The PICTEX Manual
The PICTEX Manual
The PICTEX Manual
The PICTEX Manual
The PICTEX Manual
The PICTEX Manual
The PICTEX Manual
The PICTEX Manual
The PICTEX Manual
The PICTEX Manual
The PICTEX Manual

Michael J. Wichura

*The University of Chicago*

*First printing: November 1986*
*Second printing: September 1987*
*Third printing with minor corrections: March 1992*

This manual was written using TEX supplemented by the PICTEX macros. The features described herein exist in Version 1.1 of PICTEX.

The PICTEX project was carried out using computer facilities supported in part by National Science Foundation Grant No. DMS-8404941 to the Department of Statistics at the University of Chicago.

The author provides no guarantee as to the correctness of this manual and the associated software; the user accepts them as is.

## *PREFACE*

In the preface to *The TEXbook*, Knuth describes TEX as a "typesetting system intended for the creation of beautiful books—and especially for books that contain a lot of mathematics". PICTEX is a collection of TEX macros by means of which TEX users can easily instruct TEX to typeset beautiful pictures as a part of their books—and especially mathematical figures, such as the one below:

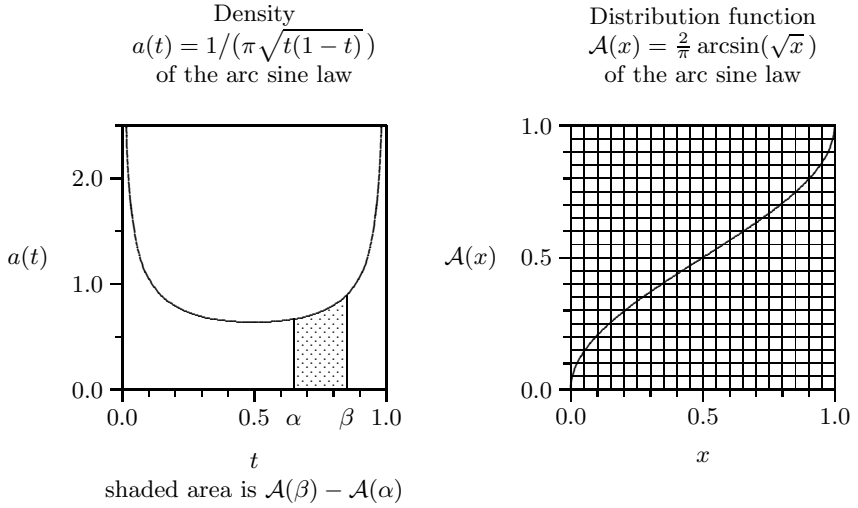



*FIGURE 1*

That figure is destined to appear in a book on probability theory that I'm writing using TEX. Indeed, the PICTEX macros are an outgrowth of the attempt I made at coercing TEX into drawing the kinds of figures I wanted to include in that book. Happily, it turned out that what I wanted to be able to do fell just within the limits of what seems feasible by way of making TEX function as a graphics device.

PICTEX offers these advantages: (1) Figures become an integral part of the typesetting process. You can avoid having to leave the proper amount of space in your document for material that has to be created on some external device and later stripped into the finished product. (2) All of TEX's formatting capabilities are available for annotating your figures. In addition, that annotation will be done (if you so desire) in the same fonts as you're using in the rest of your document. (3) Just as TEX is machine independent, so too is PICTEX. It doesn't matter whether you're working on a PC or mainframe computer. (4) Since typeset figures are embedded in the `dvi` file along with the

rest of your document, all the advantages of TEX's device independent output accrue to them. In particular, you can revise away to your heart's content on your local system until things are just the way you want them, and then you can have the final copy elegantly (but perhaps expensively) printed on a high resolution output device. (5) PICTEX can easily be extended using TEX's macro facilities.

On the other hand, PICTEX has several limitations: (1) PICTEX was expressly designed to facilitate the construction of pictures such as Figure 1. It simply is not the right tool for producing illustrations such as the lions that grace the title pages of *The TEXbook*. (2) Within the realm of mathematical figures, PICTEX doesn't make 3D pictures or other complex things. Considering that TEX provides less arithmetic capabilities than the simplest pocket calculator, that would be asking for too much. (3) PICTEX takes a while to draw a PICture. Figure 1 initially took $3\frac{1}{2}$ minutes on a Sun-2/120, all but 30 seconds of the effort going into producing the two curves. In subsequent drafts PICTEX replotted the curves in 40 seconds. (4) PICtures take up a sizeable amount of computer memory, both within TEX and within the software that processes the `dvi` file. For example, Figure 1 occupies about 15 Kilobytes. A larger PICture, with several more curves, could easily exceed the constraints of a small system.

Examples and exercises are the life blood of any instruction manual, so this manual has lots of them (maybe too many). Answers to all the exercises are given in Appendix A. Appendix B provides some extended examples by giving the code that was used to produce various figures throughout the manual. In particular, the code for Figure 1 is there; leave perusal of that for last, since it's the most complicated.

The word 'point' presented some difficulty in early drafts of the manual. Frequently it meant 'coordinate point', while at other times it meant a printer's point (1/72.27 inch, by TEX's convention). I chose to resolve the ambiguity by consistently shortening the typographical term to 'pt'.

From time to time, parts of this manual are set in smaller type, like this. Like the dangerous bend sections of *The TEXbook*, such passages pertain to fine points that are best skipped over on first reading.

In closing, I would like to thank Ron Thisted for much advice and encouragement; Diana Wilson for enlarging the input buffer of our `iptex` program to 48K so the pages of Section 5 could be printed; Leslie Lamport for his advice on Section 10; and, of course, Donald Knuth for making it all possible, and for the seminal Dirty Trick on page 389 of *The TEXbook*.

*Michael J. Wichura*

Chicago, Illinois
November, 1986

*CONTENTS*

# 1. INTRODUCTION

## 1.1. PICTEX COMMANDS

To draw a PICture you first have to load the PICTEX macros into TEX's memory—see your local system guru for details. To start drawing a PICture you type the command `\beginpicture`, and to finish it off you type the command `\endpicture`. The overall structure is thus

```
\beginpicture
  additional PICTEX commands
\endpicture
```

All this goes in your TEX input file.

The PICTEX commands are described in detail in the following pages. A few words need to be said here about the syntax with which sample commands are presented. Consider, e.g., the `\put` command of Subsection 2.1:

`\put {`*text*`}` $\big[$`[`$o_x$`][`$o_y$`]`$\big]$ $\big[$`<`*xshift*`,`*yshift*`>`$\big]$ `at` *xcoord* *ycoord*

First of all, notice the matched pairs of thin brackets: $\big[\,\big]$ and [ ]. In contrast to the thick brackets '`[`' and '`]`', these are <u>not</u> part of the command; rather they indicate that the phrases contained therein may be omitted. Secondly, note the blank spaces separating the various phrases of the command. These are <u>essential</u>. However to enhance the readability of your input, you can use as many spaces as you like (provided there is at least one) wherever a sample command shows a single space. Moreover, at least one blank <u>must</u> follow every PICTEX command. Do try hard to get the syntax of the PICTEX commands right, paying particular attention to the delimiters `{`, `}`, `[`, `]`, `<`, `>`, `(`, `)`, and `/`. If you mess up, TEX will get confused and you will get a lot of error messages—and quite possibly no PICture. Appendix C lists all the PICTEX commands alphabetically, so you can easily check the syntax of any command you write.

**Exercise 1.** B. L. User didn't pay attention to the three words that were underlined in the preceding paragraph. What price did he pay in consequence?

It is important that you understand TEX's concept of grouping (see Chapter 5 of *The TEXbook*). The point is that any change you make to one of PICTEX's parameters is local to the group in which that change is made. In particular, since `\beginpicture` and `\endpicture` mark the start and end of a group, TEX will undo the changes you've made while drawing a PICture once it reaches the terminating `\endpicture` command. Consider, e.g., the effect of PICTEX's `\setcoordinatesystem` command, which is discussed in the next subsection. There is a major difference between typing

```
\beginpicture %  PiCture A
  \setcoordinatesystem units <1in,1in>
    ⋮
\endpicture
```

and typing

```
\setcoordinatesystem units <1in,1in>
\beginpicture % PiCture B
  ⋮
\endpicture
```

In the first case, the specified coordinate system is in effect during the fabrication of PICture A, upon the completion of which the coordinate system reverts to the one in effect before PICture A was begun. On the other hand, in the second case the specified coordinate system is in effect not only during PICture B but for all subsequent PICtures, until such time as it is overridden by another `\setcoordinatesystem` command or the enclosing group ends.

The PICTEX macros include the control sequence `\PiCTeX` which produces the logo 'PICTEX', and also the control sequence `\PiC` which produces the fragment 'PIC'. Type, e.g., '`\PiC\-ture`' to get 'PICture', and '`\PiC\-to\-ri\-al`' to get 'PICtorial', with discretionary breaks as indicated.

The names of PICTEX's internal control sequences begin with an exclamation point of category code 11. Since in ordinary usage '`!`' is of category 12, you can't inadvertently redefine any internal control sequence. All of PICTEX's external control sequences (which you can redefine, inadvertently or not) are listed in Appendix C.

### *1.2. COORDINATE SYSTEMS*

Objects are positioned in a PICture by specifying their $x$ and $y$ coordinates with respect to a coordinate system set up by the `\setcoordinatesystem` command, which takes the form

`\setcoordinatesystem units <`*xunit*`,`*yunit*`> point at` *xcoord ycoord*

Here *xunit* is the length of one unit on the $x$-axis, *yunit* the length of one unit on the $y$-axis. *xcoord* and *ycoord* are the $x$ and $y$ coordinates of the so-called *reference point* of the system. Reference points have several uses, one of which will be brought up in a moment. For example, the command

```
\setcoordinatesystem  units <.5in,.25in>  point at 1.5 -2
```

establishes the coordinate system shown in Figure 2 below.

In drawing a PICture, you may (re)set the coordinate system as often as you like. This feature is advantageous if, e.g., you want to make two side-by-side graphs, using a different coordinate system for each graph. Now the significance of reference points emerges—PICTEX links multiple coordinate

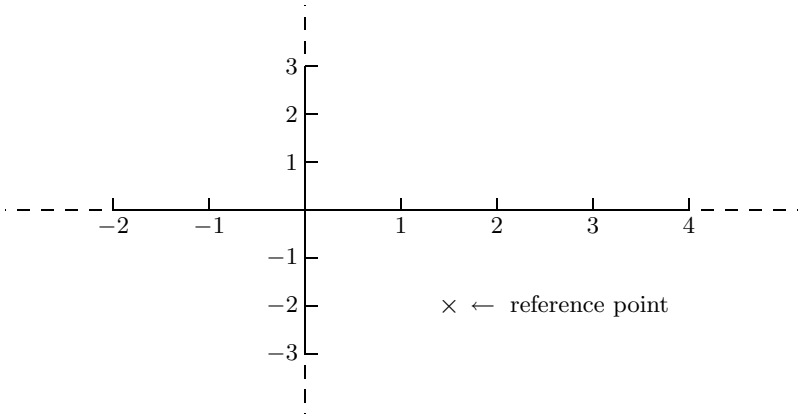



*FIGURE 2*

systems together in such a way that their reference points occupy the same physical location. If each coordinate system were to be drawn in the manner of Figure 2, the ×'s would coincide.

- The complete syntax of the `\setcoordinatesystem` command is

  `\setcoordinatesystem` [`units <`*xunit*`,`*yunit*`>`] [`point at` *xcoord ycoord*]

  Each of the fields '`units <`*xunit*`,`*yunit*`>`' and '`point at` *xcoord ycoord*' is thus optional.
- If '`point at` *xcoord ycoord*' is omitted from the command, PICTEX retains the coordinates of the previous reference point, using *xcoord* = 0 and *ycoord* = 0 as default values.
- If '`units <`*xunit*`,`*yunit*`>`' is omitted, PICTEX retains the previous units, using *xunit* = 1 pt and *yunit* = 1 pt as default values.
- *xunit* and *yunit* may be any strictly positive TEX dimensions; see Chapter 10 of *The TEXbook*.
- A coordinate system remains in effect until a new one is set, or the enclosing group terminates (see Subsection 1.1).
- ○ The fact that TEX deals only with dimensions less than 16384 pts (slightly over 18 feet) in magnitude imposes some minor TEXnical restrictions on the PICtures PICTEX can draw. Each time you give a `\setcoordinatesystem` command, you are effectively asking PICTEX for a new sheet of graph paper 36 feet wide and 36 feet tall, ruled according to the values of *xunit* and *yunit*, and having the coordinate point (0, 0)—the origin—smack in the center. Everything you intend to place in the PICture using that coordinate system has to fit on that piece of paper. When you've finished drawing the PICture, PICTEX takes the sheets of graph paper you've used, lines them up on their reference points, and copies everything to a fresh sheet of 36 ft × 36 ft paper with the common reference points smack in the center; again, everything has to fit on that piece of paper.

PICTEX then trims away the unused borders of the paper and hands it over to TEX to paste into the page layout. Clearly, you have a lot of room to maneuver in, but you can't, e.g., set the unit lengths to 1 inch and work with coordinate values in the 1000's.

**Exercise 2.** What are the relative locations of the origins of the coordinate systems specified by

```
% First system:
  \setcoordinatesystem  units <10pt,5pt>  point at 0 0
% Second system:
  \setcoordinatesystem  point at -20 0   ?
```

**Exercise 3.** Intending to establish a coordinate system with unit lengths of 1 and 2 inches and a reference point at $(3, -2)$, B. L. User typed

```
\setcoordinatesystem [units<1inch,2inches>] [point at (3,-2)]
```

What mistakes did he make?

## 2. PUTTING TEXT INTO A PICTURE

To understand how PICTEX places text in a PICture you need to be aware that TEX automatically encloses each character destined to appear on the printed page in an invisible rectangular box, and that it pastes these character boxes together to form word boxes, math formula boxes, paragraph boxes, and the like. The point to keep in mind is that each block of text has an associated enclosing box. For example, the box enclosing the word 'put' in the font cmssdc10 scaled 4300 looks like this:


*FIGURE 3*

Chapter 11 of *The TEXbook* gives several other examples.

