Using EPS Graphics in $\LaTeX 2_\varepsilon$ Documents

Keith Reckdahl
reckdahl@am-sun2.stanford.edu

Version 1.9
February 20, 1997

Summary

This document explains how to use Encapsulated PostScript (EPS) files in $\LaTeX 2_\varepsilon$ documents. The length of this document should not make one think that it is difficult to including EPS graphics in LaTeX documents. On the contrary, with the $\LaTeX 2_\varepsilon$ graphics bundle, inserting EPS graphics has never been easier. EPS files can be inserted by specifying

```
\usepackage{graphicx}
```

in the document's preamble and then using the command

```
\includegraphics{file.eps}
```

Optionally, the height and width can be specified

```
\includegraphics[height=4cm]{file.eps}
\includegraphics[width=3in]{file.eps}
```

Additionally, the `\includegraphics` can rotate the included graphic

```
\includegraphics[angle=45]{file.eps}
```

While these `\includegraphics` are sufficient for most uses, this document is divided into four parts, which cover other more-complex aspects of using graphics in LaTeX.

Background Information
: This part provides historical information and describes some basic LaTeX terminology. It also describes the EPS BoundingBox and differences between EPS and PS files.

Graphics Inclusion Commands
: This part describes the commands in the graphics bundle which insert, scale, and rotate graphics. Although there is much overlap, this part is not as complete as much the excellent graphics bundle documentation [4].

Importing EPS Graphics
: This part describes graphics rotation and alignment. It also covers three situations where graphics importation is modified

- Compressed EPS files and non-EPS graphic formats (TIFF, GIF, JPEG, PICT, etc.) can also be inserted when `dvips` is used with an operating system which supports pipes (such as Unix). When the compressed or non-EPS graphic files are not in the current directory, the `kpsewhich` command can

---

be used to make `dvips` look for the file in the directories on the TEX search path.
It is important to note that since neither LATEX nor `dvips` has any built-in decompression or graphics-conversion capabilities, that software must be provided by the user.

- Since many graphics applications support only ASCII text, the PSfrag system allows text in EPS files to be replaced with LATEX symbols or mathematical expressions.
- When an EPS graphic is inserted multiple times, the final PostScript includes multiple copies of the graphics, making the file large. When the graphics are not bitmapped, a smaller final PostScript file can be obtained by defining a PostScript command for the graphics. An example of inserting an EPS graphic in the page header with the fancyhdr package is provided.

Related LATEX Commands

This part describes various commands which are often used in conjunction with EPS graphics. This includes the following

- the insertion of graphics in `figure` environments allows the graphics to float for better formatting and allows graphics to be referenced,
- the lscape and rotating packages provide methods for producing figures with landscape orientation in a portrait document,
- methods for arranging side-by-side graphics. The graphics can be placed in a single figure, in multiple figures contained in a single float, or in subfigures.
- boxed figures can be created with `\fbox` or the commands from the fancybox package.
- the figure captions can be placed next to the figure or table, instead of the conventional above or below placement,
- the caption2 package adds flexibility to the caption formatting, allowing users to modify the style, width, and font of captions.

Acknowledgements

I would like to thank the contributors to the `comp.text.tex` newsgroup, whose posts and replies provided me with the information for this document. In particular, David Carlisle has provided a great deal of assistance. I would also like to acknowledge Jim Hafner for providing the procedure in section 9.1. Finally, I thank the readers of previous versions of this document who provided me with feedback to improve this document.

Contents

Part I
Background Information

1 Introduction

Inserting Encapsulated PostScript (EPS) graphics in LaTeX originally required the low-level `\special` command. To make graphic-insertion easier and more portable, two higher-level packages epsf and psfig were written for LaTeX2.09. In epsf, the graphics insertion was done by the `\epsfbox` command, while three other commands controlled graphic scaling. In psfig, the `\psfig` command not only inserted graphics, it also scaled and rotated them. While the `\psfig` syntax was popular, its code was not as robust as `\epsfbox`. As a result, the epsfig package was created as a hybrid of the two graphics packages, with its `\epsfig` command using the `\psfig` syntax and much of the more-robust `\epsfbox` code. Unfortunately, `\epsfig` still used some of the less-robust `\psfig` code.

With the release of LaTeX 2_ε, the LaTeX3 team addressed the general problem of inserting graphics in LaTeX 2_ε. Their efforts produced the "graphics bundle," which contains totally re-written commands which are more efficient, more robust, and more portable than other graphics-insertion commands.

The graphics bundle contains the "standard" graphics package and the "extended" graphicx package. While both packages contain an `\includegraphics` command, the packages contain *different* versions of `\includegraphics`. The graphicx version uses "named arguments" (similar to the `\psfig` syntax) which, although convenient, violate the LaTeX syntax guidelines which require that optional arguments be positional. As a compromise, two versions of `\includegraphics` were written, with the graphics package following the LaTeX syntax guidelines and the graphicx package using the more convenient named arguments. The graphicx `\includegraphics` supports scaling and rotating, but the graphics `\includegraphics` command must be nested inside `\rotatebox` or `\scalebox` commands to produce scaling or rotating.

This document uses the graphicx package because its syntax is more convenient than the graphics syntax. Since both packages have the same capabilities, the examples in this document can also be performed with the graphics package, although the resulting syntax may be more cumbersome and slightly less efficient. For a more-detailed description of the packages, see the graphics bundle documentation [4].

For backward-compatibility, the graphics bundle also includes the epsfig package which replaces the original LaTeX 2_ε epsfig package. The new epsfig package defines the `\epsfbox`, `\psfig`, and `\epsfig` commands as wrappers which simply call the `\includegraphics` command. Since these wrappers are less efficient than `\includegraphics`, these wrapped packaged should be used only for old documents, with `\includegraphics` used for all new documents.

2 LaTeX Terminology

A *box* is any LaTeX object (characters, graphics, etc.) that is treated as a unit (see [1, page 103]). Each box has a *reference point* on its left side. The box's *baseline* is a horizontal line which passes through the reference point (see Figure 1). When LaTeX forms lines of text, characters are placed left-to-right with their reference points aligned on a horizontal line called the *current baseline*, aligning the characters' baselines with the current baseline. LaTeX follows the same process for typesetting graphics or other objects; the reference point of each object is placed on the current baseline.

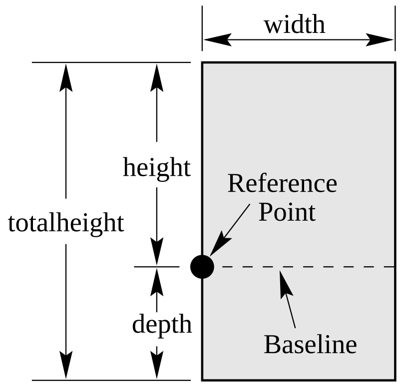


Figure 1: Sample LaTeX Box

The size of each box is described by three lengths: *height*, *depth*, *width*. The *height* is the distance from the reference point to the top of the box. The *depth* is the distance from the reference point to the bottom of the box. The *width* is the width of the box. The *totalheight* is defined as the distance from the bottom of the box to the top of the box, or totalheight=height+depth.

The reference point of a non-rotated EPS graphic is its lower-left corner (see left box in Figure 2), giving it zero depth and making its totalheight equal its height. The middle box in Figure 2 shows a rotated graphic where the height is not equal to the totalheight. The right box in Figure 2 shows a rotated graphic where the height is zero.

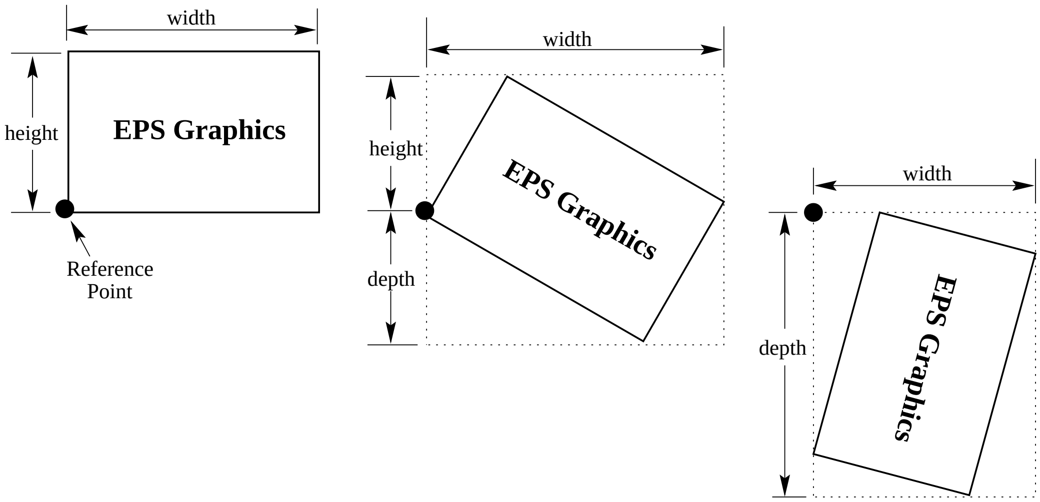


Figure 2: Rotated LaTeX Boxes

3 The EPS BoundingBox

In addition to PostScript graphics language commands which draw the graphics, EPS files contain a BoundingBox line which specifies the natural size of the graphics. By convention, the first line of a PostScript file specifies the type of PostScript and is then followed by a series of comments called the *header* or *preamble*. (Like LaTeX, PostScript's comment character is `%`). One of these comments specifies the BoundingBox. The BoundingBox line contains four integers

1. The x-coordinate of the lower-left corner of the BoundingBox.
2. The y-coordinate of the lower-left corner of the BoundingBox.
3. The x-coordinate of the upper-right corner of the BoundingBox.
4. The y-coordinate of the upper-right corner of the BoundingBox.

For example, the first 5 lines of an EPS file created by gnuplot are

```
%!PS-Adobe-2.0 EPSF-2.0
%%Creator: gnuplot
%%DocumentFonts: Times-Roman
%%BoundingBox: 50 50 410 302
%%EndComments
```

Thus the gnuplot EPS graphic has a lower-left corner with coordinates $(50, 50)$ and an upper-right corner with coordinates $(410, 302)$. The BoundingBox parameters have units of PostScript points which are $1/72$ of an inch, making the above graphic's natural width 5 inches and its natural height 3.5 inches. Note that a PostScript point is is slightly larger than a TeX point which is $1/72.27$ of an inch. In TeX and LaTeX, PostScript points are called "big points" and abbreviated `bp` while TeX points are called "points" and abbreviated `pt`.

3.1 Converting PS files to EPS

While most PostScript files (without BoundingBox information) can be converted to EPS, there are restrictions on the PostScript commands which can be used in EPS files. For example, EPS files cannot include the following PostScript operators

`a3`	`a4`	`a5`	`banddevice`	`clear`
`cleardictstack`	`copypage`	`erasepage`	`exitserver`	`framedevice`
`grestoreall`	`initclip`	`initgraphics`	`initmatrix`	`letter`
`legal`	`note`	`prenderbands`	`quit`	`renderbands`
`setdevice`	`setglobal`	`setpagedevice`	`setpageparams`	`setsccbatch`
`setshared`	`startjob`	`stop`		

Additionally, if not used properly, the following PostScript operators may cause problems in EPS files

`nulldevice`	`setcolortransfer`	`setgstate`	`sethalftone`	`setmatrix`
`setscreen`	`settransfer`			

Single-page PostScript files without any such offending commands can be converted to EPS by one of the following methods

1. The best option is to use a utility such as ghostscript's `ps2epsi` which reads the PostScript file, calculates the BoundingBox parameters, and creates an EPS file (complete with a BoundingBox) which contains the PostScript graphics. Unfortunately, ghostscript is a large package which is not trivial to install.

2. Alternatively, the BoundingBox parameters can be calculated and then either entered in the `bb` option of `\includegraphics` or a text editor can be used to insert them directly in the PostScript file's BoundingBox line. There are several ways to calculate the BoundingBox

   (a) The `bbfig` script uses a PostScript printer to calculate the BoundingBox. `bbfig` adds some PostScript commands to the beginning of the PostScript file and sends it to the printer. At the printer, the added PostScript commands calculate the BoundingBox of the original PostScript file, printing the BoundingBox coordinates superimposed on the PostScript graphic.

   (b) Use ghostview to display the PostScript graphic. As you move the ghostview pointer around the graphic, ghostview displays the pointer's coordinates (with respect the the lower-left corner of the page). To determine the BoundingBox parameters, record the pointer coordinates at the lower-left corner of the graphic and the upper-right corner of the graphic.

   (c) Print out a copy of the PostScript graphics and measure the horizontal and vertical distances (in inches) from the lower-left corner of the paper to the lower-left corner of the graphics. Multiply these measurements by 72 to get the BoundingBox's lower-left coordinates. Likewise, measure the distances from the lower-left corner of the paper to the upper-right corner of the graphics to get the BoundingBox's upper-right coordinates.

3.2 Non-standard EPS files

Some applications produce non-standard EPS files which cannot be used in other programs such as LaTeX. Sometimes the applications have developed their own flavor of PostScript with additional features, while other times this is due to poor PostScript programming.

Mathematica EPS files produced by Mathematica are written in Mathematica's extended PostScript. To use the EPS file in non-Mathematica programs, the file must be fixed to remove the non-standard extensions. DOS versions of Mathematica include a utility called either `printps.exe` or `rasterps` which removes the non-standard extensions. On Unix versions of Mathematica, this can be done with the `psfix` utility. Reference your Mathematica documentation or contact Wolfram Research for more information.

FrameMaker FrameMaker produces PostScript which fails to follow Adobe's specification for page independence. You can fix the PostScript files using the scripts

```
ftp://ftp.irisa.fr/pub/FrameMaker/Filters/fixfm4-1.3.tar.gz
ftp://ftp.irisa.fr/pub/FrameMaker/Filters/fixfm5-2.0.tar.gz
```

Use the one which corresponds to your version of FrameMaker. Another source for correction scripts is

```
ftp://ftp.frame.com/pub/techsup/framers/platform.ind/filters/fixfm3ps.sh
ftp://ftp.frame.com/pub/techsup/framers/platform.ind/filters/fixfm4ps.sh
```

4 Graphics in DVI Files

When LaTeX documents are compiled, the `\includegraphics` commands do not insert the EPS graphics file into the DVI file. Rather, they do two things

1. They reserve the proper amount of space for the graphic in the LaTeX document.
2. They place a file-specification command in the DVI file which specifies the name of the EPS file.

When a DVI-to-PS converter (such as `dvips`) converts the DVI file to PostScript, the file-specification command causes the converter to read the EPS graphic file and insert it into the PostScript file. Therefore,

- the EPS graphics do not appear in many DVI-viewers. To help the user with placement of the graphics, most DVI viewers display the BoundingBox in which the graphics will be inserted.
- the EPS files must be present when the DVI file is converted to PS. Thus the EPS files must accompany DVI files whenever they are moved.

