Illustrations in TEX documents

Rundqwist, Lars

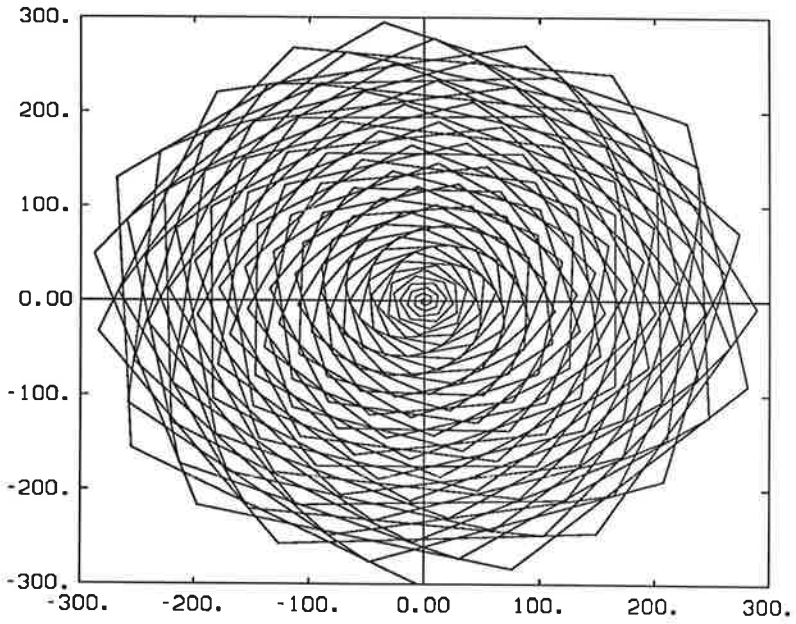
1987

Document Version:
Publisher's PDF, also known as Version of record

CODEN: LUTFD2/(TFRT-7372)/1-15/(1987)

Illustrations in TEX documents

Lars Rundqwist



Department of Automatic Control
Lund Institute of Technology
October 1987

Department of Automatic Control
Lund Institute of Technology
P.O. Box 118
S-221 00 Lund Sweden

Document name
INTERNAL REPORT

Date of issue
October 1987

Document Number
CODEN: LUTFD2/(TFRT–7372)/1-15/(1987)

Author(s)
Lars Rundqwist

Supervisor

Sponsoring organisation

Title and subtitle
Illustrations in TEX documents

Abstract

This document is a department manual about inclusion of illustrations (from PostScript files) in TEX documents.

Key words
TEX, PostScript

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title

ISBN

Language
English

Number of pages
15

Recipient's notes

Security classification

1. Introduction

This document is intended as a department manual about inclusion of illustrations in TEX documents. It covers 'all' known ways of generating PostScript files from other programs like MacDraft, Simnon etc, which are to be inserted in a TEX document and printed on a PostScript device, e.g. an Apple LaserWriter.

The first sections describe step by step, for each application program, how a PostScript file is generated. The file is inserted in the TEX document by the macro PSFIG. A short PSFIG manual and a number of examples are given.

Macintosh programs supported are MacDraft, MacDraw and Adobe Illustrator. Adobe files are stored as PostScript files, while MacDraft and MacDraw both generate PostScript files on request. The PostScript files are then transferred to the Vax, and processed by MDBBOX in order to insert a proper BoundingBox comment. For MacDraw and MacDraft PostScript files the user must manually measure the BoundingBox (in *mm*) and enter the values to MDBBOX.

Vax programs supported are Simnon (and other departemental programs), Control-C, GNUPLOT, and PRO-MATLAB. These programs output a meta coordinate file on request. With the exception of PRO-MATLAB, the files are to be processed by either HCOPY2PS or CC2PS, generating a PostScript file with a BoundingBox. PRO-MATLAB has an own postprocessor, GPP, which outputs PostScript on request, but these files don't contain BoundingBoxes. If the PostScript file is to be inserted in a TEX document, use RGPP instead, which will include a BoundingBox.

Note however that this way of inserting illustrations does not always give the desired result. E.g. spacings between figure and text is sometimes small and other times large. and the figure may seem "uncentered" due to an uncentered BoundingBox. The remedy is manual work (in the TEX file) in cases of high quality requirements. In other cases just leave the result as it is.