### 2.1. SINGLE PUTS

To have PICTEX place a block of text, say *text*, into a PICture with the box enclosing *text* centered both horizontally and vertically about the coordinate point (*xcoord*, *ycoord*), type

```
\put {text} at xcoord  ycoord
```

Other orientations of the enclosing box can be obtained by typing

```
\put {text} [o_x o_y] at xcoord  ycoord
```

where $o_x$ is either `l` or `r` (or not present), and $o_y$ is either `t`, or `B`, or `b` (or not present). Specifying

$$\left\{\begin{array}{c}\texttt{l}\\ \texttt{r}\\ \texttt{t}\\ \texttt{B}\\ \texttt{b}\end{array}\right\} \quad \text{orients the box so that } (\textit{xcoord}, \textit{ycoord}) \text{ lies on its} \quad \left\{\begin{array}{c}\text{left edge}\\ \text{right edge}\\ \text{top edge}\\ \text{baseline}\\ \text{bottom edge}\end{array}\right\}.$$

Omitting $o_x$ retains horizontal centering, and omitting $o_y$ retains vertical centering. To shift the box a distance *xshift* to the right and a distance *yshift* up from where it would otherwise go, type

`\put {`*text*`} [`$o_x o_y$`] <`*xshift*`,`*yshift*`> at` *xcoord ycoord*

For example, suppose you have already entered

```
\font\bigletters=cmssdc10 scaled 4300
\def\bigput{{\bigletters put}}
```

Then the command

```
\put {\bigput} [rt] <3.1mm,-2mm> at 1 2
```

places a big 'put' in the PICture with the top right corner of its enclosing box 3.1 millimeters to the right and 2 millimeter below the coordinate point $(1, 2)$, like this:


**Exercise 4.** The PICture above has the • (a `$\bullet$`) centered at $(1, 2)$, and the text '$(1, 2)$' vertically centered 10 pts to the right of $(1, 2)$. Give `\put` commands to place these objects in the PICture.

- The syntax of the `\put` command is

  `\put {`*text*`}` $\big[$`[`$[o_x]$`[`$o_y$`]]`$\big]$ $\big[$`<`*xshift*`,`*yshift*`>`$\big]$ `at` *xcoord ycoord*

- *text* can be anything TEX will allow you to put into an hbox (i.e., almost anything).
- If both $o_x$ and $o_y$ are specified, they can appear in either order.
- *xshift* and *yshift* can be any TEX dimension. Remember to type a zero value as, say, `0pt`, not simply `0`.
- $(\textit{xcoord}, \textit{ycoord})$ is a coordinate point in the current coordinate system. In particular, *xcoord* and *ycoord* are dimensionless quantities.

It is good practice to annotate a PICture using type which is slightly smaller than that used in the surrounding text; this helps set the PICture off from that text. Most of the PICtures in this manual are annotated in nine point type (this size), while the body of the text is in ten point type. The

switches to nine point were made using the `\ninepoint` macro described in Appendix E of *The TEXbook*. Should you wish to use the `\ninepoint` macro yourself, you'll have to type in its definition, since it's not included in plain TEX, nor in PICTEX.

**Exercise 5.** Redo the preceding exercise, using nine point type for the bullet and the text '(1, 2)'.

### *2.2. MULTIPLE PUTS*

On occasion you may want to place the same block of text several places in a PICture. This can be done efficiently with the construction

`\multiput {`*text*`} [`$o_x o_y$`] <`*xshift*`,`*yshift*`> at`
  ... *xcoord ycoord* ... `*`*n dxcoord dycoord* ... `/`

Between '`at`' and the terminating '`/`' each occurrence of

*xcoord ycoord*

gives the effect of

$x =$ *xcoord*
$y =$ *ycoord*
`\put {`*text*`} [`$o_x o_y$`] <`*xshift*`,`*yshift*`> at` $x$ $y$

and each occurrence of

`*`$n$ *dxcoord dycoord*

gives the effect of $n$ repetitions of

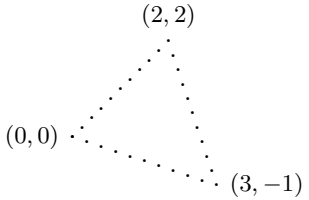
$x = x +$ *dxcoord*
$y = y +$ *dycoord*
`\put {`*text*`} [`$o_x o_y$`] <`*xshift*`,`*yshift*`> at` $x$ $y$

- An arbitrary number of coordinate points $(x, y)$ can be specified. Coordinate values are separated by at least one blank, and at least one blank must precede the terminator signal '`/`'.
- *xcoord*, *ycoord*, *dxcoord*, and *dycoord* refer to the current coordinate system.
- $n$ is an integer; if $n \le 0$, '`*`$n$ *dxcoord dycoord*' has no effect.
- A '*xcoord ycoord*' specification must precede the first '`*`$n$ *dxcoord dycoord*' specification.
- As with the `\put` command itself, the orientation and shift specifications '`[`$o_x o_y$`]`' and '`<`*xshift*`,`*yshift*`>`' are optional.

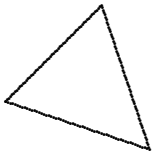
For example

```
\setcoordinatesystem units <.25in, .25in>
\multiput {.} at
  0 0   *10 .2 .2   *10 .1 -.3   *10 -.3 .1   /
```

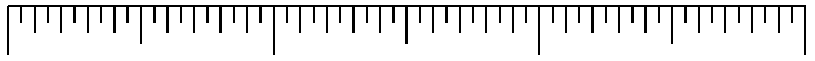
draws in the outline of a triangle between the points $(0,0)$, $(2,2)$, and $(3,-1)$:




Changing the repetition counts from 10 to 60 and scaling the increments down by a factor of 6 produces



**Exercise 6.** Give PICTEX commands to draw the following ruler:




The construction '`\vrule height .4pt width 300pt`' will give you the horizontal line, while '`\vrule height 18pt`' will give you the major divisions.

The coordinate specifications for a `\multiput` command can be stored without the terminator '`/`' in an external file, say *file name*, and accessed with the command

`\multiput {`*text*`} [`$o_x o_y$`] <`*xshift*`,`*yshift*`> at "`*file name*`"`

For example, if the specifications

```
0 0   *150  .58779 -1.80902   *150 -1.53884  1.11803
      *150 1.90211  0.0       *150 -1.53884 -1.11803
      *150  .58779  1.80902
```

are stored in the file `Star.tex`, then the instructions

```
\setcoordinatesystem units <.2pt,.2pt>
\multiput {\fiverm .} at "Star.tex"
```

produce

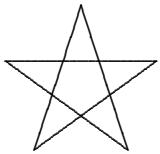


*FIGURE 4*

In principle you could draw any curve you wanted to by `\multiput`ting enough dots into the PICture at the right places. In general that would involve writing a computer program to generate a file containing the appropriate coordinates. You can spare yourself the bulk of that effort by making use of PICTEX's `\plot`, `\setlinear`, and `\setquadratic` commands discussed later on (see also `\replot`).

### *2.3. STACKS OF LETTERS AND ROWS OF LINES*

It would be nice if TEX wrote words vertically as well as horizontally, but it doesn't. PICTEX provides a facility for 

letters up vertically. Such things are produced by the command

`\stack` ⟦`[`*o*`]`⟧ ⟦`<`*leading*`>`⟧ `{`*list*`}`

- *list* is a list of textual items (e.g., letters or words) to be stacked, from top to bottom. In the list items are separated by commas, without intervening blanks — if an item contains a comma, enclose the comma (or the entire item) in `{}`'s.
- Stacked items are left justified if $o$ = `l`, right justified if $o$ = `r`, and centered if '`[`*o*`]`' is omitted from the command.
- *leading* is the distance separating the enclosing boxes of the items in the stack. If '`<`*leading*`>`' is omitted from the command, *leading* defaults to the value of the dimension register `\stackleading`. This register is set to `.17\baselineskip` when the PICTEX macros are loaded; you can change its value by typing, e.g., '`\stackleading=1pt`'.
- The baseline of a stack is the baseline of the bottom item.

You can use `\stack` outside the `\beginpicture` ... `\endpicture` environment. Indeed the first sentence of this paragraph was set with

'... `for \stack <1pt> {s,t,a,c,k,i,n,g}\ letters` ...'.

The '`\ `' after '`{s,t,a,c,k,i,n,g}`' is needed because the PICTEX macros instruct TEX to skip over any blanks following their arguments; this keeps spurious blanks from lousing up a PICture. To place a stack in a PICture, just `\put` it there, as with the command

```
\put {\stack [r] <2pt> {stacking,is,as,simple,%
  as ABC}} [lt] <10pt,0pt> at 3 2
```

**Exercise 7.** Describe the result of the preceding command.

`\stack` was designed for tall scraggly things like the label for a $y$-axis. You may prefer labels consisting of rows of lines, like this: 

. Such

structures are produced by the command

`\lines` [`[`*o*`]`] {*line*$_1$`\cr` *line*$_2$`\cr` ... }

- Any number of *line*s may be specified; they will be arrayed from top to bottom.
- *line*s are left justified if $o =$ `l`, right justified if $o =$ `r`, and centered relative to one another if '`[`*o*`]`' is omitted from the command.
- TEX's usual vertical spacing conventions apply within the array; in particular the baselines of the various *line*s ordinarily will be a fixed distance apart. It is primarily in this respect that `\lines` differs from `\stack`.
- The baseline of the array is the baseline of the bottom *line*.

Like `\stack`, `\lines` can be used outside a PICture. For example, the author typed '`\lines {rows of\cr lines\cr}`' to produce the structure above. The command

`\Lines` [`[`*o*`]`] {*line*$_1$`\cr` *line*$_2$`\cr` ... }

is identical to `\lines`, except that the baseline of the array is the baseline of the top line.

**Exercise 8.** Describe the result of

```
\put {\lines {Rows of lines\cr are easy to put\cr
  into a PiCture\cr}} [lt] <10pt,0pt> at 3 2
```

**Exercise 9.** (a) Do the instructions '`\put {\lines {Two\cr lines\cr}} [B] at 3 2`' and '`\put {\Lines {Two\cr lines\cr}} [B] at 3 2`' produce the same result? (b) What if '`[B]`' is changed to '`[b]`'?

**Exercise 10.** How does the `\lines` command differ from the `\line` command of plain TEX?

`\stack` and `\lines` provide two ways to annotate a PICture. You can put any paragraph you like into a PICture by building the paragraph in a vbox (where all of TEX's formatting capabilities are at your disposal) and `\put`ting the vbox in the PICture.

**Exercise 11.** Describe the result of

```
\put {\vbox{\hsize=1.25in \raggedright \noindent It's easy
to put a paragraph into a PiCture.}} [Bl] <10pt,0pt> at 3 2
```

## *3. GRAPH SETUP*

This section presents PICTEX's commands `\axis` for drawing $x$ and $y$ axes with tick marks, tick values, and axis labels; `\plotheading`, for attaching a heading to a graph; and `\grid`, for superimposing grid lines on a graph. Unless indicated otherwise, the examples given use a coordinate system with unit lengths of 1 pt.

### *3.1. ESTABLISHING A PLOT AREA*

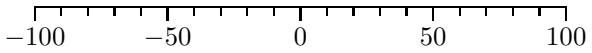
Before you can use any of the commands mentioned above you have to tell PICTEX where you intend to place your graph in the current PICture. The command

```
\setplotarea x from xcoord_l to xcoord_r,
  y from ycoord_b to ycoord_t
```

sets up a rectangular 'plot area' with the coordinate point $(xcoord_l, ycoord_b)$ at its bottom left corner and $(xcoord_r, ycoord_t)$ at its top right corner. You can use `\setplotarea` more than once in the same PICture; `\axis`, `\grid`, and `\plotheading` always refer to the most recently defined plot area.

### *3.2. DRAWING AXES*

Because of its many options, `\axis` is PICTEX's most versatile command. By the same token, it takes the most time to assimilate (not all that much time, though). If you study the following examples carefully, you can pretty much avoid reading the long list of rules given later on. For starters, here are some simple $x$-axes:




First example

results from

```
\setplotarea x from -100 to 100, y from 0 to 0
\axis bottom label {First example} ticks
  numbered from -100 to 100 by 50
  unlabeled short quantity 21 /
```

while


results from

```
\setplotarea x from 0 to 125, y from 0 to 0
\axis top ticks
  withvalues $10^0$ $10^1$ $10^2$ {} {} $10^n$ /
    quantity 6 /
\setplotarea x from 175 to 300, y from 0 to 0
\axis top label {Third example}  /
```

Here are some more extended examples. The instructions

```
\setcoordinatesystem units <40pt,80pt>
\setplotarea x from -2 to 2, y from 0 to 1
```

```
\ninepoint%   (See Subsection 2.1)
\axis bottom ticks
  numbered from -2 to 2 by 1
  length <0pt> withvalues $\alpha^2$ $\beta^2$ /
    at -.5 .5 /  /
\axis top ticks
  length <8pt> width <.2pt> numbered from -1.5 to 1.5 by 1.0
  length <4pt> unlabeled from -2.00 to 2.00 by .25  /
\axis left label {\stack {I,N,C,O,M,E}} ticks
  withvalues \$0 \$10,000 \$20,000 \$30,000 \$40,000 /
   quantity 5  /
\axis right label {\lines {Value\cr of $A_n$\cr}} ticks
  withvalues {$1/1$} {$1/2$} {$1/3$} {$1/n$} /
    at 1 .5 .33333 .125 /
  unlabeled at .25 .2 .16667 .14286 /  /
```

produce Figure 5 below.

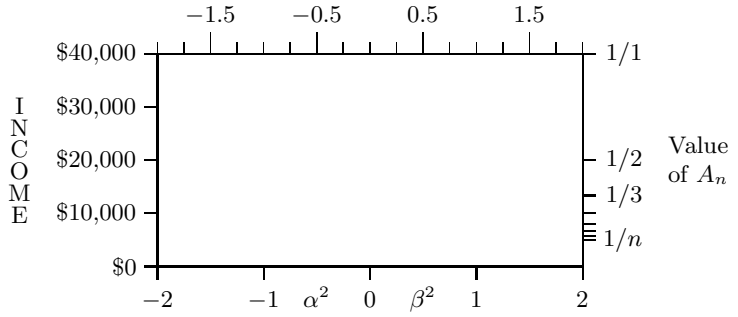



*FIGURE 5*

Moreover the instructions

```
\setcoordinatesystem units <1pt,100pt>
\setplotarea x from 0 to 100, y from 0 to 1
\ninepoint
\axis bottom invisible ticks
  andacross width <.25pt> quantity 5
  length <0pt> from 12.5 to 87.5 by 25.0  /
\axis left invisible ticks
  andacross width <.25pt> logged
    numbered at 1 2 3 5 10 /
    unlabeled short at 4 / from 6 to 9 by 1
      length <0pt> from 1.5 to 4.5 by .5
        from 1.25 to 2.75 by .50 /
\setcoordinatesystem units <1pt,1pt> point at -150 0
\setplotarea x from 0 to 100, y from 0 to 100
```

```
\axis bottom shiftedto y=50 ticks
  in withvalues $-2$ $-1$ {} 1 2 /  quantity 5 /
\axis left   shiftedto x=50 ticks
  in withvalues $-2$ $-1$ {} 1 2 /  quantity 5 /
```

produce Figure 6 below.

