.e. a single byte, so it only requires one key. As just remarked, a key must be placed under the backslash to indicate the font set required as below.

```
This is what n tilde looks like \361
00000000000000000000000000000008
```

which will plot as

This what n tilde looks like ñ

2.5 A detail about PostScript fonts

It has to be emphasised that the Windows bit-map display is using TrueType fonts, while the Postscript file will use PostScript fonts. Because these are not identical, it is possible that arrows and other graphical objects added to graphs will not have exactly the same appearance. The objects will have the same starting coordinates, but the lengths of the character strings will not be exactly equal, which can lead to unwanted overlap of objects and character strings in the PostScript display. Similarly the colours and line thickness will not be the same, and so it is vital to check a PostScript visual display in your PostScript viewer before saving a PostScript file.

3 PostScript procedures

The best way to use SIMFIT graphics is to archive standard sized SIMFIT PostScript files in portrait orientation and then, when required, manipulate them, followed by printing hardcopy, or by transforming into other formats, such as .png, for pasting into documents.

Most simple editing operations can be done using a text editor as described later, but for more extensive manipulations program **editps** can be used as now described.

3.1 Using editps to manipulate PostScript files

Several points must be mentioned concerning SIMFIT EPS files and program **editps**.

1. An encapsulated PostScript file (*.eps) is a special type of self-contained one page PostScript file containing a BoundingBox with dimensions, so that the file can be easily manipulated for re-sizing and inclusion within documents.
2. All PostScript files created by SIMFIT adhere to this convention.
3. Program **editps** will accept any such files, but some features will only work with SIMFIT .eps files.

3.2 Editing Simfit Postscript files

After a SIMFIT file has been loaded into **editps** it is possible to search for such items as the title, or legends, etc., and edit as required. However, as described later, it is much easier to do such editing using a simple text editor.

Further, note that this type of editing is restricted to SIMFIT PostScript files which contain special markers to locate titles, legends, etc.

3.3 Rotating, re-sizing, and changing aspect ratios.

In addition program **editps** can be used to rotate or re-size SIMFIT EPS PostScript files and perform shearing transformations. However, the main use is create various types of collages such as these.

- Simple collages.

  Here all the constituent files have identical BoundingBoxes, i.e., they are the same size, and the collage can be created directly.

- Freestyle collages.

  If the constituent files do not have identical BoundingBoxes then hand-crafting is required.

- Adding new files as insets to existing files.

  Here it must be realized that there are two cases.

  - The inserted child file can have an invisible background, when the parent file will show through.
  - The inserted child file can have an opaque background, when the parent graph will be obliterated where the child overlaps.

3.4 Creating simple collages

Figure 1 illustrates a simple collage created using **editps** from a set of SIMFIT PostScript files assembled into a library file. If required, titles, labels, and extra text can be added by program **editps** to identify the sub-graphs or add further details. To create such collages it is advisable that all the files supplied should have the same dimensions and orientation, as this facility can only generate collages with fixed cell dimensions.

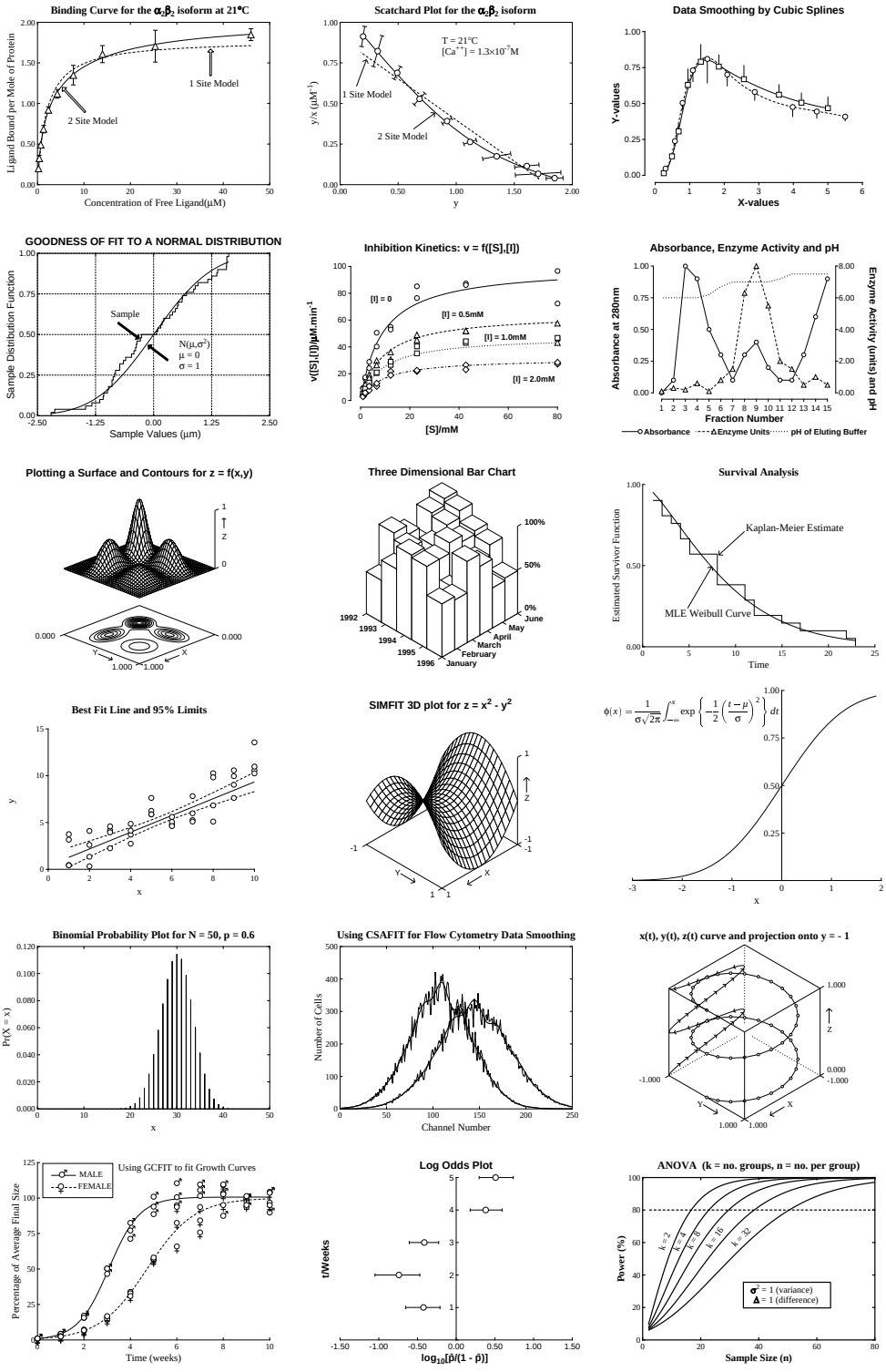


Figure 1: Collage 1

3.5 Creating freestyle collages

Plots that are incorporated into simple collages as just described must all have the standard default SimFIT portrait format, i.e., with identical standard BoundingBoxes. However, users often wish to collect plots of varying sizes together into an arbitrary pattern, as in figure 2.

This illustrates the value of changing font size and line thickness as graphs are re-sized. In general, it will be clear from figure 2 that, if a graph is to be reduced in size before building into a freestyle collage, it is a good idea to increase the size of fonts and thickness of lines.

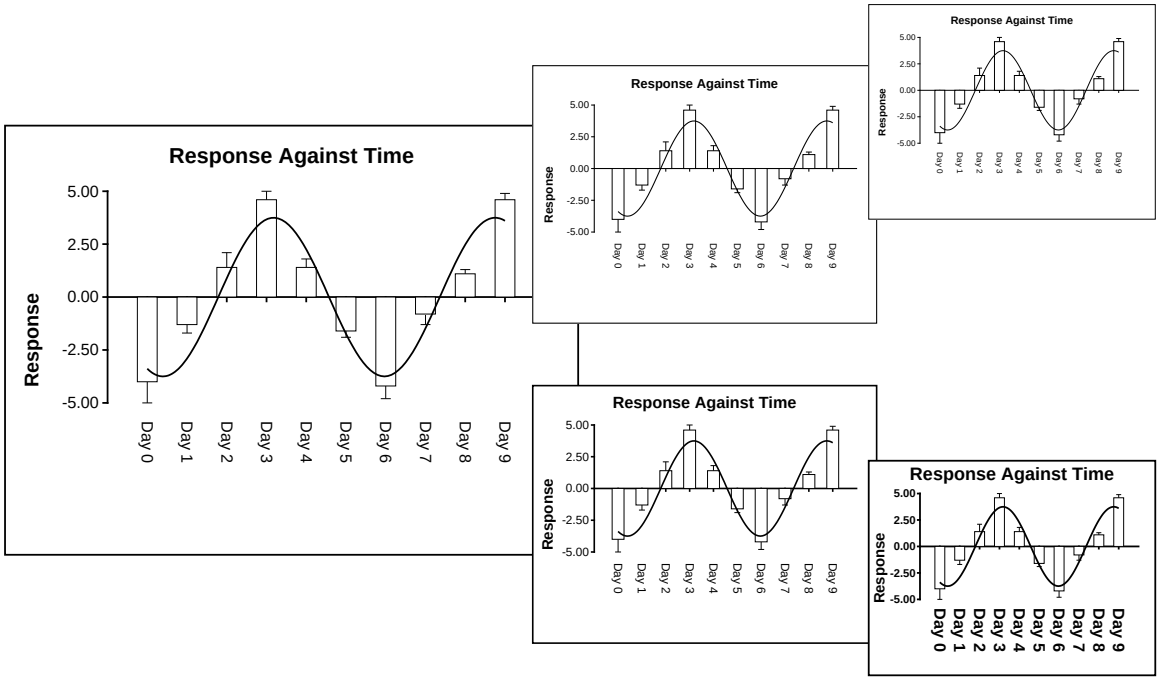


Figure 2: Collage 2

In the above figure the upper sub–figures are derived from the large figure by reduction, so the text becomes progressively more difficult to read as the figures scale down. In the lower sub-figures, however, line thicknesses and font sizes have been increased as the figure is reduced, maintaining legibility. Such editing can be done interactively, but SimFIT PostScript files are designed to make such retrospective editing easy. Clearly, the graphs should be edited individually before assembling into the final collage.

Figure 3 and figure 4 show further examples of how such collages can be assembled using **editps** in freestyle mode with a set of graphs that can have arbitrary dimensions and rotations. In addition to being able to move the sub–graphs into any positions, this procedure also allows interactive differential re-sizing of individual graphs.

There is an extremely important point to remember when creating freestyle collages: it is possible to create PostScript files from SimFIT where the background, if white, can be either transparent or opaque. Note that PostScript files with opaque white backgrounds, as in the above sub–figures, will obscure any graphs they overlay. Of course, sometimes this is desired, but often a transparent white background may be preferred. This choice is determined, of course, by the configuration option in use when the PostScript file is created