Leif Andersson is acknowledged for valuable discussions on how to streamline the use of PostScript files in TEX documents. Further, I stole parts of HCOPY2PS (written by Bengt Mårtensson) during creation of MDBBOX, and I have used some figures created by Ms Britt-Marie Carlsson. Tomas Schöntal quickly received message from John Little at the MathWorks Inc., which resulted in proper handling of PRO-MATLAB PostScript files.

2. MacDraft figures

MacDraft figures are easily inserted in TEX documents using the PSFIG-macro. The figures can be either upright (portrait) or tilted (landscape), see Figure 1. After drawing the figure, a PostScript file is generated. This file is then transferred to the Vax. The PSFIG macro requires a `%%BoundingBox` comment in the PostScript file. An `%%Orientation` comment is required for landscape figures. These two comment lines are inserted by MDBBOX, based on the figure's corner coordinates and data in the PostScript file. The corner coordinates are to be measured by the user. The PostScript file is then ready to be inserted in your TEX file.

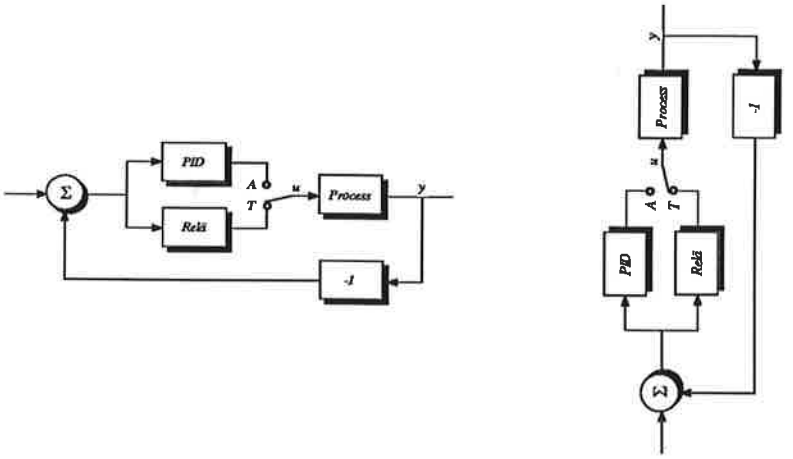


Figure 1. Portrait figure (left) and Landscape figure (right). Block diagrams with permission from Britt-Marie Carlsson.

Preparing the figure

1. Draw the figure in MacDraft!
2. Select the 'PAGE SETUP' menu, and then select papersize A4, orientation Portrait or Landscape (usually Portrait) and a scale small enough to print the figure on a single paper. Print the figure on the LaserWriter. This print is necessary later in the procedure.

Creating the PostScript file

3. To create a PostScript file in MacDraft, hold down ⌘ while pressing first P, then RETURN and finally F, i.e. ⌘ + P + RETURN + F. Keep F depressed until MacDraft verifies that a PostScript file is created.
4. Quit MacDraft.
5. The PostScript file is always named PostScript and is usually located in the same directory as the MacDraft program. Using a Macintosh with a hard disk , you find it in the BILD-directory, otherwise on your MacDraft diskette. If, however, there is a file named PostScript on your application diskette or directory, this file is overwritten by the new PostScript file. If you want to create and transfer more than one PostScript file to the Vax, you must rename the file before you create your next PostScript file. If you want a file named PostScript to remain on your diskette, make a copy instead.
6. Transfer the file(s) to the Vax, see Section 5.

3. MacDraw figures

The procedure for a MacDraw figure is similar to a MacDraft figure, but there is a difference in creating a PostScript file.

Preparing the figure

1. Draw the figure in MacDraw!
2. Select the 'PAGE SETUP' menu, and then select papersize A4, orientation Portrait or Landscape (usually Portrait) and a scale small enough to print

the figure on a single paper. Print the figure on the LaserWriter. This print is necessary later in the procedure.

Creating the PostScript file

3. To create a PostScript file in MacDraw, select PRINT and then press ⌘ F as soon as the mouse button is released. Keep ⌘ F depressed until MacDraw verifies that a PostScript file is created.
4. Quit MacDraw.
5. The PostScript file is always named POSTSCRIPT and is usually located in the same directory as the MacDraw program. Using a Macintosh with a hard disk , you find it in the BILD-directory, otherwise on your MacDraw diskette. If, however, there is a file named POSTSCRIPT on your application diskette or directory, this file is overwritten by the new PostScript file. If you want to create and transfer more than one PostScript file to the Vax, you must rename the file before you create your next PostScript file. If you want a file named POSTSCRIPT to remain on your diskette, make a copy instead.
6. Transfer the file(s) to the Vax, see Section 5.