Part II
Graphics Inclusion Commands

5 The Commands in the graphicx Package

The best reference for the graphics and graphicx packages is the graphics guide [4]. The coverage of the graphicx package in the standard LaTeX references is sporadic: [3] covers both the graphics and graphicx packages, [1] only covers the graphics package and [2] describes neither.

The graphicx package has five main commands

```
\includegraphics[options]{filename}
\rotatebox[options]{angle}{argument}
\scalebox{h-scale}[v-scale]{argument}
\resizebox{width}{height}{argument}
\resizebox*{width}{totalheight}{argument}
```

5.1 The includegraphics Command

Syntax: `\includegraphics[options]{filename}`

where the options are listed in Tables 1, 2, and 3. Since `\includegraphics` does not end the current paragraph, it can place graphics within text such as or . The commands

```
\documentclass{article}
\usepackage{graphicx}
\begin{document}
   \includegraphics{file.eps}
\end{document}
```

include the graphics from `file.eps` at its natural size.

Specifying Width

The command

```
\includegraphics[width=3in]{file.eps}
```

includes the graphics from `file.eps` scaled such that its width is 3 inches. The command

```
\includegraphics[width=\textwidth]{box.eps}
```

scales the included graphic such that it is as wide as the text. The command

```
\includegraphics[width=0.80\textwidth]{box.eps}
```

makes the included graphic 80% as wide as the text. When the calc package is used, the following command make the graphic width 2 inches less than the width of text

```
\includegraphics[width=\textwidth-2.0in]{box.eps}
```

A graphicx version earlier than *12/95* requires the following commands

```
\newlength{\epswidth}
\setlength{\epswidth}{\textwidth -2.0in}
\includegraphics[width=\epswidth]{box.eps}
```

Table 1: `includegraphics` Options

`height`	The height of the graphics (in any of the accepted TEX units).
`totalheight`	The totalheight of the graphics, in any of the accepted TEX units. *(Added 6/95)*
`width`	The width of the graphics, in any of the accepted TEX units.
`scale`	Scale factor for the graphic. Specifying `scale=2` makes the graphic twice as large as its natural size.
`angle`	Specifies the angle of rotation, in degrees, with a counter-clockwise (anti-clockwise) rotation being positive.
`origin`	The `origin` command specifies what point to use for the rotation origin. By default, the object is rotated about its reference point. *(Added 12/95)* The possible origin points are the same as those for the `\rotatebox` command in section 5.4. For example, `origin=c` rotates the graphic about its center.
`bb`	Specifies BoundingBox parameters. For example `bb=10 20 100 200` specifies that the BoundingBox has its lower-left corner at (10,20) and its upper-right corner at (100,200). Since `\includegraphics` automatically reads the BoundingBox parameters from the EPS file, the `bb` option is usually not specified. It is useful if the BoundingBox parameters in the EPS file are missing or are incorrect. The BoundingBox can be specified with the `natheight` and `natwidth` options instead of the `bb` options. `natheight=h, natwidth=w` is equivalent to `bb=0 0 h w`. For backward compatibility, the BoundingBox coordinates can also be individually specified with `bbllx`, `bblly`, `bburx`, `bbury` options.

The `\newlength` command only needs to be issued once. Subsequent graphics can be scaled without re-issuing the `\newlength` command. The length name `\epswidth` is not special. Any other name (which isn't already used by LATEX) could have been used.

Specifying Height

Users must be careful when using the `height` option, as they often mean the overall height which is set by the `totalheight` option (see Figure 1). When the object has zero depth, the `totalheight` is the same as the `height` and specifying `height` works fine. When the object has a non-zero depth, specifying `height` instead of `totalheight` causes either an incorrectly-sized graphic or a divide-by-zero error.

Graphic Justification

The placement of the graphic is controlled by the current text justification. To center the graphic, put it inside a center environment

```
\begin{center}
   \includegraphics[width=2in]{box.eps}
\end{center}
```

Table 2: `includegraphics` Cropping Options

`viewport`	Specify what portion of the graphic to view. Like a BoundingBox, the area is specified by four numbers which are the coordinates of the lower-left corner and upper-right corner. The coordinates are relative to lower-left corner of the BoundingBox. *(Added 6/95)* For example, `viewport=0 0 72 72` displays the 1-inch square from the lower left of the graphic. Note that some early `graphicx` versions may have a broken `viewport` option in which `viewport=a b c d` produces an upper-right corner of (a+c,b+d) instead of (c,d).
`trim`	An alternate method for specifying what portion of the graphic to view. The four numbers specify the amount to remove from the left, bottom, right, and top side, respectively. Positive numbers trim from a side, negative numbers add to a side. *(Added 6/95)* For example, `trim=1 2 3 4` trims the graphic by 1 `bp` on the left, 2 `bp` on the bottom, 3 `bp` on the right, 4 `bp` on the top.

If the `\includegraphics` command is inside an environment (such as `minipage` or `figure`), the `\centering` declaration centers the remaining output of the environment. For example

```
\begin{figure}
   \centering
   \includegraphics[width=2in]{box.eps}
\end{figure}
```

is similar to

```
\begin{figure}
\begin{center}
   \includegraphics[width=2in]{box.eps}
\end{center}
\end{figure}
```

The difference between these examples is that the `center` environment produces extra vertical space above and below the environment, while `\centering` produces no extra space.

5.2 The scalebox Command

Syntax: `\scalebox{h-scale}[v-scale]{argument}`

The `\scalebox` command scales an object, making its width be its original width multiplied by `h-scale`. The object can be text, EPS graphic, or any other LaTeX object. The object's height is its original height multiplied by `v-scale`. Negative values reflect the object. If `v-scale` is omitted, it defaults to h-scale, which keeps the aspect ratio constant.

5.3 The resizebox Commands

Syntax: `\resizebox{width}{height}{argument}`
`\resizebox*{width}{totalheight}{argument}`

Table 3: `includegraphics` Boolean Options

`clip`	When `clip` is specified, any graphics outside of the viewing area are clipped and do not appear.
`keepaspectratio`	When `keepaspectratio` is not specified, specifying both the `width` and either `height` or `totalheight` causes the graphic to be scaled anamorphically to fit both the specified height and width. When `keepaspectratio` is specified, specifying both the `width` and either `height` or `totalheight` makes the graphic as large as possible such that its aspect ratio remains the same and the graphic does exceeds neither the specified height nor width. *(Added 9/95)*
`draft`	When `draft` is specified, the graphic's BoundingBox and filename are displayed in place of the graphic, making it faster to display and print the document. The `draft` package option `\usepackage[draft]{graphicx}` causes all the graphics in a document to be inserted in draft mode.

The `\resizebox` command resizes an object to a specified size. The object can be any LaTeX object: letter, paragraph, EPS graphic, etc. Specifying `!` as either height or width makes that length be such that the aspect ratio remains constant. The standard LaTeX 2_ε arguments `\height`, `\width`, `\totalheight`, `\depth` can be used to refer to the original size of `argument`. So `\resizebox{2in}{\height}{argument}` makes `argument` keep its same height but have a width of 2 inches.

The `\resizebox*` command is identical to `\resizebox`, except the second argument specifies the `totalheight` of the object.

5.4 The rotatebox Command

Syntax: `\rotatebox[options]{angle}{argument}`

The `\rotatebox` command rotates an object by an angle given in degrees, with a counterclockwise rotation being positive. The object can be any LaTeX object: letter, paragraph, EPS graphic, etc. By default, the object is rotated about its reference point. The options allow the user to specify the point of rotation

1. Specifying the `[x=xdim,y=ydim]`, the object is rotated about the point whose coordinates relative to the reference point are (`xdim,ydim`).
2. The `origin` option specifies one of 12 special points shown in in Figure 3.

   The horizontal position of the `origin` points is specified by one of three letters: `lcr` (which stand for left, center, right, respectively), while the vertical position is specified by one of four letters: `t,c,B,b` (which stand for top, center, Baseline, bottom, respectively). For example

   `[rb]` specifies the bottom-right corner

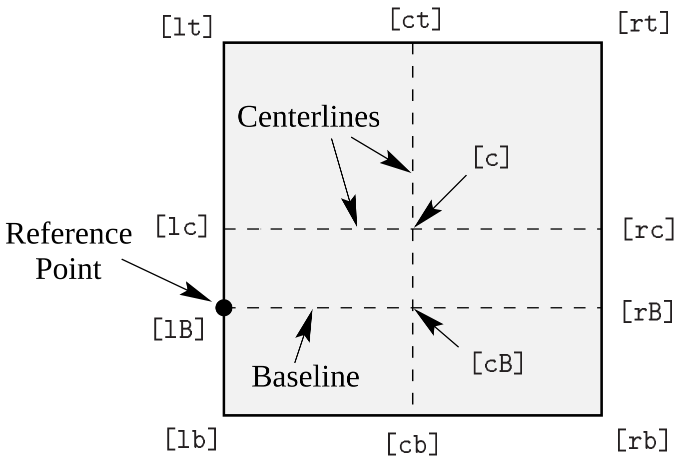


Figure 3: Available Origin Points

[lt] specifies the top-left corner

[cB] specifies the center of the graphic's Baseline

[lc] specifies the midpoint of the left side

[ct] specifies the midpoint of the top side

Note that

- The order of the letters is not important, making [br] equivalent to [rb].
- c represents either the horizontal center or vertical center depending what letter is used with it.
- If only one letter is specified, the other is assumed to be c, making [c] equivalent to [cc], [l] equivalent to [lc], [t] equivalent to [tc], etc.

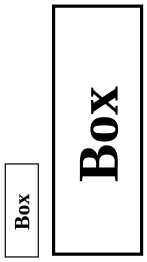
Part III
Importing EPS Graphics

6 Rotation, Scaling, and Alignment

Since the `\includegraphics` options are interpreted from left to right, the order in which the angle and size are specified makes a difference. For example

```
\begin{center}
  \includegraphics[angle=90,totalheight=0.5in]{box.eps}
  \includegraphics[totalheight=0.5in,angle=90]{box.eps}
\end{center}
```

produces



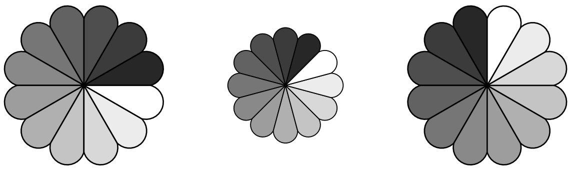
The first box is rotated 90 degrees and then scaled such that its height is a half inch. The second box is scaled such that its height is a half inch and then it is rotated 90 degrees.

6.1 Scaling of Rotated Graphics

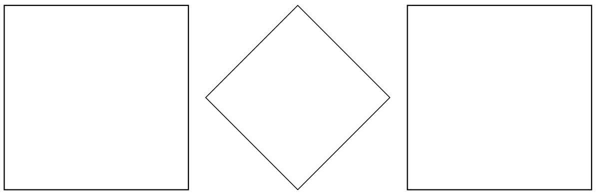
When the height or width of a graphic is specified, the specified size is not the size of the graphic but rather of its BoundingBox. This distinction is especially important in order to understand the scaling of rotated graphics. For example

```
\begin{center}
  \includegraphics[totalheight=1in]{rosette.eps}
  \includegraphics[angle=45,totalheight=1in]{rosette.eps}
  \includegraphics[angle=90,totalheight=1in]{rosette.eps}
\end{center}
```

produces



Although it may seem strange that the graphics have different sizes, it makes sense after viewing the BoundingBoxes



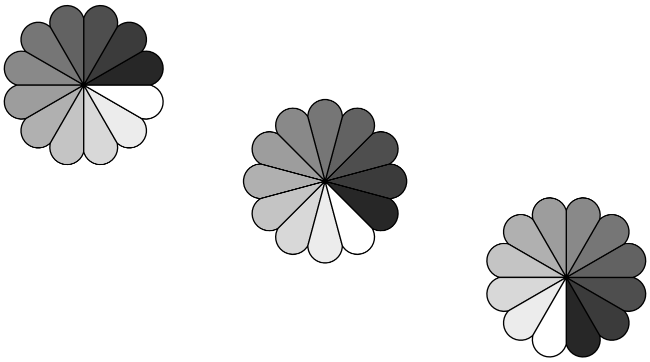
Each graphic is scaled such that its rotated BoundingBox is 1 inch tall.

6.2 Alignment of Rotated Graphics

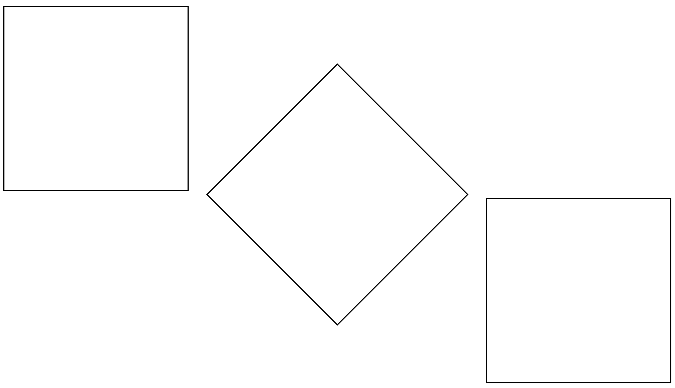
When graphics are rotated, the objects may not align properly. For example

```
\begin{center}
  \includegraphics[totalheight=1in]{rosette.eps}
  \includegraphics[totalheight=1in,angle=-45]{rosette.eps}
  \includegraphics[totalheight=1in,angle=-90]{rosette.eps}
\end{center}
```

produces



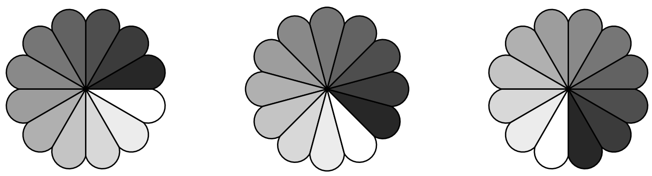
Again, this is better illustrated by the BoundingBoxes



In this case, the objects' reference points (original lower-left corners) are aligned on a horizontal line. If it is desired to instead have the centers aligned, the `origin` option of `\includegraphics` can be used

```
\begin{center}
  \includegraphics[totalheight=1in]{rosette.eps}
  \includegraphics[totalheight=1in,origin=c,angle=-45]{rosette.eps}
  \includegraphics[totalheight=1in,origin=c,angle=-90]{rosette.eps}
\end{center}
```

This aligns the centers of the graphics



If the version of graphicx is before *12/95*, the following commands must be used.

```
\begin{center}
   \includegraphics[totalheight=1in]{rosette.eps}
   \rotatebox[origin=c]{-45}{\includegraphics[totalheight=1in]{rosette.eps}}
   \rotatebox[origin=c]{-90}{\includegraphics[totalheight=1in]{rosette.eps}}
\end{center}
```

Similarly, the commands

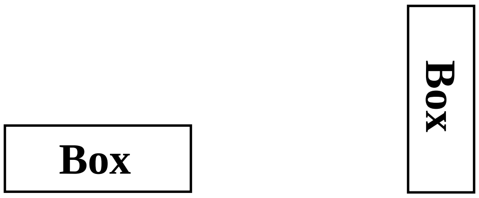
```
\begin{center}
    \includegraphics[width=1in]{box.eps}
    \hspace{1in}
    \includegraphics[width=1in,angle=-90]{box.eps}
\end{center}
```

rotate the right graphic around its lower-left corner, producing

To align the bottoms of the graphics, use the following commands

```
\begin{center}
   \includegraphics[width=1in]{box.eps}
   \hspace{1in}
   \includegraphics[width=1in,origin=br,angle=-90]{box.eps}
\end{center}
```

which rotate the right graphic about its lower-right corner, producing



6.3 Minipage Placement Option Details

Placing graphics inside minipage environments often makes it easier to properly space side-by-side graphics (Section 12 covers this in more detail). However, the manner in which `minipage` environments are vertically aligned may be confusing. For example, one might think the commands

```
\begin{center}
   \begin{minipage}[b]{.25\textwidth}
      \centering \includegraphics[width=1in]{box.eps}
   \end{minipage}
   \begin{minipage}[b]{.25\textwidth}
      \centering \includegraphics[width=1in,angle=-90]{box.eps}
   \end{minipage}
\end{center}
```

which use the minipage `[b]` options would align the bottoms of the graphics. Instead they produce Figure 4.