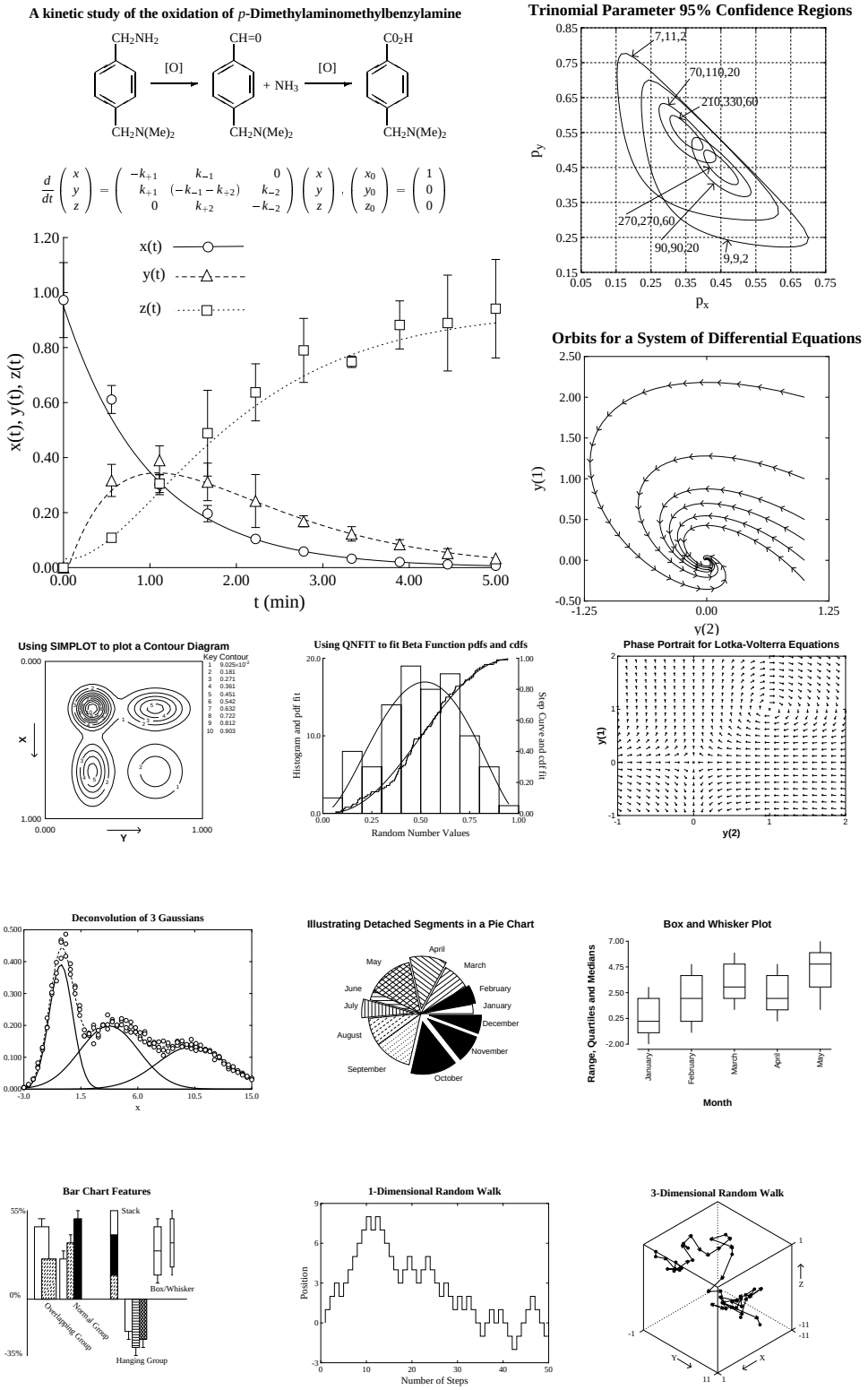


Figure 3: Collage 3

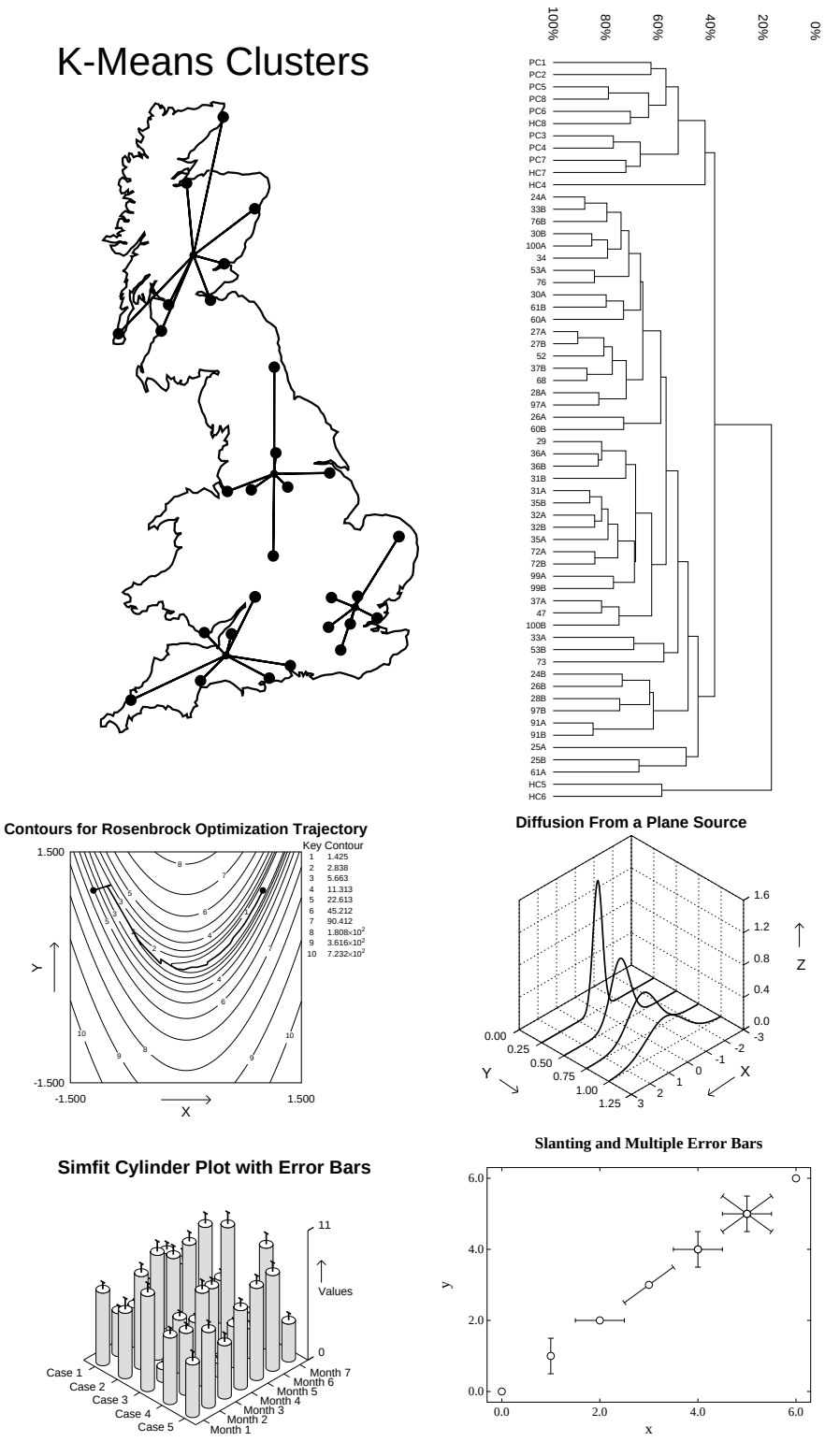


Figure 4: Collage 4

3.6 Subsidiary figures as insets

Figure 5 illustrates a special type of freestyle collage where a sub-graph is placed inside a parent graph. Sometimes it is best to enlarge the fonts and increase the line thicknesses when a sub-graph is going to be reduced in size in this way, and it is always important to remember the effects of opaque and transparent backgrounds that SIMFIT allows.

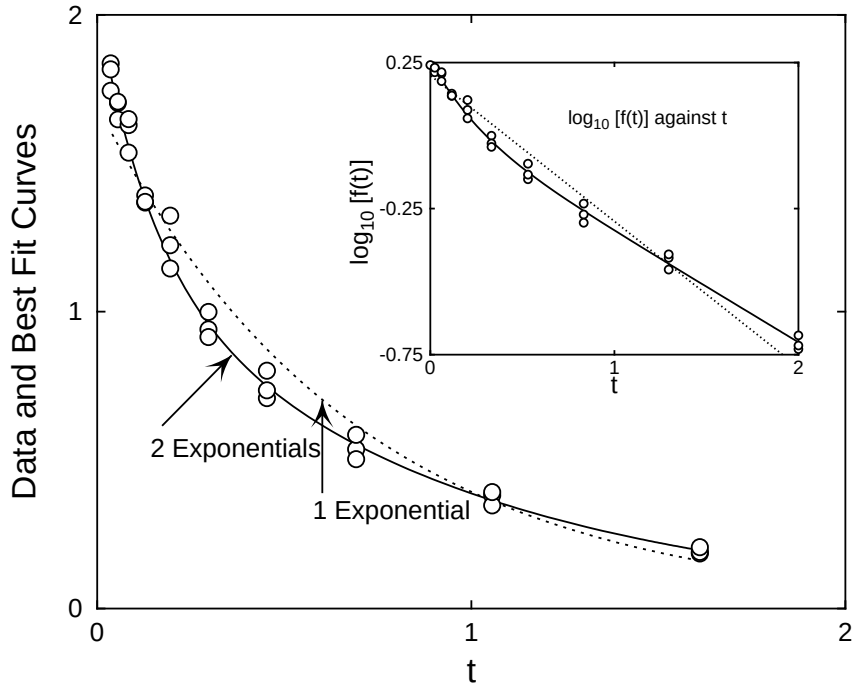


Figure 5: Subsidiary figures as insets

First of all the plots in figure 6 were created using **exfit** with test file `exfit.tf4` after fitting 1 exponential then 2 exponentials. Note that the line thickness and font size have been increased in the transformed plot as it is going to be reduced in the inset. Figure 7 was then created by **editps** using the option to create a

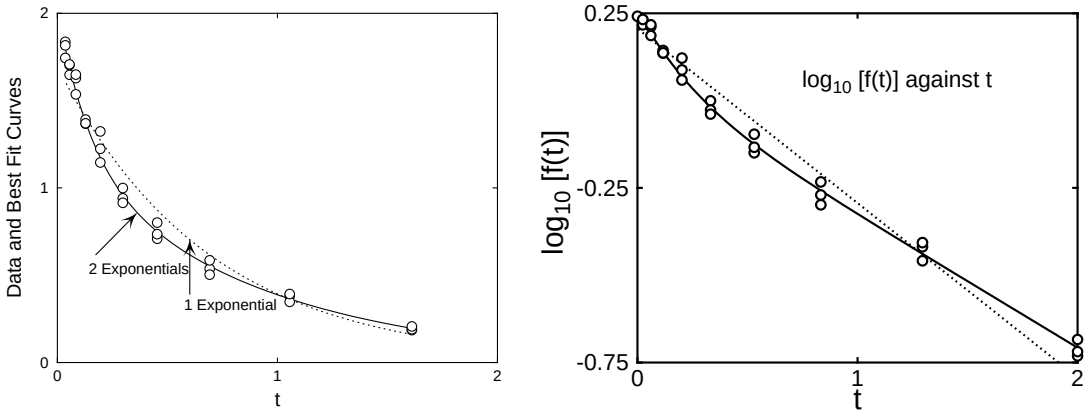


Figure 6: Insets 1: Exponential fitting and semilog transforms

freestylecollage. Note how, in the left hand plot the option to plot an opaque background even when white was selected and the transformed plot obscures the underlying main plot. In the right hand plot the option for a transparent background was used so that the main plot was not obscured. Both techniques are valuable when creating insets, and all that is now necessary to create figure 5 is to shrink the transformed plot and translate it to a more convenient location. A further point to note is that SIMFIT plots have a border, which is obscuring more of the left hand main figure in figure 7 than seems necessary. When subsidary figures are going to be used in this way it is often advisable to use the option to clip the plot to trim away extra white space, or else use GsView to calculate a new BoundingBox in a transparent subsidiary plot by transforming ps into eps.

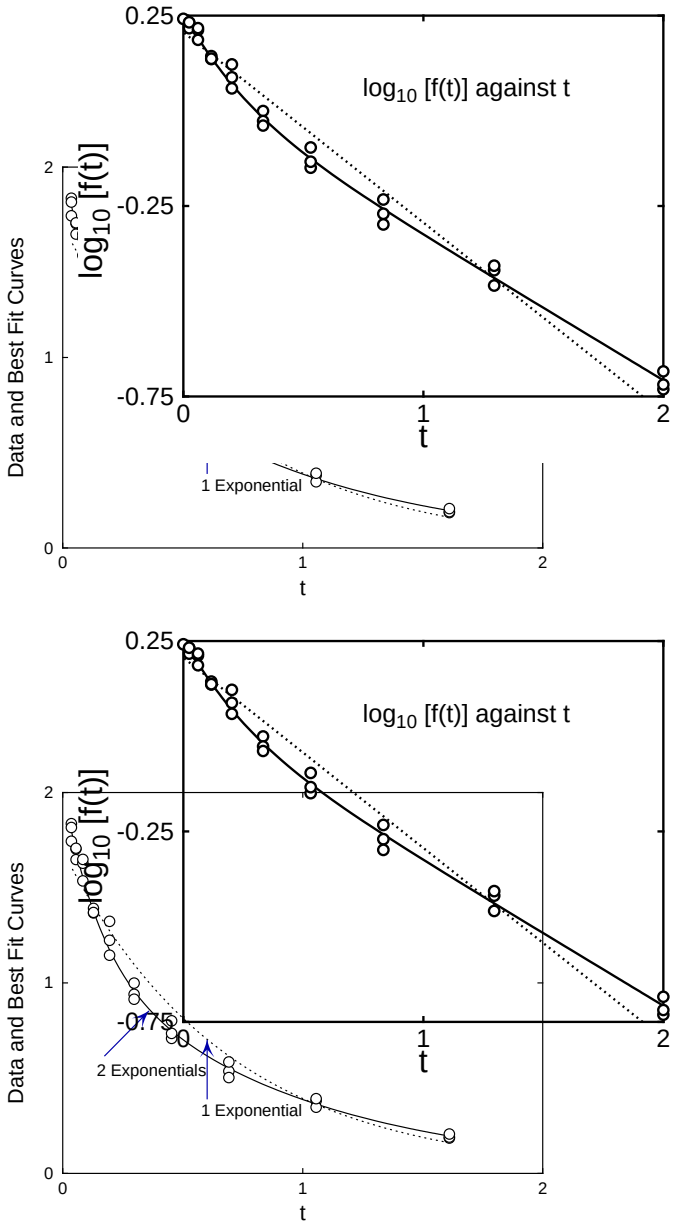
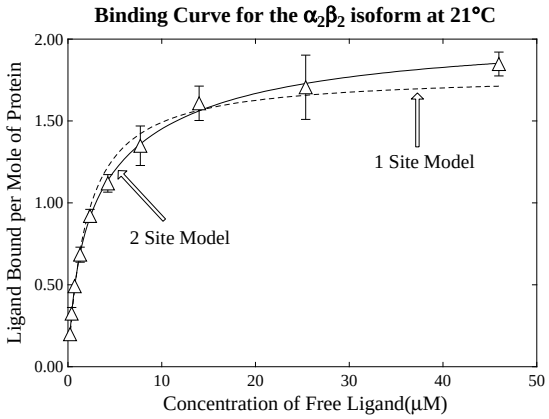
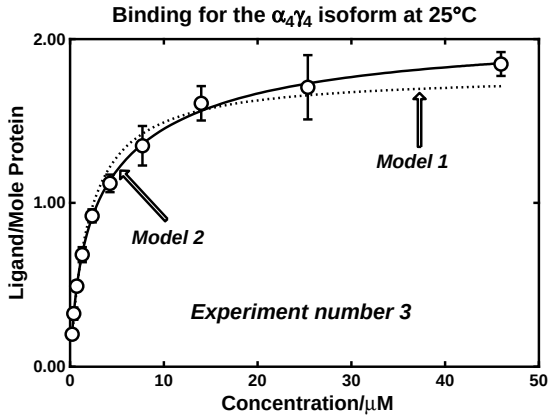


Figure 7: Insets 2: Opaque and tranparent backgrounds in insets

4 Editing Simfit PostScript files

One of the unique features of SimFIT PostScript files is that the format is designed to make retrospective editing easy. A typical example of when this could be useful would be when a graph needs to be changed for some reason. Typically an experimentalist might have many plots stored as .eps files and want to alter one for publication or presentation. SimFIT users are strongly recommended to save all their plots as .ps or .eps files, so that they can be altered in the way to be described. Even if you do not have a PostScript printer it is still best to save as .ps, then use GSview/Ghostscript to print or transform into another graphics format. Consider these next two figures, showing how a graph can be transformed by simple editing in a text editor, e.g. NOTEPAD.





