Using EPS Graphics in LaTeX2ε Documents

Keith Reckdahl *

reckdahl@leland.stanford.edu

Version 1.8c
July 28, 1996

Abstract

This document explains how to use Encapsulated PostScript (EPS) files in LaTeX2ε documents. The graphics and graphicx packages provide commands which insert, scale, and rotate EPS graphics. The following EPS-inclusion topics are covered

- Compressed EPS files and non-EPS graphic formats (TIFF, GIF, JPEG, PICT, etc.) can also be inserted when `dvips` is used. Since neither LaTeX nor `dvips` has any built-in decompression or graphics-conversion capabilities, that software must be provided by the user.
- Since many applications which produce EPS files 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. A smaller final PostScript file results from defining a PostScript command for the graphics. An example of inserting an EPS graphic in the page header with the fancyhdr package is provided.

Various commands are often used in conjunction with EPS graphics. The following topics are covered in this document

- 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.

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 ps g were written for LaTeX2.09. In epsf, the graphics insertion was done by the `\epsfbox` command, while three other commands controlled graphic scaling. In ps g, 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`. The eps g 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.

The eps g package was updated to LaTeX2ε as a stop-gap measure while the LaTeX3 team addressed the general problem of inserting graphics in LaTeX2ε. The commands in the totally re-written "graphics bundle" are more efficient and more robust than those in other packages.

The graphics bundle contains the "standard" graphics package and the "extended" graphicx package. While both packages contain an `\includegraphics` command which includes graphics, the packages contain *different* versions of `\includegraphics`. The syntax of the graphicx `\includegraphics` is modeled after `\psfig`, while the syntax of the graphics `\includegraphics` is modeled after the `\epsfbox` command. The graphicx `\includegraphics` supports scaling and rotating, but the graphics `\includegraphics` command must be nested inside `\rotatebox` and/or `\scalebox` commands for scaling and/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. The syntax of the graphicx commands is described in section 5. The syntax of the graphics commands is described in section 6. For a more-detailed description of the packages, see David Carlisle's graphics bundle documentation [1].

For backward-compatibility, the graphics bundle also includes the `eps g` package which replaces the original LaTeX2ε `eps g` package. The new `eps g` package defines the `\epsfbox`, `\psfig`, and `\epsfig` commands as wrappers which simply call the `\includegraphics` command.

2 LaTeX Terminology

A *box* is any LaTeX object (characters, graphics, etc.) that is treated as a unit (see [4, 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.

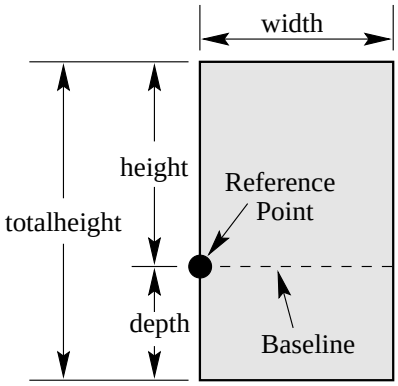


Figure 1: Sample LaTeX Box

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.

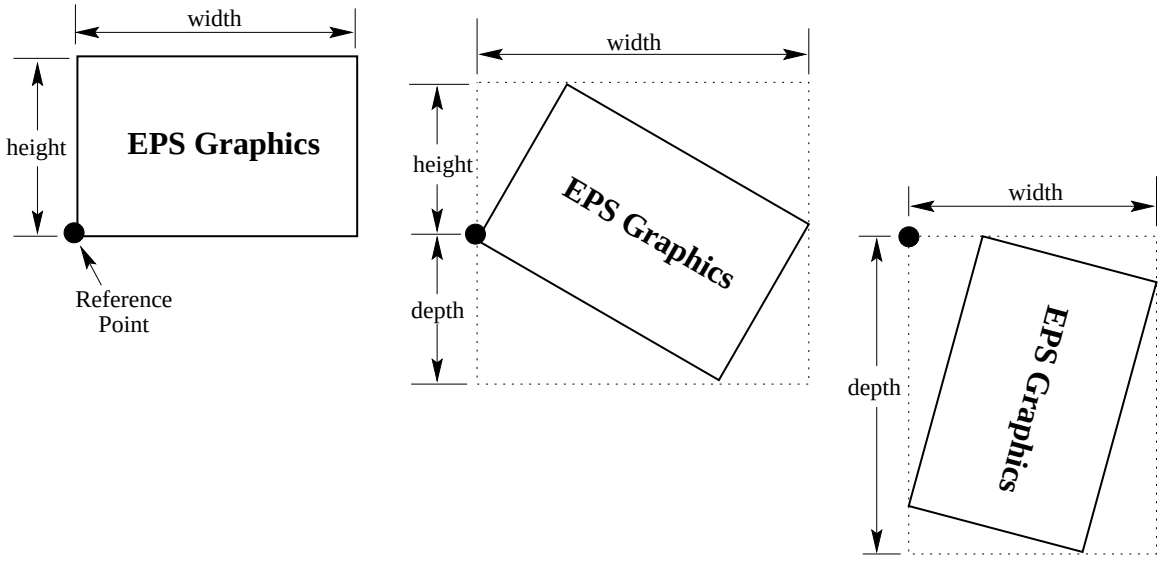


Figure 2: Rotated LaTeX Boxes

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 Post-

Script 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 `setpagedevice`, `letter`, or `a4` PostScript operators. 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 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
   1. 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.
   2. 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.
   3. 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 coordinates of the BoundingBox's lower-left corner. Likewise, measure the distances from the lower-left corner of the paper to the upper-right corner of the graphics to get the coordinates of the BoundingBox's upper-right corner.