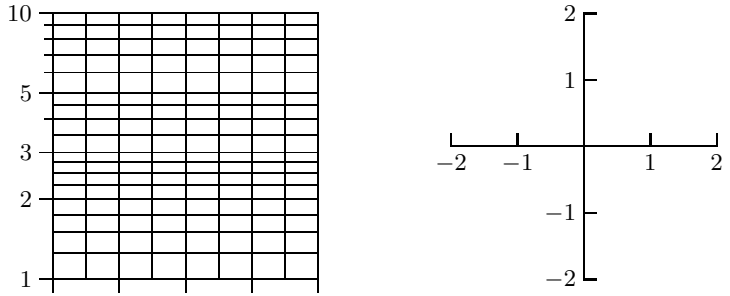



*FIGURE 6*

That's the last of the introductory examples. It's time now for the formalities. The syntax of the `\axis` command is

```
\axis [bottom] [top] [left] [right]
  [shiftedto y=ycoord] [shiftedto x=xcoord]
  [visible] [invisible]
  [label {axis label}]
  [ticks]
    [out] [in]
    [long] [short] [length <length>]
    [width <width>]
    [butnotacross] [andacross]
    [unlabeled] [numbered] [withvalues value1 value2 ... /]
    [unlogged] [logged]
    [quantity q] [from coords to coorde by dcoord]
      [at coord1 coord2 ... /]
  /
```

- The axis is ordinarily drawn along the `bottom`, `top`, `left`, or `right` edge of the current plot area, according to which one of these keywords is specified (one of them must be specified). Tick marks, tick values, and the axis label are drawn, in that order, down from a bottom axis, up from a top axis, to the left from a left axis, and to the right from a right axis.
- The `shiftedto` option causes a bottom or top axis to be drawn at the specified $y$-coordinate, and a left or right axis to be drawn at the specified $x$-coordinate.

- The keyword `invisible` suppresses the drawing of the axis (but not the tick marks, etc., if any). `visible` is the default.
- The text specified by *axis label* is centered with respect to the appropriate edge of the plot area. *axis label* can be anything TEX will let you put in an hbox. In particular, it can be a `\stack` or `\lines` construction (see Subsection 2.3).
- No ticks are drawn unless the keyword `ticks` is specified.
- The keywords discussed so far can appear in any order, except that the positioning keyword (e.g., `bottom`) must come first, and `ticks` last. The remaining keywords describe the ticks and their labels, and can be specified only if `ticks` itself is.
- Ticks ordinarily point `out` from the plot area; `in` makes them point into the plot area.
- Ticks are ordinarily `long` (like this: '|'). However they can be `short` (like this: '˙'), or of an arbitrary *length* specified by the `length` option.
- Ticks ordinarily have the width shown above. The `width` option can be used to set their width to any desired dimension. A tick is invisible if either its length or its width is 0 pt.
- The keyword `andacross` causes ticks to be drawn across the plot area, as well as pointing out (if `out` is in effect) from it. In effect, the tick mark is augmented by a grid line. The keyword `butnotacross` annuls `andacross`, and is the default.
- Ticks are ordinarily `unlabeled`. However when `numbered` is specified, the `at` and `from` options discussed below assign numeric values to them. Alternatively, arbitrary tick labels can be specified by the `withvalues` option; the labels $value_1$, $value_2$, ... are assigned, in that order, to subsequent ticks until either the list of values is exhausted, or a `unlabeled` or `numbered` keyword is encountered. Values in the list must be separated by at least one blank, and at least one blank must precede the '`/`' that terminates the list. If a value contains a blank or a '`/`', enclose the entire value in `{}`'s. Null values are permitted: type '`{}`'.
- The option `quantity` $q$ draws $q$ ticks equally spaced from left to right along an $x$-axis, or from bottom to top along a $y$-axis. The first and last ticks are at the ends of the axis. $q$ is an integer; if $q \leq 1$, no ticks are produced.
- The `from` option draws ticks at the indicated coordinates. $coord_s$, $coord_e$, and *dcoord* must be fixed point numbers, with the same number of digits (if any) to the right of the decimal point (if any), and *dcoord* must be positive. If the `numbered` option is in effect, the coordinate of the tick is used as the tick label (with the same number of digits to the right of the decimal point as in *dcoord*).
- The `at` option draws ticks at the specified coordinates. As with the `from` option, the coordinates must be fixed point numbers, which are used as tick labels when `numbered` is in effect. The list of coordinates must be terminated by a '`/`'.

- The `logged` option applies only to the positioning of subsequent ticks specified by the `from` and `at` options. Ticks are placed at the $\log_{10}$'s of the specified locations; the original unlogged locations are used as labels if `numbered` is in effect. (PICTEX's $\log_{10}$ function is too slow to be used for general purposes. If a logarithmic scale is right for your data, you have to log-transform the data before asking PICTEX to plot it.) `unlogged` annuls `logged`, and is the default.
- Descriptive tick keywords (`in`, `short`, `andacross`, `logged`, `withvalues`, etc.) must precede the positioning keywords (`quantity`, `from`, and `at`) to which they apply. Any number of descriptive and positioning keywords can be specified. For an $x$-axis, tick values are placed in the PICture along a common baseline, centered horizontally with respect to the corresponding ticks. For a $y$-axis, tick values are placed right-justified in the PICture, centered vertically with respect to the corresponding ticks.
- Finally, the entire `\axis` command is terminated by a '`/`'. This terminator, as well as those for the `withvalues` and `at` options, is very important—don't forget it.

**Exercise 12.** B. L. User made 7 mistakes on his first attempt:

```
\axis label 'My First Axis' ticks
  length 10pt with values $x_1$ $x_2$ $x_3$ $x_4$
  from 7.5 to 52.5 by 15
```

at an axis. What were they?

### *3.3. HEADINGS AND GRIDS*

The command

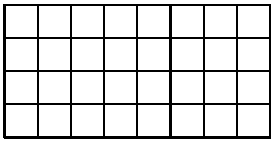
`\plotheading {`*heading*`}`

places the text specified by *heading* centered above the plot area (and above any top axis tick marks, tick values, or axis label). The command

`\grid {`*c*`} {`*r*`}`

partitions the plot area into $c$ columns and $r$ rows. When $c$ or $r$ is a single digit, the `{}`'s bracketing it may be omitted. '`grid 1 1`' just frames the plot area. To produce

A grid



enter

```
\setplotarea x from 0 to 100, y from 0 to 50
\plotheading {A grid}
\grid 8 4
```

More complicated grids can be drawn using the `andacross` option of the `\axis` command (see, e.g., Figure 6).

**Exercise 13.** `\grid` is defined in terms of `\axis`. Guess how.

**Exercise 14.** Give commands to draw Figure 7 below. Assume that the coordinates

```
1 4.1576   2 3.6378   ...   5 3.0183   ...   100 0.6990
```

of the circles are stored in the file `Shakespeare.tex`.

*Number of words Shakespeare used exactly $n$ times, for $n = 1$ to 100 by 1.*

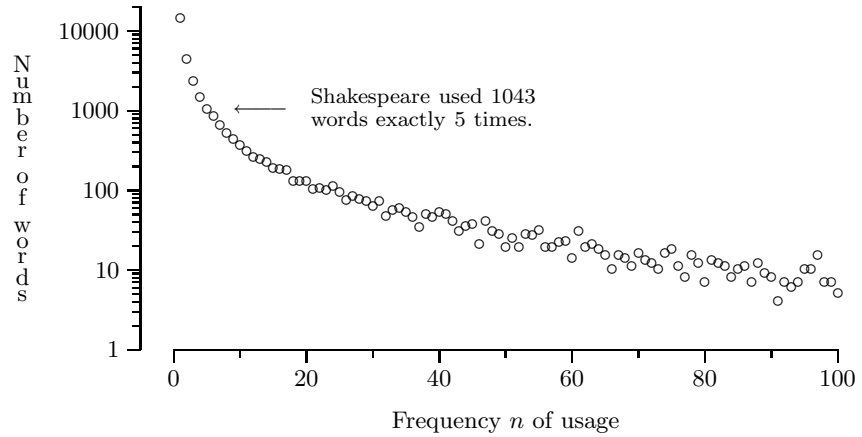



*FIGURE 7*

### *3.4. MODIFYING THE DEFAULTS FOR GRAPH SETUP*

You can modify PICTEX's defaults for the length of ticks, etc., by assigning nonnegative dimensions to the following parameters. The first five apply to any axis, whether it be bottom, top, left, or right.

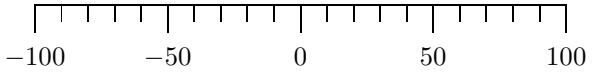
- `\longticklength` — the default length of long ticks.
- `\shortticklength` — the default length of short ticks.
- `\tickstovaluesleading` — the distance separating the ticks and the box enclosing the tick values.

- `\valuestolabelleading` — the distance separating the box enclosing the tick values and the box enclosing the axis label.
- `\linethickness` — the default thickness of axes, tick marks, and grid lines.
- `\headingtoplotskip` — the distance between the baseline of the heading and the top of the plot area, or the top of the box enclosing the top axis structure (if any), whichever is higher.

For example, when prefixed by

```
\longticklength=10pt
\shortticklength=6pt
\tickstovaluesleading=6pt
\linethickness=.25pt
```

the instructions for the first example of an $x$-axis in Subsection 3.2 produce




First example (redone)

Moreover the following commands can be used to set the defaults concerning the `visible`/`invisible`, `out`/`in`, `unlogged`/`logged`, and `butnotacross`/`andacross` keywords of the `\axis` command:

```
\visibleaxes      \invisibleaxes
\ticksout         \ticksin
\unloggedticks    \loggedticks
\nogridlines      \gridlines
```

For example, after specifying '`\ticksin`', all subsequent ticks will point into their plot areas, except when `out` is specified in the `\axis` command.

The command `\normalgraphs` specifies the first member of each of the four pairs of commands in the preceding display and sets the graph setup parameters and `\stackleading` (see Subsection 2.3) to the default values specified below:

| *parameter* | *default value* |
|---|---|
| `\longticklength` | `.4\baselineskip` |
| `\shortticklength` | `.25\baselineskip` |
| `\tickstovaluesleading` | `.25\baselineskip` |
| `\valuestolabelleading` | `.8\baselineskip` |
| `\stackleading` | `.17\baselineskip` |
| `\headingtoplotskip` | `1.5\baselineskip` |
| `\linethickness` | `.4pt` |

The use of TEX's `\baselineskip` register to some extent adapts the spacing to the size of the fonts you're using. However the parameter values are static, so

they won't change when you change `\baselineskip` unless you subsequently issue a `\normalgraphs` command. The default values you get to start with are determined by the value of `\baselineskip` when the PICTEX macros are loaded.

**Exercise 15.** Use `\axis` to draw the ruler of Exercise 6.

## 4. RULES AND OBJECTS MADE FROM THEM

TEX calls horizontal and vertical lines *rules*. This section presents PICTEX's commands dealing with rules and objects like rectangles, histograms, and bar graphs that are built up out of rules. The thickness of the rules drawn by these commands is governed by the `\linethickness` parameter of the preceding subsection.

### 4.1. RULES

The command

`\putrule from` $xcoord_s$ $ycoord_s$ `to` $xcoord_e$ $ycoord_e$

draws a rule in the current PICture from the point $(xcoord_s, ycoord_s)$ to the point $(xcoord_e, ycoord_e)$; either the starting and ending $x$-coordinates should be the same, or the starting and ending $y$-coordinates should be the same. For example the little 'firecracker'  is produced by

```
\setcoordinatesystem units <1pt,1pt>
\putrule from 0 0 to 0 15
\linethickness=6pt
\putrule from 0 0 to 0 12
```

Note the effect of changing `\linethickness`. The shift option of `\put` commands can be used with `\putrule`:

`\putrule <`*xshift*`,`*yshift*`> from` $xcoord_s$ $ycoord_s$ `to` $xcoord_e$`,`$ycoord_e$

moves the rule a distance *xshift* to the right and *yshift* up from where it would otherwise go.

**Exercise 16.** Construct the ruler of Exercise 6 making use of `\putrule`.

*4.2. RECTANGLES*

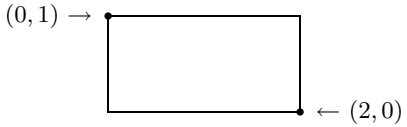
PICTEX provides several ways to construct rectangles. The command

`\putrectangle corners at` *xcoord*$_s$ *ycoord*$_s$ `and` *xcoord*$_e$ *ycoord*$_e$

draws a rectangle with opposite corners at the points $(xcoord_s, ycoord_s)$ and $(xcoord_e, ycoord_e)$. For example

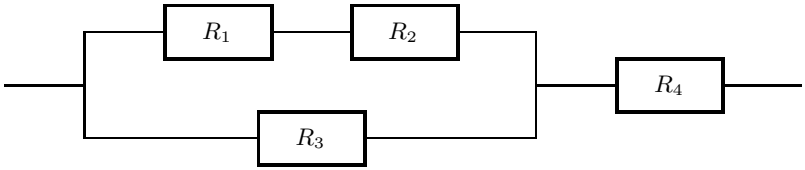
```
\setcoordinatesystem units <.5in,.5in>
\putrectangle corners at 0 1 and 2 0
```

produces




If you look closely, you'll see that the rules outlining the rectangle overlap at the corners. The shift option '`<`*xshift*`,`*yshift*`>`' of `\put` commands can be placed between '`\putrectangle`' and '`corners`'.

**Exercise 17.** Give commands to draw the circuit diagram




The command

`\putbar breadth <`$\beta$`> from` *xcoord*$_s$ *ycoord*$_s$ `to` *xcoord*$_e$ *ycoord*$_e$

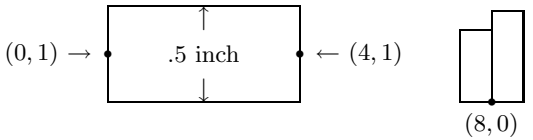
draws a rectangle having $(xcoord_s, ycoord_s)$ and $(xcoord_e, ycoord_e)$ as the midpoints of opposite sides of length $\beta$.

- $(xcoord_s, ycoord_s)$ and $(xcoord_e, ycoord_e)$ are points in the current coordinate system. Either the starting and ending $x$-coordinate must be the same, or the starting and ending $y$-coordinate must be the same.
- $\beta$ is a dimension. If $\beta = 0\,\text{pt}$, `\putbar` has the same effect as `\putrule`.
- The shift option of `\put` commands can be used with `\putbar`.

For example the instructions

```
\setcoordinatesystem units <.25in,.25in>
\putbar breadth <.5in> from 0 1 to 4 1
\putbar <-6pt,0pt> breadth <12pt> from 8 0 to 8 1.5
\putbar < 6pt,0pt> breadth <12pt> from 8 0 to 8 1.9
```

were used in making the following PICture:




You can frame a textual item, inside or outside a PICture, with the command

`\frame` [`<`*separation*`>`] {*text*}

Here

- *text* can be anything you can put in an hbox.
- *separation* is a (not necessarily positive) dimension specifing the amount of space between the box enclosing *text* and the framing rectangle. If '`<`*separation*`>`' is omitted from the command, you get the effect of '`<0pt>`'.