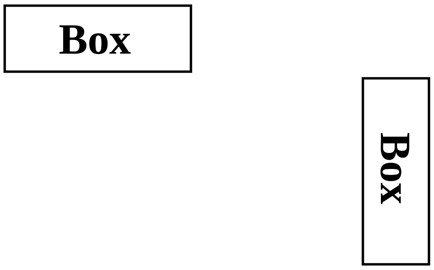


Figure 4: `minipage` with `[b]` option

Similarly, one might think the commands

```
\begin{center}
   \begin{minipage}[t]{.25\textwidth}
      \centering \includegraphics[width=1in]{box.eps}
   \end{minipage}
   \begin{minipage}[t]{.25\textwidth}
      \centering \includegraphics[width=1in,angle=-90]{box.eps}
   \end{minipage}
\end{center}
```

which use the minipage `[t]` options would align the tops of the graphics. Instead they produce a figure which is *exactly* the same as Figure 4.

The `[b]` and `[t]` options produce the same figure because the `minipage` environment's `[b]` option does *not* align the bottoms of the minipages. Rather, it aligns the baselines of the minipages' bottom lines. Similarly, the `[t]` option aligns the baselines of the minipages' top lines. Since the minipages in the above examples only have one line, the [t] and [b] use the same line for alignment. In this case, the reference point of the minipage is the reference point (original lower-left corner) of the EPS graphic.

6.3.1 Aligning the Bottoms of Minipages

One method for aligning the bottoms of minipages is to make the bottom of the minipage be the baseline of the minipage. If a line with zero height and zero depth is added inside the minipage after the graphics then the [b] option makes the bottom of the minipage be minipage's baseline. The command `\par\vspace{0pt}` creates such a zero-height, zero-depth line. Since the baseline of this zero-depth line is the bottom of the minipage, the [b] option now aligns the bottom of the minipage. For example

```
\begin{center}
   \begin{minipage}[b]{.25\textwidth}
      \centering \includegraphics[width=1in]{box.eps}
      \par\vspace{0pt}
   \end{minipage}
   \begin{minipage}[b]{.25\textwidth}
      \centering \includegraphics[width=1in,angle=-90]{box.eps}
      \par\vspace{0pt}
   \end{minipage}
\end{center}
```

produces Figure 5.

Figure 5: Minipages with Bottoms Aligned

6.3.2 Aligning the Tops of Minipages

To align the tops of the minipages, one must add a zero-height, zero-depth line to the top of the minipage. Then the [t] option makes the top of the minipage be the baseline of the minipage. Preceding `\includegraphics` command by `\vspace{0pt}` inserts a zero-height, zero-depth line above the graphic. Since the baseline of this zero-height line is the top of the minipage, the [t] option now aligns the top of the minipage. For example

```
\begin{center}
   \begin{minipage}[t]{.25\textwidth}
      \vspace{0pt}
      \centering \includegraphics[width=1in]{box.eps}
   \end{minipage}
   \begin{minipage}[t]{.25\textwidth}
      \vspace{0pt}
      \centering \includegraphics[width=1in,angle=-90]{box.eps}
   \end{minipage}
\end{center}
```

produces Figure 6.

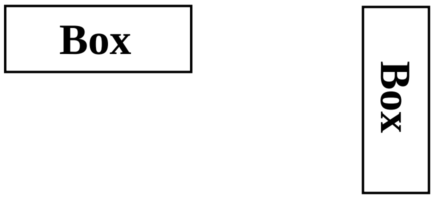


Figure 6: Minipages with Tops Aligned

This aligns the tops of the minipages with the current baseline. If it is instead desired to align the tops of the minipages with the top of the current line of text, replace `\vspace{0pt}` with `\vspace{-\baselineskip}`. This topic is mentioned in [2, pages 456-457].

7 Compressed and Non-EPS Graphics Files

When using `dvips`, users can specify an operation to be performed on the file before it is inserted. Making this operation a decompression command, allows compressed graphics files to be used. Making this operation a graphics-conversion command, allows non-EPS graphics files can be used. Since `dvips` is currently the only DVI-to-PS converter with this capability, everything in this section requires `dvips`. Users need to pass the `dvips` option to the graphicx package. This can be done by either specifying the `dvips` global option in the `\documentclass` command

```
\documentclass[dvips,11pt]{article}
```

or by specifying the `dvips` option in the `\usepackage` command

```
\usepackage[dvips]{graphicx}
```

Since specifying the `dvips` option in `\documentclass` passes it to all packages, it is generally preferred.

The `\DeclareGraphicsRule` and `\DeclareGraphicsExtensions` commands control how LaTeX deals with files specified in `\includegraphics`. In particular, `\DeclareGraphicsRule` specifies a command which operates on the file. The execution of this command requires that the operating system support pipes. For example, Unix supports pipes while DOS does not. For operating systems which do not support piping, the decompression or conversion cannot be done on-the-fly and the user must store all graphics as uncompressed EPS files.

7.1 Compressed EPS Example

The steps for using compressed EPS files are

1. Create an EPS file (`file1.eps` for example)
2. Store the BoundingBox line in another file (`file1.eps.bb`)
3. Compress the EPS file. For example, the Unix command

   ```
   gzip -9 file1.eps
   ```

   creates the compressed file `file1.eps.gz`. The `-9` (or `-best`) option specifies maximum compression.

4. Include the proper `\DeclareGraphicsRule` command before the `\includegraphics` command in the LaTeX file. The `\DeclareGraphicsRule` command informs LaTeX how to treat the particular suffix (see section 7.2). For example

```
\documentclass[dvips]{article}
\usepackage{graphicx}
\begin{document}
   \DeclareGraphicsRule{.eps.gz}{eps}{.eps.bb}{`gunzip -c #1}
   \begin{figure}
        \centering
        \includegraphics[width=3in]{file1.eps.gz}
        \caption{Compressed EPS Graphic}
        \label{fig:compressed:eps}
   \end{figure}
\end{document}
```

In this case, the `\DeclareGraphicsRule` command is actually not necessary because it happens to be one of the pre-defined graphics rules. If another compression program or suffixes were used, the `\DeclareGraphicsRule` command would be mandatory. For example, if the BoundingBox file had been stored in `file1.bb`, the corresponding `\DeclareGraphicsRule` would be

```
\DeclareGraphicsRule{.eps.gz}{eps}{.bb}{`gunzip -c #1}
```

If many compressed EPS files are used, the BoundingBox extraction and EPS file compression may be tedious. This process can be automated with a `perl` script named `epsbb` which was included with the old `epsfig` package and is still available from CTAN.

7.2 The DeclareGraphicsRule Command

The `\DeclareGraphicsRule` command specifies how `\includegraphics` treats files depending on their extensions. Multiple `\DeclareGraphicsRule` commands may be issued.

Syntax: `\DeclareGraphicsRule{ext}{type}{sizefile}{command}`

Table 4: `DeclareGraphicsRule` **Arguments**

`ext`	The file extension.
`type`	The graphics type for that extension.
`sizefile`	The extension of the file which contains the BoundingBox information for the graphics. If this option is blank `{}`, the size information must be specified by an `\includegraphics` option.
`command`	The command to be applied to the file (often left blank `{}`). The command must be preceded by a single backward quote (not to be confused with the more common forward single quote.)

For example, the command

```
\DeclareGraphicsRule{.eps.gz}{eps}{.eps.bb}{`gunzip -c #1}
```

specifies that any file with a `.eps.gz` extension is treated as compressed EPS file, with the the BoundingBox information stored in the file with a `.eps.bb` extension, and the `gunzip`

-c command uncompresses the file. (Since LaTeX cannot read BoundingBox information from a compressed file, the BoundingBox line must be stored in an uncompressed file.)

Users generally do not need to use the \DeclareGraphicsRule command because the following graphics rules are defined by default in dvips.def

```
\DeclareGraphicsRule{.eps}{eps}{.eps}{}
\DeclareGraphicsRule{.ps}{eps}{.ps}{}
\DeclareGraphicsRule{.pz}{eps}{.bb}{`gunzip -c #1}
\DeclareGraphicsRule{.eps.Z}{eps}{.eps.bb}{`gunzip -c #1}
\DeclareGraphicsRule{.ps.Z}{eps}{.ps.bb}{`gunzip -c #1}
\DeclareGraphicsRule{.eps.gz}{eps}{.eps.bb}{`gunzip -c #1}
\DeclareGraphicsRule{.ps.gz}{eps}{.ps.bb}{`gunzip -c #1}
\DeclareGraphicsRule{.pcx}{bmp}{}{}
\DeclareGraphicsRule{.bmp}{bmp}{}{}
\DeclareGraphicsRule{.msp}{bmp}{}{}
\DeclareGraphicsRule{*}{eps}{*}{}
```

The first two commands define the .eps and .ps extensions as EPS files. The next five commands define extensions for compressed EPS files. The next three commands define extensions for bitmaps (see section 7.5.2). The last command defines any other suffix as an EPS file.

7.3 TeX Search Path and dvips

When LaTeX encounters an \includegraphics command, it looks in the current directory for the file. If it does not find the file in the current directory, it searches through the TeX path for the file. When the DVI file is converted to PostScript, dvips performs the same search routine and everything works well. When the file is an uncompressed EPS file, dvips directly includes the specified file. However, when the file is compressed or has a non-EPS format, the file must be operated on by the command specified in \DeclareGraphicsRule. For example, the rule

```
\DeclareGraphicsRule{.eps.gz}{eps}{.eps.bb}{`gunzip -c #1}
```

specifies that the gunzip -c command be used on files having a .eps.gz suffix. Suppose the following command is used

```
\includegraphics{file.eps.gz}
```

If file.eps.gz and file.eps.bb are in the current directory, LaTeX uses file.eps.bb and dvips executes gunzip -c file.eps.gz to uncompressed the file.

If file.eps.gz and file.eps.bb are in the directory /a/b/c/ (on the TeX path), LaTeX searches the path to find /a/b/c/file.eps.bb. However, dvips does not know what part of the command construction is the filename, and thus cannot find the file `gunzip -c file.eps.gz in the TeX search path. If TeX is using a recent kpathsea library (such as the teTeX distribution), this problem can be solved by the following graphics rule

```
\DeclareGraphicsRule{.eps.gz}{eps}{.eps.bb}%
                    {`gunzip -c `kpsewhich -n latex tex #1`}
```

This rule specifies that any file ending with .eps.gz is an EPS file whose BoundingBox file ends with .eps.bb and is uncompressed with gunzip -c. The kpsewhich -n latex tex command makes dvips look for the compressed file in the directories on the TeX search path. This rule allows dvips to operate on any file which TeX can find. This replaces the default definition in dvips.def which is essentially

```
\DeclareGraphicsRule {.eps.gz}{eps}{.eps.bb}{`gunzip -c #1}
```

The rules for other suffixes (such as `.eps.gz` or `.ps.Z`) can be modified similarly. While the new `\DeclareGraphicsRule` command can be placed at the beginning of every document, it may be more convenient to add the following to the `graphics.cfg` file

```
\AtEndOfPackage{%
\DeclareGraphicsRule{.eps.gz}{eps}{.eps.bb}%
                    {`gunzip -c `kpsewhich -n latex tex #1`}}
```

and leaving the existing `\ExecuteOptions{dvips}` line.

7.4 The DeclareGraphicsExtensions Command

The `\DeclareGraphicsExtensions` command tells LaTeX which extensions to try if a file with no extension is specified in the `\includegraphics` command. The following graphic extensions are defined by default in `dvips.def`

```
\DeclareGraphicsExtensions{.eps,.ps,.eps.gz,.ps.gz,.eps.Z}
```

With the above graphics extensions specified, `\includegraphics{file}` first looks for `file.eps`, then `file.ps`, then `file.eps.gz`, etc. until a file is found. This allows the graphics to be specified with

```
\includegraphics{file}
```

instead of

```
\includegraphics{file.eps}
```

The first syntax has the advantage that if you later decide to compress `file.eps`, you need not edit the LaTeX file.

7.5 Non-EPS Graphic Files

While it is easy to insert EPS graphics into LaTeX documents, it is not as straight-forward to insert non-EPS graphics (GIF, TIFF, JPEG, PICT, etc.). A simple solution is to determine whether the application which generated the non-EPS graphic also generates EPS output. If not, a graphics-conversion program must be used to convert the graphics to PostScript.

- ImageMagick is a very good graphics-conversion utility for Unix, Linux, and VMS that is distributed for free from `ftp.wizards.dupont.com` and other sites. See

  `http://www.wizards.dupont.com/cristy/ImageMagick.html`

- `xv` is an X-Windows graphics viewing and conversion program distributed as shareware. See

  `http://www.sun.com/sunsoft/catlink/xv/note.html`

  Unlike ImageMagick, `xv` does not provide command-line conversion capabilities for on-the-fly graphics conversion.

- `WMF2EPS` is a freeware WMF-to-EPS conversion program which runs on Windows95 and WindowsNT. See

  `ftp://ftp.tex.ac.uk/tex-archive/support/wmf2eps/readme.txt`

- A JPEG graphic can be printed on PostScript *level 2* printer by placing the JPEG graphic inside a PostScript wrapper. The C program `jpeg2ps` does this and is available from

```
http://www.muc.de/~tm/free/free.html
```

jpeg2ps can be compiled for Unix, DOS, and other systems. While this results in a smaller PostScript file, it requires a level 2 printer.