This type of editing should always be done if you want to use one figure as a reduced size inset figure inside another, or when making a slide, otherwise the SimFIT default line thickness will be too thin. Note that most of the editing to be described below can actually be done at the stage of creating the file, or by using program EDITPS. In this hypothetical example, we shall suppose that the experimentalist had realized that the title referred to the wrong isoform and temperature, and also wanted to add extra detail, but simplify the graph in order to make a slide using thicker lines and a bolder font. In the following sections the editing required to transform the SimFIT example file `simfig1.ps` will be discussed, following a preliminary warning.

4.1 Warning about editing PostScript files

In the first place the technique to be described can only be done with SimFIT PostScript files, because the format was developed to facilitate the sort of editing that scientists frequently need to perform. Secondly, it must be realized that PostScript files must conform to a very strict set of rules. If you violate these rules, then GSview/Ghostscript will warn you and indicate the fault. Unfortunately, if you do not understand PostScript, the warning will be meaningless. So here are some rules that you must keep in mind when editing.

- ❏ Always keep a backup copy at each successful stage of the editing.
- ❏ All text after a single percentage sign % to the line end is ignored in PostScript.
- ❏ Parentheses must always be balanced as in (figure 1(a)) not as in (figure 1(a).
- ❏ Fonts must be spelled correctly, e.g. Helvetica-Bold and not helveticabold.
- ❏ Character strings for displaying must have underneath them a vector index string of EXACTLY the same length.
- ❏ When introducing non-keyboard characters each octal code represents one byte.
- ❏ The meaning of symbols and line types depends on the function, e.g. `da` means dashed line while `do` means dotted line.

A review of the PostScript colours, fonts and conventions is also in the `w_readme` files. In the next sections it will be assumed that are running SIMFIT and have a renamed copy of `simfig1.ps` in your text editor (e.g. notepad), and after each edit you will view the result using GSview/Ghostscript. Any errors reported when you try to view the edited file will be due to violation of a PostScript convention. The most usual one is to edit a text string without correctly altering the index below it to have exactly the same number of characters.

4.2 The percent-hash escape sequence

Later versions of SIMFIT create PostScript files that can be edited by a stretch, clip, slide procedure, which relies on each line containing coordinates being identified by a comment line starting with %#. All text extending to the right from the first character of this sequence can safely be ignored and is suppressed for clarity in the following examples.

4.3 Changing line thickness and plot size

The following text will be observed in the original `simfig1.ps` file.

```
72.00  252.00 translate    0.07    0.07 scale    0.00 rotate
   11.00 setlinewidth 0 setlinecap 0 setlinejoin [] 0 setdash
    2.50 setmiterlimit
```

The postfix argument for setlinewidth alters the line width globally. In other words, altering this number by a factor will alter all the linewidths in the figure by this factor, irrespective on any changes in relative line thicknesses set when the file was created. The translate, scale and rotate are obvious, but perhaps best done by program EDITPS. Here is the same text edited to increase the line thickness by a factor of two and a half.

```
72.00  252.00 translate    0.07    0.07 scale    0.00 rotate
   27.50 setlinewidth 0 setlinecap 0 setlinejoin [] 0 setdash
    2.50 setmiterlimit
```

4.4 Changing PostScript fonts

In general the Times-Roman fonts may be preferred for readability in diagrams to be included in books, while Helvetica may look better in scientific publications. For making slides it is usually preferable to use Helvetica-Bold. Of course any PostScript fonts can be used, but in the next example we see how to change the fonts in `simfig1.ps` to achieve the effect illustrated.

```
/ti-font /Times-Bold  D%plot-title
/xl-font /Times-Roman D%x-legend
/yl-font /Times-Roman D%y-legend
/zl-font /Times-Roman D%z-legend
/tc-font /Times-Roman D%text centred
/td-font /Times-Roman D%text down
/tl-font /Times-Roman D%text left to right
/tr-font /Times-Roman D%text right to left
/ty-font /Times-Roman D%text right y-mid
/tz-font /Times-Roman D%text left  y-mid
```

The notation is obvious, the use indicated being clear from the comment text following the percentage sign % at each definition, denoted by a D. This is the editing needed to bring about the font substitution.

```
/ti-font /Helvetica-Bold D%plot-title
/xl-font /Helvetica-Bold D%x-legend
/yl-font /Helvetica-Bold D%y-legend
/zl-font /Helvetica-Bold D%z-legend
```

```
/tc-font /Helvetica-Bold D%text centred
/td-font /Helvetica-Bold D%text down
/tl-font /Helvetica-Bold D%text left to right
/tr-font /Helvetica-Bold D%text right to left
/ty-font /Helvetica-Bold D%text right y-mid
/tz-font /Helvetica-Bold D%text left  y-mid
```

Observing the scheme for colours (just before the fonts in the file) and text sizes (following the font definitions) will make it obvious how to change colours and text sizes.

4.5 Changing title and legends

Observe the declaration for the title and legends in the original file.

```
(Binding Curve for the a2b2 isoform at 21@C) 3514 4502 ti
(000000000000000000000061610000000000000060) fx
(Concentration of Free Ligand(lM)) 3514 191 xl
(000000000000000000000000000300) fx
(Ligand Bound per Mole of Protein) 388 2491 yl
(000000000000000000000000000000) fx
```

Note that, for each of the text strings displayed, there is a corresponding index of font substitutions. For example a zero prints the letter in the original font, a one denotes a subscript, while a six denotes bold maths. Since the allowed number of index keys is open-ended, the number of potential font substitutions is enormous. You can have any accent on any letter, for instance. This is the editing required to change the text. However, note that the positions of the text do not need to be changed, the font display functions work out the correct position to centre the text string.

```
(Binding for the a4c4 isoform at 25@C) 3514 4502 ti
(000000000000000061610000000000000060) fx
(Concentration/lM) 3514 191 xl
(0000000000000030) fx
(Ligand/Mole Protein) 388 2491 yl
(000000000000000000) fx
```

Note that the \ character is an escape character in PostScript so, if you want to have something like an unbalanced parenthesis, as in `Figure 1 a)` you would have to write `Figure 1a\).` When you create a PostScript file from SIMFIT it will prevent you from writing a text string that violates PostScript conventions but, when you are editing, you must make sure yourself that the conventions are not violated, e.g. use `c:\\simfit` instead of `c:\simfit.`

4.6 Deleting graphical objects

It is very easy to delete any text or graphical object by simply inserting a percentage sign % at the start of the line to be suppressed. In this way an experimental observation can be temporarily suppressed, but it is still in the file to be restored later if required. Here is the PostScript code for the notation on the left hand vertical, i.e. *y* axis in the file `simfig1.ps`.

```
910 1581 958 1581 li
6118 1581 6070 1581 li
(0.50) 862 1581 ty
(0000) fx
910 2491 958 2491 li
6118 2491 6070 2491 li
(1.00) 862 2491 ty
```

```
(0000) fx
910 3401 958 3401 li
6118 3401 6070 3401 li
(1.50) 862 3401 ty
(0000) fx
```

This is the text, after suppressing the tick marks and notation for $y = 0.5$ and $y = 1.5$ by inserting a percentage sign. Note that the index must also be suppressed as well as the text string.

```
%910 1581 958 1581 li
 %6118 1581 6070 1581 li
 %(0.50) 862 1581 ty
 %(0000) fx
910 2491 958 2491 li
6118 2491 6070 2491 li
(1.00) 862 2491 ty
(0000) fx
 %910 3401 958 3401 li
 %6118 3401 6070 3401 li
 %(1.50) 862 3401 ty
 %(0000) fx
```

4.7 Changing line and symbol types

This is simply a matter of substituting the desired line or plotting symbol key.

```
Lines    : li (normal) da (dashed) do (dotted) dd (dashed dotted) pl (polyline)
Circles  : ce (empty)  ch (half)   cf(full)
Triangles: te (empty)  th (half)   tf (full)
Squares  : se (empty)  sh (half)   sf (full)
Diamonds : de (empty)  dh (half)   df (full)
Signs    : ad (add)    mi (minus)  cr (cross)  as (asterisk)
```

Here is the original text for the dashed line and empty triangles.

```
5697 3788 120 da
933 1032 72 te
951 1261 72 te
984 1566 73 te
1045 1916 72 te
1155 2346 72 te
1353 2708 73 te
1714 3125 72 te
2367 3597 72 te
3551 3775 72 te
5697 4033 72 te
```

Here is the text edited for a dotted line and empty circles.

```
5697 3788 120 do
933 1032 72 ce
951 1261 72 ce
984 1566 73 ce
1045 1916 72 ce
1155 2346 72 ce
1353 2708 73 ce
```

```
1714 3125 72 ce
2367 3597 72 ce
3551 3775 72 ce
5697 4033 72 ce
```

4.8 Adding extra text

Here is the original extra text section.

```
/font /Times-Roman D /size   216 D
GS font F size S   4313  2874 M    0 rotate
(1 Site Model)
(000000000000) fx
/font /Times-Roman D /size   216 D
GS font F size S   1597  2035 M    0 rotate
(2 Site Model)
(000000000000) fx
```

Here is the above text after changing the font.

```
/font /Helvetica-BoldOblique D /size   216 D
GS font F size S   4313  2874 M    0 rotate
(Model 1)
(0000000) fx
/font /Helvetica-BoldOblique D /size   216 D
GS font F size S   1597  2035 M    0 rotate
(Model 2)
(0000000) fx
```

Here is the additional code required to add another label to the plot.

```
/font /Helvetica-BoldOblique D /size   240 D
GS font F size S   2250  1200 M    0 rotate
(Experiment number 3)
(00000000000000000000) fx
```

4.9 Changing colors

The definition of the 72 colors c0 to c71 in terms of red, green, blue components is in the file header and any color can be edited. Note that the colors c0 and c15 should be defined as black and white respectively, and perhaps the main editing involved would be to replace all appearances of colors other than c0 (and c15 for the background) by c0 in order. to transform a colored file into a monochrome file, as these tend to look better in Word documents and Power Point. For instance, changing

```
c4
(Survival Analysis) 3195 4467 ti%#title
(00000000000000000) fx
```

into

```
c0
(Survival Analysis) 3195 4467 ti%#title
(00000000000000000) fx
```

would change a red title into a black title.

5 Standard fonts

All PostScript printers have a basic set of 35 fonts and it can be safely assumed that graphics using these fonts will display in GSview/Ghostscript and print on all except the most primitive PostScript printers. Of course there may be a wealth of other fonts available. The Times and Helvetica fonts are well known, and the monospaced Courier family of typewriter fonts are sometimes convenient for tables.

Times-Roman
!"#$%&’()*+,-./0123456789:;<=>?@
ABCDEFGHIJKLMNOPQRSTUVWXYZ[]^_‘
abcdefghijklmnopqrstuvwxyz{|}~

Times-Bold
!"#$%&’()*+,-./0123456789:;<=>?@
ABCDEFGHIJKLMNOPQRSTUVWXYZ[]^_‘
abcdefghijklmnopqrstuvwxyz{|}~

Times-Italic
!"#$%&’()+,-./0123456789:;<=>?@*
ABCDEFGHIJKLMNOPQRSTUVWXYZ[]^_‘
abcdefghijklmnopqrstuvwxyz{|}~

Times-BoldItalic
!"#$%&’()*+,-./0123456789:;<=>?@
ABCDEFGHIJKLMNOPQRSTUVWXYZ[]^_‘
abcdefghijklmnopqrstuvwxyz{|}~

Helvetica
!"#$%&’()*+,-./0123456789:;<=>?@
ABCDEFGHIJKLMNOPQRSTUVWXYZ[]^_‘
abcdefghijklmnopqrstuvwxyz{|}~

Helvetica-Bold
!"#$%&’()*+,-./0123456789:;<=>?@
ABCDEFGHIJKLMNOPQRSTUVWXYZ[]^_‘
abcdefghijklmnopqrstuvwxyz{|}~

Helvetica-Oblique
!"#$%&’()+,-./0123456789:;<=>?@*
ABCDEFGHIJKLMNOPQRSTUVWXYZ[]^_‘
abcdefghijklmnopqrstuvwxyz{|}~

Helvetica-BoldOblique
!"#$%&’()*+,-./0123456789:;<=>?@
ABCDEFGHIJKLMNOPQRSTUVWXYZ[]^_‘
abcdefghijklmnopqrstuvwxyz{|}~

5.1 Decorative fonts

Sometimes decorative or graphic fonts are required, such as pointing hands or scissors. It is easy to include such fonts using program Simplot, although the characters will be visible only if the plot is inspected using GSview/Ghostscript.