4. Adobe Illustrator figures

Since Adobe Illustrator files are stored as PostScript files, this section only discusses how to save a proper file.

Preparing the figure

1. Draw the figure in Adobe Illustrator!
2. The figure must (to my belief) be printed on one paper in order to appear correctly in TEX documents. This is accomplished in two steps. First, select the 'PAGE SETUP' menu, and select papersize A4 and a scale small enough. Then you must 'DEFINE PAGE EDGES' to make sure the figure is printed on a single paper. In the the latter procedure, you can place the figure at an arbitrary position on a page.

Creating the PostScript file

This is an almost trivial task.

3. In the 'SAVE AS' menu, specify a file name and select the store option 'PostScript only' (default). Save the file.
4. Quit Adobe Illustrator.
5. Transfer the Adobe Illustrator file(s) to the Vax, see Section 5.

Comment

The Adobe Illustrator PostScript file does not contain any `showpage` command. The figure may be located outside the 'normal PostScript page' on the LaserWriter. Thus the file is not printable by itself, using `PSdump` on a Macintosh or `lwprint` on the Vax.

When Adobe Illustrator prints the page, the program adds translation, rotation and scaling to the file, and sends a **showpage** command. Sofar, landscape figures have not been tested. The PostScript file contains a `%%BoundingBox` comment, but it is not directly readable for PSFIG. Processing the PostScript file by MDBBOX takes care of this.

5. File transfer from Macintosh to Vax

This section desribes file transfer from Macintosh computers to the Vax.

1. Select the Vax-symbol, which invokes MacKermit on the Macintosh. The Macintosh is now a VT100 terminal connected to the Vax. Hit the Return-key for Vax-login, select the desired target directory for your PostScript file and then invoke Kermit on the Vax too. Set the Vax Kermit in `server` mode. This may look like:

```
Username: xxxxx
Password: -----

$ set default [.xxx]
$ kermit
Kermit-32> server
```

2. Now select 'SEND FILE' in the FILE menu. Select drive and directory, and then the desired PostScript file. In the 'As:'-box, give a reasonable Vax/VMS file name, e.g. FIGURE.PS. Then click at the 'SEND' box. If the communication works alright each packet requires less than one second for transfer.
3. Repeat item 2 until all files are transferred.
4. Finish the Vax communication by clicking 'BYE' in the REMOTE menu. If you click 'FINISH' instead of 'BYE', you continue the terminal session at the Vax. Then you must log out in the standard way, i.e. `$ logout`.
5. Quit MacKermit.
6. Insert a BoundingBox comment in your Vax PostScript file, see Section 6.

6. Inserting a proper BoundingBox comment

This section applies to PostScript files from MacDraft, MacDraw and Adobe Illustrator which have been transferred to the Vax.

1. Process the PostScript file by MDBBOX.

MDBBOX inserts a BoundingBox and an Orientation comment into MacDraft and MacDraw PostScript files, taking scaling and orientation into account. Default file name: POSTSCRIPT, default file type .PS.

MDBBOX prompts for the corner coordinates (in *mm*), of the lower left and upper right corners of the figure. These coordinates are all measured from the lower left corner of the paper. The x-direction is to right (along the short side of the paper), and the y-direction is upwards (along the long side of the paper). For a landscape figure, the lower left corner of the paper is located to the upper left of the figure. Compare the landscape figure in Figure 1. An example:

```
$ mdbbox figure
This is MDBBOX dated October 13, 1987
Enter BoundingBox values (in mm)
Lower left x: 20
Lower left y: 50
Upper right x: 185
Upper right y: 250
[*==> Done <==*]
```

MDBBOX does not prompt for any coordinates if the file is created by Adobe Illustrator.

2. The file is ready for insertion by PSFIG in any TEX document.

7. Simnon, Idpac, etc.

These programs may produce a meta coordinate file for the hardcopies. This file is converted to a PostScript file by HCOPY2PS, and is then ready for insertion in any TEX document.