Since a non-EPS graphics file may be smaller than the corresponding EPS file, it may be desirable to keep the graphics in a non-EPS format and convert them to PostScript when the DVI file is converted to PostScript. If `dvips` is used, this on-the-fly conversion can be specified by the command option in `\DeclareGraphicsRule`. For example, using on-the-fly conversion to insert `file2.gif` into a LaTeX document requires the following steps

1. Find a GIF-to-PS conversion program (assume it's called `gif2ps`)
2. One needs to create a BoundingBox file which specifies the natural size of the `file2.gif` graphics. To do this, convert `file2.gif` to PostScript and
   (a) If the Postscript file is EPS, save the BoundingBox line in `file2.gif.bb`
   (b) If the Postscript file is not EPS, determine the appropriate BoundingBox (see section 3) and place those numbers in a `%%BoundingBox:` line in `file2.gif.bb`
3. Keep `file2.gif` and `file2.gif.bb` and delete the PostScript file.
4. Include `\DeclareGraphicsRule{.gif}{eps}{.gif.bb}{`gif2ps #1}` before the `\includegraphics` command in the LaTeX file.

When `\includegraphics{file.gif}` is issued, LaTeX reads the BoundingBox from `file.gif.bb` and tells `dvips` to use `gif2ps` to convert `file.gif` to EPS.

7.5.1 GIF Example

While the commands necessary for including non-EPS graphics are dependent on the operating system and the graphics conversion program, this section provides examples for two common Unix conversion programs. The commands

```
\DeclareGraphicsRule{.gif}{eps}{.gif.bb}{`convert #1 'eps:-' }
\begin{figure}
   \centering
   \includegraphics[width=3in]{file2.gif}
   \caption{GIF Graphic}
\end{figure}
```

use the `convert` program (part of the ImageMagick package) package to translate the GIF file into EPS. The command

```
convert file2.gif 'eps:-'
```

translates `file2.gif` into EPS format (specified by the "`eps:`" option), sending the result to standard output (specified by the "`-`" specification).

Alternatively, one can use the `ppm` utilities in which `giftoppm`, `ppmtopgm`, and `pgmtops` convert the GIF file to EPS via the `ppm` and grayscale `pgm` formats. In Unix, the piping between these programs is specified by the following `\DeclareGraphicsRule` command

```
\DeclareGraphicsRule{.gif}{eps}{.gif.bb}{`giftoppm #1 | ppmtopgm | pgmtops}
```

7.5.2 Direct Support for Non-EPS Graphics

It is often requested that LaTeX and `dvips` support the direct inclusion of non-EPS graphic formats, making it as easy as inserting EPS files. While this would be convenient, there unfortunately are problems with this.

For example, most non-EPS graphic formats use binary files which cannot be read by TeX. This prevents LaTeX from reading the file to determine the size of the non-EPS graphics. Furthermore, supporting non-EPS graphics would also require `dvips` to incorporate graphics-conversion capabilities (GIF-to-PS, TIFF-to-PS, etc.). This would not only require a lot of programming, it would also require more future maintenance.

Rather than directly incorporating graphics-conversion routines, `dvips` provides a mechanism of calling external conversion programs. This mechanism can be accessed from LaTeX by use the the command argument of `\DeclareGraphicsRule`. This has the benefit of being more flexible than direct support, and since it keeps the graphics-conversion uncoupled from the DVI-to-PS conversion, users are free to choose their own graphics-conversion program.

While LaTeX and `dvips` generally do not support the direct inclusion of non-EPS graphics, there are some exceptions

1. If `dvips` is compiled with `-Demtex`, it supports some EmTeX `\special` commands, allowing it to include PCX, BMP, or MSP bitmaps.
2. Oztex 2.1, a shareware Macintosh TeX/LaTeX distribution, includes the DVI-to-PS converter OzDVIPS, which allows MacPaint and PICT files to be included via `\special` commands. See

   `http://www.kagi.com/authors/akt/oztex.html`
3. Some commercial versions of LaTeX support non-EPS graphics
   (a) Textures for the Macintosh supports PICT graphics. See
   `http://www.bluesky.com/`
   (b) Y&Y's TeX package for Windows includes the DVI-to-PS converter DVIPSONE which supports TIFF files. However, TeX cannot read the binary TIFF files, preventing LaTeX from reading the TIFF tags the same way it reads EPS BoundingBox information. Since LaTeX cannot determine the natural size of TIFF graphics, the user must still use a `.bb` file or specify the `bb` parameters explicitly in the `\includegraphics` command. See
   `http://www.YandY.com/`

   Check your documentation or contact the company's customer service for the correct syntax.

8 The PSfrag System

While there are many drawing and analysis packages which produce EPS files, most of them do not support symbols and equations as well as LaTeX. The PSfrag system allows LaTeX users to replace text strings in EPS files with LaTeX text or equations.

PSfrag 3.0, which was released in November 1996, has been totally re-written. Previous versions of PSfrag required running a preprocessor (such as `ps2frag` or `ps2psfrag`) to identify and tag all the text in the EPS file. Since PSfrag 3.0 requires no such preprocessing, it does not require any external programs such as `perl` or `ghostscript`. PSfrag 3.0 only requires a recent LaTeX (*12/95* or later) and the graphics bundle distributed with LaTeX. Reference [7] provides complete documentation on PSfrag 3.0.

An additional benefit of PSfrag rewrite is that it now supports compressed EPS graphics. However, the `\tex` command (described in section 8.3) cannot be used to embed LaTeX text in compressed graphics.

To use PSfrag, create an EPS file and then perform the following steps

1. Include `\usepackage{psfrag}` in the preamble of the LaTeX document.
2. In the document, use the `\psfrag` command to specify the EPS text to replace and the LaTeX string to replace it. This makes the specified substitution occur in any subsequent `\includegraphics` command issued in the same environment.
3. Use the `\includegraphics` command as usual.

The LaTeX `\psfrag` command has the following syntax

```
\psfrag{PStext}[posn][PSposn][scale][rot]{text}
```

with its arguments described in Table 5.

Table 5: `psfrag` Options

`PStext`	Text in EPS file to be replaced.
`posn`	*(Optional, Defaults to [Bl].)* Position of placement point relative to new LaTeX text.
`PSposn`	*(Optional, Defaults to [Bl].)* Position of placement point relative to existing EPS text.
`scale`	*(Optional, defaults to 1.)* Scaling factor for the text. For best results, avoid using the scaling factor and instead use LaTeX type-size commands such as `\small` and `\large`
`rot`	*(Optional, defaults to zero.)* This option is especially useful when dealing with applications which only allow horizontal text in their EPS files. When this rotation angle is zero, the new text is inserted at the same angle as the existing EPS text. When an angle is specified here, it is the angle of rotation of the new text relative to the existing text. The angle is in degrees with a counter-clockwise rotation being positive.
`text`	The LaTeX text to insert into the EPS graphic. Like regular LaTeX text, math formulas must be enclosed by dollar signs (e.g., `$\frac{1}{2}$` or `$x^2$`) and special symbols can be used (e.g., `\%` produces %).

The `posn` and `PSposn` options are one of the 12 points (such as `[tl]`, `[br]`, `[cc]`) shown in Figure 3 on page 14. If the optional arguments are not issued, the point defaults to `[Bl]`. Any missing letters default to `c` (e.g., `[\,]` and `[c]` are equivalent to `[cc]`, `[l]` is equivalent to `[lc]`). See [7] for examples of various combinations of placement points.

8.1 PSfrag Example #1

The commands

```
\includegraphics{pend.eps}
```

include the graphic without any PSfrag replacement, producing Figure 7. The commands

```
\psfrag{q1}{$\theta_1$}
\psfrag{q2}{$\theta_2$}
\psfrag{L1}{$L_1$}
\psfrag{L2}{$L_2$}
\psfrag{P1}[][]{$P_1$}
\psfrag{P2}[][]{\Large $P_2$}
\includegraphics{pend.eps}
```

include the graphic with PSfrag replacement, producing Figure 8. The first four `\psfrag` commands position the new LaTeX text such that its left baseline point corresponds to the left baseline point of the EPS text. The last two `\psfrag` commands use the `[][]` options to position the LaTeX text such that its center corresponds to the center of the EPS text.

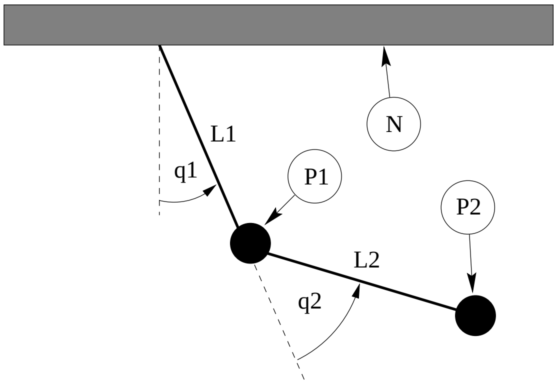


Figure 7: Without PSfrag Replacement

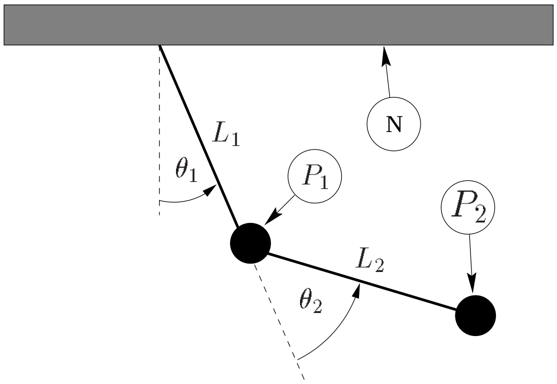


Figure 8: With PSfrag Replacement

Note that all EPS text need not be replaced. For example, the `N` tag is left unchanged in Figure 8. Also note that `\psfrag` matches *entire* text strings. Thus the command

```
\psfrag{pi}{$\pi$}
```

replaces the string `pi` with π, but does not affect the strings `pi/2` or `2pi`. Separate `\psfrag` commands must entered for these strings.

8.2 PSfrag Example #2

This example demonstrates how the `\shortstack`, `\colorbox`, and `\fcolorbox` commands can be used with `\psfrag`.

shortstack The `\shortstack` command allows text to be stacked vertically, which can be used to substitute multiple lines of text for a single line of text. The lines of text are separated by the `\\` command.

colorbox The `\colorbox` command (part of the color package, which is distributed with LaTeX) places a color background behind an object. For example,

```
\colorbox{white}{text}
```

places a rectangular white background behind `text`. See reference [4] for more details on `\colorbox`.

With PSfrag, `\colorbox` is useful for placing text at a location where lines or shading would make it difficult to view the text. Placing a white background behind the text will give the text an unobstructed view.

fcolorbox The `\fcolorbox` command (also part of the color package) is similar to the `\colorbox` command, except that a frame is drawn around the background. The command `\fcolorbox{black}{white}{text}` puts a white background with a black frame behind `text`.

Figure 9 and 10 demonstrate the use of these commands with PSfrag. Figure 9 shows the graphic without PSfrag substitution. The commands

```
\psfrag{q1}[][]{\colorbox{white}{$q_1$}}
\psfrag{base}{\fcolorbox{black}{white}{Base}}
\psfrag{Actuator}[l][l]{\shortstack{Hydraulic\\ Actuator}}
\includegraphics{mass.eps}
```

use PSfrag to produce Figure 10.

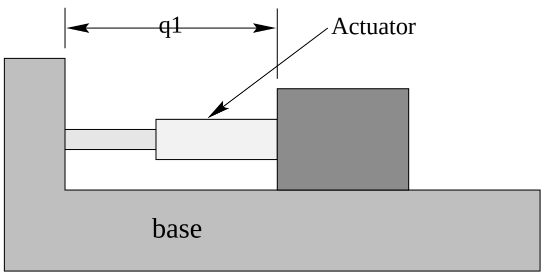


Figure 9: Without PSfrag Replacement

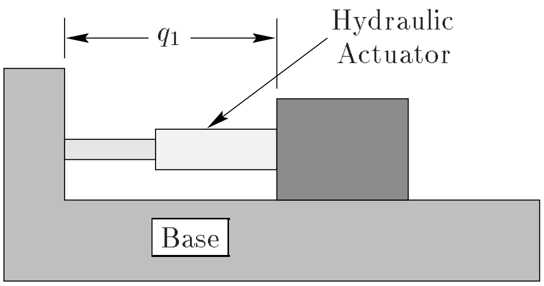


Figure 10: With PSfrag Replacement

8.3 LaTeX Text in EPS File

The recommended and most popular method for using PSfrag is the `\psfrag` command described in the previous section. An alternative, albeit less efficient, method for using PSfrag is the `\tex` command, which embeds the LaTeX text directly in the EPS file. The `\tex` command has the following syntax

```
\tex[posn][PSposn][scale][rot]{text}
```

which is the same as `\psfrag` command, except there is no `{PStext}` argument. Unlike the `\psfrag` command, the `\tex` command is placed in the EPS file. For example, if an EPS file contains the text `\tex{$x^2$}` PSfrag automatically replaces it with x^2. Since no optional arguments were specified, the left-baseline point of x^2 is aligned with the left-baseline point of `\tex{$x^2$}`.

Unlike the `\psfrag` command, the `\tex` command requires LaTeX to scan the EPS file looking for `\tex` commands. Since this scanning slows down the LaTeX processing, it is not turned on by default. There are two methods for activating the replacement of `\tex` commands

1. Use the `\psfragscanon` command to cause all subsequent `\includegraphics` commands in the current environment to perform `\tex` scanning.
2. Use the `scanall` option

   ```
   \usepackage[scanall]{psfrag}
   ```

   which causes *all* `\includegraphics` commands to perform `\tex` scanning.

The `\tex` command must be *entire* text string. For example, the text

```
Transfer Function \tex{$\frac{s+a}{s+b}$}
```

produces an error. Instead use

```
\tex{Transfer Function $\frac{s+a}{s+b}$}
```

The `\tex` command has three disadvantages.

1. The `\tex` scanning slows down the LaTeX processing.
2. The EPS file cannot be used for non-LaTeX purposes, while the EPS graphic in Figure 7 could be used without replacement.
3. If a `\tex` command contains complicated formulas, the text can extend beyond the edge of the graphics, enlarging the EPS BoundingBox. This oversized BoundingBox causes incorrect graphic placement/alignment in LaTeX.

Because of these disadvantages, it is recommended that the `\psfrag` command be used instead of the embedded `\tex` command.

8.4 Figure and Text Scaling with PSfrag

If a graphic using PSfrag is scaled, the PSfrag text is scaled along with the graphic. As a result, a subtlety of the graphicx package affects the size of the text.

- When the `width`, `height`, or `totalheight` options are used to size the graphic

  ```
  \includegraphics[width=3in]{file.eps}
  ```

  the PSfrag text is inserted *after* the scaling. Conversely,

  ```
  \resizebox{3in}{!}{\includegraphics{file.eps}}
  ```

  Includes the graphic at its natural size, inserts the PSfrag text, and then scales both the graphics and the text.
- Similarly, when scaling options are specified *before* rotation

  ```
  \includegraphics[width=3in,angle=30]{file.eps}
  ```

  the scaling is implicitly handled by the graphics inclusion function. However, when scaling options are specified *after* rotation

  ```
  \includegraphics[angle=30,width=3in]{file.eps}
  ```

  the graphic is first included at its natural size, then rotated, and then scaled. Since PSfrag replaces the new text during the graphics inclusion, the second command scales the new PSfrag text while the first command does *not*. When the included size of the EPS graphic greatly differs from its natural size, the two commands produce very different results.