The framed text has the same baseline as *text*. For example, the first line of this paragraph was set with '`You can \frame <2pt> {frame}\ a ...`'. To place a framed item into a PICture, `\put` it there, as with '`\put {\frame <3pt> {\frame <3pt> {LEGEND}}} at 100 20` '.

**Exercise 18.** Describe the result of the preceding command.

The command

`\rectangle <`*w*`> <`*h*`>`

produces a rectangle of a prescribed width $w$ and height $h$, having its baseline along the bottom edge. Such an object can be `\put` into a PICture just like any block of text.

**Exercise 19.** Describe the result of '`\put {\rectangle <1in> <.25in> } [tr] at 1 2` '.

**Exercise 20.** `\rectangle` is defined in terms of `\frame`. Guess how.

### *4.3. HISTOGRAMS*

The commands

`\sethistograms`
`\plot` $xcoord_0$ $ycoord_0$ $xcoord_1$ $ycoord_1$ $xcoord_2$ $ycoord_2$
$xcoord_3$ $ycoord_3$ ... `/`

construct a histogram composed of rectangles having opposite corners at the points

$(xcoord_0, ycoord_0)$–$(xcoord_1, ycoord_1)$
$(xcoord_1, ycoord_0)$–$(xcoord_2, ycoord_2)$

$(xcoord_2, ycoord_0)$–$(xcoord_3, ycoord_3)$
⋮

- An arbitrary number of coordinate points $(xcoord_i, ycoord_i)$ can be specified with the `\plot` command. Coordinate values are separated by at least one blank, and at least one blank must precede the terminator signal '`/`'.

For example the histogram in Figure 8 below was constructed with

```
\setcoordinatesystem units <.03125in,.25in>
\sethistograms
\plot 0 0   10 1.5   20 3.5   40 1.5   80 0.5  /
```

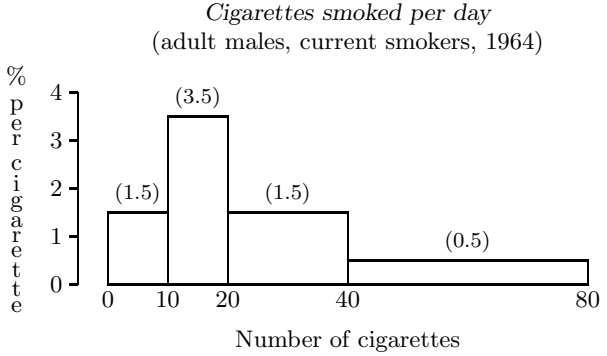
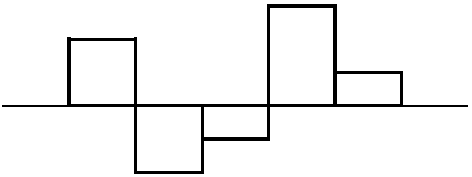



*FIGURE 8*

**Exercise 21.** Give a complete set of instructions to draw Figure 8.

**Exercise 22.** Show how to produce



The list of coordinate values that `\plot` acts upon can be stored without the terminator '`/`' in an external file, say '*file name*', and accessed simply by

`\plot "`*file name*`"`

For example, the histogram in Figure 8 could have been produced by entering

```
0 0   10 1.5   20 3.5
      40 1.5   80 0.5
```

into, say, file `cigarettes.tex`, and then specifying

```
\sethistograms
\plot "cigarettes.tex"
```

### *4.4. BAR GRAPHS*

The commands

```
\setbars breadth <β> baseline at z = zcoord
\plot xcoord1 ycoord1  xcoord2 ycoord2  ...  /
```

have the effect of

```
\putbar breadth <β> from xcoord1 zcoord to xcoord1 ycoord1
\putbar breadth <β> from xcoord2 zcoord to xcoord2 ycoord2
  ⋮
```

when $z$ is the letter `y`, and the effect of

```
\putbar breadth <β> from zcoord ycoord1 to xcoord1 ycoord1
\putbar breadth <β> from zcoord ycoord2 to xcoord2 ycoord2
  ⋮
```

when $z$ is the letter `x`. These commands thus make it easy to draw bar graphs.

- When $z$ is `y`, the bars rest on the horizontal line `y` = *zcoord*, while when $z$ is `x`, the bars rest on vertical line `x` = *zcoord*. (There are no other possibilities for $z$.)
- The specification '`<`*xshift*`,`*yshift*`>`' may be placed between '`\setbars`' and '`breadth`' to shift the bars in the usual manner.
- The "*file name*" option of the `\plot` command may be used.
- Labels can be attached to the base of the bars by continuing the `\setbars` command with

  `baselabels (`$\big[$`[[`$o_x$`][`$o_y$`]]`$\big]$ $\big[$`<`*xshift*`,`*yshift*`>`$\big]$`)`

  and by following each coordinate specification in the `\plot` command by the appropriate label, enclosed in quotation marks, like this: "*label*". For each $i$, the $i^{\text{th}}$ base *label* is positioned relative to ($xcoord_i$, *zcoord*) when $z$ is `y`, or relative to (*zcoord*, $ycoord_i$) when $z$ is `x`, just as though it were `\put` there with the options specified by the `baselabels` field.
- Similarly labels can be attached to the end of the bars by continuing the `\setbars` command with

  `endlabels (`$\big[$`[[`$o_x$`][`$o_y$`]]`$\big]$ $\big[$`<`*xshift*`,`*yshift*`>`$\big]$`)`

  and by following each coordinate specification in the `\plot` command by the appropriate label, enclosed in double quotes. For each $i$, the $i^{\text{th}}$ end *label* is positioned relative to ($xcoord_i$, $ycoord_i$) just as though it were `\put` there with the options specified by the `endlabels` field.
- Base label specifications must precede end label specifications if both are used.

For example the instructions

```
\setcoordinatesystem units <7pt,11pt>
\setbars breadth <0pt> baseline at x = 0
  baselabels ([Br] <-5pt,-2pt>)
\linethickness=2pt  \def\Yr#1{{\sevenrm 7#1}}%
\plot
 24.1   0 "Austria      \Yr5"  23.8  -1 "Denmark     \Yr3"
 21.0  -2 "West Germany \Yr4"  15.4  -3 "France      \Yr0"
 14.9  -4 "Belgium      \Yr3"  10.6  -5 "Luxembourg  \Yr5"
  9.2  -6 "Netherlands  \Yr4"   8.6  -7 "Portugal    \Yr4"
  7.9  -8 "England      \Yr4"   5.8  -9 "Italy       \Yr2"
  4.0 -10 "Spain        \Yr4"   1.5 -11 "Switzerland \Yr5" /
```

were used in constructing Figure 9.

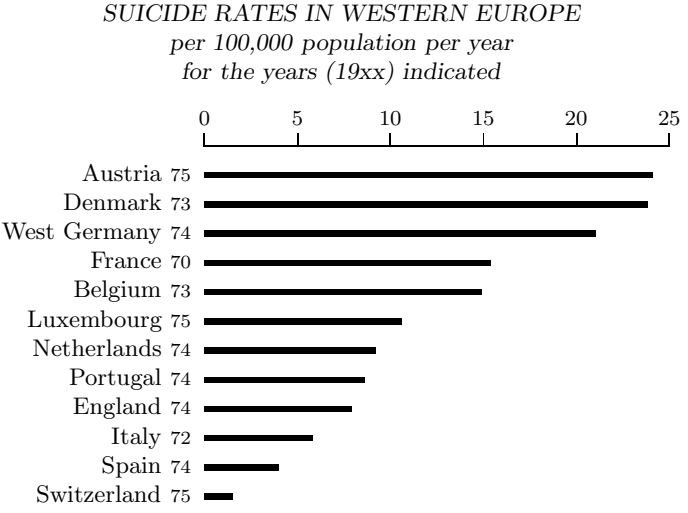



*FIGURE 9*

**Exercise 23.** Give commands to draw Figure 10. Suppose that the coordinates $k$ $b_k$, $k = 0, 1, \ldots, 12$, are stored in the file `Binomial.tex`, and that the corresponding coordinates $k$ $p_k$ are stored in the file `Poisson.tex`.

### *4.5. PICTEX'S INTERPOLATION MODES*

What the `\plot` command does to a list of coordinate values depends on what *interpolation mode* PICTEX is in. `\sethistograms` puts PICTEX into a histogram drawing mode, and `\setbars` puts it into a bar graph drawing mode. There are two other modes: piecewise linear interpolation and piecewise quadratic interpolation, which are initiated by `\setlinear` and

*The Binomial distribution for 20 trials of success probability 0.2 and its Poisson approximation. The height of the bar just to the left of $k$ is the Binomial probability $b_k = \binom{20}{k}(.2)^k(.8)^{20-k}$, while the height of the bar just to the right of $k$ is the Poisson probability $p_k = e^{-4}4^k/k!$.*

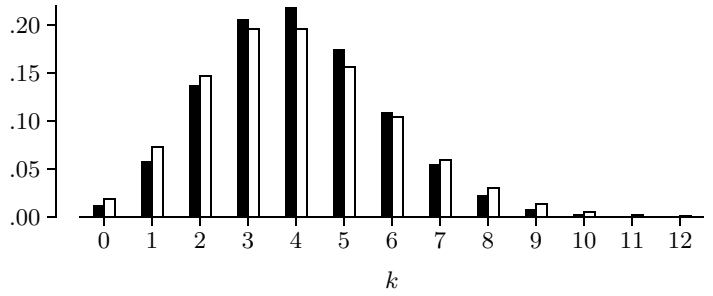



*FIGURE 10*

`\setquadratic` respectively (see the next section). Piecewise linear interpolation is the default. Regardless of the mode, coordinates for the `\plot` command can be specified either via the "*file name*" option or as an explicit list terminated by a '`/`'.

## 5. LINES AND CURVES

This section presents PiCTeX's basic commands `\plot`, `\setlinear`, and `\setquadratic` for drawing lines and curves, and its higher level commands `\circulararc`, `\ellipticalarc`, `\arrow`, and `\betweenarrows` for drawing circles, ellipses, and arrows.

### 5.1. PIECEWISE LINEAR AND QUADRATIC INTERPOLATION

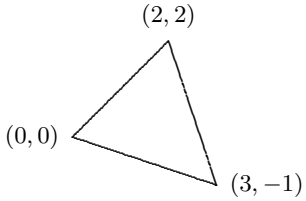
The commands

```
\setlinear
\plot xcoord0 ycoord0  xcoord1 ycoord1  xcoord2 ycoord2
  xcoord3 ycoord3  ...  /
```

connect the points $(xcoord_{i-1}, ycoord_{i-1})$ and $(xcoord_i, ycoord_i)$, $i = 1, 2, \ldots$ by straight lines. Notice that the `\plot` command of Subsections 4.3–4.5 is put to a new use here. For example

```
\setcoordinatesystem units <.25in, .25in>
\setlinear   \plot  0 0   2 2   3 -1   0 0  /
```

reproduces the triangle of Subsection 2.2:



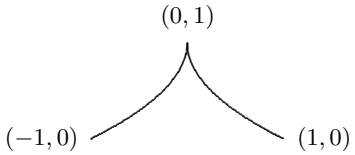

Similarly, the commands

```
\setquadratic
\plot xcoord0 ycoord0  xcoord1 ycoord1  xcoord2 ycoord2
  xcoord3 ycoord3  xcoord4 ycoord4  ...  /
```

draw quadratic arcs through the points $(xcoord_j, ycoord_j)$, $j = 2i-2, 2i-1, 2i$, for $i = 1, 2, \ldots$ . (The number of points must be odd.) Thus

```
\setcoordinatesystem units <.5in, .5in>
\setquadratic
\plot   -1 0   -0.25 .5   0 1   .25 .5   1 0 /
```

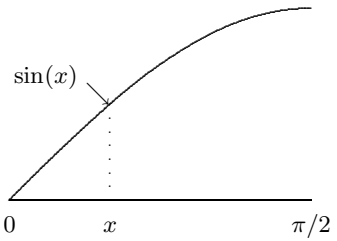
yields




and

```
\setcoordinatesystem units <1.5708in, 1in>
\setquadratic
\plot   0 0   .16667 .25882   .33333 .5
  .5 .70711   .66667 .86603   .83333 .96593   1 1  /
```

draws the graph of $\sin(x)$ for $0 \le x \le \pi/2$:




**Exercise 24.** Use `\plot` to produce the star in Figure 4.

**Exercise 25.** Give commands to draw Figure 11 below.

The density $\phi(\zeta) = e^{-\zeta^2/2}/\sqrt{2\pi}$ of the standard normal distribution.

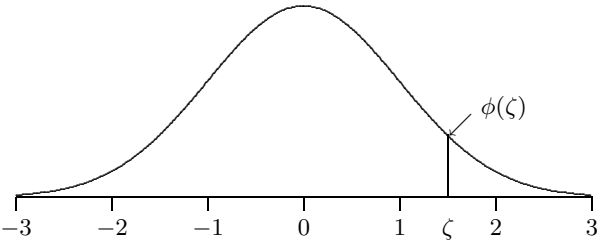



*FIGURE 11*

### *5.2. SPECIFYING THE PLOT SYMBOL*

PICTEX draws lines and curves by placing a 'plot symbol' every $\delta$ units along the arc under construction. The default plot symbol is a `\fiverm` period; $\delta$ defaults to 0.4 pt. You can change the plot symbol with the command

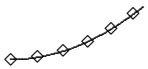
`\setplotsymbol ({`*plot symbol*`}` `[[`$[o_x]$`[`$o_y$`]]]` `[<`*xshift*`,`*yshift*`>]``)`

- The new plot symbol '*plot symbol*' typically is a single character, but in fact it can be anything that can go in an hbox.
- $o_x$, $o_y$, *xshift*, and *yshift* have the same significance as they do for a `\put` command.

A plot symbol remains in effect until a new one is set. You can change $\delta$ simply by assigning a new value to the dimension register `\plotsymbolspacing`. For example

```
\setcoordinatesystem units <1pt,1pt>
\setquadratic  \plot  0 0  25 5  50 20  /
\setplotsymbol ({$\diamond$})
\plotsymbolspacing=10pt
\plot  0 0  25 5  50 20  /
```

yields



Ordinarily you shouldn't set `\plotsymbolspacing` to such a large value; the `\setdots` command of the next section is the right tool to use to space out the plot symbol.