Symbol
!∀#∃%&ə()*+,−./0123456789:;<=>?≅
ΑΒΧΔΕΦΓΗΙϑΚΛΜΝΟΠΘΡΣΤΥςΩΞΨΖ[]⊥_
αβχδεφγηιφκλμνοπθρστυϖωξψζ{|}~

ZapfDingbats
✁✂✃✄✆✇✈✉☛☞✌✍✎✏✑✒✓✔✕✖✗✘✙✚✛✜✝✞✟✠✡✢✣✤✥✦✧★☆✪✫✬✭✮✯✰✱✲✳✴✵✶✷✸✹✺✻✼✽✾✿❀❁
❂❃❄❅❆❇❈❉❊❋●❍■❑❒❒❑❒▲▼◆❖◗❙❚❛❜❝❞

ZapfChancery-MediumItalic
!"#$%&'()+,-./0123456789:;<=>?@*
ABCDEFGHIJKLMNOPQRSTUVWXYZ[]^_‘
abcdefghijklmnopqrstuvwxyz{|}~

Some extra characters in Times, Helvetica, etc.
æ(361)•(267)†(262)‡(263)¡(241)ƒ(246)œ(372)¿(277)˚(312)§(247)£(243)

Some extra characters in Symbol
∠(320)〈(341)〉(361)≈(273)↔(253)⇔(333)⇐(334)⇒(336)←(254)→(256)|(174)
⊗(304)⊕(305)°(260)÷(270)∈(316)…(274)∅(306)≡(272)ƒ(246)∇(321)≥(263)
∞(245)ȷ(362)≤(243)×(264)≠(271)∏(325)∂(266)±(261)√(326)∑(345)∪(310)

5.2 Plotting characters outside the keyboard set

To use characters outside the keyboard set you have to use the corresponding octal codes. Note that these codes represent just one byte in PostScript so, in this special case, four string characters need only one key character. For example, such codes as `\277` for an upside down question mark in standard encoding, or `\326` for a square root sign in Symbol, only need one index key. You might wonder why, if Simplot can put any accent on any character and there are maths and bold maths fonts, you would ever want alternative encodings, like the ISOLatin1Encoding. This is because the ISOLatin1Encoding allows you to use specially formed accented letters, which are more accurately proportioned than those generated by program Simplot by adding the accent as a second over-printing character, e.g. using `\361` for n tilde is more professional than overprinting.
All the characters present in the coding vectors to be shown next can be used by program Simplot, as well as a special Maths/Greek font and a vast number of accented letters and graphical objects, but several points must be remembered.

All letters can be displayed using GSview/Ghostscript and then Adobe Acrobat after distilling to pdf. Although substitutions can be made interactively from Simplot, you can also save a .eps file and edit it in a text editor. When using an octal code to introduce a non-keyboard character, only use one index key for the four character code. If you do not have a PostScript printer, save plots as .eps files and print from GSview/Ghostscript or transform into graphics files to include in documents.

Some useful codes follow, then by examples to clarify the subject. You will find it instructive to view `simfonts.ps` in the SIMFIT viewer and display it in GSview/Ghostcsript.

5.2.1 The StandardEncoding Vector

octal	0	1	2	3	4	5	6	7
\00x								
\01x								
\02x								
\03x								
\04x		!	"	#	$	%	&	’
\05x	(	)	*	+	,	-	.	/
\06x	0	1	2	3	4	5	6	7
\07x	8	9	:	;	<	=	>	?
\10x	@	A	B	C	D	E	F	G
\11x	H	I	J	K	L	M	N	O
\12x	P	Q	R	S	T	U	V	W
\13x	X	Y	Z	[	\	]	^	_
\14x	‘	a	b	c	d	e	f	g
\15x	h	i	j	k	l	m	n	o
\16x	p	q	r	s	t	u	v	w
\17x	x	y	z	{	|	}	~	
\20x								
\21x								
\22x								
\23x								
\24x		¡	¢	£	⁄	¥	*ƒ*	§
\25x	¤	'	“	«	‹	›	fi	fl
\26x		–	†	‡	·		¶	•
\27x	‚	„	”	»	…	‰		¿
\30x		`	´	ˆ	˜	¯	˘	˙
\31x	¨		°	¸		˝	˛	ˇ
\32x	—							
\33x								
\34x		Æ		ª				
\35x	Ł	Ø	Œ	º				
\36x		æ				ı		
\37x	ł	ø	œ	ß				

5.2.2 The ISOLatin1Encoding Vector

octal	0	1	2	3	4	5	6	7
\00x								
\01x								
\02x								
\03x								
\04x		!	"	#	$	%	&	’
\05x	(	)	*	+	,	−	.	/
\06x	0	1	2	3	4	5	6	7
\07x	8	9	:	;	<	=	>	?
\10x	@	A	B	C	D	E	F	G
\11x	H	I	J	K	L	M	N	O
\12x	P	Q	R	S	T	U	V	W
\13x	X	Y	Z	[	\	]	^	_
\14x	‘	a	b	c	d	e	f	g
\15x	h	i	j	k	l	m	n	o
\16x	p	q	r	s	t	u	v	w
\17x	x	y	z	{	|	}	~	
\20x								
\21x								
\22x	ı	`	´	ˆ	˜	¯	˘	˙
\23x	¨		°	¸		˝	˛	ˇ
\24x		¡	¢	£	¤	¥	¦	§
\25x	¨	©	ª	«	¬	-	®	¯
\26x	°	±	2	3	´	µ	¶	·
\27x	¸	1	º	»	¼	½	¾	¿
\30x	À	Á	Â	Ã	Ä	Å	Æ	Ç
\31x	È	É	Ê	Ë	Ì	Í	Î	Ï
\32x	Đ	Ñ	Ò	Ó	Ô	Õ	Ö	×
\33x	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
\34x	à	á	â	ã	ä	å	æ	ç
\35x	è	é	ê	ë	ì	í	î	ï
\36x	ð	ñ	ò	ó	ô	õ	ö	÷
\37x	ø	ù	ú	û	ü	ý	þ	ÿ

5.2.3 The SymbolEncoding Vector

octal	0	1	2	3	4	5	6	7
\00x								
\01x								
\02x								
\03x								
\04x		!	∀	#	∃	%	&	∍
\05x	(	)	*	+	,	−	.	/
\06x	0	1	2	3	4	5	6	7
\07x	8	9	:	;	<	=	>	?
\10x	≅	Α	Β	Χ	Δ	Ε	Φ	Γ
\11x	Η	Ι	ϑ	Κ	Λ	Μ	Ν	Ο
\12x	Π	Θ	Ρ	Σ	Τ	Υ	ς	Ω
\13x	Ξ	Ψ	Ζ	[	∴	]	⊥	_
\14x	‾	α	β	χ	δ	ε	ϕ	γ
\15x	η	ι	φ	κ	λ	μ	ν	ο
\16x	π	θ	ρ	σ	τ	υ	ϖ	ω
\17x	ξ	ψ	ζ	{	|	}	~	
\20x								
\21x								
\22x								
\23x								
\24x		ϒ	′	≤	⁄	∞	*f*	♣
\25x	♦	♥	♠	↔	←	↑	→	↓
\26x	°	±	″	≥	×	∝	∂	•
\27x	÷	≠	≡	≈	…	|	—	↵
\30x	ℵ	ℑ	ℜ	℘	⊗	⊕	∅	∩
\31x	∪	⊃	⊇	⊄	⊂	⊆	∈	∉
\32x	∠	∇	®	©	™	∏	√	·
\33x	¬	∧	∨	⇔	⇐	⇑	⇒	⇓
\34x	◊	〈	®	©	™	∑	⎛	⎜
\35x	⎝	⎡	⎢	⎣	⎧	⎨	⎩	⎪
\36x		〉	⌠	⌡	⎪	⎭	⎫	⎪
\37x	⎭	⎤	⎥	⌋	⎫	⎬	⌋	

5.2.4 The ZapfDingbatsEncoding Vector

octal	0	1	2	3	4	5	6	7
\00x								
\01x								
\02x								
\03x								
\04x								
\05x								
\06x				✓	✔	✕	✕	✗
\07x	✘		✚	✛	✜	†		
\10x		✡						
\11x	★	☆	✪	✯	✰	✰	✰	✰
\12x	✰	✶	✷	✸	✶		✶	✶
\13x	✶	✹	✺	✻	✻	✻		
\14x			✼					✳
\15x	✳	✼	✳	✳	●	❍	■	❑
\16x	❒	❑	❒	▲	▼	◆	❖	◗
\17x				❛	❜	❝	❞	
\20x	❨	❩	❪	❫	❬	❭	❮	❯
\21x	❰	❱	[	]	❴	❵		
\22x								
\23x								
\24x		❡	❢	❣	❤	❥		
\25x	♣	♦	♥	♠	①	②	③	④
\26x	⑤	⑥	⑦	⑧	⑨	⑩	❶	❷
\27x	❸	❹	❺	❻	❼	❽	❾	❿
\30x	①	②	③	④	⑤	⑥	⑦	⑧
\31x	⑨	⑩	❶	❷	❸	❹	❺	❻
\32x	❼	❽	❾	❿	➔	→	↔	↕
\33x	➘	→	➘	→	➔	→	→	➙
\34x	➙	➡	➢	➢	➤	➥	➦	➧
\35x	➡	➪	➪	➪	➪	➪	➪	➪
\36x		➪	➲	➳	➴	➵	➶	➷
\37x	➸	➹	➺	➻	➼	➽	➾	

6 Simfit character display codes

0 Standard font
1 Standard font subscript
2 Standard font superscript
3 Maths/Greek
4 Maths/Greek subscript
5 Maths/Greek superscript
6 Bold Maths/Greek
7 ZapfDingbats (PostScript) Wingding (Windows)
8 ISOLatin1Encoding (PostScript), Standard (Windows, almost)
9 Special (PostScript) Wingding2 (Windows)
A Grave accent
B Acute accent
C Circumflex/Hat
D Tilde
E Macron/Bar/Overline
F Dieresis
G Maths/Greek-hat
H Maths/Greek-bar
I Bold maths/Greek-hat
J Bold Maths/Greek-bar
K Symbol font
L Bold Symbol font

You will need non-keyboard characters from the standard font for such characters as a double dagger (‡) or upside down question mark (¿), e.g. typing `\277` in a text string would generate the upside down question mark (¿) in the PostScript output. If you want to include a single backslash in a text string, use `\\`, and also cancel any unpaired parentheses using `\(` and `\)`. Try it in program SIMPLOT and it will then all make sense. The ISOLatin1Encoding vector is used for special characters, such as `\305` for Angstrom (Å), `\361` for n-tilde (ñ), or `\367` for the division sign (÷), and, apart from a few omissions, the standard Windows font is the same as the ISOLatin1Encoding. The Symbol and ZapfDingbats fonts are used for including special graphical characters like scissors or pointing hands in a text string.

A special font is reserved for PostScript experts who want to add their own character function. Note that, in a document with many graphs, the prologue can be cut out from all the graphs and sent to the printer just once at the start of the job. This compresses the PostScript file, saves memory and speeds up the printing. Examine the manuals source code for this technique.

If you type four character octal codes as character strings for plotting non-keyboard characters, you do not have to worry about adjusting the character display codes, program SIMPLOT will make the necessary corrections. The only time you have to be careful about the length of character display code vectors is when editing in a text editor. If in doubt, just pad the character display code vector with question marks until it is the same length as the character string.

7 editps text formatting commands

Program **editps** uses the SIMFIT convention for text formatting characters within included SIMFIT .eps files but, because this is rather cumbersome, a simplified set of formatting commands is available within **editps** whenever you want to add text, or even create PostScript files containing text only. The idea of these formatting commands is to allow you to introduce superscripts, subscripts, accented letters, maths, dashed lines or plotting symbols into PostScript text files, or into collage titles, captions, or legends, using only ASCII text controls. To use a formatting command you simply introduce the command into the text enclosed in curly brackets as in: {raise}, {lower}, {newline}, and so on. If {anything} is a recognized command then it will be executed when the .eps file is created. Otherwise the literal string argument, i.e. anything, will be printed with no inter-word space. Note that no {commands} add interword spaces, so this provides a mechanism to build up long character strings and also control spacing; use {anything} to print anything with no trailing inter-word space, or use { } to introduce an inter-word space character. To introduce spaces for tabbing, for instance, just use {newline}{ }start-of-tabbing, with the number of spaces required inside the { }. Note that the commands are both spelling and case sensitive, so, for instance, {21}{degree}{C} will indicate the temperature intended, but {21}{degrees}{C} will print as 21degreesC while {21}{Degree}{C} will produce 21DegreeC.

7.0.5 Special text formatting commands, e.g. left

{left} ... use {left} to print a {
{right} ... use {right} to print a }
{%!command} ... use {%!command} to issue command as raw PostScript

The construction {%!command} should only be used if you understand PostScript. It provides PostScript programmers with the power to create special effects. For example {%!1 0 0 setrgbcolor}, will change the font colour to red, and {%!0 0 1 setrgbcolor} will make it blue, while {%!2 setlinewidth} will double line thickness. In fact, with this feature, it is possible to add almost any conceivable textual or graphical objects to an existing .eps file.