Preparing the figure

1. Prepare the figure by `plot`, `ashow`, etc. in Simnon, Idpac, etc.
2. Create a meta file `filename.p` by the `hcopy` command.

```
> hcopy meta /filename
```

Creating the PostScript file

3. Process `filename.p` by HCOPY2PS.

```
$ hcopy2ps/noprint/... filename
```

The option `/noprint` is necessary if the PostScript file is to be saved on disk. A number of other options for scaling, text fonts, etc. are available. Text fonts and the plot itself can be independently scaled. Thus the best place for choosing figure size is probably here. The output file is `filename.plo`. For further information see TFRT-7352.

4. The file is ready for insertion by PSFIG in any TEX document.

8. Control-C

Control-C may produce a meta coordinate file (a `pen`-file) for hardcopies. This file is converted to a PostScript file by CC2PS, and is then ready for insertion in any TEX document.

Preparing the figure

1. Plot the desired variables.
2. Set up Control-C to store a `pen` file in `filename.pen` and then `replot` the figure.

```
[> hard = 'pen'            ! hardcopies stored in pen-files
[> redhard >filename.pen   ! redirect output to 'filename.pen'
[>                         ! default file name is 'ctrlc.pen'
[> replot                  ! create the pen file
[> redhard -close          ! close the pen file
[> redhard >ctrlc          ! redirect again to protect
[>                         !  'filename.pen'
```

If the file type (`.pen`) isn't specified in the `redhard` command, the plot is stored in `filename.dat`.

Creating the PostScript file

3. Process `filename.pen` by CC2PS.

```
$ cc2ps/noprint/... filename
```

The option `/noprint` is necessary if the PostScript file is to be saved on disk. Default input file type is `.pen`. The output file is `filename.plo`. For further information see TFRT-7352.

4. The file is ready for insertion by PSFIG in any TEX document.

9. GNUPLOT

GNUPLOT is an interactive program for plotting mathematical functions or data files. You can get a PostScript file produced directly by GNUPLOT, but it is better to store a meta file in 'Simnon format' and process the meta file by HCOPY2PS instead. This is the procedure described below.

Preparing the figure

1. Plot the functions you desire.
2. The the output must be redirected to a file and replotted in the following way.

```
gnuplot> set terminal hcopymeta
gnuplot> set output 'filename.p'
gnuplot> replot
gnuplot> set output 'junk.p'
```

The `replot` command can be substituted by a real plot command, e.g. `plot sin(x)/x`, but the result is directly stored in the file. You may select any file type, but `.p` is the default for HCOPY2PS. The last `set output` command is for protection of `filename.p`. Otherwise you get more than one figure in the same file.

Creating the PostScript file

3. Process `filename.p` by HCOPY2PS.

```
$ hcopy2ps/noprint/... filename
```

The option `/noprint` is necessary if the PostScript file is to be saved on disk. The output file is `filename.plo`. For further information see TFRT-7352.

4. The file is ready for insertion by PSFIG in any TEX document.

10. PRO-MATLAB

PRO-MATLAB outputs a meta coordinate file on request. To get a PostScript file including a BoundingBox comment, you should not use the postprocessor GPP, described in the PRO-MATLAB manual, directly. RGPP is a special department version of GPP, which calls GPP for creation of a PostScript file and then includes a BoundingBox.

Preparing the figure

1. Using the `meta` command, the plot is sent both to your terminal and the desired meta file. This may look like:

```
>> meta filename
>> plot(....)
>> meta
```

The output file is `filename.met`. The last `meta` command redirects the output in order to protect `filename.met`.

Creating the PostScript file

2. Call RGPP to get a PostScript file containing a BoundingBox.

```
$ rgpp filename
```

RGPP outputs `filename.ps`. Note that the file type `.met` should not be specified when calling RGPP. `filename.ps` is ready for insertion by PSFIG in any TEX document.

11. Special cases

This section contains special information for 'wizards' on how to possibly overcome problems. There is no guarantee for success.

If the BoundingBox is missing or the the figure doesn't print or some other mysterious error occurs, then there are two ways of possibly overcoming this.

`pscoord.pos`

`pscoord.pos` is a PostScript file which generates a coordinate system on the current page. The file is to be included in another PostScript file, which then is to be printed by `lwprint`. The result is (hopefully) at least a coordinate system printed on the page. The coordinates may help you diagnosing, and if the figure is printed on the page then you may check the BoundingBox limits. `pscoord.pos` is located in `ps$inputs`.