See [6] and [7] for more information on the scaling of PSfrag text.

9 Including An EPS File Multiple Times

When the same EPS graphic is inserted multiple times, its EPS code appears multiple times in the final PS file. In particular, this often happens when a logo or other graphics are inserted into a document's header or footer. This section describes improved methods for inserting a graphic multiple times.

There are four common methods for including the same EPS graphics many times

1. Using `\includegraphics{file.eps}` wherever you want the graphic has two problems
   (a) LaTeX must find and read the file every time `\includegraphics` is used.
   (b) The EPS graphics commands are repeated in the final PS file, producing a large file.
2. Save the graphics in a LaTeX box, using the box wherever you want the graphic. This saves LaTeX time since it must only find and read the file once. However, it does not reduce the size of the final PostScript file.

   At the beginning of the file, include the following commands

   ```
   \newsavebox\mygraphic
   \sbox\mygraphic{\includegraphics{file.eps}}
   ```

   Then use the command `\usebox{\mygraphic}` wherever you want the graphic.
3. When the EPS file contains vector graphics (as opposed to bitmapped graphics), it is possible to write a PostScript command which draws the graphics. The graphic can then be included by issuing the PostScript command wherever the graphic is needed. Section 9.1 describes this procedure.

   Since the final PostScript file includes the graphics commands only once, the final PostScript file is much smaller. Note that since the graphics commands are stored in printer memory while the final PostScript file is being printed, this method *may* cause the printer to run out of memory and not print the document.

   Although this method results in a small final PostScript file, it still requires LaTeX to find and read the file containing the PostScript commands.
4. Like the previous method, define a PostScript command which draws the graphics, but include this command in a LaTeX box. This results in a small final PostScript file and only requires LaTeX to find and read the file once.

9.1 Defining a PostScript Command

This section describes how to created a PostScript command which draws the graphics from an EPS file containing vector graphics. This procedure will not work if the EPS file contains *bitmapped* graphics.

To convert the EPS graphics into a PostScript command, the EPS file must be broken into two files, one which defines the PostScript dictionary and the graphics commands, and another which includes the header information and the uses the previously-defined PostScript command. For example, an EPS file created by xfig has the form

```
%!PS-Adobe-2.0 EPSF-2.0
%%Title: /tmp/xfig-fig017255
%%Creator: fig2dev Version 2.1.8 Patchlevel 0
%%CreationDate: Sun Sep  3 15:36:01 1995
%%Orientation: Portrait
%%BoundingBox: 0 0 369 255
%%Pages: 0
%%EndComments
/$F2psDict 200 dict def
$F2psDict begin
...
%%EndProlog

$F2psBegin
...
$F2psEnd
```

Where ... indicates unlisted commands. The EPS file generally contains three parts

1. The header commands which begin with `%`
2. The Prolog section which starts with

   ```
   /$F2psDict 200 dict def
   ```

   and ends with

   ```
   %%EndProlog
   ```

   The Prolog defines the commands in the PostScript dictionary used by the EPS file. In this example, the dictionary is named `$F2psDict` although other names can be used.
3. The last part contains the commands used to draw the graphics.

Suppose the above EPS file is named `file.eps`. Create the files `file.h` and `file.ps` where `file.h` contains

```
/$F2psDict 200 dict def
$F2psDict begin
...
%%EndProlog

/MyFigure {
$F2psBegin
...
$F2psEnd
} def
```

and `file.ps` contains

```
%!PS-Adobe-2.0 EPSF-2.0
%%Title: /tmp/xfig-fig017255
%%Creator: fig2dev Version 2.1.8 Patchlevel 0
%%CreationDate: Sun Sep  3 15:36:01 1995
%%Orientation: Portrait
%%BoundingBox: 0 0 369 255
%%Pages: 0
%%EndComments
$F2psDict begin MyFigure end
```

`file.h` defines the dictionary and defines the PostScript command `/MyFigure`, while `file.ps` contains the header information and uses the PostScript command defined in `file.h`. In particular, it is important that the `file.ps` header includes the `%!PS...` line and the BoundingBox line. The graphics can then be used in the LaTeX document as

```
\documentclass{article}
\usepackage{graphicx}
\special{header=file.h}
\begin{document}
   ...
   \includegraphics[width=2in]{file.ps}
   ...
   \includegraphics[totalheight=1in]{file.ps}
   ...
\end{document}
```

Note that the original file `file.eps` is not used. Since the graphics commands in `file.h` are only included once, the final PostScript file remains small. However, this still requires LaTeX to find and read `file.ps` whenever the graphics are used. The following commands save the graphics in a LaTeX box to produce a small final PostScript file while reading `file.ps` only once.

```
\documentclass{article}
\usepackage{graphicx}
\special{header=file.h}

\newsavebox\mygraphic
\sbox\mygraphic{\includegraphics[width=2in]{file.ps}}

\begin{document}
   ...
   \usebox{\mygraphic}
   ...
   \resizebox*{1in}{!}{\usebox{\mygraphic}}
   ...
\end{document}
```

Like the previous example, these commands produce a 2-inch wide graphic and another graphic whose totalheight is 1 inch.

9.2 Graphics in Page Header or Footer

An easy method of including graphics in the heading is to use the fancyhdr package (an improved version of the old fancyheadings package) which is documented by [8]. The header consists of three parts: its left field, its center field, and its right field. The `\fancyhead` command specifies the contents of the header fields, with the `L,C,R` options specifying which field(s) the command modifies. For example

```
\pagestyle{fancy}
\fancyhead[C]{My Paper}
```

causes the center header field to be "My Paper", while

```
\pagestyle{fancy}
\fancyhead[L,R]{\textbf{Confidential}}
```

causes both the left and right header fields to be "**Confidential**". If no `L,C,R` option is specified, it applies to all three header fields. Thus `\fancyhead{}` is used to clear all the header fields. The `\fancyfoot` command similarly specifies the left, center, and right footer fields.

Note that the above `\fancyhead` commands only apply to pages whose style are "fancy". Even though `\pagestyle{fancy}` causes the document to have a fancy page style, some pages (title pages, table of contents pages, the first page of chapters, etc.) are still given a plain pagestyle by default.

Graphics in Page Header/Footer

The commands in the fancyhdr package can insert graphics in the headers and footers. For example, after splitting the EPS file `file.eps` into the two file `file.h` and `file.ps` as described in section 9.1, the commands

```
\documentclass{article}
\usepackage{fancyhdr,graphicx}

\renewcommand{\headheight}{0.6in}  %% must be large enough for graphic
\renewcommand{\textheight}{7.5in}

% Define PostScript graphics command
\special{header=file.h}

% Save graphics in LaTeX box
\newsavebox\mygraphic
\sbox\mygraphic{\includegraphics[totalheight=0.5in]{file.ps}}

\pagestyle{fancy}
\fancyhead{}   % clear all header fields
\fancyhead[L]{\usebox{\mygraphic}}
\fancyfoot{}   % clear all footer fields
\fancyfoot[C]{\thepage}
\renewcommand{\headrulewidth}{0.5pt}
\renewcommand{\footrulewidth}{0pt}

\begin{document}
...
\end{document}
```

places the graphics at the top left of each "fancy" page with a 0.5 pt horizontal line drawn under the header. Additionally, the page number is placed at the bottom center of each page, with no horizontal line drawn above the footer. Note that this does not affect "plain" pages.

Odd/Even Headings

When the `[twoside]` documentclass option is used, one may want to individually specify the odd and even page headers/footers. The `E,O` `\fancyhead` options specify the even and odd page headers, respectively. If the `E,O` options are not specified, the command applies to both even and odd pages. Likewise the `E,O` `\fancyfoot` options specify the even and odd page footers. For example,

```
\pagestyle{fancy}
```

```
\fancyhead[LE]{My Paper}
\fancyhead[RO]{My Name}
\fancyfoot[C]{\thepage}
```

places “My Paper” in the upper left of even fancy pages, “My Name” in the upper right of odd fancy pages, and the page number in the bottom center of all fancy pages. Replacing the

```
\fancyhead[L]{\usebox{\mygraphic}}
```

command in the above example with

```
\fancyhead[LE,RO]{\usebox{\mygraphic}}
```

places the graphic at the top outside (the left side of even pages, right side of odd pages) of all fancy pages.

Modifying Plain Pages

Although the above commands do not affect pages with plain pagestyles, the `\fancypagestyle` command can be used to modify the plain pagestyle. For example

```
\documentclass{article}
\usepackage{fancyhdr,graphicx}

\renewcommand{\headheight}{0.6in}  %% must be large enough for graphic
\renewcommand{\textheight}{7.5in}

% Define PostScript graphics command
\special{header=file.h}

% Save graphics in LaTeX box
\newsavebox\mygraphic
\sbox\mygraphic{\includegraphics[totalheight=0.5in]{file.ps}}

\pagestyle{fancy}
\fancyhead{}   % clear all header fields
\fancyhead[L]{\usebox{\mygraphic}}
\fancyfoot{}   % clear all footer fields
\fancyfoot[C]{\thepage}
\renewcommand{\headrulewidth}{0.5pt}
\renewcommand{\footrulewidth}{0pt}

\fancypagestyle{plain}{%
   \fancyhead{}   % clear all header fields
   \fancyhead[L]{\usebox{\mygraphic}}
   \fancyfoot{}   % clear all footer fields
   \fancyfoot[C]{\thepage}
   \renewcommand{\headrulewidth}{0.5pt}
   \renewcommand{\footrulewidth}{0pt}}

\begin{document}
...
\end{document}
```

place the graphic at the upper left of every page (both plain and fancy). Likewise, when the `twoside` documentclass option is used, replacing both of the

```
\fancyhead[L]{\usebox{\mygraphic}}
```

commands with

```
\fancyhead[LE,RO]{\usebox{\mygraphic}}
```

places the graphic at the top outside of every page (both plain and fancy).

Part IV
Related LaTeX Commands

10 The figure Environment

Graphics can be inserted as part of a LaTeX `\figure` environment, which allows the graphics to float for better formatting, especially for large graphics. The `\figure` environment also makes it easy to reference the graphic. The commands

```
\begin{figure}
   \centering
   \includegraphics[totalheight=2in]{graph.eps}
   \caption{This is an inserted EPS graphic}
   \label{fig:graph}
\end{figure}

The graph in Figure~\ref{fig:graph} is
from an EPS file generated by gnuplot.
```

insert the graphic in a figure and place a caption under the graphic. The optional `\label` command specifies a label which is used by the `\ref` command to reference the figure. Note that the `\label` command must be *after* the `\caption` command (but must still be in the same group/environment). Note that the figure environment can only be used in *outer paragraph mode* and thus cannot be used inside any box (such as parbox or minipage).

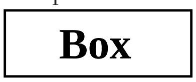
10.1 Caption Vertical Spacing

While the figure caption is usually placed below the graphic, it can be placed above the graphics simply by placing the `\caption` command before the graphics-inclusion command. For example, the commands

```
\begin{figure}
   \centering
   \caption{Caption Above Graphic}
   \includegraphics[width=1in]{box.eps}
\end{figure}
```

produce Figure 11.

Figure 11: Caption Above Graphic

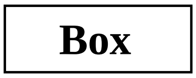


Since captions are generally placed below the graphic, LaTeX places more vertical spacing above the caption than below it. As a result, the caption in Figure 11 is placed quite close to the graphic. The spacing above and below the caption is controlled by the two lengths `\abovecaptionskip` (which is `10pt` by default) and `\belowcaptionskip` (which is zero by default). The standard LaTeX commands `\setlength` and `\addtolength` are used to modify these lengths. For example, the commands

```
\begin{figure}
   \setlength{\abovecaptionskip}{0pt}
   \setlength{\belowcaptionskip}{10pt}
   \centering
   \caption{Caption Above Graphic}
   \includegraphics[width=1in]{box.eps}
\end{figure}
```

produce Figure 12, which has has no extra space above the caption and 10 points of space between the caption and the graphic.

Figure 12: Caption Above Graphic



10.2 Figure Placement Options

Figures are "floats" whose placement is determined by LaTeX in order to avoid blank whitespace and to follow certain esthetic guidelines. Since users' taste in figure-placement may differ from that of LaTeX, the `\figure` environment has placement options which allow users to specify possible figure locations

`h` *Here:* Place the figure in the text where the figure command is located.

`t` *Top:* Place the figure at the top of a page.

`b` *Bottom:* Place the figure at the bottom of a page.

`p` *Page of Floats:* Place the figure on a separate page which contains only floats.

The placement options `[htb]` cause LaTeX to first try to place the figure at that location, then try to place the figure at the top of a page, and finally try to place the figure at the bottom of a page. When LaTeX "tries" to place a figure, it checks how many figures are already on the page and other esthetic concerns. If LaTeX determines that the figure wouldn't look good, it tries the next placement option.

The order in which the placement options are specified does *not* make any difference. The placement options are attempted in the order `h-t-b-p` regardless of the order in which the options are specified. Thus `[hb]` and `[bh]` are both attempted as `h-b`.

To make LaTeX "try really hard" in its float placement, specify an exclamation point in the placement options (e.g., `\begin{figure}[!ht]`) which makes LaTeX suspend its esthetic rules and do its best to make the requested placement. Even with the `!` option, LaTeX will override the "here" request in order to avoid whitespace. For example, if the commands

```
\begin{figure}[!ht]
   \includegraphics[totalheight=4in]{graph.eps}
\end{figure}
```

occur 3 inches from the bottom of the page, LaTeX objects to leaving 3 inches of whitespace at the bottom of the page and thus the bottom 3 inches of the page is filled with the text which is after the figure in the .tex file.

If you feel LaTeX is making poor float placement decisions, you may need to tweak its placement algorithm by modifying the float parameters (see [1, pages 199-200], [2, pages 141-143], or [3, pages 174-175]).

10.2.1 The float Package's [H] Placement Option

The float package adds an [H] option to the \figure environment which *always* places the float "here". However, the [H] option should generally be avoided, as the [!ht] option is a better way of producing the desired behavior.

To use the [H] option, include a \usepackage{float} command in the preamble and issue the \restylefloat{figure} command *before* the \begin{figure}[H] command is used (See [2, page 149]). When using the [H] option, the user is responsible for managing the document to avoid large sections of whitespace.

While the figure environment defined by the float package allows the [H] option, it also places the figure caption below the figure environment. While this does not affect simple figures, it prevents captions above graphics as in Figure 11 or the construction of side-by-side and other complex figure arrangements.

11 Landscape Figures

In a document with portrait orientation, there are three methods for producing figures with landscape orientation.

1. The lscape package provides a landscape environment, which treats the left edge of the paper as the top of the page, causing any text, tables, or figures in the landscape environment to have landscape orientation.
2. The rotating package provides a sidewaysfigure environment which is similar to the figure environment except that the figures have landscape orientation.
3. The rotating package provides a \rotcaption command which is similar to the \caption command except that caption has landscape orientation.