7.0.6 Coordinate text formatting commands, e.g. raise

{raise} ... use {raise} to create a superscript or restore after {lower}
{lower} ... use {lower} to create a subscript or restore after {raise}
{increase} ... use {increase} to increase font size by 1 point
{decrease} ... use {decrease} to decrease font size by 1 point
{expand} ... use {expand} to expand inter-line spacing by 1 point
{contract} ... use {contract} to contract inter-line spacing by 1 point

7.0.7 Currency text formatting commands, e.g. dollar

{dollar} $ {sterling} £ {yen} Y

7.0.8 Maths text formatting commands, e.g. divide

{divide} ÷ {multiply} × {plusminus} ±

7.0.9 Scientific units text formatting commands, e.g. Angstrom

{Angstrom} Å {degree} ∘ {micron} μ

7.0.10 Font text formatting commands, e.g. roman

{roman} {bold} {italic} {helvetica}
{helveticabold} {helveticaoblique} {symbol} {zapfchancery}
{zapfdingbats} {isolatin1}

Note that you can use octal codes to get extra-keyboard characters, and the character selected will depend on whether the StandardEncoding or IOSLatin1Encoding is current. For instance, \ 361 will locate an {ae} character if the StandardEncoding Encoding Vector is current, but it will locate a {ñ} character if the ISOLatin1Encoding Encoding Vector is current, i.e. the command {isolatin1} has been used previously. The command {isolatin1} will install the ISOLatin1Encoding Vector as the current Encoding Vector until it is cancelled by any font command, such as {roman}, or by any shortcut command such as {ntilde} or {alpha}. For this reason, {isolatin1} should only be used for characters where shortcuts like {ntilde} are not available.

7.0.11 Poor man's bold text formatting command, e.g. pmb?

The command {pmb?} will use the same technique of overprinting as used by the Knuth TEX macro to render the argument, that is ? in this case, in bold face font, where ? can be a letter or an octal code. This is most useful when printing a boldface character from a font that only exists in standard typeface. For example, {pmbb} will print a boldface letter b in the current font then restore the current font, while {symbol}{pmbb}{roman} will print a boldface beta then restore roman font. Again, {pmb\ 243} will print a boldface pound sign.

7.0.12 Punctuation text formatting commands, e.g. dagger

{dagger} †	{daggerdbl} ‡	{paragraph} ¶	{subsection} §
{questiondown} ¿			

7.0.13 Letters and accents text formatting commands, e.g. Aacute

{Aacute} Á	{agrave} à	{aacute} á	{acircumflex} â
{atilde} ã	{adieresis} ä	{aring} å	{ae} æ
{ccedilla} ç	{egrave} è	{eacute} é	{ecircumflex} ê
{edieresis} ë	{igrave} ì	{iacute} í	{icircumflex} î
{idieresis} ï	{ntilde} ñ	{ograve} ò	{oacute} ó
{ocircumflex} ô	{otilde} õ	{odieresis} ö	{ugrave} ù
{uacute} ú	{ucircumflex} û	{udieresis} ü	

All the other special letters can be printed using {isolatin1} (say just once at the start of the text) then using the octal codes, for instance {isolatin1}{\ 303} will print an upper case ntilde.

7.0.14 Greek text formatting commands, e.g. alpha

{alpha} α	{beta} β	{chi} χ	{delta} δ
{epsilon} ϵ	{phi} ϕ	{gamma} γ	{eta} η
{kappa} κ	{lambda} λ	{mu} μ	{nu} ν
{pi} π	{theta} θ	{rho} ρ	{sigma} σ
{tau} τ	{omega} ω	{psi} ψ	

All the other characters in the Symbol font can be printed by installing Symbol font, supplying the octal code, then restoring the font, as in {symbol}{\ 245}{roman} which will print infinity, then restore Times Roman font.

7.0.15 Line and Symbol text formatting commands, e.g. ce

{li} = line
{da} = dashed line
{do} = dotted line
{dd} = dashed dotted line
{ce}, {ch}, {cf} = circle (empty, half filled, filled)
{te}, {th}, {tf} = triangle (empty, half filled, filled)
{se}, {sh}, {sf} = square (empty, half filled, filled)

{de}, {dh}, {df} = diamond (empty, half filled, filled)

These line and symbol formatting commands can be used to add information panels to legends, titles, etc. to identify plotting symbols.

7.0.16 Examples of text formatting commands

{TGF}{beta}{lower}{1}{raise} is involved
TGFβ_1 is involved

y = {x}{raise}{2}{lower} + 2
$y = x^2 + 2$

The temperature was {21}{degree}{C}
The temperature was 21°C

{pi}{r}{raise}{decrease}{2}{increase}{lower} is the area of a circle
πr^2 is the area of a circle

The {alpha}{lower}{2}{raise}{beta}{lower}{2}{raise} isoform
The $\alpha_2\beta_2$ isoform

{[Ca}{raise}{decrease}{++}{increase}{lower}{]} = {2}{mu}{M}
$[Ca^{++}] = 2\mu M$

8 Scaling, rotating, and stretching

One of the features SIMFIT provides for EPS PostScript files is the ability to manipulate the metric features of the files as they are created, or retrospectively using program **editps**. The following procedures, for instance, are frequently required.

- Changing the aspect ratio and clipping;
- Scaling, rotating, and shearing;
- Adjusting fonts and line thicknesses if graphs are enlarged or reduced; and
- Stretching the white space between items plotted without altering the aspect ratios of characters or symbols.

This is possible because of two special features in SIMFIT PostScript files.