4 Graphics in DVI Files

When LaTeX documents are compiled, the graphics-inclusion 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 insert the EPS graphics into the PostScript file. Therefore,

- the EPS graphics do not appear in most 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 [1]. The coverage of the graphicx package in the standard LaTeX references is sporadic: [6] covers both the graphics and graphicx packages, [4] only covers the graphics package and [5] 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 EPS 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. The following commands make the graphic width 2 inches less than the width of text

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

If the calc package is available, this is shortened to

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

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. The calc package with the 12/95 graphicx package shortens this further to

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

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

`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.

Table 1: `includegraphics` 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.

Table 2: `includegraphics` Cropping Options

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

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
```

`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.

Table 3: `includegraphics` Boolean Options

```
   \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 any LaTeX object: letter, paragraph, EPS graphic, etc. 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}`

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 LaTeX2ε 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 counter-clockwise 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`).

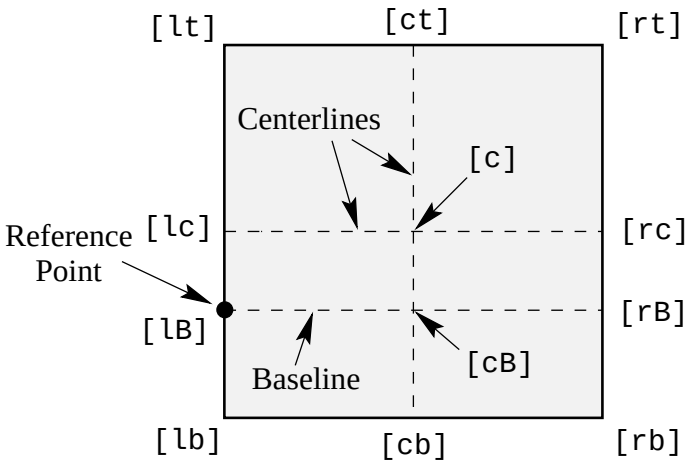


Figure 3: Available Origin Points

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
   `[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.

6 The graphics Version of includegraphics

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.

The graphics package contains two commands `\includegraphics` and `\includegraphics*` which are identical except that `\includegraphics*` clips (does not show) graphics outside the BoundingBox. The syntax for `\includegraphics` is

```
\includegraphics[llx,lly][urx,ury]{filename}
```

`[llx,lly]` are the *x* and *y* coordinates of the lower-left corner of the image. `[urx,ury]` are the *x* and *y* coordinates of the upper-right corner of the image. If no coordinates are given, the BoundingBox in the file is used. If only one set of coordinates is listed, it is assumed to be `[urx,ury]`, with `[llx,lly]` set to zero. The default units for the coordinates are `bp`, although any valid TeX units can be used.

The graphics package's `\rotatebox`, `\scalebox`, `\resizebox` commands are the same as the corresponding graphicx commands except the graphics version of `\rotatebox` does not allow any of the options which the graphicx version offers (see section 5.4).

The following commands use the graphicx version of `\includegraphics`

```
\documentclass{article}
\usepackage{graphicx}
\begin{document}
   %% include file1.eps with a width of 3 inches
   \includegraphics[width=3in]{file1.eps}
```

```
    %% include file2.eps with a width of 3 inches, then rotate 45 degrees
    \includegraphics[width=3in,angle=45]{file2.eps}

    %% include file3.eps, rotate 45 degrees, and resize to a width of 3 inches
    \includegraphics[angle=45,width=3in]{file3.eps}
\end{document}
```

The same output can be produced by the graphics version of `\includegraphics`

```
\documentclass{article}
\usepackage{graphics}
\begin{document}
   %% include file1.eps with a width of 3 inches
   \resizebox{3in}{!}{\includegraphics{file1.eps}}

   %% include file2.eps with a width of 3 inches, then rotate 45 degrees
   \rotatebox{45}{\resizebox{3in}{!}{\includegraphics{file2.eps}}}

   %% include file3.eps, rotate 45 degrees, and resize to a width of 3 inches
   \resizebox{3in}{!}{\rotatebox{45}{\includegraphics{file3.eps}}}
\end{document}
```

Part III

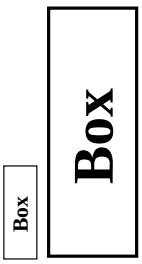
Importing EPS Graphics

7 Rotation and Scaling

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



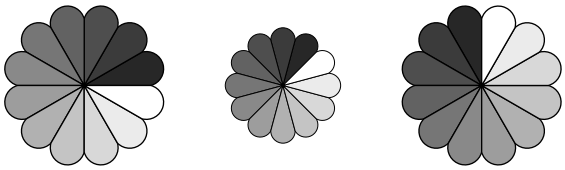
The first box is rotated 90 degrees and then scaled such that its height is a half inch. The second box is scaled such that its height is a half inch and then it is rotated 90 degrees.