**Exercise 26.** Suppose you've changed the plot symbol and its spacing. How could you reestablish PICTEX's defaults?

### 5.3. CIRCLES AND ELLIPSES

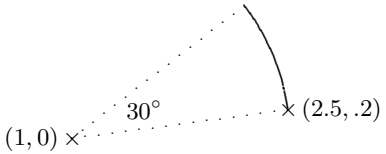
The command

\circulararc $\theta$ degrees from $xcoord_s$ $ycoord_s$
center at $xcoord_c$ $ycoord_c$

draws an arc of a circle with center at $(xcoord_c, ycoord_c)$; the arc starts from $(xcoord_s, ycoord_s)$ and extends counterclockwise through an angle of $\theta$ degrees.

- $\theta$ can have any real value between $-360$ and 360.
- If $\theta$ is negative, the arc is drawn clockwise through an angle of $|\theta|$ degrees.
  - The radius of the circle involved must be less than 512 pts (slightly over 7 inches).

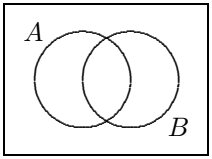
For example the following arc




was constructed with

```
\setcoordinatesystem units <.75in,.75in>
\circulararc 30 degrees from 2.5 .2  center at 1 0
```

**Exercise 27.** Give commands to draw the Venn diagram




The command

\ellipticalarc axes ratio $\xi$:$\eta$ $\theta$ degrees from $xcoord_s$ $ycoord_s$
center at $xcoord_c$ $ycoord_c$

functions like \circulararc, but applies to an ellipse whose major and minor axes are parallel[1] to the $x$ and $y$ axes.

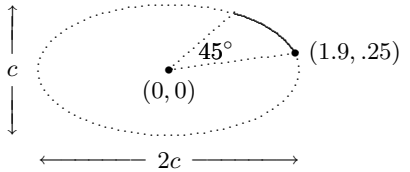
- $\xi$ and $\eta$ are numbers proportional to the lengths of the horizontal and vertical axes of the ellipse.
  - The quantity $\sqrt{((xcoord_s - xcoord_c)/\xi)^2 + ((ycoord_s - ycoord_c)/\eta)^2}$ must be less than 512 pts.

For example

```
\setcoordinatesystem units <25pt,25pt>
\ellipticalarc  axes ratio 2:1  45 degrees  from 1.9 .25
  center at 0 0
```

[1] PiCTeX can draw arbitrary ellipses; see Subsection 9.1.

produces the solid arc below:




**Exercise 28.** How could you produce the dotted elliptical arc above (ignoring the fact that it is dotted)?

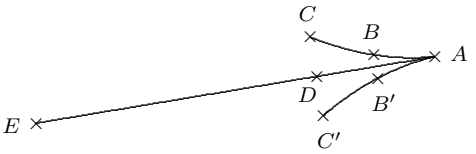
**Exercise 29.** What is the value of $c$ above?

### *5.4. ARROWS*

The fonts that come with plain TEX have a variety of arrows (see Appendix F of *The TEXbook*) which you can easily `\put` into a picture. You will probably find that these arrows suffice for most purposes. Nonetheless, you may find uses for PICTEX's command

`\arrow <`$\ell$`> [`$\beta$`,`$\gamma$`]` [`<`*xshift*`,`*yshift*`>`]
`from` *xcoord*$_s$ *ycoord*$_s$ `to` *xcoord*$_e$ *ycoord*$_e$

which draws an arrow of the form




where

$E = (xcoord_s, ycoord_s)$,
$A = (xcoord_e, ycoord_e)$,
$\ell$ is the distance between $A$ and $D$,
$\beta\ell$ is the distance between $B$ and $B'$,
$\gamma\ell$ is the distance between $C$ and $C'$.

As with a `\put` command, if the optional field '`<`*xshift*`,`*yshift*`>`' is specified, the arrow is shifted a distance *xshift* to the right and *yshift* up from where it would otherwise go. The arrow above was set with

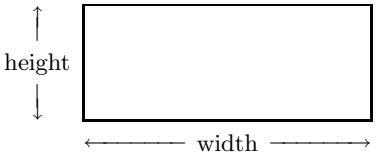
```
\setcoordinatesystem units <1pt,1pt>
\arrow <45pt> [.2,.67] from  -150 -25  to  0 0
```

PICTEX draws its arrows by `\plot`ting a straight line and two quadratic arcs. Consequently, `\arrow` uses considerably more time and computer memory than `\put`ting a character arrow does.

**Exercise 30.** How could you produce

 ?

Sometimes you may want to place some text between arrows that go between given points in a PICture, as below:




(Those arrows are extra long versions of TEX's arrows.) Such constructions are easily handled by the command

```
\betweenarrows {text} [[[ox][oy]]] [<xshift,yshift>]
  from xcoords ycoords to xcoorde ycoorde
```

- *text* can be anything that can go in an hbox.
- $o_x$, $o_y$, *xshift*, and *yshift* have the same significance as they do for a `\put` command.
- $(xcoord_s, ycoord_s)$ and $(xcoord_e, ycoord_e)$ are points in the current coordinate system. Either $xcoord_s$ and $xcoord_e$ must be the same, or $ycoord_s$ and $ycoord_e$ must be the same.

For example the PICture above was produced by

```
\setcoordinatesystem units <1in,1in>
\putrectangle corners at 0 0 and 1.5 .6
\ninepoint
\betweenarrows {width}  [t] <0pt,-5pt> from 0 0 to 1.5 0
\betweenarrows {height} [r] <-5pt,0pt> from 0 0 to 0  .6
```

### *5.5. THE ROLE OF PLOT AREAS*

To save time, PICTEX doesn't keep track of where `\plot` places the current plot symbol. To avoid adverse effects like this

you should draw your lines, curves, circles, and arrows within a plot area specified by a prior `\setplotarea` command. For instance, it is the `\setplotarea` command in the following code

```
\setcoordinatesystem units <6pt,12pt>
\setplotarea x from 0 to 2, y from 0 to 2
\setquadratic  \plot  0 0  1 1.3  2 2  /
```

that keeps the curve

from straying into regions where it doesn't belong.

You can instruct PICTEX not to place plot symbols at coordinate points lying outside the current plot area. The command `\inboundscheckon` activates this feature, while `\inboundscheckoff` deactivates it. Use `\inboundscheckon` sparingly, since PICTEX runs faster under `\inboundscheckoff`. The commands

```
\setcoordinatesystem units <1.5708in,.25in>
\setplotarea x from 0 to 1, y from 0 to 4
\setquadratic  \plot  0 0
  .15  .24008  .30  .50953   .43  .80115  .55 1.17085
  .63 1.52235  .70 1.96261   .75 2.41421  .80 3.07768  /
\inboundscheckon %   (next \plot goes out of range)
\plot          .80 3.07768  .825 3.54573  .85 4.16530  /
```

were used in constructing Figure 12 below.

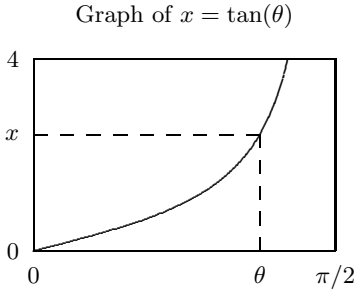



*FIGURE 12*

**Exercise 31.** What do you get from

```
\inboundscheckon
\setplotarea x from 0 to 100, y from 0 to 100
\setlinear
\plot   200 0   300 100   / %     (Line A)
\setplotarea x from 200 to 300, y from 0 to 100
\plot     0 0   100 100   / %     (Line B)    ?
```

### *5.6. REPLOTTING LINES AND CURVES*

You may well find PICTEX's curve drawing algorithm to be frustratingly slow, especially for documents that undergo many revisions. In cases where the PICtures don't change from one revision to another you can speed things up considerably as follows. During one of the early runs you can instruct PICTEX to save the locations at which it places plot symbols while constructing lines and curves. Then in subsequent runs you can redraw saved lines and curves

using the \replot command discussed below. A curve can be \replotted about 4 times faster than it can be \plotted from scratch. \replotting saves some time for dashed lines (see the next section), but essentially none for solid lines, for which PICTEX uses a streamlined algorithm.

Subsequent to the command

\savelinesandcurves on "*file name*"

PICTEX writes out on file *file name* the locations at which it places plot symbols while \plotting lines (not rules) and curves. The command

\dontsavelinesandcurves

stops PICTEX from saving plot locations. The command

\replot "*file name*"

draws all the lines and curves that were saved on file *file name* in a previous run.

- The term 'lines and curves' includes (PICTEX) arrows, circular and elliptical arcs, and anything you might have constructed using \plot with either \setlinear and/or \setquadratic in effect.
- In any given run, you must specify a different *file name* each time you use \savelinesandcurves.
- You can specify \savelinesandcurves as many times as you like within a PICture. A single specification will suffice unless you choose not to save some lines and/or curves, or change the plot symbol, or construct lines and curves within a subPICture of the PICture.
- Lines and curves that are drawn with different plot symbols have to be saved on different files, and each such file has to be \replotted with the appropriate plot symbol in effect.
- Lines and curves that were saved within a PICture can only be \replotted within that PICture. In particular, lines and curves that were saved within a subPICture of a PICture (see Subsection 8.4) have to be \replotted within that subPICture.
- Insofar as it applies to whether or not PICTEX saves lines and/or curves, the effect of a \savelinesandcurves command is local to the group containing that command. The same is true of a \dontsavelinesandcurves command. However, the file specification of a \savelinesandcurves command is global, i.e., pertains to all existing groups, not just the one containing the save command.
  - PICTEX saves plot locations using a coordinate system with unit lengths of 1 scaled pt (= 1/65536 pt) and with the origin at the reference point of the PICture.

  The command

  \writesavefile {*message*}

writes out the text of the message *message* on the file specified by the most recent `\savelinesandcurves` command; the message is preceded in the file by the comment character '`%`' so that it won't interfere with `\replot`ting. You can use this feature to provide a reminder of what's saved on the file in question.

For example, to save the plot locations for the sine curve in Subsection 5.1 the author used the code

```
\savelinesandcurves on "Sine.tex"
\writesavefile {Sine curve for Subsection 5.1}
\setquadratic
\plot   0 0   .16667 .25882      ...      1 1  /
```

while debugging the PiCture involved. Subsequently he replaced those instructions with the single command '`\replot "Sine.tex"`'.

**Exercise 32.** What happened when B. L. User specified the same *file name* for two `\savelinesandcurves` commands?

**Exercise 33.** Is there a way to stop PiCTeX from saving lines and curves without using the `\dontsavelinesandcurves` command?

**Exercise 34.** Consider the following instructions:

```
\beginpicture %   (main PiCture)
  \savelinesandcurves on "main.tex"
  \plot  0 0  100 100  / %  (Line 1)
  \put {\beginpicture %  (subPiCture)
          \savelinesandcurves on "sub.tex"
          \plot  0 0  50 75  / %  (Line 2)
          \dontsavelinesandcurves
        \endpicture} at 100 200
  \plot  0 200  100 150 / %  (Line 3)
\endpicture
```

Which of Lines 1, 2, and 3 will be saved, and on which files?

**Exercise 35.** `\replot`ting speeds up the processing of your document during revisions, but it is still somewhat time-consuming. Show how you can speed things up even more by using TeX `\if` constructions (see pages 207–211 of *The TeXbook*) to instruct PiCTeX not to draw certain lines and/or curves except during production runs.

## *5.7. THE CURVE DRAWING ALGORITHM*

This subsection discusses the algorithm PiCTeX performs to draw an arc through three points, call them

$(x_0, y_0)$, $(x_1, y_1)$, and $(x_2, y_2)$.

Let $x(t)$ and $y(t)$ be the quadratic functions on the interval $[0, 2]$ such that

$$x(t) = x_t \text{ and } y(t) = y_t, \quad \text{for } t = 0,\ 1,\ 2,$$

as illustrated in Figure 13.

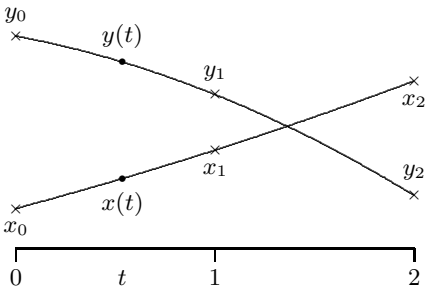



*FIGURE 13*

The length of the parametric curve $\big(x(s), y(s)\big)$, $0 \le s \le t$, is

$$A(t) = \textstyle\int_0^t a(s)\, ds$$

where

$$a(s) = \sqrt{\big(x'(s)\big)^2 + \big(y'(s)\big)^2}\,;$$

PICTEX uses Simpson's rule to approximate $A(1)$ by

$$\alpha_1 = \frac{1}{6}\big(a(0) + 4a(.5) + a(1)\big)$$

and $A(2)$ by

$$\alpha_2 = \alpha_1 + \frac{1}{6}\big(a(1) + 4a(1.5) + a(2)\big).$$

Let $B(u)$ be the inverse function to $A(t)$; PICTEX approximates $B(u)$ by the quadratic function $B^\circ(u)$ such that

$$B^\circ(0) = 0,\ B^\circ(\alpha_1) = 1, \text{ and } B^\circ(\alpha_2) = 2.$$

Finally, working from a sequence $(u_i)$ of $u$ values spaced a distance $\delta$ apart in the interval $[0, \alpha_2]$, where $\delta$ is the current value of `\plotsymbolspacing`, PICTEX places the current plot symbol at the points

$$p_i = \big(x(B^\circ(u_i)), y(B^\circ(u_i))\big).$$

The $p_i$'s lie along the curve $C = \big\{(x(t), y(t))\big\}$, and, provided $B^\circ(u)$ is a decent approximation to $B(u)$, are spaced out nearly a distance $\delta$ apart as in Figure 14.[2]

---

[2] The $p_i$'s in Figure 14 are nominally 10 pts apart, but a discriminating eye can see that in fact the distances between them vary between 9 and 11 pts. However, the curve in that figure was constructed by interpolating just 3 points. When 5 points are interpolated, the spacing is uniform to within .3 pt — the resolution of a 300 dot per inch printer.

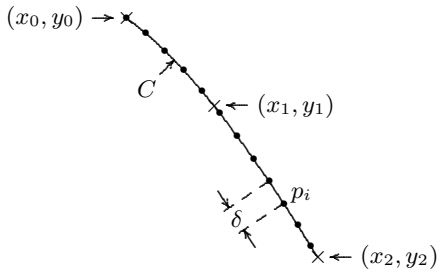



*FIGURE 14*

Considering that TEX only provides facilities for integer arithmetic, you can well understand why `\plot` takes a while to complete its task when `\setquadratic` is in effect. On the other hand, since both $x$ and $y$ are interpolated, vertical turning points present no problem. Moreover with trivial modifications the algorithm produces dashed arcs with dashes (nearly) of a prescribed length; no simpler algorithm could accomplish that.