1. The BoundingBox and clipping coordinates, as in the typical header section shown next.

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox:    72   252   520   588
%%Creator: Simfit Version 7.2.8 (simfit.org.uk)
%%Title: colours=72/ISOLatin1Encoding/Accents/special/PSfrag/dict=300
%%CreationDate: Saturday, 15 April 2017
%%EndComments
%
%Start of SIMFIT PostScript file
%
save %save current state before clipping, etc.
    70   250   522   250   522   590    70   590%#clipping
newpath moveto lineto lineto lineto closepath clip newpath
   72.00  252.00 translate    0.07    0.07 scale    0.00 rotate%#portrait
   12.00 setlinewidth 0 setlinecap 1 setlinejoin [] 0 setdash
    2.50 setmiterlimit
%
```

   The BoundingBox holds the overall dimensions of the EPS file in PostScript units (1/72 inches) while the rest defines the clipping and rotating parameters.

2. The special sections indicated by %#, as in the typical data section shown below.

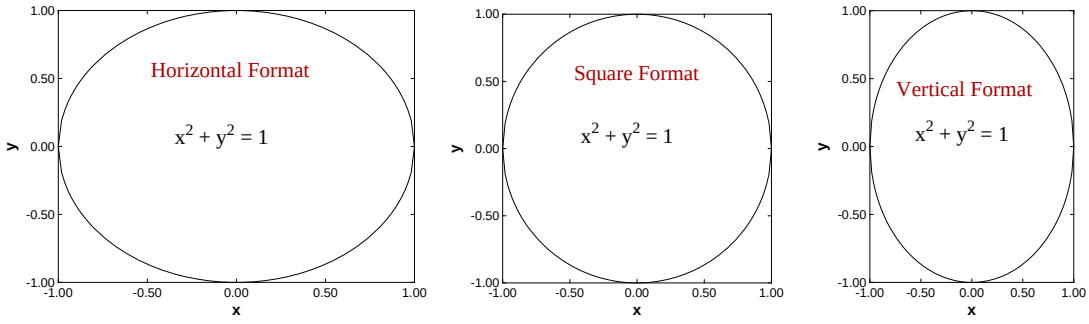
```
c0
1070 671 59 ce%#2
3514 2443 59 ce%#2
5959 4215 58 ce%#2
```

   Text following any % sign are taken as comments in PostScript files, so the parameters following the %# are specific SIMFIT stretching factors to alter the preceding coordinates. These stretch white space and alter the starting coordinates for symbols, lines, and character strings but not the inter-word separation, so preserving the aspect ratio of fonts.

Of course the brave may wish to alter these themselves using a text editor but, as this is very tedious and error-prone, the SIMFIT package provides functions to perform the editing

8.1 Alternative sizes, shapes and clipping

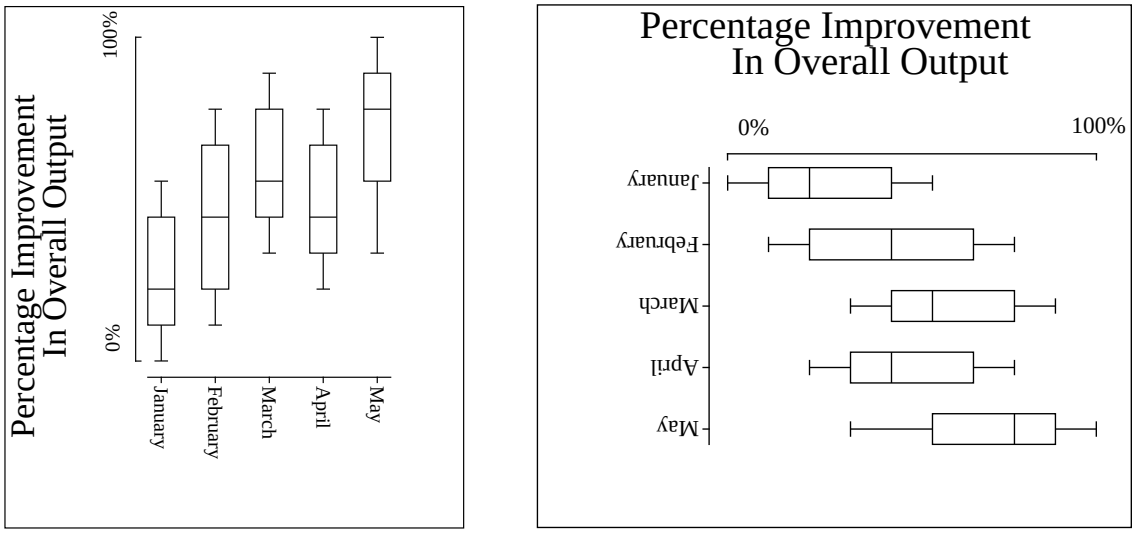
Plots can have horizontal, square or vertical format as in the next figure, and user-defined clipping schemes can be used. After clipping, SIMFIT adds a standard BoundingBox so all plots with the same clipping scheme will have the same absolute size but, when **GSview/Ghostscript** transforms ps into eps, it clips individual files to the boundary of white space and the desirable property of equal dimensions will be lost.



There is also a stretched format for long horizontal ribbon type graphs and a PostScript option to stretch white space without altering symbols and fonts, which is very useful with dense data sets such as dendrograms.

8.2 Rotated and re-scaled graphs

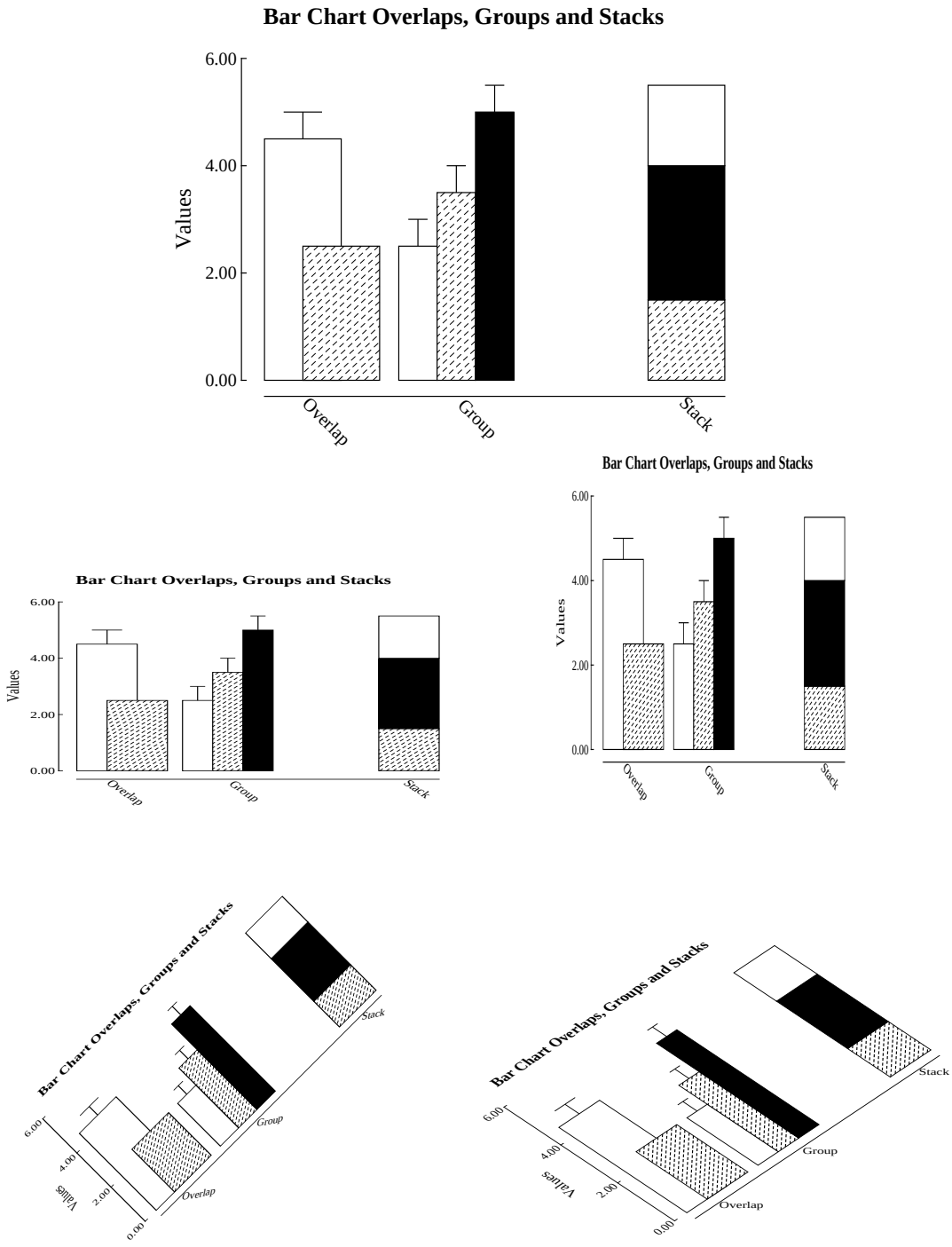
PostScript files can be read into **editps** which has options for re-sizing, re-scaling, editing, rotating, making collages, etc. In the next figure the box and whisker plot was turned on its side to generate a side-on barchart. To do this sort of thing you should learn how to browse a SIMFIT PostScript file in the SIMFIT viewer to read BoundingBox coordinates, in PostScript units of 72 to one inch, and calculate how much to translate, scale, rotate, etc.



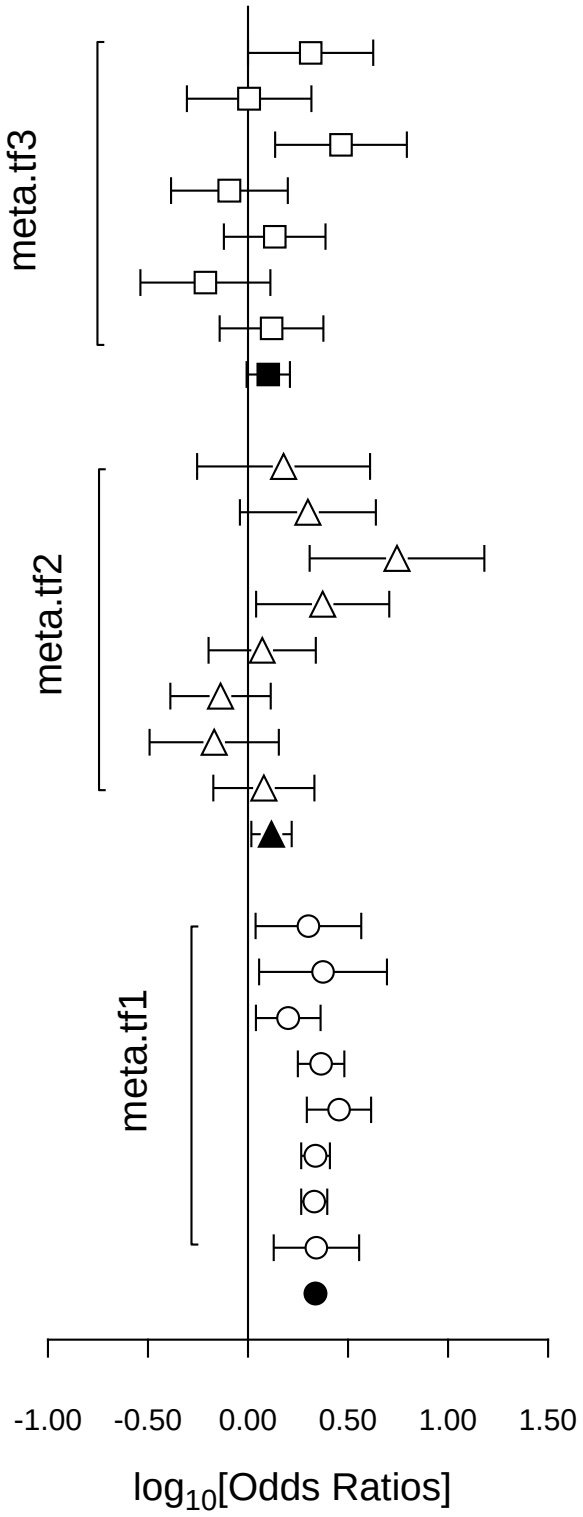
PostScript users should be warned that the special structure of SIMFIT PostScript files that allows extensive retrospective editing using **editps**, or more easily if you know how using a simple text editor like **notepad**, is lost if you read such graphs into a graphics editor program like Adobe Illustrator. Such programs start off by redrawing vector graphics files into their own conventions which are only machine readable.

8.3 Changed aspect ratios and shear transformations

The barchart in the figure below was scaled to make the X-axis longer than the Y-axis and vice-versa, but note how this type of differential scaling changes the aspect ratio as illustrated. Since rotation and scaling do not commute, the effect created depends on the order of concatenation of the transformation matrices. For instance, scaling then rotation cause shearing which can be used to generate 3-dimensional perspective effects as in the last sub-figure.



8.4 Plotting combined meta analysis results



It is often useful to plot Log-Odds-Ratios, so the creation of the adjacent figure will be outlined.

(1) **The data**
Test files `meta.tf1`, `meta.tf2`, and `meta.tf3` were analyzed in sequence using the SIMFIT Meta Analysis procedure. Note that, in these files, column 3 contains spacing coordinates so that data will be plotted consecutively.

(2) **The ASCII coordinate files**
During Meta Analysis, $100(1-\alpha)\%$ confidence limits on the Log-Odds-Ratio resulting from a 2 by 2 contingency tables with cell frequencies n_{ij} can be constructed from the approximation $\hat{e}$ where

$$\hat{e} = Z_{\alpha/2}\sqrt{\frac{1}{n_{11}} + \frac{1}{n_{12}} + \frac{1}{n_{21}} + \frac{1}{n_{22}}}.$$

When Log-Odds-Ratios with error bars are displayed, the overall values (shown as filled symbols) with error bars are also plotted with a x coordinate one less than smallest x value on the input file. For this figure, error bar coordinates were transferred into the project archive using the [Advanced] option to save ASCII coordinate files.

(3) **Creating the composite plot**
Program **simplot** was opened and the six error bar coordinate files were retrieved from the project archive. Experienced users would do this more easily using a library file of course. Reverse y-semilog transformation was selected, symbols were chosen, axes, title, and legends were edited, then half bracket hooks identifying the data were added as arrows and extra text.

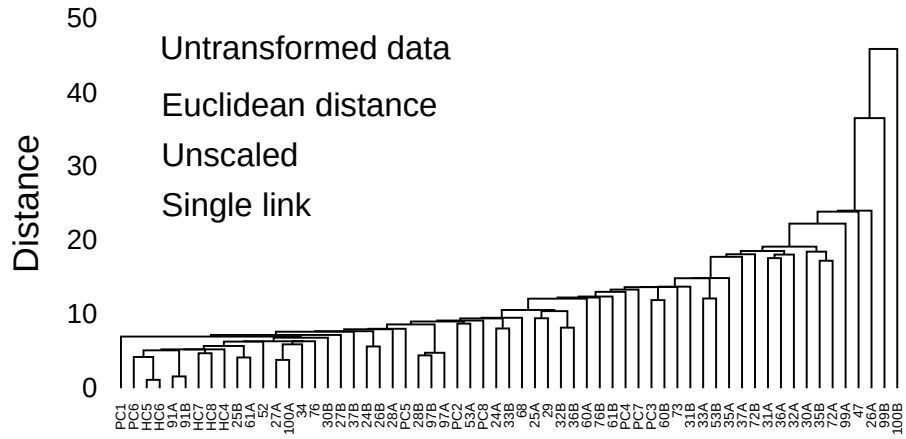
(4) **Creating the PostScript file**
Vertical format was chosen then, using the option to stretch PostScript files, the y coordinate was stretched by a factor of two.

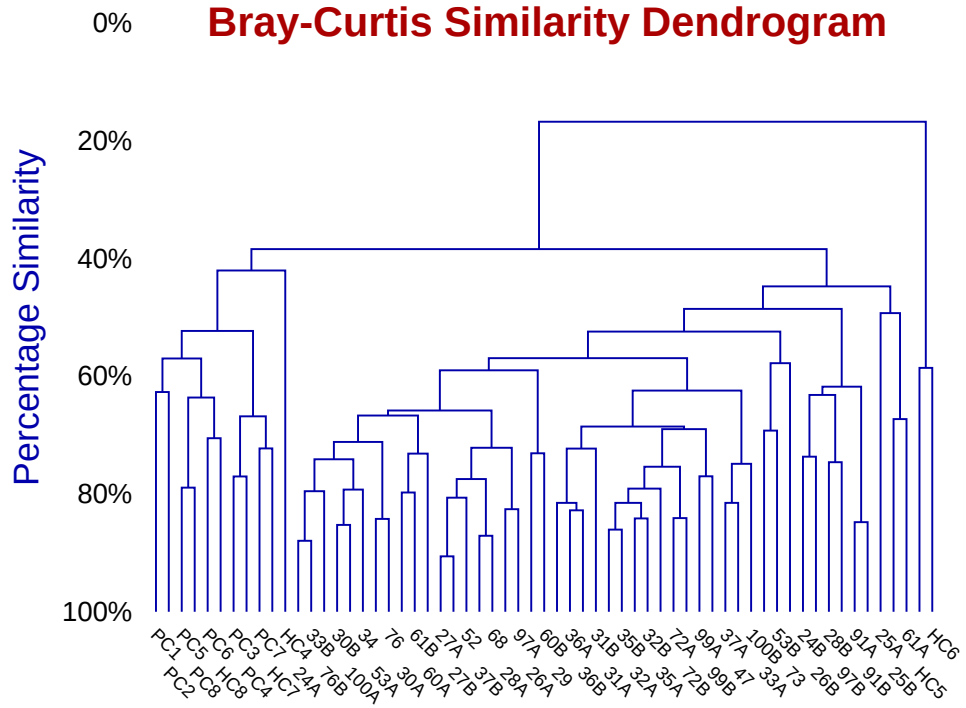
(5) **Editing the PostScript file**
To create the final PostScript file for LaTeX a tighter bounding box was calculated using **gsview** then, using **notepad**, clipping coordinates at the top of the file were set equal to the BoundingBox coordinates, to suppress excess white space. This can also be done using the [Style] option to omit painting a white background, so that PostScript files are created with transparent backgrounds, i.e. no white space, and clipping is irrelevant.

8.5 Plotting dendrograms: standard format

Dendrogram shape is arbitrary in two ways; the x axis order is arbitrary as clusters can be rotated around any clustering distance leading to 2^{n-1} different orders, and the distance matrix depends on the settings used. For instance, a square root transformation, Bray-Curtis similarity, and a group average link generates the second dendrogram in this figure from the first. The y plotted are dissimilarities, while labels are $100 - y$, which should be remembered when changing the y axis range.

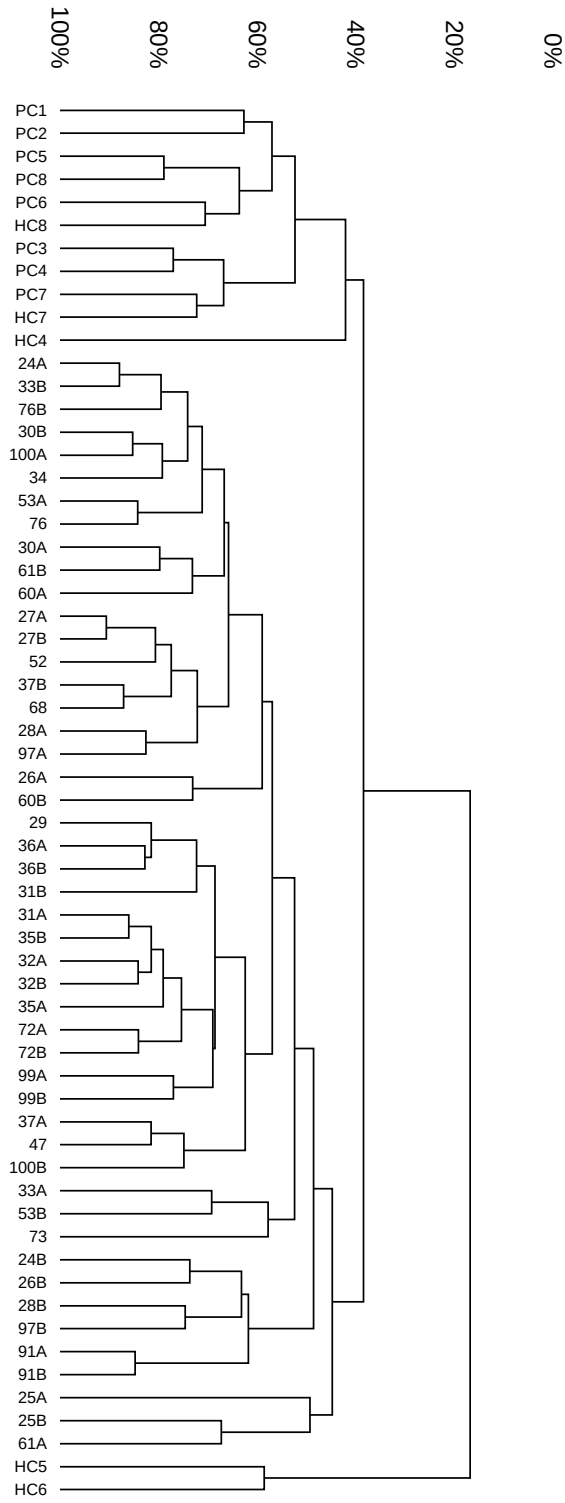
Users should not manipulate dendrogram parameters to create a dendrogram supporting some preconceived clustering scheme. You can set a label threshold and translation distance from the [X-axis] menu so that, if the number of labels exceeds the threshold, even numbered labels are translated, and font size is decreased.





8.6 Plotting dendrograms: stretched format

Sometimes dendrograms are more readable if the white space is stretched without distorting the labels. So SIMFIT PostScript graphs have a very useful feature: you can stretch or compress the white space between plotted lines and symbols without changing the line thickness, symbol size, or font size and aspect ratio. For instance, stretching, clipping and sliding procedures are valuable in graphs which are crowded due to overlapping symbols or labels, as in previous figures. If such dendrograms are stretched retrospectively using **editps**, the labels will not separate as the fonts will also be stretched so letters become ugly due to altered aspect ratios. SIMFIT can increase white space between symbols and labels while maintaining correct aspect ratios for the fonts in PostScript hardcopy and, to explain this, the creation of this stretrched figure will be described.

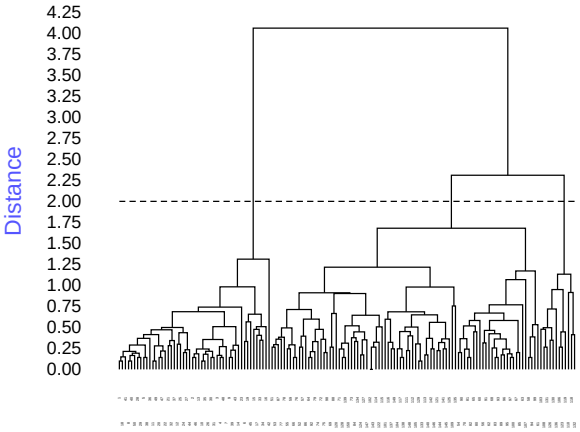


The title, legend and double x labeling were suppressed, and landscape mode with stretching, clipping and sliding was selected from the PostScript control using the [Shape] then [Landscape +] options, with an x stretching factor of two. Stretching increases the space between each symbol, or the start of each character string, arrow or other graphical object, but does not turn circles into ellipses or distort letters. As graphs are often stretched to print on several sheets of paper, sub-sections of the graph can be clipped out, then the clipped sub-sections can be slid to the start of the original coordinate system to facilitate printing.

If stretch factors greater than two are used, legends tend to become detached from axes, and empty white space round the graph increases. To remedy the former complication, the default legends should be suppressed or replaced by more closely positioned legends while, to cure the later effect, GSview can be used to calculate new BoundingBox coordinates (by transforming .ps to .eps). If you select the option to plot an opaque background even when white (by mistake), you may then find it necessary to edit the resulting .eps file in a text editor to adjust the clipping coordinates (identified by %#clip in the .eps file) and background polygon filling coordinates (identified by %#pf in the .ps file) to trim away unwanted white background borders that are ignored by GSview when calculating BoundingBox coordinates. Another example of this technique is with meta analysis plots, where it is also pointed out that creating transparent backgrounds by suppressing the painting of a white background obviates the need to clip away extraneous white space.

8.7 Plotting dendrograms: subgroups

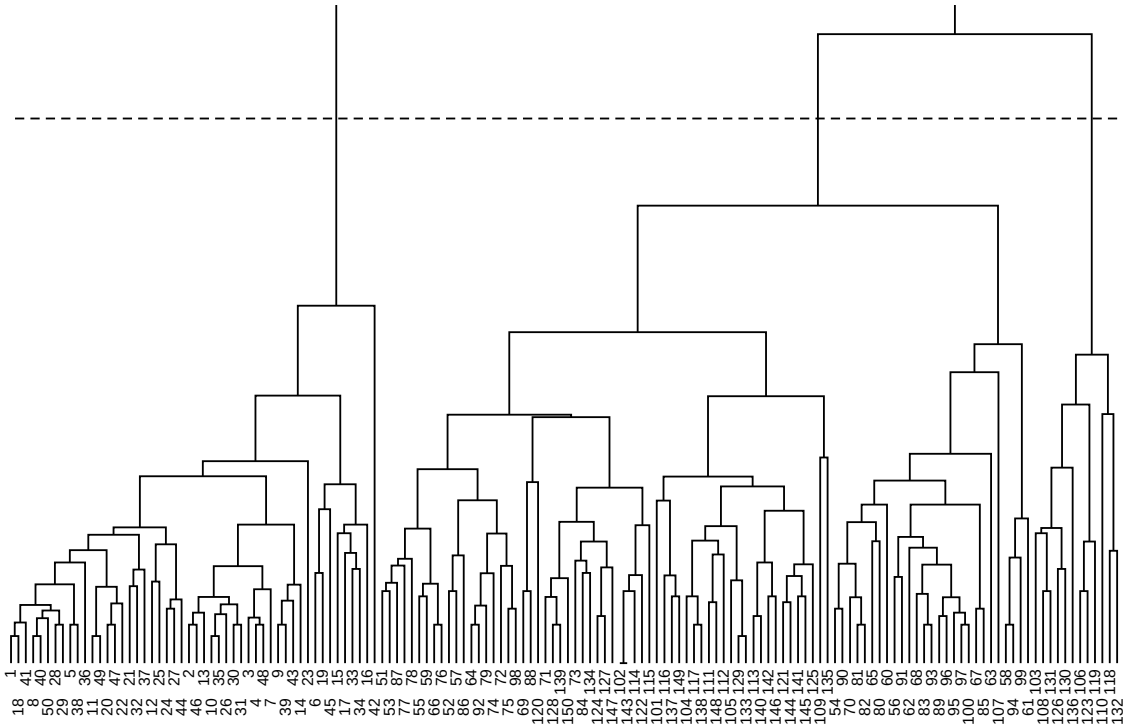
The procedures described can also be used to improve the readability of dendrograms where subgroups have been assigned by partial clustering. The next figure shows a graph from `iris.tf1` when three subgroups are requested, or a threshold is set corresponding to the horizontal dotted line. The figure was created by these steps.



First the title was suppressed, the y-axis range was changed to $(0, 4.25)$ with 18 tick marks, the (x, y) offset was canceled as this suppresses axis moving, the label font size was increased from 1 to 3, and the x-axis was translated to 0.8.
Then the PostScript stretch/slide/clip procedure was used with these parameters

$$\begin{aligned} x_{\text{stretch}} &= 1.5 \\ y_{\text{stretch}} &= 2.0 \\ x_{\text{clip}} &= 0.15, 0.95 \\ y_{\text{clip}} &= 0.10, 0.60. \end{aligned}$$

This generates the following graph.



Windows users without PostScript printing facilities must create a `*.eps` file using this technique, then use the SimFIT procedures to create a graphics file they can use, e.g. `*.jpg`. Use of a larger font and increased x-stretching would be required to read the labels, of course.

9 PostScript specials

SimFIT PostScript files are designed to facilitate editing, and one important type of editing is to be able to specify text files, known as specials, that can modify the graph in an almost unlimited number of ways. This technique will now be described but, if you want to do it and you are not a PostScript programmer, do not even think about it; get somebody who has the necessary skill to do what you want. An example showing how to display a logo will be seen on page 45 and further details follow.

9.1 What specials can do

First of all, here are some examples of things you may wish to do with SimFIT PostScript files that would require specials.

- ❏ Replace the 35 standard fonts by special user-defined fonts.
- ❏ Add a logo to plots, e.g. a departmental heading for slides.
- ❏ Redefine the plotting symbols, line types, colours, fill styles, etc.
- ❏ Add new features, e.g. outline or shadowed fonts, or clipping to non-rectangular shapes.

When SimFIT PostScript files are created, a header subsection, called a prologue, is placed at the head of the file which contains all the definitions required to create the SimFIT dictionary. Specials can be added, as independent text files, to the files after these headings in order to re-define any existing functions, or even add new PostScript plotting instructions. The idea is is very simple; you can just modify the existing SimFIT dictionary, or even be ambitious and add completely new and arbitrary graphical objects.

9.2 The technique for defining specials

Any SimFIT PostScript file can be taken into a text editor in order to delete the existing header in order to save space in a large document, as done with the SimFIT manual, or else to paste in a special. However, this can also be done interactively by using the font option, accessible from the SimFIT PostScript interface. Since this mechanism is so powerful, and could easily lead to the PostScript graphics being permanently disabled by an incorrectly formatted special, SimFIT always assumes that no specials are installed. If you want to use a special, then you simply install the special and it will be active until it is de-selected or replaced by another special. Further details will be found in the on-line documentation and `w_readme` files, and examples of specials are distributed with the SimFIT package to illustrate the technique. You should observe the effect of the example specials before creating your own. Note that any files created with specials can easily be restored to the default configuration by cutting out the special. So it makes sense to format your specials like the SimFIT example specials `pspecial.1`, etc. to facilitate such retrospective editing. The use of specials is controlled by the file `pspecial.cfg` as now described. The first ten lines are Booleans indicating which of files 1 through 10 are to be included. The next ten lines are the file names containing the special code. There are ten SimFIT examples supplied, and it is suggested that line 1 of your specials should be in the style of these examples. You simply edit the file names in `pspecial.cfg` to install your own specials. The Booleans can be edited interactively from the advanced graphics PS/Fonts option. Note that any specials currently installed are flagged by the SimFIT program manager and specials only work in advanced graphics mode. In the event of problems with PostScript printing caused by specials, just delete `pspecial.cfg`. To summarise.

- ❏ Create the special you want to insert.
- ❏ Edit the file `psecial.cfg` in the SimFIT folder.
- ❏ Attach the special using the Postscript Font option.

9.3 Example codes for PostScript specials

To clarify the structure of SimFIT PostScript specials, just consider the code for the first three examples distributed with the SimFIT package. The file `psecial.1` simply adds a monochrome logo, the file `psecial.2` shows how to add color, while the file `psecial.3` makes more sweeping changes to the color scheme by reversing the definitions for black and white.

- **The PostScript special** `pspecial.1`