7.1 Scaling of Rotated Graphics

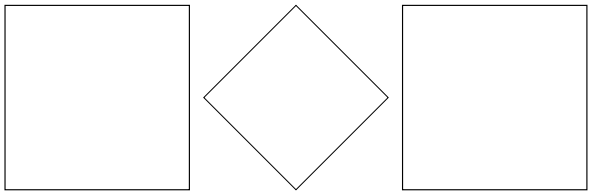
When the height or width of a graphic is specified, the specified size is not the size of the graphic but rather of its BoundingBox. This distinction is especially important in order to understand the scaling of rotated graphics. For example

```
\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



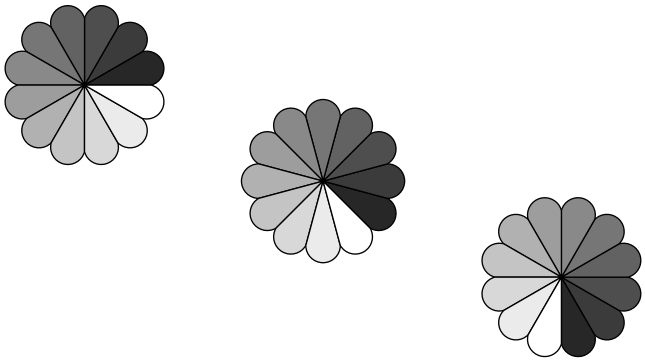
Each graphic is scaled such that its rotated BoundingBox is 1 inch tall.

7.2 Alignment of Rotated Graphics

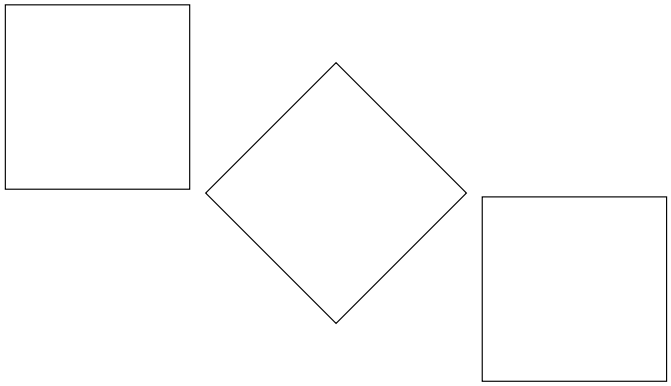
When graphics are rotated, the objects may not align properly. For example

```
\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



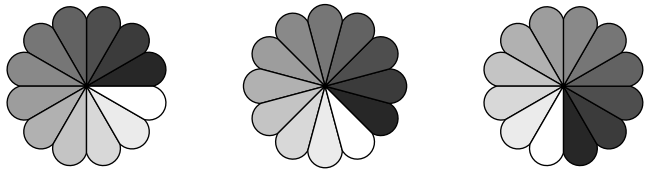
In this case, the objects' reference points (original lower-left corners) are aligned on a horizontal line. If it is desired to instead have the centers aligned, the `minipage` environment can be used

```
\begin{center}
   \begin{minipage}[c]{1.25in}
      \centering
      \includegraphics[totalheight=1in,angle=0]{rosette.eps}
   \end{minipage}
   \begin{minipage}[c]{1.25in}
      \centering
      \includegraphics[totalheight=1in,angle=-45]{rosette.eps}
   \end{minipage}
   \begin{minipage}[c]{1.25in}
      \centering
      \includegraphics[totalheight=1in,angle=-90]{rosette.eps}
   \end{minipage}
\end{center}
```

However, it is easier to use the `\rotatebox` command to rotate the graphic about its center

```
\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}
```

This aligns the centers of the graphics



If the 12/95 version of graphicx is used, the `origin` option can be used in `\includegraphics`

```
\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}
```

Similarly, the commands

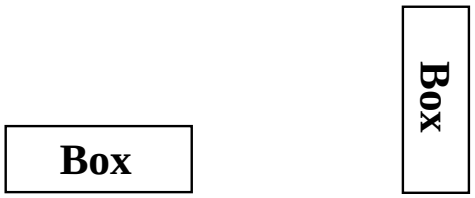
```
\begin{center}
    \includegraphics[width=1in]{box.eps}
    \hspace{1in}
    \includegraphics[width=1in,angle=-90]{box.eps}
\end{center}
```

rotate the right box 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}
    \rotatebox[origin=br]{-90}{\includegraphics[width=1in]{box.eps}}
 \end{center}
```

to rotate the right box around its lower-right corner



If the 12/95 version of graphicx is used, the `origin` option can be used in `\includegraphics`

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

8 Compressed and Non-EPS Graphics Files

When using `dvips`, users can specify an operation to be performed on the file before it is inserted. By making this operation a decompression command, compressed graphics files can be used. By making this a graphics-conversion command, 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. Without piping, the decompression or conversion cannot be done on-the-fly and the user must store all graphics as uncompressed EPS files.

8.1 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 8.2). 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 `eps g` package and is still available from CTAN.

8.2 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}`

`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.)

Table 4: `DeclareGraphicsRule` Arguments

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 8.5.2). The last command defines any other suffix as an EPS file.

8.3 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.

8.4 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. The `\DeclareGraphicsExtensions` command 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.

8.5 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. See

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

for information on ImageMagick, 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.sun.com/sunsoft/catlink/xv/note.html
```

for information on `xv`, an X-Windows graphics viewing and conversion program distributed from `ftp.cis.upenn.edu` as shareware. Unlike ImageMagick, `xv` does not provide command-line conversion capabilities for on-the-fly graphics conversion.

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
   1. If the Postscript file is EPS, save the BoundingBox line in `file2.gif.bb`
   2. 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.