Differences between methods

- Both options 1 and 2 place the rotated figure on a separate page. Option 3 produces an individual float which need not be on its own page.
- The full-page figure produced by Option 2 will float to provide better document formatting. Since the figure(s) produced by Option 1 can only float within the landscape pages, it may result in a partially-empty page before the figure.
- The landscape environment in Option 1 can be used to produce landscape pages containing any combination of text, tables, and figures. Option 2 produces only rotated figures.

11.1 Landscape Environment

The lscape package (which is part of the standard "graphics bundle" distributed with LaTeX) defines the landscape environment, which provides a method of placing landscape pages in a portrait document. The landscape pages are rotated such that the left edge of the portrait page is the top edge of the landscape page.

Entering \begin{landscape} prints all unprocessed portrait floats and then switches to landscape orientation. Likewise, \end{landscape} prints all unprocessed landscape floats and then switches back to portrait orientation.

The entire contents of the `landscape` environment is typeset with landscape orientation. This may include any mixture of text, figures, and tables. If the landscape environment contains only a figure environment

```
\begin{landscape}
  \begin{figure}
     \centering
     \includegraphics[width=4in]{box.eps}
     \caption{Landscape Figure}
  \end{figure}
\end{landscape}
```

the `landscape` environment produces a landscape figure. Note that since the `landscape` environment starts a new page, it may result in a partially-blank page.

11.2 Sidewaysfigure Environment

The rotating package provides the `sidewaysfigure` environment which produces figures with landscape orientation. For example

```
\begin{sidewaysfigure}
   \centering
   \includegraphics[width=4in]{box.eps}
   \caption{Sidewaysfigure Figure}
\end{sidewaysfigure}
```

produces Figure 13.

Unlike the `landscape` environment, the figure produced by `sidewaysfigure` can float within the portrait pages to avoid the partially-blank page that the `landscape` environment may produce. (The rotating package also provides a `sidewaystable` environment for producing tables with landscape orientation.) Note that the `landscape` environment is much more flexible, allowing the landscape pages to consist of a mixture of text, tables, and figures.

The default orientation of the figures produced by `sidewaysfigure` depends on whether the document is processed with the `oneside` or `twoside` documentclass option

- When the `oneside` option is chosen, the bottom of graphic is towards the the right edge of the portrait page.
- When the `twoside` option is chosen, the bottom of graphic is towards the the outside edge of the portrait page.

This default behavior can be overridden by options to the `\usepackage{rotating}` command.

```
\usepackage[rotateleft]{rotating}
```

causes the bottom of the `sidewaysfigure` graphics to be towards the left edge of the portrait page (regardless of `oneside` or `twoside` options). Similarly,

```
\usepackage[rotateright]{rotating}
```

causes the bottom of the `sidewaysfigure` graphics to be towards the right edge of the portrait page.

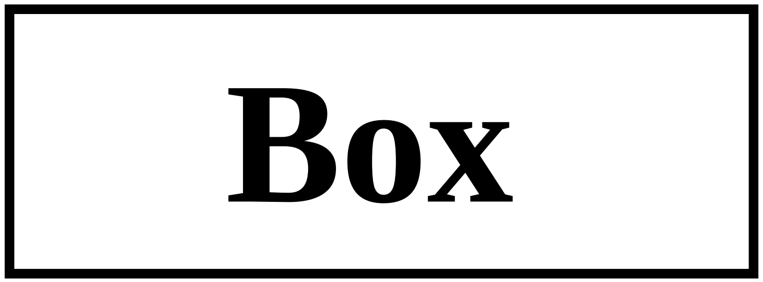


Figure 13: Sidewaysfigure Figure

11.3 Rotcaption Command

The methods in Sections 11.1 and 11.2 both produce full-page landscape figures, which may not be necessary for smaller landscape figures. The `rotating` package's `\rotcaption` command can be used to construct smaller landscape figures. For example

```
\begin{figure}
  \centering
  \begin{minipage}[c]{1in}
     \includegraphics[angle=90,width=\textwidth]{box.eps}
  \end{minipage}
  \begin{minipage}[c]{0.5in}
     \rotcaption{Rotcaption Caption}
     \label{fig:rotcaption}
  \end{minipage}
\end{figure}
```

produces Figure 14.

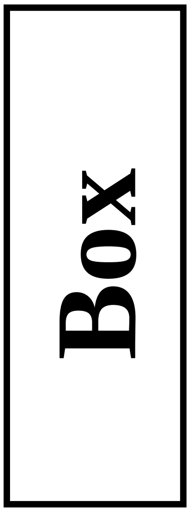


Figure 14: Rotcaption Caption

The caption produced by `\rotcaption` is always rotated such that its bottom is towards the right edge of the paper. Unlike the methods in Sections 11.1 and 11.2, the `\rotcaption` command does not rotate the graphics. Therefore, the `\includegraphics` command in the above example required the `angle=90` option. See Figure 31 on page 53 for a side-by-side similar arrangement which has a horizontal caption.

12 Side-by-Side Graphics

The commands necessary for side-by-side graphics depend on how the user wants the graphics organized. This section covers three common methods of organizing side-by-side graphics

1. The side-by-side graphics are combined into a single figure.
2. The side-by-side graphics each form their own figure (e.g., Figure 17 and Figure 18).
3. The side-by-side graphics each form a subfigure (e.g., Subfigure 25(a) and Subfigure 25(b)) which are part of a single figure (Figure 25).

While this section specifically discusses side-by-side graphics, most of the information is also valid for vertically-stacked graphics and complex figures such as Figures 32-38 on Page 55.

12.1 Side-by-Side Graphics in a Single Figure

The two most common methods for placing side-by-side graphics in a figure are

1. Multiple `\includegraphics` commands
2. Multiple `minipage` environments, each of which contain a `\includegraphics` command

12.1.1 Side-by-Side includegraphics Commands

While spacing side-by-side graphics in a figure is as simple as

```
\begin{figure}
   \centering
   \includegraphics[width=1in]{file1.eps}
   \includegraphics[width=2in]{file2.eps}
   \caption{Two Graphics in One Figure}
\end{figure}
```

there usually are horizontal-spacing commands such as `\hspace{1in}` or `\hfill` between the `\includegraphics` commands. For example

```
\begin{figure}
   \centering
   \includegraphics[width=1in]{box.eps}%
   \hspace{1in}%
   \includegraphics[width=2in]{box.eps}
   \caption{Two Graphics in One Figure}
\end{figure}
```

produces Figure 15 which is 4 inches wide (1 inch for `file1.eps`, 1 inch for the `\hspace`, and 2 inches for `file2.eps`). This 4-inch-wide figure is centered on the page. If `\hfill` is used instead of `\hspace`, the graphics are pushed to the margins.

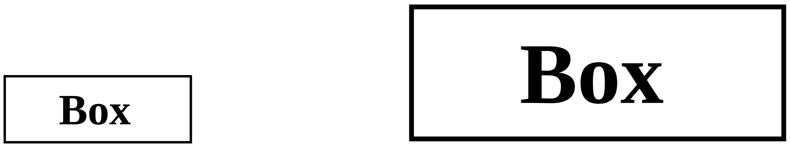


Figure 15: Two Graphics in One Figure

12.1.2 Side-by-Side Minipage Environments

Placing the `\includegraphics` commands inside `minipage` environments provides the user more control over the graphics' horizontal and vertical placement. For example

```
\begin{figure}
   \centering
   \begin{minipage}[c]{0.5\textwidth}
```

```
      \centering \includegraphics[width=1in]{box.eps}
   \end{minipage}%
   \begin{minipage}[c]{0.5\textwidth}
      \centering \includegraphics[width=2in]{box.eps}
   \end{minipage}
   \caption{Centers Aligned Vertically}
\end{figure}
```

produces Figure 16.

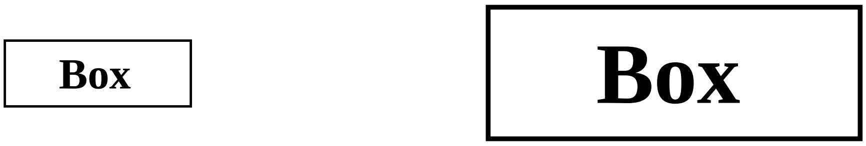


Figure 16: Centers Aligned Vertically

Notes on this example

- Like any other LaTeX object, minipages are positioned such that their baseline is aligned with the current baseline. The minipage `[c]` option defines the minipage's baseline as its centerline. The `[b]` option defines the minipage's baseline as the baseline of the bottom line of the minipage (which is not necessarily the bottom of the minipage). The `[t]` option defines the minipage's baseline as the baseline of the top line of the minipage (which is not necessarily the top of the minipage). See section 6.3 for information on the `minipage` environment and its placement options.
- The `%` after the first `\end{minipage}` command prevents a space from being inserted between the `minipage` boxes. Such a space would use some horizontal space, preventing both minipages from fitting on the same line.

When the widths of the minipages do not add to `1.0\textwidth`, the `\hspace` or `\hfill` commands can be used to specify to horizontal spacing. For example

```
\begin{figure}
   \centering
   \begin{minipage}[c]{1in}
      \centering \includegraphics[width=\textwidth]{box.eps}
   \end{minipage}
   \hspace{1in}
   \begin{minipage}[c]{2in}
      \centering \includegraphics[width=\textwidth]{box.eps}
   \end{minipage}
   \caption{Centers Aligned Vertically}
\end{figure}
```

produces a figure with the same horizontal spacing as Figure 15, but the centers of the boxes are aligned vertically.

12.2 Side-by-Side Figures

In the previous section, multiple `minipage` environments were used inside a `figure` environment to produce a single figure consisting of multiple graphics. Placing `\caption` statements inside the minipages makes the minipages themselves become figures. For example

```
\begin{figure}
   \begin{minipage}[b]{0.5\linewidth}
      \centering \includegraphics[width=1in]{box.eps}
      \caption{Small Box}  \label{fig:side:a}
   \end{minipage}%
   \begin{minipage}[b]{0.5\linewidth}
     \centering \includegraphics[width=1.5in]{box.eps}
     \caption{Big Box}  \label{fig:side:b}
   \end{minipage}
\end{figure}
```

produces Figures 17 and 18.

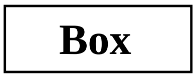


Figure 17: Small Box

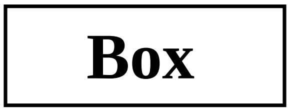


Figure 18: Big Box

Although the above commands include *one* figure environment, the commands produce *two* figures. Since the `\caption` command actually produces the figure, figure environments with multiple `\caption` commands produce multiple figures.

12.2.1 Alignment Problems with Side-by-Side Figures

The `[b]` options aligned the bottoms of Figures 17 and 18. However, long captions may affect this alignment. For example

```
\begin{figure}
   \begin{minipage}[b]{.333\linewidth}
      \centering \includegraphics[width=1in]{box.eps}
      \caption{Small Box with a Long Caption} \label{fig:side:c}
   \end{minipage}%
   \begin{minipage}[b]{.333\linewidth}
      \centering \includegraphics[width=1.5in]{box.eps}
      \caption{Medium Box}  \label{fig:side:d}
   \end{minipage}%
   \begin{minipage}[b]{.333\linewidth}
      \centering \includegraphics[width=2.0in]{box.eps}
      \caption{Big Box}  \label{fig:side:e}
   \end{minipage}
\end{figure}
```

produces Figures 19, 20, and 21.

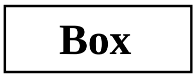


Figure 19: Small Box with a Long Caption

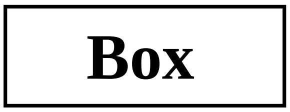


Figure 20: Medium Box

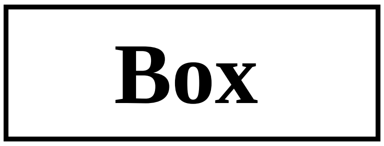


Figure 21: Big Box

The long caption of Figure 19 makes it unaligned with the other figures. In this case, the baselines of all the figures are their bottoms, so the alignment can be corrected by changing the `minipage` positioning option from `[b]` to `[t]` which aligns the baselines of the graphics (see Section 6.3 for information). If the baselines of the graphics do not correspond to their bottoms, the `[t]` option does not produce the desired positioning. Instead, invisible vertical lines (called *struts*) can be placed in the captions of the other figures to make LaTeX think that all the captions are two lines long.

```
\begin{figure}
   \begin{minipage}[b]{.333\linewidth}
      \centering \includegraphics[width=1in]{box.eps}
      \caption{Small Box with a Long Caption}  \label{fig:side:cc}
   \end{minipage}%
   \begin{minipage}[b]{.333\linewidth}
      \centering \includegraphics[width=1.5in]{box.eps}
      \caption[Medium Box]
          {Medium Box \protect\rule[-\baselineskip]{0pt}{2\baselineskip}}
      \label{fig:side:dd}
   \end{minipage}%
   \begin{minipage}[b]{.333\linewidth}
      \centering \includegraphics[width=2.0in]{box.eps}
      \caption[Big Box]
          {Big Box \protect\rule[-\baselineskip]{0pt}{2\baselineskip}}
      \label{fig:side:ee}
   \end{minipage}
\end{figure}
```

which produces Figures 22, 23, and 24.

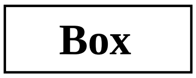


Figure 22: Small Box with a Long Caption

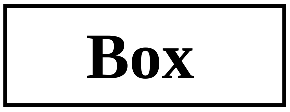


Figure 23: Medium Box

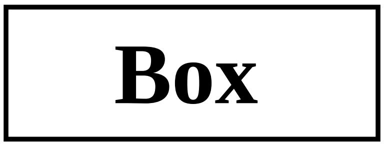


Figure 24: Big Box

The command `\rule[start]{width}{height}` produces an vertical line with a width of `width` starting `start` above the baseline and with a height `height`. When the width is zero, the line becomes invisible and is called a *strut*. In the above captions, the strut

```
\rule[-\baselineskip]{0pt}{2\baselineskip}}
```

starts one line below the baseline and continues to the top of the current line. This makes LaTeX think that, like the Figure 22 caption, the captions for Figures 23 and 24 are two lines tall. Since the `\rule` command is fragile, the `\protect` command must be used so `\rule` can be used in the `\caption` command. The `\caption[Big Box]` option specifies that the text “Big Box” should be used in the list of figures (where the extra vertical space is not desired).

12.3 Side-by-Side Subfigures

It is often desirable to refer to side-by-side graphics both individually and as a group. The `\subfigure` command (from the subfigure package) defines the group of side-by-side graphics as a single figure and defines each graphics as a subfigure. For example

```
\begin{figure}
   \centering
   \subfigure[Small Box with a Long Caption]{
        \label{fig:subfig:a}          %% label for first subfigure
        \includegraphics[width=1.0in]{box.eps}}
   \hspace{1in}
   \subfigure[Big Box]{
        \label{fig:subfig:b}          %% label for second subfigure
        \includegraphics[width=1.5in]{box.eps}}
   \caption{Two Subfigures}
   \label{fig:subfig}                 %% label for entire figure
\end{figure}
```

produces Figure 25. The label `{fig:subfig:a}` refers to subfigure 25(a), while `{fig:subfig:b}` refers to subfigure 25(b), and `{fig:subfig}` refers to to Figure 25.