The algorithm performs poorly unless $(x_1, y_1)$ lies in the middle third of the arc between $(x_0, y_0)$ and $(x_2, y_2)$, i.e., unless

$$\alpha_2/3 \le \alpha_1 \le 2\alpha_2/3.$$

When this condition is not met, PICTEX writes a warning message to that effect in your log file. To remedy the situation you should appropriately revise the values you're `\plot`ting. As a general rule you should choose $(x_0, y_0)$, $(x_1, y_1)$, and $(x_2, y_2)$ so that the curve you're trying to interpolate is approximately linear between $(x_0, y_0)$ and $(x_2, y_2)$, and so that $(x_1, y_1)$ lies approximately midway between these two points.

**Exercise 36.** Hoping to draw the cubic in Figure 15, B. L. User specified

```
\setquadratic
\plot  -1 0   -.5 .375   0 0   .5 -.375   1 0  /
```

but the curve didn't come out right. Why not?

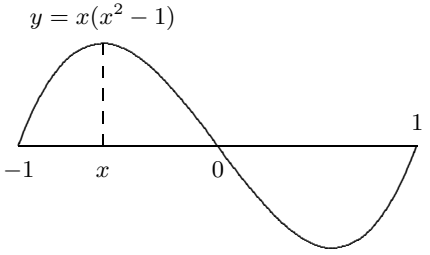



*FIGURE 15*

**Exercise 37.** Another time B. L. User specified

```
\plotsymbolspacing=5pt
\setcoordinatesystem units <50pt,50pt>
```

```
\setplotarea x from -1 to 1, y from 0 to 1
\setquadratic  \plot  -1 1  0 0  1 1  /
```

and got

Why did the spacing come out so uneven?

## 6. DOTS AND DASHES

In drawing a rule, line, or curve, PICTEX uses solid line fill unless instructed to use dots, dashes, or some general user-specified pattern. Once selected, one of these so-called interrupted line patterns applies to all subsequent axes, tick marks, grid lines, rules, lines, and curves, until a new pattern is set.

### 6.1. SIMPLE DOT AND DASH PATTERNS

The command

`\setdots` [`<`$\ell$`>`]

specifies an interrupted line pattern consisting of dots spaced a distance $\ell$ apart.

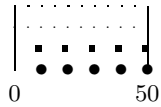
- $\ell$ is a positive dimension.
- Omitting '`<`$\ell$`>`' has the effect of '`<5pt>`'.
- For rules, a 'dot' is a rule of length `\plotsymbolspacing`.
- For lines and curves, a 'dot' is a single occurrence of the current plot symbol.
  - Rules start at coordinate locations, whereas plot symbols (typically) are centered about coordinate locations; this results in a slight difference in the placement of dots for rules and for lines and curves.
  - If you change `\plotsymbolspacing` within the scope of a `\setdots` command, you have to reissue the `\setdots` command.

For example

```
\setcoordinatesystem units <1pt,1pt>
\setplotarea x from 0 to 50, y from 0 to 24
\linethickness=.25pt
\putrule from  0 0  to  0 24  \putrule from  50 0  to  50 24
\linethickness=.4pt
\setdots
\putrule from  0 24  to  50 24
\setlinear
```

```
\plot     0 16   50 16  /
\linethickness=2pt
\setplotsymbol ({$\bullet$})  \plotsymbolspacing=2pt
\setdots <10pt>
\putrule from  50 8  to  0 8
\plot     50 0   0 0  /
```

yields




To save time when you're debugging some pictures, apply '`\setdots`' (or maybe even '`\setdots <20pt>`') to your lines and curves.

The command

`\setdashes` [`<`$\ell$`>`]

specifies an interrupted line pattern composed of dashes of length $\ell$ separated by gaps of length $\ell$.

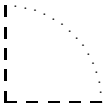
- $\ell$ is a positive dimension.
- Omitting '`<`$\ell$`>`' has the effect of '`<5pt>`'.
  - For lines and curves, the size of the current plot symbol is not taken into account — the larger the plot symbol, the longer will be the dashes and the shorter the gaps.

For example

```
\setcoordinatesystem units <1pt,1pt>
\setplotarea x from 0 to 75, y from 0 to 20
\linethickness=.75pt
\setdashes
\putrule from  0 0  to  75 0
\setlinear
\plot  0 0   37.5 20   75 0  /
```

produces

**Exercise 38.** Construct



The radius of the circle is half an inch.

The command

`\setsolid`

returns PICTEX to the default solid line fill mode.

*6.2. GENERAL DASH PATTERNS*

The command

```
\setdashpattern <d1,g1,d2,g2,...>
```

specifies an interrupted line pattern composed of a dash of length $d_1$, followed by a gap of length $g_1$, followed by a dash of length $d_2$, and so on.

- There is no limit on the number of dashes and gaps in the pattern.
- $d_1$, $g_1$, $d_2$, $g_2$, ... are nonnegative dimensions.
- The length of any dash or gap can be expressed either as an explicit dimension (e.g., '`5pt`'), or in terms of one of TEX's dimension registers (e.g., '`.5\dimen0`'). However, dash patterns are static, so subsequent changes to any dimension register do not affect them.
- A pattern is repeated as many times (perhaps fractional) as needed to complete the rule, line, or curve under construction.

For example

```
\setcoordinatesystem units <1pt,1pt>
\setplotarea x from 0 to 100, y from 0 to 25
\setdashpattern <10pt, 2pt, 3pt, 2pt, 3pt, 2pt>
\setquadratic  \plot  0 0   50 8   100 25  /
```

yields

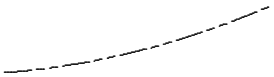
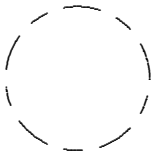


*FIGURE 16*

while

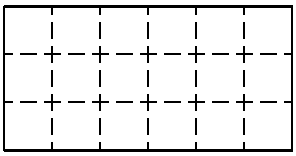
```
\dimen0=.375in
\setcoordinatesystem units <\dimen0,\dimen0>
\setplotarea x from -1 to 1, y from -1 to 1
\dimen2=6.28319\dimen0  \divide\dimen2 by 5
\setdashpattern <.4\dimen2, .2\dimen2, .2\dimen2, .2\dimen2>
\circulararc  360  degrees from  1 0  center at  0 0
```

produces



**Exercise 39.** How could you use `\setdashpattern` to get the effect of
(i) `\setdots`;
(ii) `\setdashes`?

**Exercise 40.** Give commands to produce



### *6.3. TAILORING A PATTERN TO THE LENGTH OF A CURVE*

The command

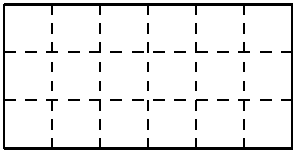
```
\setdotsnear <ℓ> for <λ>
```

sets a pattern of dots evenly spaced a distance about $\ell$ apart, the exact spacing being determined so that a curve of length $\lambda$ will both start and end with a dot. Similarly after the command

```
\setdashesnear <ℓ> for <λ>
```

a curve of length $\lambda$ will be drawn so that it both starts and ends with a complete dash.

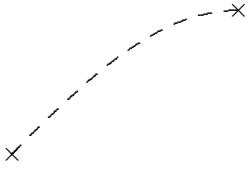
**Exercise 41.** Produce



To use `\setdotsnear` and `\setdashesnear` you need to know the length of the curve you're constructing. To find out how long PICTEX thinks a given curve is (and that's really what counts), use the command

```
\findlength {curve commands}
```

PICTEX executes the curve drawing commands comprising *curve commands* without plotting anything, and then places the result of its length calculations in the dimension register `\totalarclength`.[3] You can use `\totalarclength` as the $\lambda$ argument to `\setdotsnear` and `\setdashesnear`. For example, to get



[3] In general, `\totalarclength` holds the length of the most recently drawn line or curve.

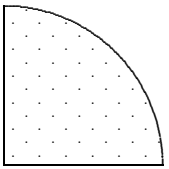
enter

```
\def\sinecurve{% For convenience, put plot cmd in a macro.
  \plot   0 0   .16667 .25882   .33333 .50000
    .5 .70711   .66667 .86603   .83333 .96593   1 1 /}%
\setcoordinatesystem units <1.1781in, .75in>
\setplotarea x from 0 to 1, y from 0 to 1
\setquadratic
\findlength {\sinecurve}  % find length of curve
\setdashesnear <5pt> for <\totalarclength>
\sinecurve                % draw curve
\put {$\times$} at 0 0   \put {$\times$} at 1 1
```

**Exercise 42.** How could you adjust the pattern used to draw the curve in Figure 16 above so that the curve starts and ends with the long dash, like this:

## 7. SHADING

PiCTeX's approach to shading a region is to place a 'shading symbol' at each point of a lattice that falls within the region, as illustrated to the right. The commands `\setshadesymbol` and `\setshadegrid` give the user control over the shading symbol and shading lattice. The commands `\vshade` and `\hshade` do the actual shading in conjunction with `\setlinear` and `\setquadratic`.



### 7.1. SPECIFYING THE SHADING SYMBOL

The default shading symbol is a `\fiverm` period. You can set the shading symbol to whatever you want with the command

`\setshadesymbol` [`<`$\epsilon_l$`,`$\epsilon_r$`,`$\epsilon_d$`,`$\epsilon_u$`>`] ({*shade symbol*}
[[$o_x$][$o_y$]]] [`<`*xshift*`,`*yshift*`>`])

- The new shading symbol '*shade symbol*' typically is a single character, but in fact it can be anything that can go in an hbox.
- $o_x$, $o_y$, *xshift*, and *yshift* have the same significance they do for a `\put` command.
- The optional 'edge effects' field `<`$\epsilon_l$`,`$\epsilon_r$`,`$\epsilon_d$`,`$\epsilon_u$`>` is used to keep a shading symbol from extending across the boundary of the region $R$ being shaded, as happens here:  . Specifically, PiCTeX will not place a shading symbol at a point within a region $R$ unless the distances from that point to

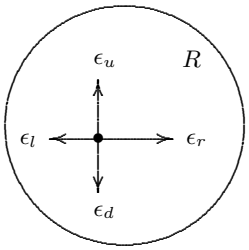



*FIGURE 17*

the boundary of $R$, measured to the left, right, down, and up, are at least $\epsilon_l$, $\epsilon_r$, $\epsilon_d$, and $\epsilon_u$ respectively, as indicated in Figure 17. If the edge effects field is omitted from the `\setshadesymbol` command, PICTEX will choose default values for the $\epsilon$'s based on the size of the shading symbol and its orientation and shift. You can override a default value by specifying its replacement in the appropriate subfield of the edge effects field. To accept a default value, leave the corresponding subfield empty.

- The $\epsilon$'s are (not necessarily positive) dimensions. 0 pt may be coded simply as `z`.

For example,

```
\setshadesymbol <,z,,5pt> ({$\scriptscriptstyle\circ$})
```

sets the shading symbol to '$\circ$', $\epsilon_r$ to 0 pt, and $\epsilon_u$ to 5 pt; $\epsilon_l$ and $\epsilon_d$ receive their default values.

**Exercise 43.** How could you use `\setshadesymbol` to specify PICTEX's default for the shading symbol?

**Exercise 44.** Guess how the author specified the shading symbol to produce: (1)  ; (2)  .

## *7.2. SPECIFYING THE SHADING LATTICE*

The lattice of points at which shading symbols may be placed is composed of all points which are a distance $j\sigma$ to the right of and a distance $k\sigma$ above a so-called anchor point $(x, y)$, $j$ and $k$ being (not necessarily positive) integers whose sum $j + k$ is even; see Figure 18.

The command

`\setshadegrid` [`span <`*s*`>`] [`point at` *xcoord ycoord*]

sets $\sigma$ to $s$ and $(x, y)$ to $(xcoord, ycoord)$.

- If '`span <`*s*`>`' is omitted, PICTEX retains the previous value of $\sigma$, using 5 pt as a default.
- If '`point at` *xcoord ycoord*' is omitted, PICTEX retains the previous anchor point, using $(0, 0)$ as a default.

A shading lattice

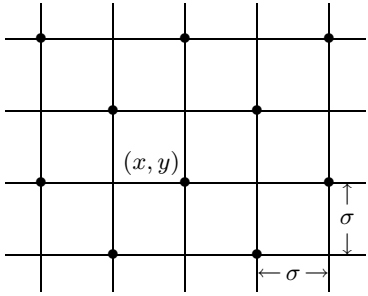



*FIGURE 18*

- $s$ must be a positive dimension.
- *xcoord* and *ycoord* refer to the coordinate system current at the time the `\setshadegrid` command is issued.

For example '`\setshadegrid span <.125in> point at 1 2`' anchors the shading lattice at the point $(1, 2)$, with a span of $\frac{1}{8}$ inch.

**Exercise 45.** How could you use `\setshadegrid` to specify PiCTeX's defaults for the shading lattice?

### *7.3. SHADING: VERTICAL MODE*

PiCTeX's shading algorithm can operate either in a 'vertical' mode or a 'horizontal' mode. In vertical mode the region being shaded can have piecewise linear/quadratic bottom and top boundaries, whereas in horizontal mode it can have piecewise linear/quadratic left and right boundaries. To shade a general region you have to subdivide it into subregions which can be shaded in vertical or horizontal mode. The technique is clearly primitive, but with enough patience you can shade fairly complex regions.

The commands

`\setlinear`
`\vshade` $x_0\ y_0^{(b)}\ y_0^{(t)}\quad \big[$`<`$\epsilon_{l;1}$`,`$\epsilon_{r;1}$`,`$\epsilon_{d;1}$`,`$\epsilon_{u;1}$`>`$\big]\ x_1\ y_1^{(b)}\ y_1^{(t)}$
$\quad \big[$`<`$\epsilon_{l;2}$`,`$\epsilon_{r;2}$`,`$\epsilon_{d;2}$`,`$\epsilon_{u;2}$`>`$\big]\ x_2\ y_2^{(b)}\ y_2^{(t)}\ \ \ldots\ \ /$

shade a region of the form in Figure 19 below.

- The coordinate points $(x_i, y_i^{(b)})$ and $(x_i, y_i^{(t)})$ satisfy $x_0 < x_1 < x_2 < \cdots$ and $y_i^{(b)} \le y_i^{(t)}$, $i = 0, 1, 2, \ldots$ . In particular the subregions $R_1$, $R_2$, ... are specified from left to right.
- For the duration of the shading of region $R_i$, the the optional edge effects field '`<`$\epsilon_{l;i}$`,`$\epsilon_{r;i}$`,`$\epsilon_{d;i}$`,`$\epsilon_{u;i}$`>`' overrides the specifications '`<`$\epsilon_l$`,`$\epsilon_r$`,`$\epsilon_d$`,`$\epsilon_u$`>`' made when the shading symbol was set with `\setshadesymbol`.