```
%file = pspecial.1: add monochrome simfit logo to plot
gsave
/printSIMFIT {0 0 moveto (SIMFIT) show} def
/Times-Italic findfont 300 scalefont setfont
300 4400 translate
.95 -.05 0
{setgray printSIMFIT -10 5 translate} for
1 1 1 setrgbcolor printSIMFIT
grestore
%end of pspecial.1
```

- **The PostScript special** `pspecial.2`

```
%file = pspecial.2: add yellow simfit logo to plot
gsave
/printSIMFIT {0 0 moveto (SIMFIT) show} def
/Times-Italic findfont 300 scalefont setfont
300 4400 translate
.95 -.05 0
{setgray printSIMFIT -10 5 translate} for
0 0 moveto (SIMFIT) true charpath gsave 1 1 0 setrgbcolor fill grestore
grestore
%end of pspecial.2
```

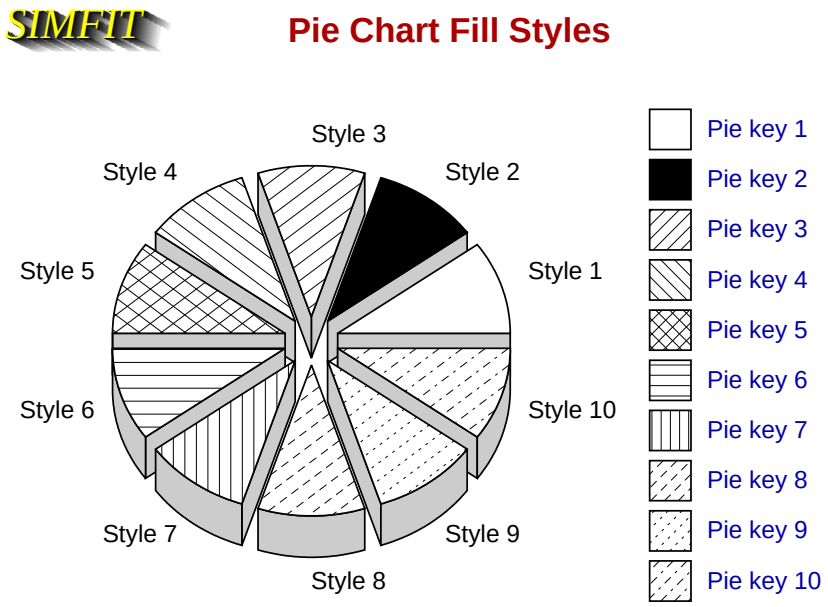
- **The PostScript special** `pspecial.3`