BBFIG

BBFIG appends the PostScript file `bb.ps` and the specified file and prints the new file. The figure and tight BoundingBox coordinates will be printed. You must still insert the BoundingBox manually. This may look like:

```
$ bbfig filename.ps
```

Note that the file type must be given. The output file is `bbfig.ps`, which must not be deleted before it is printed by `lwprint`. Also note that this doesn't always work, e.g. on PRO-MATLAB figures the BoundingBox coordinates will be invisible.

The BoundingBox comment

The BoundingBox comment must look exactly like this.

```
%%BoundingBox: llx lly urx ury
```

`llx`, `lly`, etc are the PostScript coordinates of the lower left and the upper right corners respectively of the BoundingBox, separated by spaces.

12. Using PSFIG in TEX

This section gives a few examples on how to insert figures and applies to all PostScript files. PSFIG is further documented in next Section.

1. Insert figures in your TEX file in the following way.

```
!style<report>
!input psfig                          %Initializes PSFIG
...
!topinsert
!centerline<!psfig<file=fig1.ps,height=50mm,clip=>>
!caption<This figure is inserted on top of a page.>
!endinsert
...
!midinsert
!centerline<!hbox<
  !psfig<figure=portrait2.ps,width=60mm,height=60mm>
  !psfig<figure=landscape2.ps,width=60mm,height=60mm>>>
!caption<Figure 1 was inserted this way.>
!endinsert
...
```

About the PSFIG options: If `width` or `height` is specified the figure will be uniformly scaled to give the desired size. If both are specified the figure will be non-uniformly scaled to the desired size. Clipping, `clip=`, hides all parts of the figure outside the BoundingBox. `file=` and `figure=` are equivalent. Note that `psfig` calls should not be split into several lines.

2. Run TEX and DVILW, i.e.

```
$ tex file
$ lw file
```

If you define the logical name `temp`, then DVILW may put the PostScript file e.g. on a disk without disk quota. The file is then printed and deleted. An example:

```
$ define temp scr:[username]
$ lw/temp file
```

Then DVILW outputs `scr:[username]file.ps`, which is printed and deleted.

13. PSFIG — A Short Documentation

This section gives a brief summary on the use of PSFIG. The full documentation can be found in the PSFIG manual in TFRT-7352.

Commands

`!input psfig`	Initialization of PSFIG. This command must precede the first figure, where PSFIG is called.
`!psdraft`	Draft mode. Correct space is reserved, but figures are not printed. Instead the file specification is printed in the reserved space. Draft mode is on until the command `!psfull` is given.
`!psfull`	Full mode. PostScript files are included. Full mode is on until the command `!psdraft` is given. Full mode is the default mode.

Draft mode is useful when including large PostScript files. Then full mode is used only when the final version is printed.

Figure input

A PSFIG call looks like this.

```
!psfig<file=filename,option1=value1,...>
```

A PSFIG call should neither be split into several lines nor contain spaces. This piece of advice is found in the PSFIG manual, but may be unnecessary.

Figure placement

Figures are usually put in a figure insertion, e.g in a `!topinsert`.

`!centerline<!psfig<...>>`	This figure is centered.
`!rightline<!psfig<...>>`	This figure is aligned with the right margin.
`!psfig<...>`	This figure is aligned with the left margin.

However, a figure may be included at any point in the text. An example is the ⌘ character, which is inserted using PSFIG. Since this character is inserted several times a TEX macro is used. The previous insertion looks like this.

```
!def!CTRL<!psfig<figure=ctrl.ps,height=3mm>>
An example is the !CTRL! character,
```

PSFIG options

The options in a PSFIG call have the form `option=value`. The standard options (including file specifications) are

`file=` The value is the name of the PostScript file to be included.

`figure=` Same as `file=`.

`height=` Desired figure height, e.g. `height=50mm`.

`width=` Desired figure width.

`clip=` Takes no value. Hides all parts of the figure outside the BoundingBox.

If `width` or `height` is specified the figure will be uniformly scaled to give the desired size. If both are specified the figure will be non-uniformly scaled to the desired size. If none of them are specified the figure will be inserted in natural size.

Advanced options

There are a number of options which may provide a number of special effects. For reasons of completeness, they are listed below. Further documentation may be found in the PSFIG manual in TFRT-7352.