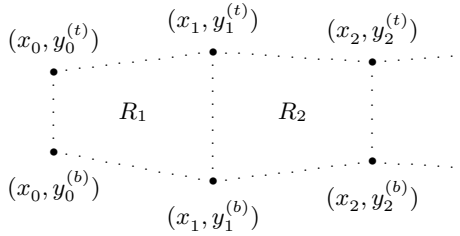
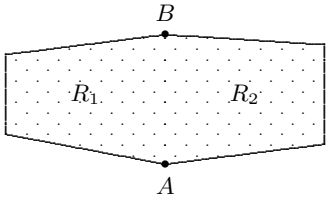



*FIGURE 19*

- The various fields of the `\vshade` command are separated by blanks, and the command ends with a '`/`'.

For example the shading below

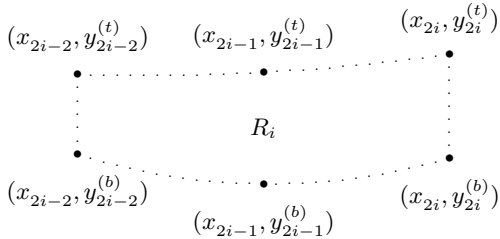



was produced by

```
\setcoordinatesystem units <.75pt,.75pt>
\setshadegrid span <4pt>
\vshade  0 0 40   <,z,,> 80 -15 50   <z,,,> 160 -5 45  /
```

$\epsilon_{r;1}$ and $\epsilon_{l;2}$ were set to `z` (i.e., 0 pt) to allow shading symbols to be placed along the line joining point $A$ to point $B$.

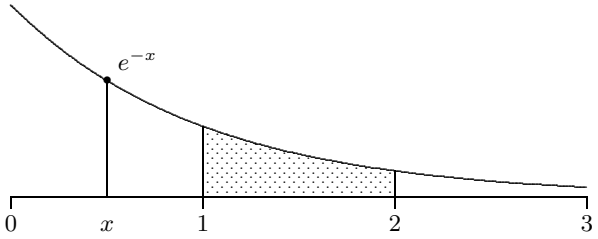
The subregions $R_i$ being `\vshade`d can have the form




the bottom and top boundaries being quadratic functions of $x$. In this case you specify

```
\setquadratic
\vshade x_0 y_0^(b) y_0^(t)  ...
  [<ε_{l;i},ε_{r;i},ε_{d;i},ε_{u;i}>] x_{2i-1} y_{2i-1}^(b) y_{2i-1}^(t)  x_{2i} y_{2i}^(b) y_{2i}^(t)  ...  /
```

**Exercise 46.** Give commands to draw

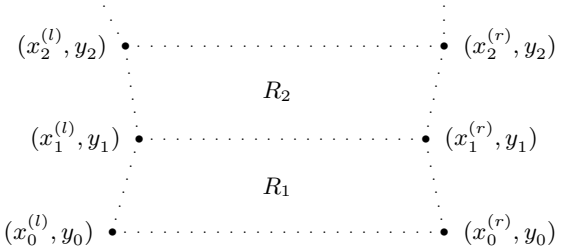



*7.4. SHADING: HORIZONTAL MODE*

The commands

```
\setlinear
\hshade y_0 x_0^(l) x_0^(r)  [<ε_{l;1},ε_{r;1},ε_{d;1},ε_{u;1}>] y_1 x_1^(l) x_1^(r)
  [<ε_{l;2},ε_{r;2},ε_{d;2},ε_{u;2}>] y_2 x_2^(l) x_2^(r)  ...  /
```

shade a region of the form




- The coordinate points $(x_i^{(l)}, y_i)$ and $(x_i^{(r)}, y_i)$ satisfy $y_0 < y_1 < y_2 < \cdots$ and $x_i^{(l)} \le x_i^{(r)}$, $i = 0, 1, 2, \ldots$ . In particular the subregions $R_1$, $R_2$, ... are specified from bottom to top.
- For the duration of the shading of region $R_i$, the the optional edge effects field '<$\epsilon_{l;i}$,$\epsilon_{r;i}$,$\epsilon_{d;i}$,$\epsilon_{u;i}$>' overrides the specifications '<$\epsilon_l$,$\epsilon_r$,$\epsilon_d$,$\epsilon_u$>' made when the shading symbol was set with `\setshadesymbol`.
- The specifications

  $$\left[\texttt{<}\epsilon_{l;i}\texttt{,}\epsilon_{r;i}\texttt{,}\epsilon_{d;i}\texttt{,}\epsilon_{u;i}\texttt{>}\right]\ y_{2i-1}\ x_{2i-1}^{(l)}\ x_{2i-1}^{(r)}\ \ y_{2i}\ x_{2i}^{(l)}\ x_{2i}^{(r)}\ \ \ldots\ \ /$$

  are used in conjunction with `\setquadratic` when the subregions $R_i$ have left and right boundaries that are quadratic in $y$.

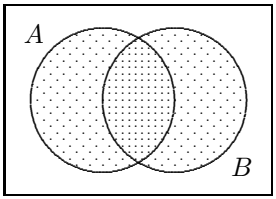
For example, here are the commands that drew the shaded quarter circle at the start of this section:

```
\setcoordinatesystem units <1pt,1pt>
\putrule from  0 60  to  0 0    \putrule from  0 0  to  60 0
\circulararc  90  degrees from  60 0  center at  0 0
\setquadratic
```

```
\setshadegrid point at 3.5 3.5
\vshade 0  0 60   <,z,,> 15  0 58.09475   30  0 51.96152  /
\hshade 0 30 60   <z,,,> 26 30 54.07402   51.96152 30 30  /
```

Note that half of the quarter circle was shaded in vertical mode, and half in horizontal mode.

**Exercise 47.** Give commands to draw the Venn diagram

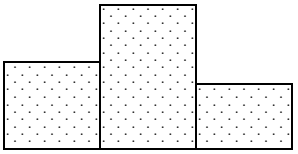



*7.5. SHADING RECTANGLES*

Subsequent to the command

```
\shaderectangleson
```

every rectangle PICTEX draws will be shaded according to the current shading symbol and lattice. `\shaderectangleson` thus affects `\putrectangle`, `\putbar`, `\frame`, and `\rectangle`, and `\plot` when PICTEX is in a histogram or bar graph drawing mode. For example



results from

```
\dimen0=.5in
\setcoordinatesystem units <\dimen0,1.5\dimen0>
\divide \dimen0 by 13
\setshadegrid  span <\dimen0>  point at .5 0
\shaderectangleson  \sethistograms
\plot  0 0   1 .6   2 1   3 .45   /
```

The command

```
\shaderectanglesoff
```

annuls `\shaderectangleson`.

# 8. USING PICTURES

This section discusses various ways in which a PICture can be positioned on a page: in a display or insert, in line with text, or as a component of a larger PICture.

## 8.1. PICTURES IN DISPLAYS AND INSERTS

A PICture that doesn't take up much vertical space can be displayed in place simply by enclosing the whole affair in `$$`'s, like this:

```
$$\beginpicture
    ⋮
  \endpicture$$
```

Larger PICtures should be handled as `\midinsert`s or `\topinsert`s so that they can float to where there's room available for them (see Chapter 15 of *The TEXbook*). Here's sample code for a `\midinsert` in which the PICture is centered horizontally on the page:

```
\midinsert
  \line \bgroup \hss
    \beginpicture
      ⋮
    \endpicture
  \hss \egroup
\endinsert
```

If you plan to present all your PICtures in this manner, you don't need to read the rest of this section.

## 8.2. THE ENCLOSING BOX FOR A PICTURE

If you want to embed a PICture in ordinary text or use it as a component of a larger PICture, you need to understand how PICTEX hands a completed PICture over to TEX. Here is what happens. First PICTEX determines the smallest box $B$ that encloses all the boxes that were put in the PICture by `\put` and its relatives. Then PICTEX assigns a baseline to $B$ as follows. If the common reference point of the coordinate system(s) used in constructing the PICture lies above $B$, the baseline lies along the top edge of $B$. If the reference point lies below $B$, the baseline lies along the bottom edge of $B$. Otherwise the baseline is set so that it passes through the reference point, as in Figure 20. Finally PICTEX hands $B$ and its associated baseline[4] over to TEX to use just as it would any other block of text.

[4] Suppose that the box $B$ in Figure 20 is a distance $w$ wide and $t$ tall, and that the reference point is a distance $r$ above the bottom edge of $B$. Then precisely what PICTEX delivers to TEX is an hbox of width $w$, height $h = \max(0, \min(t - r, t))$, and depth $t - h$.

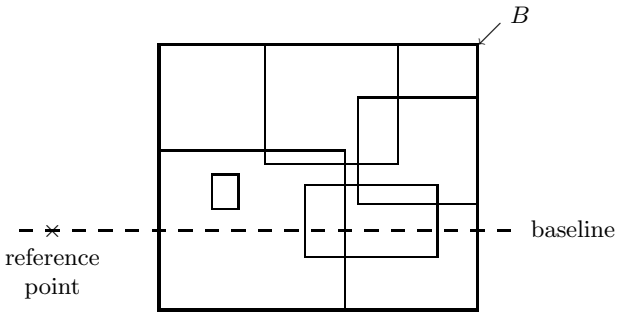



*FIGURE 20*

PICTEX actually updates box $B$ as each new box is placed in the PICture by `\put`, `\multiput`, `\putrule`, `\putrectangle`, and `\putbar`, and by `\setplotarea`. However, you can suspend PICTEX's updating procedure with the command `\accountingoff` and reinstate it later with the command `\accountingon`. This feature is advantageous if, e.g., you're `\multiput`ting many copies of an object into a region that PICTEX is already aware of (perhaps from a prior `\setplotarea`). You can speed things up a bit by typing `\accountingoff` before `\multiput` and `\accountingon` after the terminating '`/`'.

If you draw an entire PICture with `\accountingoff`, or only `\plot` lines and curves, PICTEX will think nothing was placed in the PICture. In this case box $B$ is of zero width and height, and is positioned at the reference point.

**Exercise 48.** How could you instruct TEX to tell you the height, depth, and width of the enclosing box of a PICture?

**Exercise 49.** How could you instruct TEX to leave space for an hbox having a specified height, depth, and width?

**Exercise 50.** Explain how to use a TEX `\if` construction to optionally draw a certain PICture, or leave the proper amount of space for it.

## *8.3. PICTURES EMBEDDED IN TEXT*

Since TEX lines boxes up horizontally on their baselines, where you place the reference point for a PICture has a bearing on how TEX will position the completed figure on a line of text. For example, consider the symbol '$\therefore$', which some people like to use as an abbreviation for 'therefore'. This symbol isn't included in the font tables of Appendix F of *The TEXbook*. The following macro constructs it as a PICture in which a period is placed at the vertices of an equilateral triangle 1 ex in height (1 ex is the height of the letter 'x' in the current font). Note that the baseline of the completed PICture is the common baseline of the bottom two periods.

```
\def\therefore{%
  \beginpicture
```

```
    \setcoordinatesystem units <1ex,1ex> point at 0 0
    \multiput {.} [B] at  -.577 0   0 1   .577 0  /
  \endpicture}
```

The second sentence of this paragraph was set with "... `consider the symbol '\therefore', which` ..."

**Exercise 51.** How would the `\therefore` macro above perform if it had said '`\multiput {.} at`' instead of '`\multiput {.} [B] at`'?

PICtures can be positioned relative to other PICtures and/or blocks of text using TEX's `\halign` command (of which `\lines` is a special case). See, e.g., the code in Appendix C for Figure 9.

### *8.4. PICTURES INSIDE PICTURES*

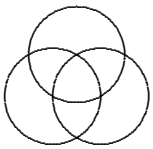
Since a completed PICture is just a block of text as far as TEX is concerned, it can be `\put` or `\multiput` into another PICture. The placement of the subPICture depends in part on its enclosing box, which was described above in Subsection 8.2.

**Exercise 52.** Consider the following instructions.

```
\beginpicture %   (main PiCture)
  \setcoordinatesystem point at 50 75
  \put {\beginpicture %   (subPiCture A)
          \putrule from  100 0  to  200 0
        \endpicture}  at  0 0
  \put {\beginpicture %   (subPiCture B)
          \setlinear  \plot  100 0  200 0  /
        \endpicture}  at  0 0
\endpicture
```

Where do the rule constructed in subPICture A and the line constructed in subPICture B appear in the main PICture?

PICTEX can copy a PICture faster than it can draw it from scratch. Consequently the Venn diagram



is more efficiently produced by

```
\multiput {\beginpicture
             \circulararc 360 degrees from 1 0 center at 0 0
           \endpicture}  at  0 0   1 0   .5 .866  /
```

than by

```
\circulararc 360 degrees from 1 0 center at 0 0
\circulararc 360 degrees from 2 0 center at 1 0
\circulararc 360 degrees from 1.5 .866 center at .5 .866
```

To use a PICture as a component of several other PICtures, you need to first store the component PICture in one of TEX's box registers, like this:

```
\newbox\picA %          (Allocate a box register)
\setbox\picA=\hbox{%    (Store the component PiCture there)
  \beginpicture
    ⋮
  \endpicture}
```

Type '`\put {\copy\picA} ...`' to place the component PICture into a subsequent PICture. Type '`\box\picA`' instead of '`\copy\picA`' the last time you place the component; this frees up the space it occupied. (`\newbox`, `\setbox`, `\copy`, and `\box` are discussed in Chapter 15 of *The TEXbook*.)

**Exercise 53.** How could you modify the `\therefore` macro of the preceding subsection so that it doesn't reconstruct the '∴' symbol each time it's invoked?

Sometimes you may want to `\put` a subPICture into a PICture so that the reference point of the subPICture is placed at a specified location. This can be accomplished by making use of a variant of the `\endpicture` command, in the following manner. Let $p$ be the reference point of the subPICture, and let $q$ be the point where the baseline and left edge of the enclosing box of the subPICture intersect. Let $\xi$ (respectively, $\eta$) be the distance $q$ is to the right of (respectively, up from) $p$. If you finish off the subPICture with the command

`\endpicturesave <`*xreg*`,`*yreg*`>`

instead of the usual `\endpicture`, PICTEX will place $\xi$ in the dimension register *xreg* and $\eta$ in the dimension register *yreg*. A `\put` of the subPICture `at` *xcoord ycoord* with the options '`[Bl] <`*xreg*`,`*yreg*`>`' will then position the subPICture in a larger PICture so that $p$ is at (*xcoord*, *ycoord*). For example

```
\put {\beginpicture
        \setcoordinatesystem point at 10 20
        \putrectangle corners at 50 50 and 75 75
      \endpicturesave <\dimen1,\dimen3>}
  [Bl] <\dimen1,\dimen3> at 0 0
```

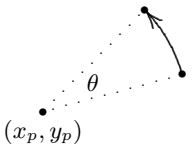
has the same effect as '`\putrectangle corners at 40 30 and 65 55` '. For an example with more substance to it, look at the instructions in Appendix C that were used to draw Figure 14.