8.5.1 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}
```

8.5.2 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, which prevents LaTeX from determining 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 using 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. Some commercial versions of LaTeX support non-EPS graphics
   1. Textures for the Macintosh supports PICT graphics.
   2. 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.

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

9 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. Currently available for both DOS and Unix, the PSfrag system consists of the LaTeX style file `psfrag.sty` and the `perl` script `ps2frag` and is well-documented by [7].

PSfrag currently does not support compressed or non-EPS graphics. This means that if PSfrag is used for even one graphic in a document, all of the document's graphics must be non-compressed EPS files.

The procedure for using `PSfrag`

1. Create an EPS file.
2. At the operating system prompt, type `ps2frag file.eps` which scans the EPS file `file.eps` for text strings and then records these locations in the EPS file. Since this added information is in the form of header comments in the EPS file, it does not change the appearance of the EPS output.

3. In the LaTeX document, use the following commands
   1. Include `\usepackage{psfrag}` in the preamble.
   2. Use the `\psfrag` command to specify the EPS text 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.

`PStext`	Text in EPS file to be replaced. PSfrag is sensitive about what type of text it replaces. For example, if the EPS file contains the text `Error (%)`, the percent sign confuses LaTeX and PSfrag *cannot* be used on the file, regardless of whether PSfrag replaces `Error (%)`. Instead, regenerate the EPS file using text such as Error (percent) which does not contain any of the LaTeX special characters.
`posn`	*(Optional, Defaults to `[Bl]`.)* Position of placement point relative to new LaTeX text. `[ ]` indicates center.
`PSposn`	*(Optional, Defaults to `[Bl]`.)* Position of placement point relative to existing EPS text. `[ ]` indicates center.
`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.)* 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 counterclockwise rotation being positive. This option is useful when dealing with EPS files generated by applications which only allow horizontal text. This option effectively adds rotated-text capabilities to those applications.
`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 %).

Table 5: `psfrag` Options

The `posn` and `PSposn` options are one of the 12 points shown in Figure 3 on page 79, except that the `c` specifier is not available (e.g., to align the left-center, use `[l]` instead of `[lc]`; to align centers, use `[ ]` instead of `[cc]`). See [7] for examples of various combinations of placement points.

9.1 PSfrag Example

The commands

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

include the graphic without any PSfrag replacement, producing Figure 4. 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}
```

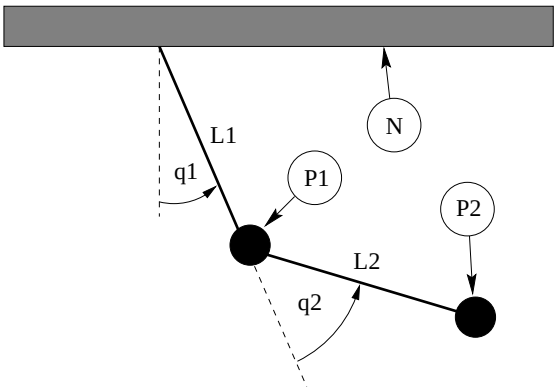


Figure 4: Without PSfrag Replacement

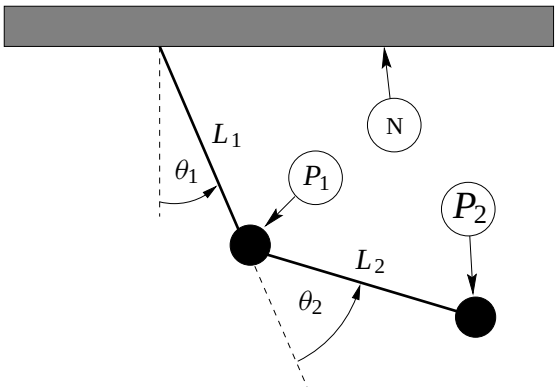


Figure 5: With PSfrag Replacement

include the graphic with PSfrag replacement, producing Figure 5. 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 position the new LaTeX text such that its center corresponds to the center of the EPS text.

Note that one need not replace all the EPS text with LaTeX text. For example, the `N` tag is left unchanged in Figure 5. 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.

9.2 LaTeX Text in EPS File

In the previous section, the `\psfrag` command specified the LaTeX text in the LaTeX file. While this is the most popular method, PSfrag's `\tex` command includes LaTeX text directly in EPS files. 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. The left-baseline point of x^2 is aligned with the left-baseline point of `\tex{$x^2$}`. Note that PSfrag does the replacement automatically; apart from the `\usepackage{psfrag}` command, it does not require any commands in the LaTeX file. Placement, scaling, and rotation options can be specified as with the `\psfrag` command. If an EPS file contains the text `\tex[][]{$x^2$}` PSfrag replaces it with a centered x^2. 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 advantage of the `\tex` command is that the LaTeX file doesn't need to be edited when an EPS file is modified. The `\tex` command has two disadvantages. First, the EPS file cannot be used for non-LaTeX purposes, while the EPS graphic in Figure 4 could be used without replacement. Second, if `\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 in LaTeX .

9.3 Text Scaling in PSfrag

A subtlety of the `\includegraphics` command (see [3, page 3]) comes into play with PSfrag. 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 [7, pages 10-11] for information.

10 Graphics in Page Header or Footer

The fancyhdr package (an improved version of the old fancyheadings package) makes it easy to customize a document's page headers and footers. It is often desired to place a logo or other EPS graphics in the header or footer, which results in the same EPS file being included multiple times.

10.1 Including An EPS File 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
   1. LaTeX must find and read the file every time `\includegraphics` is used.
   2. The EPS graphics commands are repeated in the final PS file, producing a large file.
3. 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.
4. Define a PostScript command which draws the graphics, and then issue the Postscript command wherever you want the graphic. Section 10.2 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.
5. 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.

10.2 Defining a PostScript Command

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.

10.3 fancyhdr Package

An easy method of including graphics in the heading is to use the fancyhdr 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 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 the command `\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 10.2, 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.

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

11 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}[htb]
   \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 (the `\label` command must be *after* the `\caption` command). 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).

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

produce Figure 6.

Figure 6: 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 6 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. The commands

```
\setlength{\abovecaptionskip}{5pt}
\setlength{\belowcaptionskip}{0.5cm}
```

provides a 5 point spacing above the caption and a 0.5 centimeter spacing below the caption. The commands

```
\addtolength{\abovecaptionskip}{5pt}
\addtolength{\belowcaptionskip}{-5pt}
```

increases the spacing above the captions by 5 points and decreases the spacing below the captions by 5 points. For example, the commands

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

produce Figure 7.