`rheight=` Specifies the reserved height of the figure. Default value is same as `height`. `rheight` is the size of the `!vskip` when reserving space for the figure. A unit must be specified.

`rwidth=` Specifies the reserved width of the figure. Default value is same as `width`.

`prolog=` A PostScript prolog file can be prepended to the figure file.

`postlog=` A PostScript postlog file can be appended to the figure file.

`bbllx=` The lower left x coordinate of the BoundingBox can be specified in the PSFIG call, using PostScript coordinates. The unit must be specified, e.g. `bbllx=300pt`. At present this value is overridden by the value in the BoundingBox comment.

`bblly=` Analogously for the lower left y coordinate.

`bburx=` Analogously for the upper right x coordinate.

`bbury=` Analogously for the upper right y coordinate.

If the PostScript file doesn't contain any BoundingBox comment, the BoundingBox coordinates can be specified in the PSFIG call instead.

One way of handling a too big BoundingBox supplied by an application program is to use `!vskip` and `!hskip` (positive or negative) to compensate for deviations from the desired appearance. Do not change the BoundingBox comment. This is as ugly as patching in an executable image.

Examples

On next page a MacDraft figure is shown. Then 3 examples are given, where the figure is inserted in different ways. Finally a number of figures from different applications are inserted.

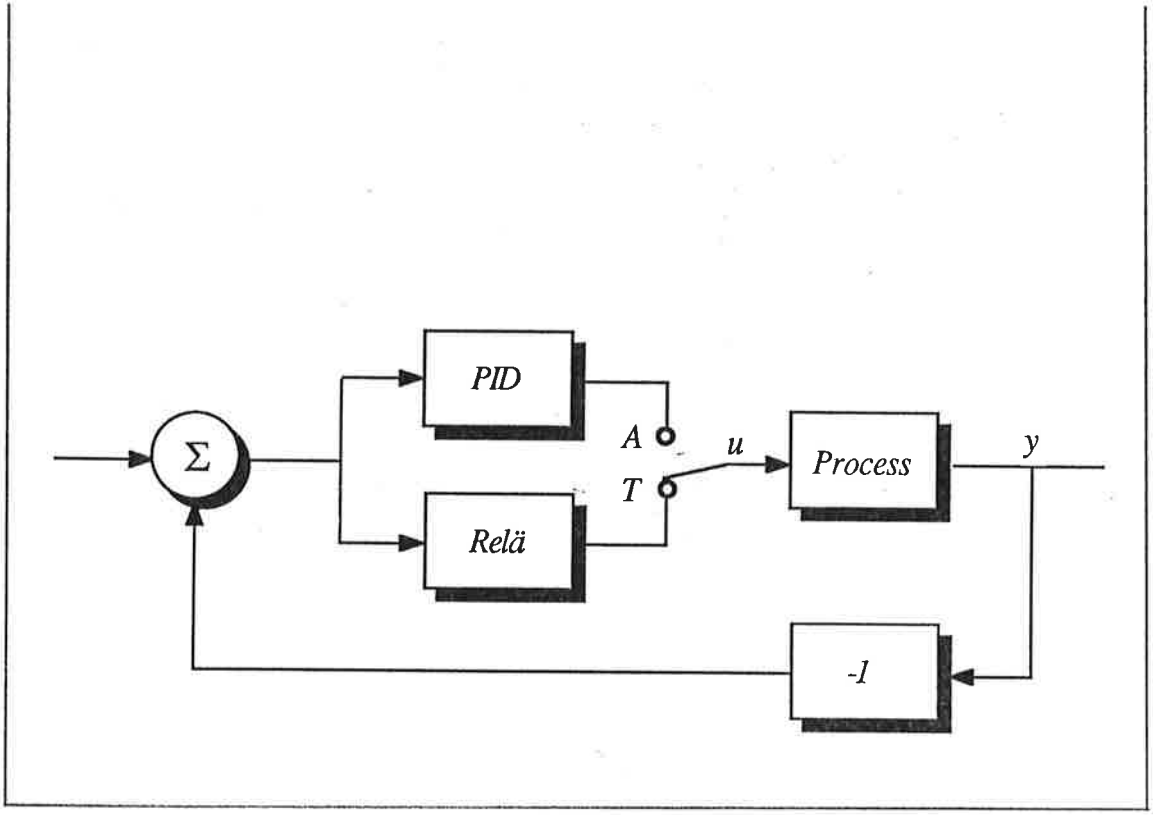


Fig. 4

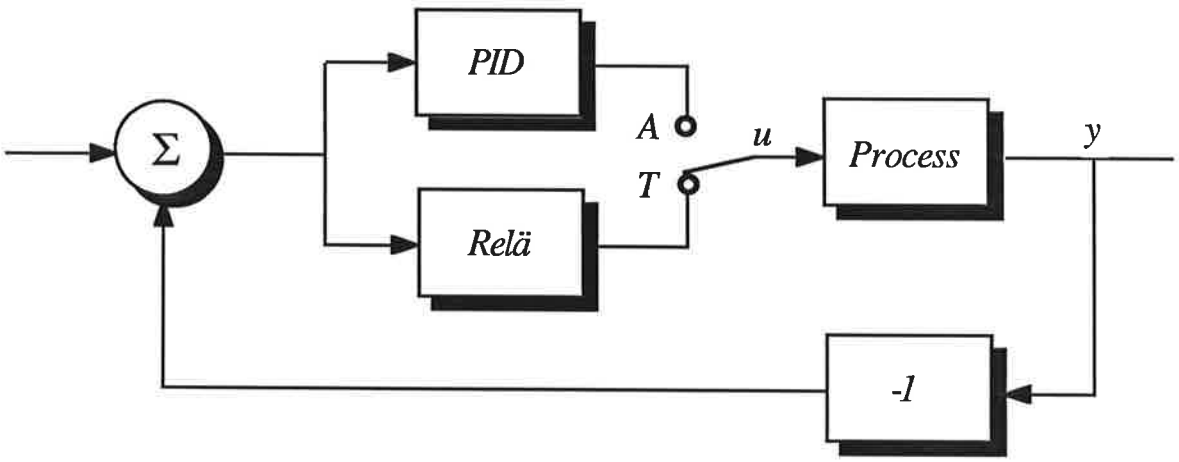
This text is glued on an original MacDraft print, which is printed in 90 % scale. The BoundingBox is roughly sketched by hand.

Example 1

The figure is inserted in the following way.

```
Text before the figure.
!midinsert
!centerline<!psfig<figure=portrait.ps>>
!endinsert
Text after the figure.
```

Text before the figure.



Text after the figure.

Fig. 4

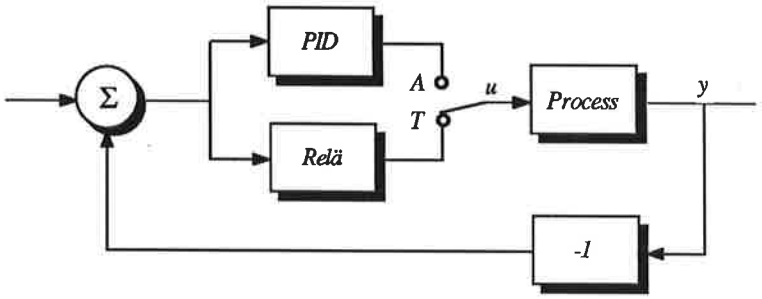
Note that the "caption" appears below the line "Text after the figure.". The reason is that the whole figure is printed, also the parts located outside the BoundingBox. Further, the figure is printed in it's natural size, 100 %, since neither `width` nor `height` is specified.

Example 2

The figure is now inserted in the following way.

```
Text before the figure.
!midinsert
!centerline<!psfig<figure=portrait.ps,width=100mm,clip=>>
!endinsert
Text after the figure.
```

Text before the figure.



Text after the figure.