```
%file = pspecial.3: yellow-logo/blue-background/swap-black-and-white
/background{.5 .5 1 setrgbcolor}def
background
0 0 0 4790 6390 4790 6390 0 4 pf
/c0{1 1 1 setrgbcolor}def
/c15{0 0 0 setrgbcolor}def
/foreground{c0}def
gsave
/printSIMFIT {0 0 moveto (SIMFIT) show} def
/Times-Italic findfont 300 scalefont setfont
300 4400 translate
.95 -.05 0
{setgray printSIMFIT -10 5 translate} for
0 0 moveto (SIMFIT) true charpath gsave 1 1 0 setrgbcolor fill grestore
grestore
%end of pspecial.3
```

Remember, the effects of these specials are only visible in the PostScript files created by SimFIT and not in any direct Windows quality hardcopy.

9.4 Example plots for PostScript specials

Figure 8 illustrates the end result of adding logos using such PostScript specials.



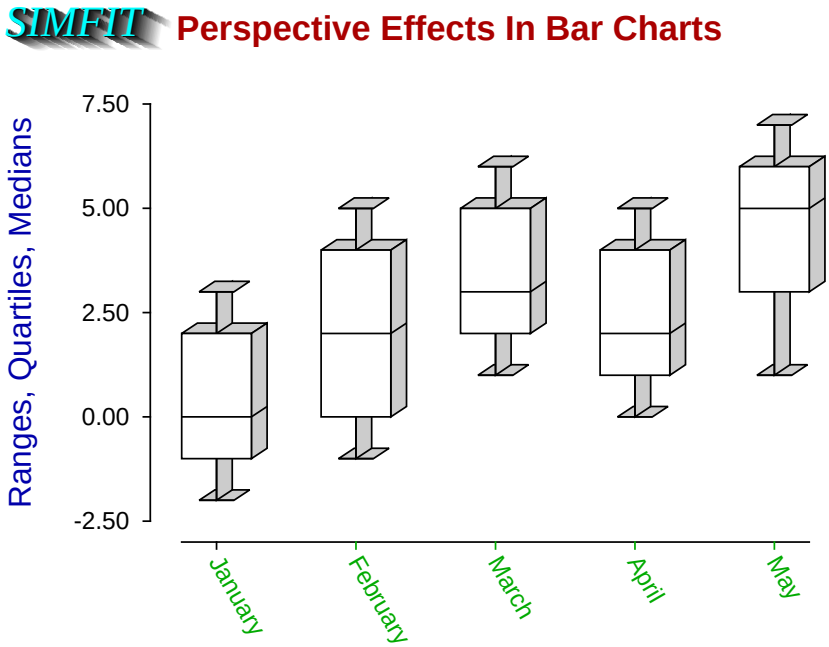


Figure 8: Using PostScript specials to add logos to plots

10 LaTeX options

10.1 Maths

You can add equations to graphs directly, but this will be a compromise, as specialized type setting techniques are required to display maths correctly. The LaTeX system is pre-eminent in the field of maths type-setting and the PSfrag system, as revised by David Carlisle and others, provides a simple way to add equations to SimFiT graphs. For figure 9, **makdat** generated a Normal cdf with $\mu = 0$ and $\sigma = 1$, then **simplot** created `cdf.eps` with the key `phi(x)`, which was then used by this stand-alone code to generate the figure, where the equation substitutes for the key. LaTeX PostScript users should be aware that SimFiT PostScript file format has been specially designed to be consistent with the PSfrag package but, if you want to then use GhostScript to create graphics file, say .png from .eps, the next section should be consulted.

```
\documentclass[dvips,12pt]{article}
   \usepackage{graphicx}
   \usepackage{psfrag}
   \pagestyle{empty}
\begin{document}
\large
\psfrag{phi(x)}{$\displaystyle
   \frac{1}{\sigma \sqrt{2\pi}}
   \int_{-\infty}^x \exp\left\{
  -\frac{1}{2} \left( \frac{t-\mu}{\sigma} \right)^2 \right\}\,dt$}
\mbox{\includegraphics[width=6.0in]{cdf.eps}}
\end{document}
```

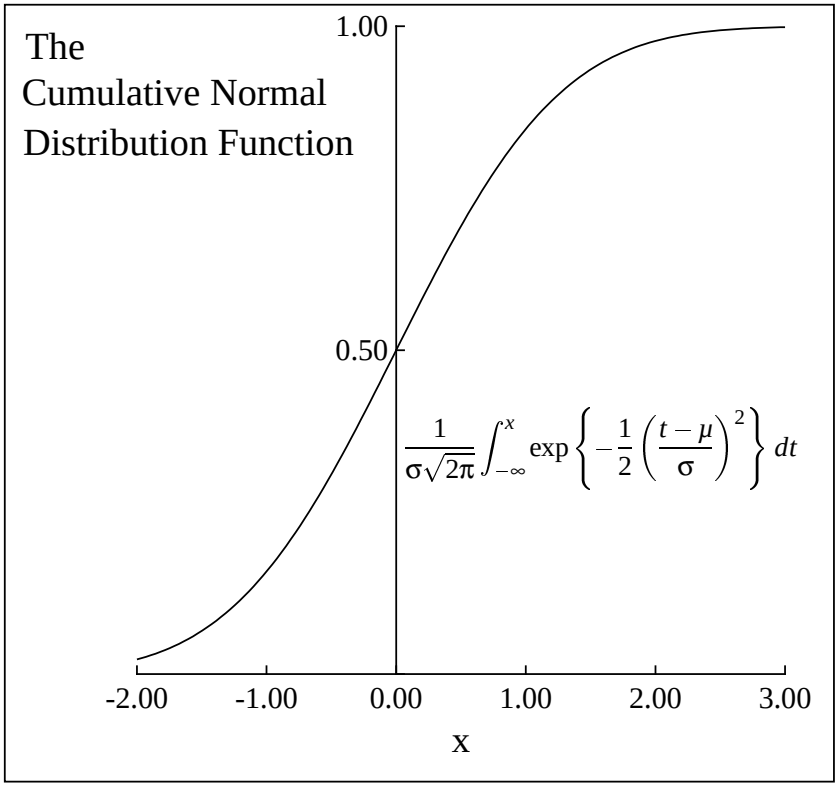


Figure 9: Plotting mathematical equations

10.2 Chemical Formulæ

LaTeX code, as below, is intended for document preparation and adds white space to the final .ps file. The easiest way round this complication is to add an outline box to the plot, as in figure 10. Then, after the .png file has been created, it can be input into, e.g., GIMP, for auto clipping to remove extraneous white space, followed by deletion of the outline box if required.

```
\documentclass[dvips,12pt]{article}
   \usepackage{graphicx}
   \usepackage{psfrag}
   \usepackage{carom}
   \pagestyle{empty}
\begin{document}
\psfrag{formula}
{\begin{picture}(3000,600)(0,0)
\thicklines
\put(0,0){\bzdrv{1==CH$_{2}$NH$_{2}$;4==CH$_{2}$N(Me)$_{2}$}}
\put(700,450){\vector(1,0){400}}
\put(820,550){[O]}
\put(1000,0){\bzdrv{1==CH=0;4==CH$_{2}$N(Me)$_{2}$}}
\put(1650,400){+}
\put(1750,400){NH$_{3}$}
\put(2000,450){\vector(1,0){400}}
\put(2120,550){[O]}
\put(2300,0){\bzdrv{1==C0$_{2}$H;4==CH$_{2}$N(Me)$_{2}$}}
\end{picture}}
\mbox{\includegraphics{chemistry.eps}}
\end{document}
```

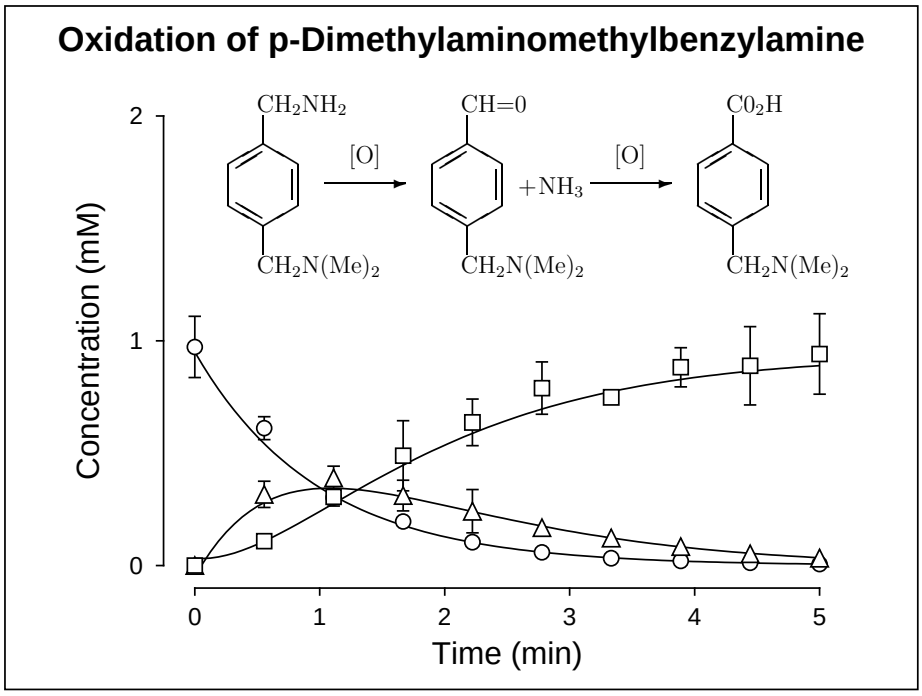


Figure 10: Plotting chemical structures

10.3 Composite graphs

The technique used to combine sub-graphs into a composite graph is easy. First use your drawing or painting program to save the figures of interest in the form of eps files. Then the SimFIT graphs and any component eps files are read into **editps** to move them and scale them until the desired effect is achieved. In figure 11, data were generated using **deqsol**, error was added using **adderr**, the simulated experimental data were fitted using **deqsol**, the plot was made using **simplot**, the chemical formulae and mathematical equations were generated using LaTeX and the final graph was composed using **editps**.

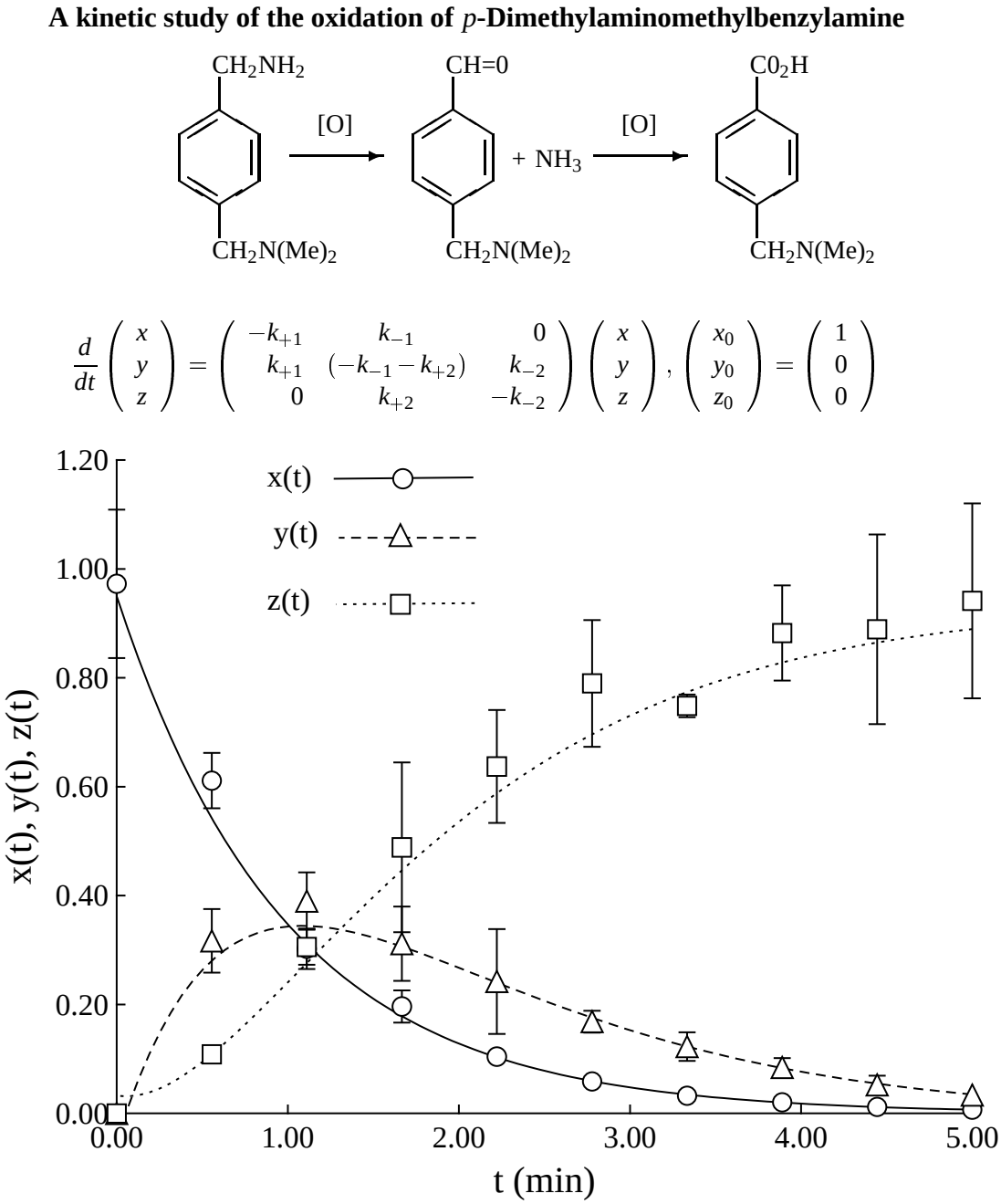


Figure 11: Chemical formulas

Index