Figure 7: Caption Above Graphic

11.2 Figure Placement Options

LaTeX figures are "floats" whose placement are decided by LaTeX. Since your taste in figure-placement may differ from that of LaTeX, the `\figure` environment has placement options

- **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 in the above example are `[htb]` which means that LaTeX first tries to place the figure at that location, then tries to place the figure at the top of a page, and finally tries 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 has the final say in the placement and reserves the right to override the request. 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 overrides the `[!h]`, with the text which is after the figure in the .tex file filling the bottom 3 inches of the page.

If you feel LaTeX is making poor float placement decisions, you may need to tweak its placement algorithm by modifying the float parameters (see [4, pages 199-200], [5, pages 141-143], or [6, pages 174-175]).

11.2.1 The oat Package's `[H]` Placement Option

The oat 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}` in the preamble and issue the `\restylefloat{figure}` command *before* the `\begin{figure}[H]` command is used (See [5, 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 oat 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 6 or the construction of side-by-side and other complex figure arrangements.

12 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.

12.1 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.

12.2 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 8.

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.

12.3 Rotcaption Command

The methods in Sections 12.1 and 12.2 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 9. 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 12.1 and 12.2, the `\rotcaption` command does not rotate the graphics. Therefore, the `\includegraphics` command in the above example required the `angle=90` option.

13 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 12, Figure 13, etc.)
3. The side-by-side graphics each form a subfigure (e.g., Figure 12a, Figure 12b, etc.) of a single figure (Figure 12).

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 33-39 on Page 110.

13.1 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

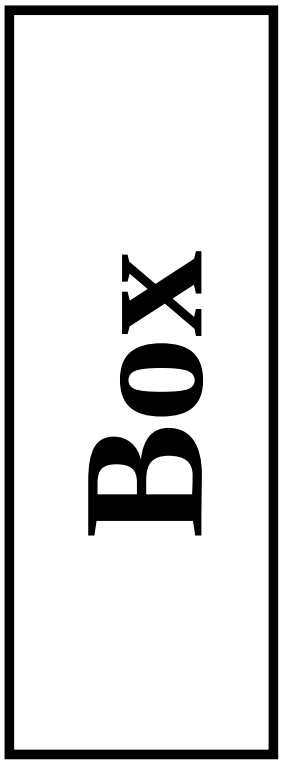


Figure 8: Sidewaysfigure Figure

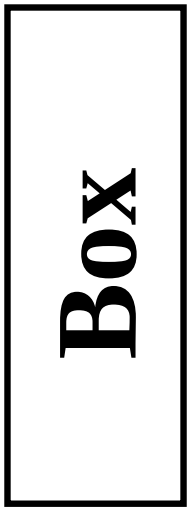


Figure 9: Rotcaption Caption

13.1.1 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 10 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.

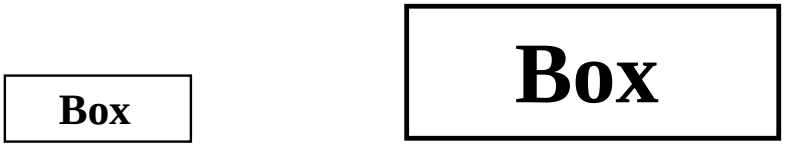


Figure 10: Two Graphics in One Figure

13.1.2 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}
```

Figure 11: Centers Aligned Vertically

produces Figure 11.

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 14 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 10, but the centers of the boxes are aligned vertically.

13.2 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 12 and 13.

Figure 12: Small Box

Figure 13: 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.

13.2.1 Alignment Problems with Side-by-Side Figures

The `[b]` options aligned the bottoms of Figures 12 and 13. 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 14, 15, and 16.

Figure 14: Small Box with a Long Caption

Figure 15: Medium Box

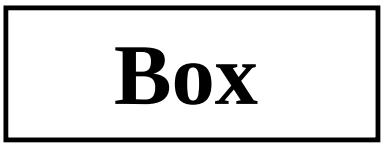


Figure 16: Big Box

The long caption of Figure 14 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 14 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 17, 18, and 19.

Figure 17: Small Box with a Long Caption

Figure 18: Medium Box

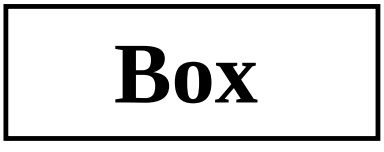


Figure 19: Big Box

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 17 caption, the captions for Figures 18 and 19 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).

13.3 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 20. The label `{fig:subfig:a}` refers to subfigure 20(a), the label `{fig:subfig:b}` refers to subfigure 20(b), and the label `{fig:subfig}` refers to to Figure 20.

(a) Small Box with a Long Caption

(b) Big Box

Figure 20: Two Subfigures

13.3.1 Subfigures Inside Minipage Environments

Like other side-by-side graphics, subfigures are often put inside `minipage` environments. For example

```
\begin{figure}
  \centering
  \begin{minipage}[b]{0.5\textwidth}
     \centering
     \subfigure[Small Box with a Long Caption]{
          \label{fig:subfig:mini:a}   %% label for first subfigure
          \includegraphics[width=1.0in]{box.eps}}
  \end{minipage}%
  \begin{minipage}[b]{0.5\textwidth}
     \centering
     \subfigure[Big Box]{
          \label{fig:subfig:mini:b}   %% label for second subfigure
          \includegraphics[width=1.5in]{box.eps}}
  \end{minipage}
  \caption{Subfigures Inside Minipages}
  \label{fig:subfig:mini}       %% label for entire figure
\end{figure}
```

produces Figure 21 which contains subfigures 21(a) and 21(b).

13.3.2 Minipage Environments Inside Subfigures

Since Subfigure 21(a) consists of only the `\includegraphics` command, the caption in subfigure 21(a) is only as wide as the included graphic. If the subfigure instead consists of the entire minipage, the caption is made as wide as the minipage. For example

(a) Small Box with a Long Caption

(b) Big Box

Figure 21: Subfigures Inside Minipages

```
\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 22. Note that the caption of subfigure 22(a) is considerably wider than that of subfigure 21(a).