(a) Small Box with a Long Caption

(b) Big Box

Figure 25: Two Subfigures

12.3.1 Minipage Environments Inside Subfigures

Like other side-by-side graphics, subfigures can be used with `minipage` environments. Depending on the situation, this may make it easier to achieve the desired spacing. For example

```
\begin{figure}
    \subfigure[Small Box with a Long Caption]{
    \label{fig:mini:subfig:a}   %% label for first subfigure
    \begin{minipage}[b]{0.5\textwidth}
       \centering \includegraphics[width=1in]{box.eps}
    \end{minipage}}%
    \subfigure[Big Box]{
    \label{fig:mini:subfig:b}   %% label for second subfigure
    \begin{minipage}[b]{0.5\textwidth}
       \centering \includegraphics[width=1.5in]{box.eps}
    \end{minipage}}
    \caption{Minipages Inside Subfigures}
    \label{fig:mini:subfig}     %% label for entire figure
\end{figure}
```

produces Figure 26, which contains subfigures 26(a) and 26(b).

(a) Small Box with a Long Caption (b) Big Box

Figure 26: Minipages Inside Subfigures

12.3.2 Changing Subfigure Numbering

The subfigure labels have two forms

1. One which appears under the subfigure as part of the caption. This is produced by the `\@thesubfigure` command.
2. One which appears when the `\ref` command is used. This is produced by concatenating the output of `\p@subfigure` to the output `\thesubfigure`.

These commands use the `subfigure` counter and the `\thefigure` command, making the subfigure label formating be controlled by the following commands

- The command `\thefigure` prints the current figure number.
- The counter `subfigure` counts the subfigures. The command `\alph{subfigure}` prints the value of the `subfigure` counter in lowercase letters, while `\roman{subfigure}` prints the value of the `subfigure` counter in lowercase Roman numerals. (see [1, page 98] or [2, page 446] for a list of counter output commands).
- The command `\thesubfigure` by default is `(\alph{subfigure})` which produces (a), (b), etc.
- The command `\@thesubfigure` by default is `\thesubfigure\space` which adds a space between the caption label and the caption.
- The command `\p@subfigure` by default is `\thefigure`

These commands make the default caption labels (a), (b), etc. and the default `\ref` labels 12(a), 12(b), etc. See [10] for controlling the size and font of the subfigure labels.

Subfigure Examples

1. To make the caption labels (i), (ii), etc. and make the `\ref` labels 12i, 12ii, etc. enter the following commands (preferably in the LaTeX file's preamble)

```
\renewcommand{\thesubfigure}{\roman{subfigure}}
\makeatletter
  \renewcommand{\@thesubfigure}{(\thesubfigure)\space}
  \renewcommand{\p@subfigure}{\thefigure}
\makeatother
```

   The `\makeatletter` command tells LaTeX to treat the `@` character as a letter, thus allowing the re-definitions of the inernal commands. The `\makeatother` command tells LaTeX return to the normal behavior of treating the `@` character as a non-letter.

2. To make the caption labels 12.1:, 12.2:, etc. and make the `\ref` labels 12.1, 12.2, etc. enter the following commands

```
\renewcommand{\thesubfigure}{\thefigure.\arabic{subfigure}}
\makeatletter
  \renewcommand{\@thesubfigure}{\thesubfigure:\space}
  \renewcommand{\p@subfigure}{}
\makeatother
```

12.3.3 Adding Subfigures to List of Figures

By default, the List of Figures generated by the `\listoffigures` command includes only figures, *not* subfigures. To add the subfigures the List of Figures, type

```
\setcounter{lofdepth}{2}
```

before the `\listoffigures` command.

Note that a change in LaTeX has caused the current *3/95* subfigure package to add "numberline1" at the beginning of any subfigure entry in the List of Figures. To fix this, include the following code in the preamble of your document.

```
\makeatletter
\renewcommand{\@subcaption}[2]{%
   \begingroup
     \let\label\@gobble
     \def\protect{\string\string\string}%
     \xdef\@subfigcaptionlist{%
       \@subfigcaptionlist,%
       {\numberline {\@currentlabel}%
     \noexpand{\ignorespaces #2}}}%
   \endgroup
\@nameuse{@make#1caption}{\@nameuse{@the#1}}{#2}}
\makeatother
```

13 Boxed Figures

The term *Boxed Figure* usually refers to one of two situations

- A box surrounds the figure's graphic but not the figure's caption.
- A box surrounds the figure's graphic and its caption.

The basic method for boxing an item is to simply place the item inside an `\fbox` command, which surrounds the object with a rectangular box. The fancybox package provides boxes of different styles.

13.1 Box Around Graphic

Placing an `\fbox` command around the `\includegraphics` command produces a box around the included graphic. For example, the commands

```
\begin{figure}
   \centering
   \fbox{\includegraphics[totalheight=2in]{file.eps}}
   \caption{Box Around Graphic, But Not Around Caption}
```

```
    \label{fig:boxed_graphic}
  \end{figure}
```

place a box around the included figure, as shown in Figure 27.

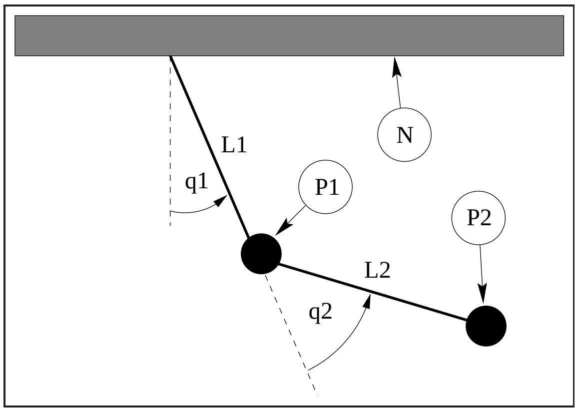


Figure 27: Box Around Graphic, But Not Around Caption

13.2 Box Around Figure and Caption

To include both the figure's graphic and its caption, one may be tempted to move the `\caption` command inside the `\fbox` command. However, this does not work because `\caption` can only be used in paragraph mode, while the contents of an `\fbox` command are processed in LR mode. (LaTeX uses three modes: LR mode, paragraph mode, and math mode. See [1, pages 36,103-5] for an explanation.)

Since the contents of minipage environments and `\parbox` commands are processed in paragraph mode, the `\caption` command can be included in the `\fbox` by enclosing the `\fbox` contents inside a minipage environment or a `\parbox` command. Since both minipages and parboxes require a width specification, there is no direct way to make the `\fbox` exactly as wide the graphic and caption.

For example, the commands

```
\begin{figure}
   \centering
   \fbox{ \begin{minipage}{4 in}
          \centering
          \includegraphics[totalheight=2in]{pend.eps}
          \caption{Box Around Figure Graphic and Caption}
          \label{fig:boxed_figure}
   \end{minipage} }
\end{figure}
```

place a box around the figure's graphic and caption, as shown in Figure 28
The determination of a proper minipage width is usually a trial-and-error process. If the caption is wider than the graphic, the minipage can be made as wide as the caption by estimating the caption width with a `\settowidth` command

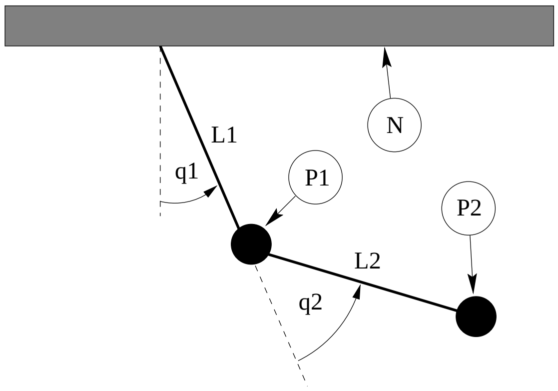


Figure 28: Box Around Figure Graphic and Caption

```
\begin{figure}
  \centering
  \newlength{\mylength}
  \settowidth{\mylength}{Figure XX:  Box Around Figure Graphic and Caption}
  \fbox{ \begin{minipage}{\mylength}
         \centering
         \includegraphics[totalheight=2in]{pend.eps}
         \caption{Box Around Figure Graphic and Caption}
         \label{fig:boxed_figure_length}
  \end{minipage} }
\end{figure}
```

13.3 Customizing fbox Parameters

In Figures 27 and 28, the box is constructed of 0.4 pt thick lines with a 3 pt space between the box and the graphic. These two dimensions can be customized by setting the LaTeX length variables `\fboxrule` and `\fboxsep`, respectively, with the `\setlength` command. For example, the commands

```
\begin{figure}
   \centering
   \setlength{\fboxrule}{3pt}
   \setlength{\fboxsep}{1cm}
   \fbox{\includegraphics[totalheight=2in]{pend.eps}}
   \caption{Graphic with Customized Box}
   \label{fig:boxed_custom}
\end{figure}
```

place a box with 3 pt thick lines which is separated from the graphic by 1 centimeter, as shown in Figure 29

13.4 The Fancybox Package

In Figures 27, 28, and 29, the `\fbox` command was used to place standard rectangular boxes around the figures. The `fancybox` package provides four commands `\shadowbox`, `\doublebox`, `\ovalbox`, and `\Ovalbox` which produce other types of boxes.

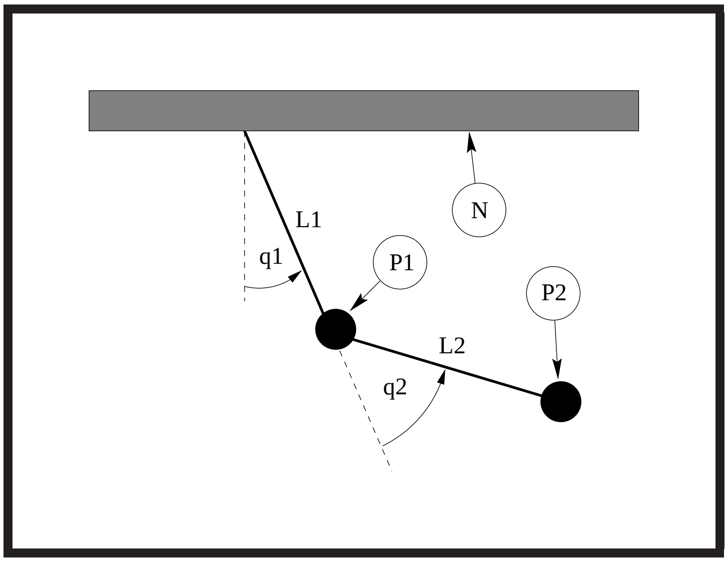


Figure 29: Graphic with Customized Box

Table 6: FancyBox Commands

Command	Parameters
`\shadowbox{Example}` **Example**	The frame thickness is `\fboxrule`. The shadow thickness is `\shadowsize` (which defaults to 4 pt).
`\doublebox{Example}` **Example**	The inner frame thickness is .75`\fboxrule` and the outer frame thickness is 1.5`\fboxrule`. The spacing between the frames is 1.5`\fboxrule` + 0.5pt.
`\ovalbox{Example}` **Example**	The frame thickness is `\thinlines`. Entering `\cornersize{x}` makes the diameter of the corners `x` times the minimum of the width and the height. The default is `\cornersize{0.5}`. The `\cornersize*` command directly sets the corner diameter. For example, `\cornersize*{1cm}` makes the corner diameters 1 cm.
`\Ovalbox{Example}` **Example**	`Ovalbox` is exactly the same as `ovalbox` except that the line thickness is controlled by `\thicklines`.

Like `\fbox`, the separation between these boxes and their contents is controlled by the LaTeX length `\fboxsep`. The length `\shadowsize` is set with the `\setlength` command, as

was done for \fboxrule and \fboxsep in section 13.3. The lines for \ovalbox and \Ovalbox have thicknesses corresponding to the picture environment's \thickline and \thinline, which are *not* lengths and thus cannot be changed with the \setlength command. The values of \thickline and \thinline depend on the size and style of the current font. Typical values are 0.8 pt for \thickline and 0.4 pt for \thinline.
For example, the commands

```
\begin{figure}
    \centering
    \shadowbox{ \begin{minipage}{3.5 in}
        \centering
        \includegraphics[totalheight=2in]{pend.eps}
        \caption{Shadowbox Around Entire Figure}
        \label{fig:boxed_fancy}
    \end{minipage} }
\end{figure}
```

place a shadow box around the figure's graphic and caption, as shown in Figure 30.

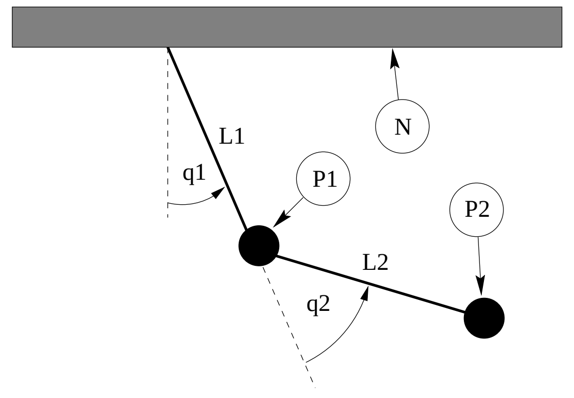


Figure 30: Shadowbox Around Entire Figure

14 Customizing Captions

14.1 Captions Next to Figures

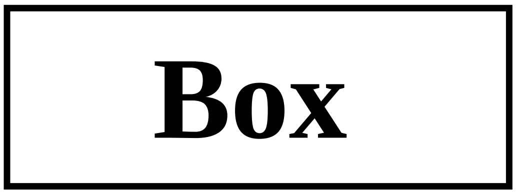
The \caption command places the caption under the figure or table. Minipage environments can be used to trick the caption command into placing the caption next to the figure. For example, the commands

```
\begin{figure}
   \centering
   \begin{minipage}[c]{.45\textwidth}
      \centering
      \caption{Caption on the Side}
      \label{fig:side:caption}
  \end{minipage}%
  \begin{minipage}[c]{.45\textwidth}
```

```
      \centering
      \includegraphics[width=\textwidth]{box.eps}
    \end{minipage}
  \end{figure}
```

produces Figure 31. Likewise, the caption can be placed to the right of the figure by changing the order of the minipages.

Figure 31: Caption on the Side



Since the figure environment defined by the float package places the caption *below* the body, Figure 31 cannot be produced with the float package's figure environment. Other aspects of the float package can be used as long as the `\restylefloat{figure}` command is not issued.

14.2 Caption Package

Since the format of LaTeX figure and table captions (especially for multi-line captions) may not be exactly what users desire, the caption package was written by Harald Axel Sommerfeldt to add flexibility to the caption formatting. Since the original caption package had some bad side-effects (particularly the requirement that it be loaded *after* other packages) it was totally re-written and renamed caption2. Although the caption2 is technically still a beta version, it is quite stable and performs well.

The caption2 package can be used with many types of floats as it officially supports the float, longtable, and subfigure packages and it also works with the floatfig, rotating, supertabular, and wrapfig packages.