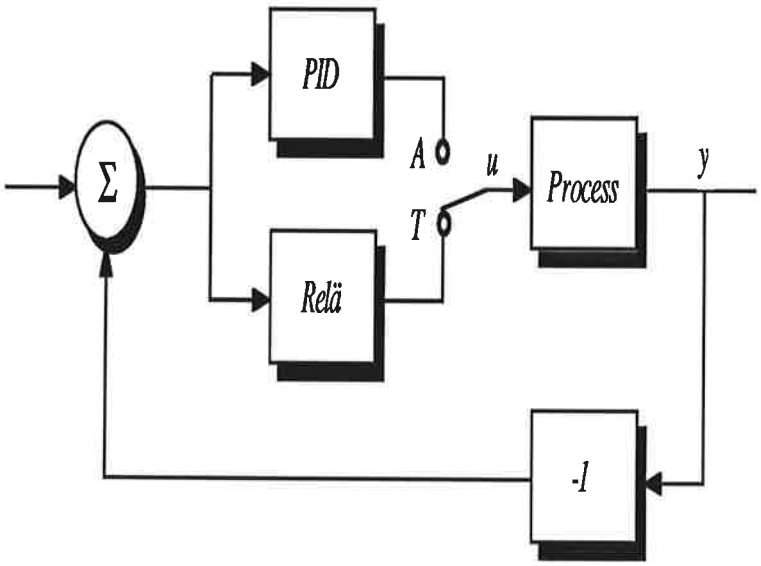
Now, using the `clip=` option, the "caption" has disappeared since it is outside the BoundingBox. Further, the figure is uniformly scaled and the resulting width is 100 mm.

Example 3

The figure is now inserted in the following way.

```
Text before the figure.
!midinsert
!centerline<
  !psfig<figure=portrait.ps,width=100mm,height=150mm,clip=>>
!endinsert
Text after the figure.
```

Text before the figure.

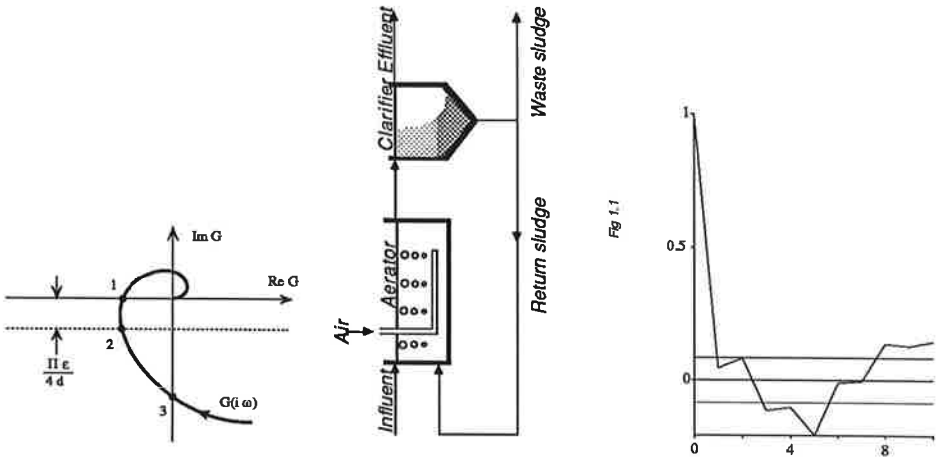


Text after the figure.

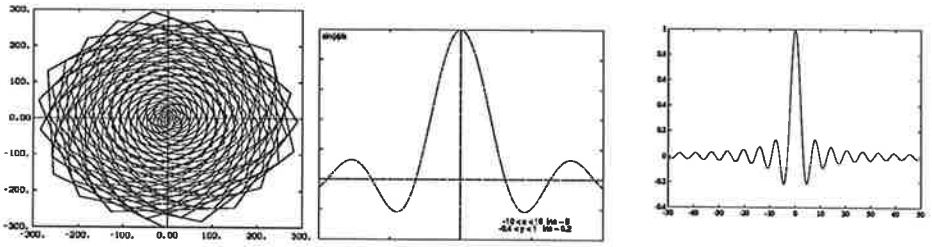
Since both `height` and `width` are specified, the figure is non-uniformly scaled to the desired size.

Example 4

A number of figures from different applications are inserted.



Figures from Adobe Illustrator, MacDraw and Idpac.



Figures from Control-C, GNUPLOT and PRO-MATLAB.

```
!midinsert
!centerline<!hbox<
   !psfig<figure=t2.ps,width=40mm>
   !psfig<figure=fig11r.ps,width=40mm>
   !psfig<figure=autocorr.plo,width=40mm>>>
!endinsert
Figures from Adobe Illustrator, MacDraw and Idpac.
!midinsert
!centerline<!hbox<
   !psfig<file=tcos_tsin.plo,width=40mm>
   !psfig<figure=gnuplot.plo,width=40mm>
   !psfig<figure=mlbbtest.ps,width=40mm>>>
!endinsert
Figures from Control-C, GNUPLOT and PRO-MATLAB.
```