(a) Small Box with a Long Caption

(b) Big Box

Figure 22: Minipages Inside Subfigures

13.3.3 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. The command `\roman{subfigure}` prints the value of the `subfigure` counter in lowercase Roman numerals. (see [4, page 98] or [5, 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` and `\makeatother` commands protect the `@` signs in the `\renewcommand` statements.
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
```

13.3.4 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.

14 Minipage Placement Option Details

The manner in which `minipage` environments are vertically aligned may be confusing. For example, one might think the commands

```
\begin{figure}
   \centering
   \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}
   \caption{\texttt{minipage} with \texttt{[b]} option}
\end{figure}
```

which use the minipage `[b]` options would align the bottoms of the graphics. Instead they produce Figure 23.

Figure 23: `minipage` with `[b]` option

Similarly, one might think the commands

```
\begin{figure}
   \centering
   \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}
   \caption{\texttt{minipage} with \texttt{[t]} option}
\end{figure}
```

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 23.

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.

14.1 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{figure}
   \centering
   \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}
   \caption{Minipages with Bottoms Aligned}
\end{figure}
```

produces Figure 24.

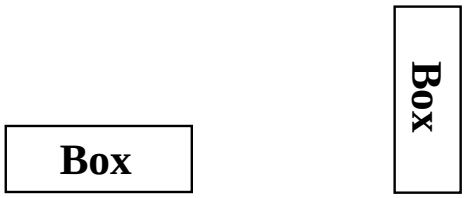


Figure 24: Minipages with Bottoms Aligned

14.2 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{figure}
   \centering
   \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}
   \caption{Minipages with Tops Aligned}
\end{figure}
```

produces Figure 25.

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 [5, pages 456-457].

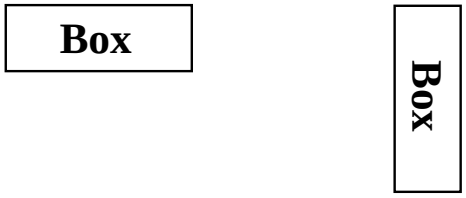


Figure 25: Minipages with Tops Aligned

15 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.

15.1 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 26.

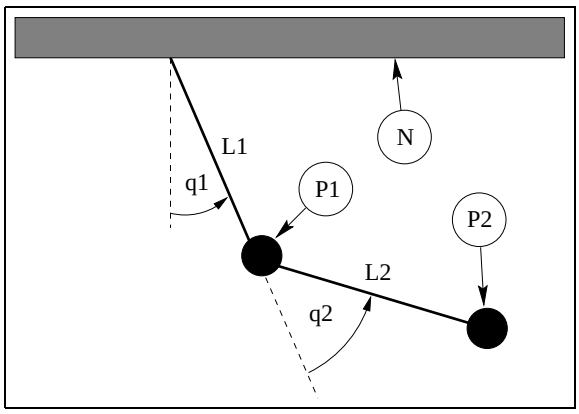


Figure 26: Box Around Graphic, But Not Around Caption

15.2 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 [4, 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 27

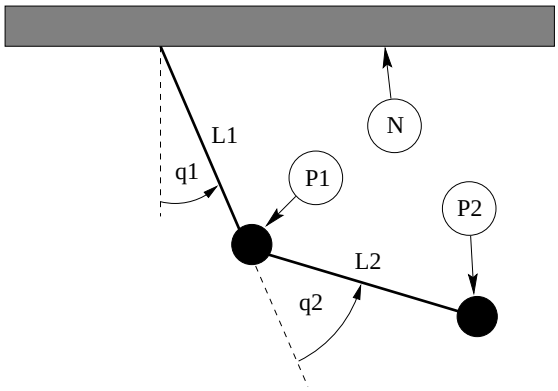


Figure 27: Box Around Figure Graphic and Caption

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