Reference [12] describes the commands for the original caption package, while the caption2 reference [13] currently includes only minimal documentation. The `test2.tex` test file demonstrates many of the caption2 capabilities.

Syntax: `\usepackage[options]{caption2}`

Where the options are described in Table 7.

14.2.1 Caption Styles

The caption2 package defines the following caption styles

normal Full lines are justified (aligned with both left and right margins) with the last line being left-justified.

center All lines of the caption are centered.

flushleft All lines of the caption are left-justified, leaving the right side ragged.

flushright All lines of the caption are right-justified, leaving the left side ragged.

Table 7: `caption2` Options

Caption Style	`normal`, `center`, `flushleft`, `flushright`, `centerlast`, `hang`, `indent`	Selects the caption style (see section 14.2.1).
Caption Fontsize	`scriptsize`, `footnotesize`, `small`, `normalsize`, `large`, `Large`	Select the fontsize for the caption label (e.g., "Figure 12:") and the caption text.
Caption Label Font Shape	`up`, `it`, `sl`, `sc`	Makes the caption label (e.g., "Figure 12:") have upright, italic, slanted, or small caps shape, respectively. Does not affect caption text.
Caption Label Font Series	`md`, `bf`	Makes the caption label (e.g., "Figure 12:") have a medium or boldface series font, respectively. Does not affect caption text.
Caption Label Font Family	`rm`, `sf`, `tt`	Makes the caption label (e.g., "Figure 12:") have roman, sans serif, or typewriter font, respectively. Does not affect caption text.
One-Line Caption Formatting	`oneline`, `nooneline`	Controls the formatting for one-line captions (see section 14.2.3)

centerlast All the lines are justified with the last line being centered.

indent Same as "normal" style except that the second and subsequent lines are indented by the length `\captionindent`. Since `\captionindent` is zero by default, a command such as `\setlength{\captionindent}{1cm}` must be used to set the indentation.

hang Same as "normal" style except that the second and subsequent lines are indented by the width of the caption label (e.g., "Figure 12:").

Usually these styles are specified as `\usepackage` options such as

```
\usepackage[centerlast]{caption2}
```

which makes all the captions in the document have `centerlast` style. Examples of the caption styles are shown in Figures 32-38.

14.2.2 Changing the Caption Style

The `\captionstyle` command changes the caption style. Placing the `\captionstyle` command inside an environment changes only those captions in that environment. For example, the commands

```
\begin{figure}
   \captionstyle{centerlast}
   \centering \includegraphics[width=3in]{box.eps}
```

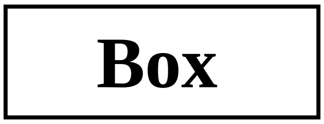


Figure 32: Normal Caption Style. Normal Caption Style.

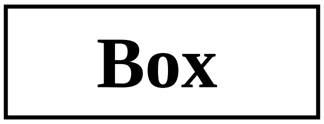


Figure 33: Center Caption Style. Center Caption Style.

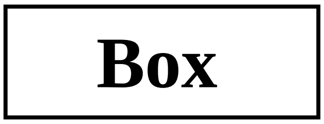


Figure 34: Centerlast Caption Style. Centerlast Caption Style.

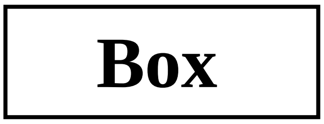


Figure 35: Flushleft Caption Style. Flushleft Caption Style.

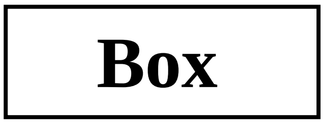


Figure 36: Flushright Caption Style. Flushright Caption Style.

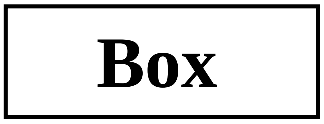


Figure 37: Indent Caption Style. Indent Caption Style.

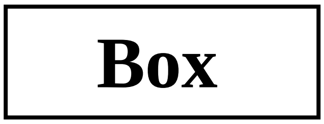


Figure 38: Hang Caption Style. Hang Caption Style.

```
   \caption{Centerlast Caption Style.  Centerlast Caption Style.}
\end{figure}
```

give only the current figure a `centerlast` style because `\captionstyle` is inside the figure environment. The commands

```
\captionstyle{centerlast}
\begin{figure}
   \centering \includegraphics[width=3in]{box.eps}
   \caption{Centerlast Caption Style.  Centerlast Caption Style.}
\end{figure}
```

give subsequent figures a `centerlast` style because `\captionstyle` is outside the figure environment.

14.2.3 One-Line Captions

If the caption is only one line, all of the above styles center the caption. To force the styles to be enforced even for one-line captions, one must include `nooneline` option

```
\usepackage[nooneline,flushleft]{caption2}
```

This formats *all* captions (including one-line captions) with the `flushleft` style. To change the `nooneline` option inside the document, `\onelinecaptionstrue` centers one-line captions while `\onelinecaptionsfalse` formats one-line captions. For example, the commands

```
\begin{figure}
   \captionstyle{flushleft}
   \onelinecaptionstrue
   \centering
   \begin{minipage}[c]{2.5in}
      \includegraphics[width=\textwidth]{box.eps}
      \caption{First Caption}
   \end{minipage}
\end{figure}
```

center one-line captions as shown in Figure 39.

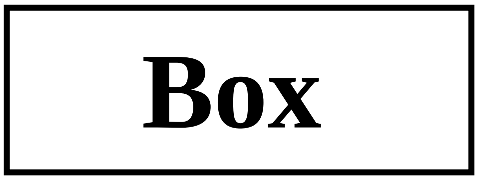


Figure 39: First Caption

The commands

```
\begin{figure}
   \captionstyle{flushleft}
   \onelinecaptionsfalse
   \centering
   \begin{minipage}[c]{2.5in}
      \includegraphics[width=\textwidth]{box.eps}
      \caption{Second Caption}
   \end{minipage}
\end{figure}
```

causes one-line captions to be left-justified as shown in Figure 40

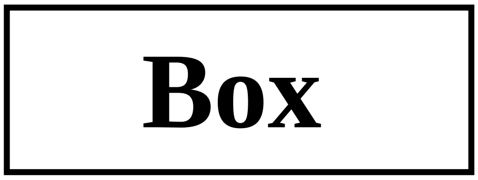


Figure 40: Second Caption

14.2.4 Linebreaks in Captions

When the caption fits in one line, it is processed in an `hbox`, which ignores any `\\` or `\par` . Thus one cannot generally not specify linebreaks in captions. However, the caption2 package provides the `\onelinecaptionsfalse` command (or `nooneline` option) to turn off this behavior. For example, the commands

```
\begin{figure}
   \centering
```

```
    \includegraphics[width=3in]{box.eps}
    \captionstyle{center}
    \onelinecaptionsfalse
    \caption{First Line of Caption \protect\\ Second Line of Caption}
    \label{fig:caption:linebreak}
  \end{figure}
```

produces the caption in Figure 41 Since `\\` is fragile, it must be preceeded by `\protect`.

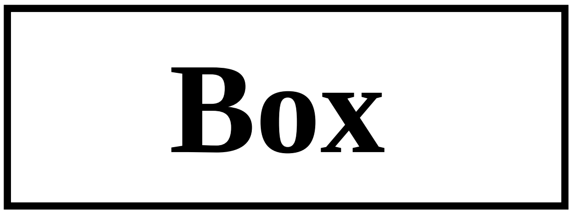


Figure 41: First Line of Caption
Second Line of Caption

14.2.5 Caption Widths

The caption2 package provides functions which directly specify the captions' width/margins.

- `\setcaptionwidth{width}` sets the width of the caption to `width`, where `width` can be in any valid TEX units.
- `\setcaptionmargin{mar}` sets the margins to `mar`, making the caption width be the standard width minus 2 times `mar`.

  If `mar` is negative, the caption is made wider than the standard width, which is useful in subfigures and minipage environments.

For example, the commands

```
  \begin{figure}
    \setcaptionwidth{3in}
    \centering
    \includegraphics[width=2in]{box.eps}
    \caption{Figure Caption Limited to Three Inches}
  \end{figure}
```

make the caption 3 inches wide, as shown in Figure 42.

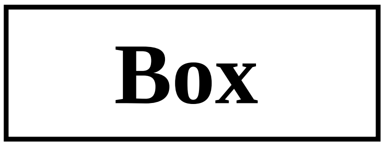


Figure 42: Figure Caption Limited to Three Inches

While the previous example directly set the width of the caption, alternatively the width can be indirectly set by specifying the spacing between the caption and each margin. For example, the commands

```
\begin{figure}
   \captionstyle{normal}
   \setcaptionmargin{1in}
   \centering
   \includegraphics[width=2in]{box.eps}
   \caption{Figure Caption Where There is One Inch of
            Spacing between the Caption and Each Margin}
\end{figure}
```

indent both sides of the caption one inches from the page margins, as shown in Figure 43.

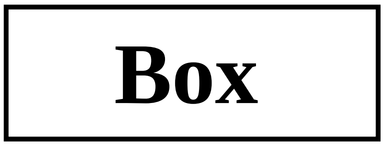


Figure 43: Figure Caption Where There is One Inch of Spacing between the Caption and Each Margin

14.2.6 Caption Font and Delimiter

While the `scriptsize,...,Large` options for `\usepackage{caption2}` change the size of both the caption label (e.g., "Figure 12:") and the caption text, the `up, it, sl, sc, md, bf, rm, sf, tt` options affect only the caption label.

Users can achieve more flexibility by redefining the `\captionfont` and `\captionlabelfont` commands. The caption is created by the following commands

```
{\captionfont%
    {\captionlabelfont \captionlabel \captionlabeldelim}%
    \captiontext}
```

where the `\captionlabel` command produces "Figure 12", the `\captionlabeldelim` command produces ":", and the `\captiontext` command produces the caption text. Thus `\captionfont` affects both the caption label and caption text, while `\captionlabelfont` affects only the caption label.

LaTeX fonts are described by size and three type style components: shape, series, and family ([1, pages 37,115], [2, pages 170-71]). All four of these characteristics can be specified in the `\captionfont` and `\captionlabelfont` commands. For example, the commands

```
\begin{figure}
   \renewcommand{\captionfont}{\Large \bfseries \sffamily}
   \renewcommand{\captionlabelfont}{}
   \centering
   \includegraphics[width=2in]{box.eps}
   \caption{Test Caption}
\end{figure}
```

produce Figure 44. In this example, the `\captionlabelfont` command does nothing. This means that it does not overwrite any font characteristics and all the `\captionfont` settings are carried over to the caption label. Since no shape declaration was specified, the entire caption has the default upright shape.
The commands

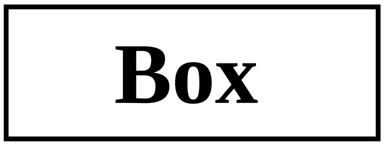


Figure 44: Test Caption

```
\begin{figure}
   \captionstyle{normal}
   \renewcommand{\captionfont}{\Large \bfseries \sffamily}
   \renewcommand{\captionlabelfont}{\small}
   \centering
   \includegraphics[width=2in]{box.eps}
   \caption{Test Caption}
\end{figure}
```

produce Figure 45. In this example, the `\small` font size in `\captionlabelfont` overwrites the `\Large` font size from `\captionfont`. However, since `\captionlabelfont` does not contain any series or family declarations, the `\bfseries` and `\sffamily` declarations carry over to the caption label.

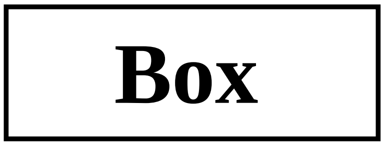


Figure 45: Test Caption

The default colon delimiter can be changed by redefining the `\captionlabeldelim` function. For example, the commands

```
\begin{figure}
   \captionstyle{normal}
   \renewcommand{\captionlabeldelim}{.\quad}
   \centering
   \includegraphics[width=2in]{box.eps}
   \caption{Caption with New Delimiter}
\end{figure}
```

change the delimiter in Figure 46 from the default colon to a period followed by a quad space.

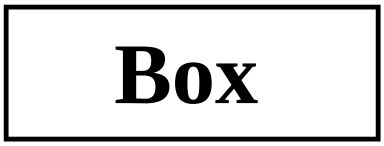


Figure 46.   Caption with New Delimiter

14.2.7 Custom Caption Styles

The caption2 package also allows users to create their own caption styles. For example, the following commands

```
\newcaptionstyle{mystyle}{%
  \usecaptionmargin\captionfont%
  {{\centering\bfseries\captionlabelfont\captionlabel\par}%
  \centering\captiontext\par}}

\begin{figure}
   \captionstyle{mystyle}
   \centering
   \includegraphics[width=2in]{box.eps}
   \caption{Customized Caption Style}
\end{figure}
```

makes the caption label boldface and places it on a separate line from the caption text, as shown in Figure 47.

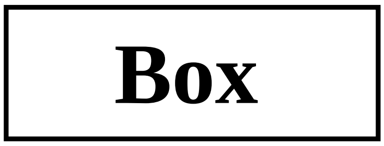


Figure 47
Customized Caption Style

See the caption2 test file [14] for more user-defined caption style examples.

References

[1] Leslie Lamport, *LaTeX: A Document Preparation System*, Addison-Wesley, Reading, Massachusetts, second edition, 1994, ISBN 0-201-52983-1

[2] Michel Goossens, Frank Mittelbach and Alexander Samarin, *The LaTeX Companion*, Addison-Wesley, Reading, Massachusetts, 1994, ISBN 0-201-54199-8

[3] Helmut Kopka and Patrick Daly, *A Guide to LaTeX 2ε*, Addison-Wesley, Reading, Massachusetts, 1995, ISBN 0-201-42777-X

[4] D. P. Carlisle, *Packages in the 'graphics' bundle*, Available from CTAN as `grfguide.tex` and `grfguide.ps`.

[5] D. P. Carlisle and S. P. Q. Rahtz, *The graphics package*, Available from CTAN as `graphics.dtx`

[6] D. P. Carlisle and S. P. Q. Rahtz, *The graphicx package*, Available from CTAN as `graphicx.dtx`

[7] Michael C. Grant and David Carlisle, *The PSfrag system, version 3*, Available from CTAN as `pfgguide.tex`

[8] Piet van Oostrum, *Page layout in LaTeX*, Available from CTAN as `fancyhdr.tex`

[9] Leonor Barroca, *The rotating package*, Available from CTAN as `rotating.dtx`

[10] Steven Douglas Cochran, *The subfigure package*, Available from CTAN as `subfigure.dtx`

[11] Timothy Van Zandt, *Documentation for fancybox.sty*, Available from CTAN as `fancybox.doc`

[12] Harald Axel Sommerfeldt, *The caption package*, Available from CTAN as `caption.dtx`

[13] Harald Axel Sommerfeldt, *The caption package*, Available from CTAN as `caption2.dtx`

[14] Harald Axel Sommerfeldt, *Test of the caption package*, Available from CTAN as `test2.tex`