```
 \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}
```

15.3 Customizing fbox Parameters

In Figures 26 and 27, 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 28

15.4 The Fancybox Package

In Figures 26, 27, and 28, 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.

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 15.3. 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

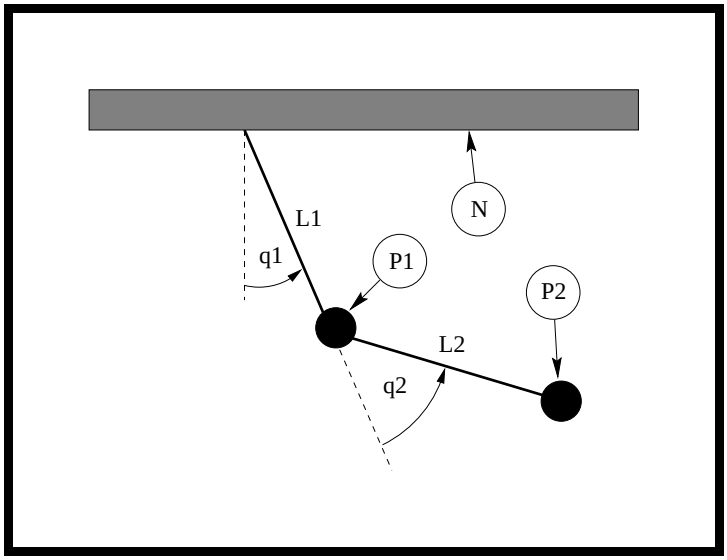


Figure 28: Graphic with Customized Box

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 corner diameter can be set directly by `\cornersize*` command. 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`.

Table 6: FancyBox 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 29.

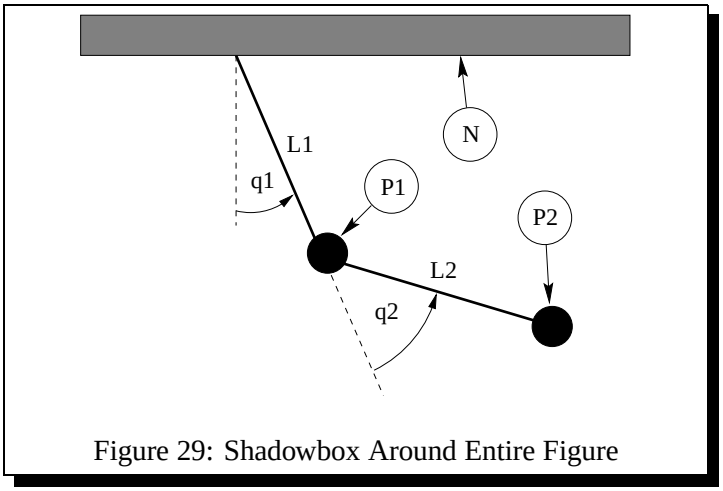


Figure 29: Shadowbox Around Entire Figure

16 Customizing Captions

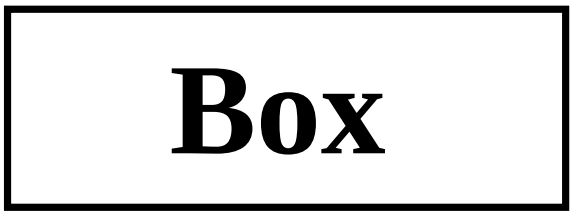
16.1 Captions Next to Figures

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]{3in}
      \centering
      \caption{Caption on the Side}
      \label{fig:side:caption}
  \end{minipage}
  \hfill
  \begin{minipage}[c]{3in}
      \centering
      \includegraphics[width=\textwidth]{box.eps}
  \end{minipage}
\end{figure}
```

produces Figure 30. Likewise, the caption can be placed to the right of the figure by changing the order of the minipages.

Figure 30: Caption on the Side



Since the figure environment defined by the oat package places the caption *below* the body, Figure 30 cannot be produced with the oat package's figure environment. Other aspects of the oat package can be used as long as the `\restylefloat{figure}` command is not issued.

16.2 Controlling Caption Width

Since placing the `\caption` command inside a minipage environment makes the caption as wide as the minipage, this can be used to control the caption width. For example, the commands

```
\begin{figure}
  \centering
  \includegraphics[width=2in]{box.eps}
  \caption{Graphic with a Very, Very, Very, Very, Very, Very Long Caption}
\end{figure}
```

produce the graphic in Figure 31.

Note that the caption in Figure 31 is as wide as the page text. The width of the caption can be limited by placing it inside a minipage environment. For example, the commands

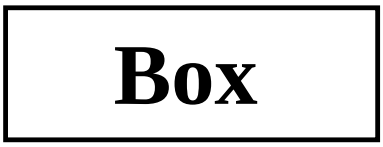


Figure 31: Graphic with a Very, Very, Very, Very, Very, Very Long Caption

```
\begin{figure}
\centering
\begin{minipage}{3in}
  \centering
  \includegraphics[width=2in]{box.eps}
  \caption{Graphic with a Very, Very, Very, Very, Very, Very Long Caption}
\end{minipage}
\end{figure}
```

produces the graphic in Figure 32. The minipage limits the width of the caption in Figure 32 to 3 inches.

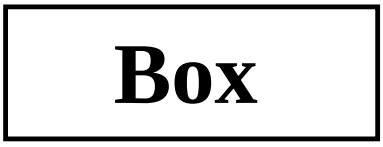


Figure 32: Graphic with a Very, Very, Very, Very, Very, Very Long Caption

A more-general approach to controlling caption width is provided by the caption package and is described in section 16.3.5.

16.3 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 oat, longtable, and sub gure packages and it also works with the oat g, rotating, supertabular, and wrap g 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.

16.3.1 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.

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

Caption Style	`normal, center, flushleft, flushright, centerlast, hang, indent`	Selects the caption style (see section 16.3.1).
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 16.3.3)

Table 7: `caption2` Options

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

which makes all the captions in the document have `centerlast` style. Examples of the caption styles are shown in Figures 33-39.

16.3.2 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}
   \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.

16.3.3 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

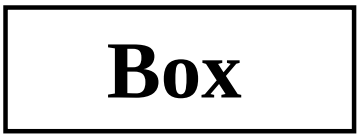


Figure 33: Normal Caption Style. Normal Caption Style.

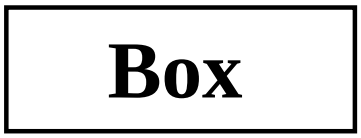


Figure 34: Center Caption Style. Center Caption Style.

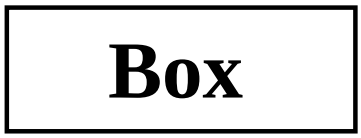


Figure 35: Centerlast Caption Style. Centerlast Caption Style.

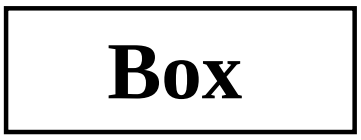


Figure 36: Flushleft Caption Style. Flushleft Caption Style.

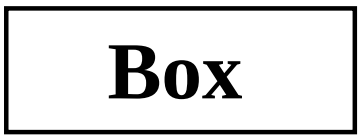


Figure 37: Flushright Caption Style. Flushright Caption Style.

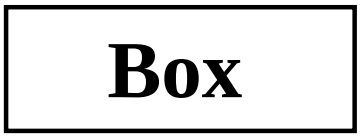


Figure 38: Indent Caption Style. Indent Caption Style.

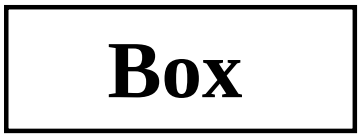


Figure 39: Hang Caption Style. Hang Caption Style.

```
\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 40.

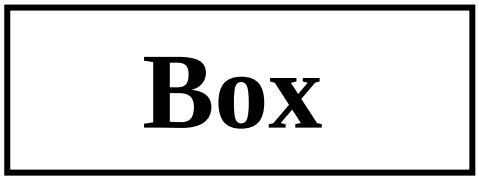


Figure 40: First Caption

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}
```

format one-line captions as shown in Figure 41

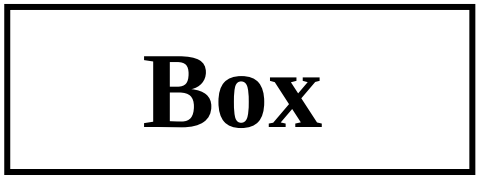


Figure 41: Second Caption

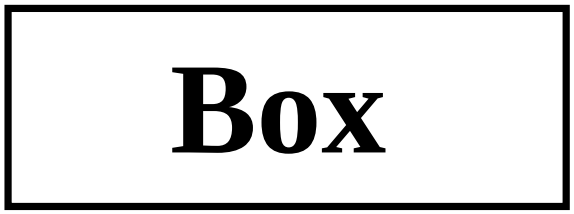


Figure 42: First Line of Caption
Second Line of Caption

16.3.4 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 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}[!ht]
   \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 42 Since `\\` is fragile, it must be preceeded by `\protect`.

16.3.5 Caption Widths

Section 16.2 demonstrated that a `\caption` command appearing in outer paragraph mode can become as wide as the page text as shown in Figure 31. Placing a `\caption` command in a minipage limits the width of the caption to the width of the minipage as shown in Figure 32. 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 43.

While the previous example directly set the width of the caption, alternatively the width can be indirectly set by specifying the caption's margin. For example, the commands

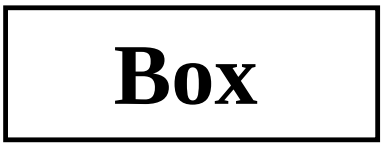


Figure 43: Figure Caption Limited to Three Inches

```
\begin{figure}
   \captionstyle{normal}
   \setcaptionmargin{2in}
   \centering
   \includegraphics[width=2in]{box.eps}
   \caption{Figure Caption With Two-Inch Margins on Each Side}
\end{figure}
```

indent both sides of the caption two inches from the page margins, as shown in Figure 44.

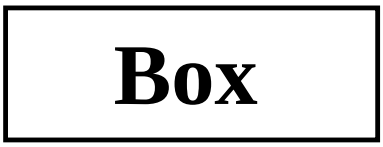


Figure 44: Figure Caption With Two-Inch Margins on Each Side

16.3.6 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 ([4, pages 37,115], [5, 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 45. 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

```
\begin{figure}
   \captionstyle{normal}
   \renewcommand{\captionfont}{\Large \bfseries \sffamily}
```

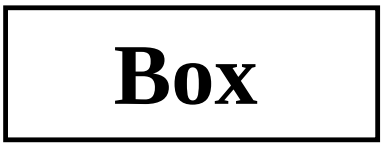


Figure 45: Test Caption

```
   \renewcommand{\captionlabelfont}{\small}
   \centering
   \includegraphics[width=2in]{box.eps}
   \caption{Test Caption}
\end{figure}
```

produce Figure 46. 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.

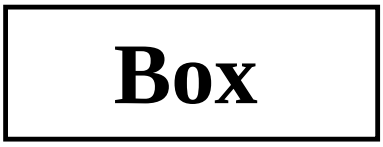


Figure 46: Test Caption

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 47 from the default colon to a period followed by a quad space.

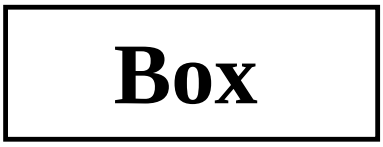


Figure 47. Caption with New Delimiter

16.3.7 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}
```

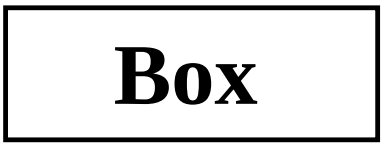


Figure 48
Customized Caption Style

makes the caption label boldface and places it on a separate line from the caption text, as shown in Figure 48.

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

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 provided a great deal of assistance. I would also like to acknowledge Jim Hafner for providing the procedure in section 10.2. Finally, I would like to thank the readers of previous versions who provided me with feedback.

References

[1] D. P. Carlisle, *Packages in the 'graphics' bundle*, Available from CTAN as `grfguide.tex` or `grfguide.ps`
[2] D. P. Carlisle and S. P. Q. Rahtz, *The graphics package*, Available from CTAN as `graphics.dtx`
[3] D. P. Carlisle and S. P. Q. Rahtz, *The graphicx package*, Available from CTAN as `graphicx.dtx`
[4] Leslie Lamport, *LaTeX: A Document Preparation System*, Addison-Wesley, Reading, Massachusetts, second edition, 1994, ISBN 0-201-52983-1
[5] Michel Goossens, Frank Mittelbach and Alexander Samarin, *The LaTeX Companion*, Addison-Wesley, Reading, Massachusetts, 1994, ISBN 0-201-54199-8
[6] Helmut Kopka and Patrick Daly, *A Guide to LaTeX2ε* , Addison-Wesley, Reading, Massachusetts, 1995, ISBN 0-201-42777-X
[7] Craig Barratt and Michael C. Grant, *The PSfrag system*, 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`