# 9. MISCELLANEOUS TOPICS

This section presents some miscellaneous topics. (1) Rotations—sometimes the easiest way to draw a PICture, or a component of a PICture, is to construct it first in one coordinate system and then rotate it to another coordinate system. (2) Dimension mode—this is what PICTEX uses in its internal workings. You can put off reading this material until you want to write macros extending PICTEX. (3) Register arithmetic—when you get around to writing those macros, you may well have need of PICTEX's commands supplementing TEX's limited facilities for register arithmetic.

## 9.1. ROTATIONS

PICTEX can rotate parts of a PICture through a given angle $\theta$ about a given pivot point $(x_p, y_p)$, as illustrated to the right. Rotation is initiated by the command




`\startrotation` [`by` $\cos(\theta)$ $\sin(\theta)$] [`about` $x_p$ $y_p$]

and terminated by the command

`\stoprotation`

(or by the end of the group enclosing the `\startrotation` command).

- $\cos(\theta)$ and $\sin(\theta)$ are the cosine and sine of the angle $\theta$ of rotation. (You can find the cosine and sine of $\theta$ on your calculator more easily than PICTEX could work them out.)
- $(x_p, y_p)$ is a point in the current coordinate system.
- If '`by` $\cos(\theta)$ $\sin(\theta)$' is omitted, PICTEX retains the previous cos and sin, using 1 and 0 as defaults.
- If '`about` $x_p$ $y_p$' is omitted, PICTEX retains the previous pivot point, using $(0, 0)$ as a default.

Actually, the situation is not as simple as the above discussion would lead one to believe. The problem is that many things in TEX, including all characters and rules, have an intrinsic horizontal and/or vertical orientation, and can't be rotated per se. The following items clarify exactly what effect `\startrotation` has.

- The '`at`' coordinates of `\put` and `\multiput` commands rotate, but the material being placed retains its original orientation.
- Plot areas, and so also axes, etc., do not rotate.
- The '`from`' coordinate of a rule rotates, but the rule retains its original orientation.
- Rectangles do not rotate (in fact, they come apart).
- All lines and curves, including (PICTEX) arrows and arcs of circles and ellipses, rotate.
- The coordinates at which shading symbols are placed rotate, but the symbols retain their original orientation.

For example the right half of Figure 21 (excluding the label) was produced by placing

```
\startrotation  by .96593 -.25882  about  0 1
```

before the instructions that drew the left half of the figure; see Appendix C for the details.


The parabola $y = x^2/4$ before rotation

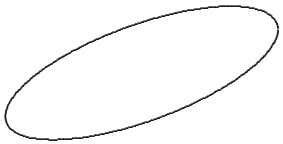
After rotation about the focus $F$ by 15 degrees

*FIGURE 21*

**Exercise 54.** Describe the result of

```
\setcoordinatesystem units <25pt,25pt>
\startrotation  by 0 1
\put {$\times$} at 1 1
\circulararc 90 degrees from 2 1 center at 1 1
\put {$\bullet$}  at 0 1
\linethickness=1pt  \putrule from 0 1 to 1 1
```

**Exercise 55.** Give instructions to draw the ellipse



The major axis of the ellipse is $1\frac{1}{2}$ inches long and makes an angle of 20 degrees with respect to the $x$-axis, while the minor axis of the ellipse is $\frac{1}{2}$ inch long.

### *9.2. DIMENSION MODE*

PICTEX ordinarily functions in *coordinate mode*, locations being specified in terms of the current coordinate system. However, PICTEX can also function in a *dimension mode* wherein locations are specified by (signed) distances to the right of and up from the origin. Since these distances can be given in terms of TEX's dimension registers, and since those registers can be manipulated using TEX's facilities for register arithmetic (see Chapter 15 of *The TEXbook*), dimension mode provides some programming capabilities, albeit rudimentary, that aren't available in coordinate mode.

Subsequent to the command

```
\setdimensionmode
```

PICTEX expects each location to be specified by how far it is horizontally and vertically from the origin: '`\put {$\bullet$} at 2in -.5in` ' places a bullet two inches to the right of and half an inch below the origin.

- When a distance is expressed in terms of a dimension register or control sequence, it must be enclosed in `{}`'s, as with '`\put {$\phi$} at {1.414\dimen0} {-\ylocation}` '.
- The `units` option of the `\setcoordinatesystem` command is irrelevant in dimension mode.
- The '`from`' and '`at`' options of the `\axis` command do not work in dimension mode.

The command

```
\setcoordinatemode
```

returns PICTEX to coordinate mode.

Suppose, e.g., that you want to devise a macro[5] which will draw an object of the form

given the coordinates $(x, y)$ of the central bullet and the difference $\Delta y$ of the $y$-coordinates of the upper cross bar and the bullet. The length $\ell$ of the cross bars is to be variable, but not a parameter of the macro. To get started, allocate some useful dimension registers:

```
\newdimen\xposition
\newdimen\yposition
\newdimen\dyposition
\newdimen\crossbarlength
```

Now begin coding the macro, letting the parameters `#1`, `#2`, and `#3` be respectively $x$, $y$, and $\Delta y$:

```
\def\puterrorbar at #1 #2 with fuzz #3 {%
  \xposition=\Xdistance{#1}
  \yposition=\Ydistance{#2}
  \dyposition=\Ydistance{#3}
```

Here `\Xdistance{`*xcoord*`}` and `\Ydistance{`*ycoord*`}` are PICTEX commands giving the horizontal and vertical distances to the origin from a point (*xcoord*, *ycoord*) in the current coordinate system. Now continue coding the macro with

```
  \setdimensionmode
  \put {$\bullet$} at {\xposition} {\yposition}
```

[5] The rest of this subsection presumes some familiarity with TEX's macro facilities (see Chapter 20 of *The TEXbook*).

This puts PICTEX into dimension mode and places the bullet. Next place the vertical bar:

```
\dimen0 = \yposition %                  ** Determine y-location of
  \advance \dimen0 by -\dyposition %    **   the lower cross bar.
\dimen2 = \yposition %                  ** Determine y-location of
  \advance \dimen2 by  \dyposition %    **   the upper cross bar.
\putrule from {\xposition} {\dimen0} % ** Place vertical rule.
  to {\xposition} {\dimen2}
```

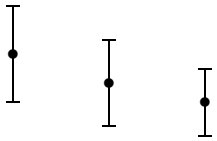
Finally, place the cross bars and return to coordinate mode:

```
\dimen4 = \xposition
  \advance \dimen4 by -.5\crossbarlength
\dimen6 = \xposition
  \advance \dimen6 by  .5\crossbarlength
\putrule from {\dimen4} {\dimen0} to {\dimen6} {\dimen0}
\putrule from {\dimen4} {\dimen2} to {\dimen6} {\dimen2}
\setcoordinatemode}
```

That completes the macro definition. Now try it out:

```
\setcoordinatesystem units <.5in,.5in>
\crossbarlength=5pt
\puterrorbar at 0 2   with fuzz .5
\puterrorbar at 1 1.7 with fuzz .45
\puterrorbar at 2 1.5 with fuzz .35
```

produces



as intended.

**Exercise 56.** Devise a `\ploterrorbars` macro such that the preceding PICture could be drawn simply with

```
\setcoordinatesystem units <.5in,.5in>
\crossbarlength=5pt
\ploterrorbars   0 2 .5   1 1.7 .45   2 1.5 .35  /
```

**Exercise 57.** Revise the `\puterrorbar` macro so that it doesn't use dimension mode.

## 9.3. REGISTER ARITHMETIC

TEX's facilities for register arithmetic are basically limited to integer arithmetic. The rationale is that for any given document TEX should find the same line breaks and page breaks regardless of what computer it's running on, and integer arithmetic is the one thing different computers can be counted on to do alike. However it is hard to write macros to do graphics without being able to multiply and divide real numbers and without being able to calculate the Euclidean distance between two points on a page. This subsection presents PICTEX's commands for doing such things.

*Multiplication.* TEX lets you scale a dimension register by a fixed point number, as in '`2.2\dimen0`'. Thus you can multiply two fixed point numbers by putting one of them in a dimension register, in units of pts say, and by scaling that register by the other; the product will be in pts. For example, after

```
\dimen0=1.5pt
\dimen2=2.2\dimen0
```

the value of `\dimen2` is $3\frac{19660}{65536}$ pt $\doteq$ 3.29999 pt, the slight difference from the expected 3.3 pt being due to conversion from decimal to binary. If you adopt this technique to multiply numbers, you will often need to extract the value, in pts, of a given dimension register in order to use that value as the scaling factor in a further multiplication. PICTEX's command

`\placevalueinpts of <`*dimension register*`> in {`*control sequence*`}`

will handle the extraction: it places the value of the dimension register *dimension register*, in units of pts, in the control sequence *control sequence*. Continuing the example above, after

```
\placevalueinpts of <\dimen2> in {\factor}
```

`\factor` expands to 3.29999, and

```
\dimen4=\factor\dimen0
```

sets `\dimen4` to 4.94998 pt. The result isn't exact, but since displacements on the order of .01 pt are indiscernible to the eye, it's more than close enough for graphics.

*Division.* PICTEX's division command is

`\Divide <`*dividend*`> by <`*divisor*`> forming <`*quotient*`>`

- *dividend* and *divisor* may be explicit dimensions or dimension registers; *quotient* must be a dimension register.
- The division is done in units of pts, and is accurate to within 1/65536 pt.
- *divisor* must be less than 2048 pt (about 28 inches) in magnitude.

For example,

```
\Divide <3.3pt> by <2.2pt> forming <\dimen4>
```

sets `\dimen4` to 1.5 pt, as does

```
\dimen0=3.3pt
\dimen2=2.2pt
\Divide <\dimen0> by <\dimen2> forming <\dimen4>
```

Don't confuse `\Divide` with TEX's `\divide` command.

*Euclidean distance.* PICTEX's command

```
\placehypotenuse for <ξ> and <η> in <ζ>
```

sets $\zeta = \sqrt{\xi^2 + \eta^2}$.

- $\xi$ and $\eta$ may be explicit dimensions or dimension registers; $\zeta$ must be a dimension register.
- The calculation is done in units of pts.
- $|\xi| + |\eta|$ must be less than 2048 pt.

For example,

```
\placehypotenuse for <0.5pt> and <-1.2pt> in <\dimen4>
```

sets `\dimen0` to 1.29999 pt, as does

```
\dimen0=  .5pt
\dimen2=-1.2pt
\placehypotenuse for <\dimen0> and <\dimen2> in <\dimen4>
```

# 10. PICTEX AND LATEX

This section deals with the relationship between PICTEX and Leslie Lamport's LATEX. Subsection 10.1 describes how LATEX's lines, vectors, circles, and ovals can be used in a PICTEX PICture. Subsection 10.2 describes how PICTEX PICtures can be incorporated into a LATEX document. Both subsections presume that you've read about LATEX's pictures in Section 5.5 of the LATEX manual.

## 10.1. USING LATEX PICTURE OBJECTS IN A PICTURE

LATEX has commands that draw certain so-called *picture objects*—among them lines, vectors, circles, and ovals—by piecing together characters from specially designed fonts. Compared to their PICTEX counterparts, LATEX's picture objects are assembled much more rapidly and take up much less space in TEX's memory. However there are only a limited number of possible slopes[6] for the lines and vectors, a limited number of diameters for the circles[7] and rounded corners[8] of the ovals, and two possible line thicknesses.

If you're using PICTEX with plain TEX, you can gain access to the aforementioned LATEX picture objects without reading all the LATEX macros into

[6] See page 198 of the LATEX manual.

[7] LATEX's open circles have diameters running from 1 pt to 16 pts in steps of 1 pt, and from 20 pts to 40 pts in steps of 4 pts. The solid circles (i.e., disks) have diameters running from 1 pt to 15 pts in steps of 1 pt.

[8] LATEX's quarter/semi/full circles and ovals have rounded corners with diameters running from 4 pts to 40 pts in steps of 4 pts.

TEX's memory simply by `\input`ting the file `latexpicobjs.tex`.[9] This file contains the LATEX macros `\line`, `\vector`, `\circle`, and `\oval` along with `\thicklines` and `\thinlines`. Consult the LATEX manual for the syntax and usage of these commands.

To place the LATEX picture object *picture object* into a PICture, type

`\put {`*picture object*`} [Bl] at` *xcoord ycoord*

This command positions *picture object* as follows:

| *picture object* | *positioning* |
|---|---|
| line | line starts at (*xcoord*, *ycoord*) |
| vector | vector's tail is at (*xcoord*, *ycoord*) |
| circle/oval | center is at (*xcoord*, *ycoord*) |
| quarter/semi circle/oval | center of the corresponding full circle/oval is at (*xcoord*, *ycoord*) |

For example the commands

```
\setcoordinatesystem units <50pt,5pt>
\unitlength=1pt
\thicklines
\put {\line(-2,-5){20}}   [Bl] at 1 0
\thinlines
\put {\line (3, 1){30}}   [Bl] at 1 0
\put {\vector(-3,-4){20}} [Bl] at 3 0
\put {\circle*{15}}       [Bl] at 5 0
\thicklines
\put {\circle{40}}        [Bl] at 5 0
\put {\oval(300,120)}     [Bl] <0pt,-5pt> at 3 0
```

were among those used to construct Figure 22 below.

*SOME LATEX PICTURE OBJECTS*

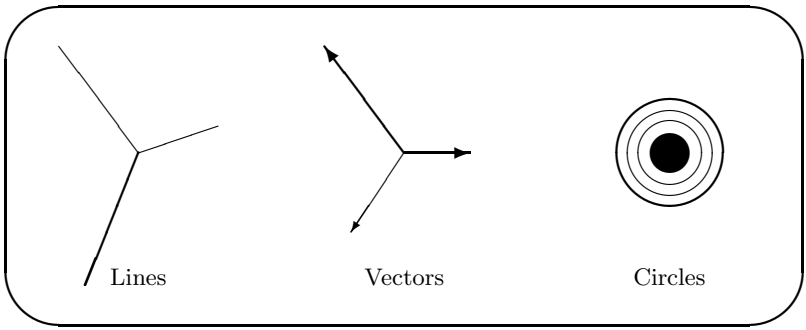


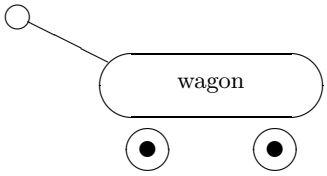

*FIGURE 22*

[9] The name of this file may vary from system to system; consult your local TEX guru.

- The size of a LaTeX picture object is controlled by the dimension register `\unitlength` in the manner described in the LaTeX manual. `\unitlength` can be different from the *xunit* and/or *yunit* specification of PiCTeX's `\setcoordinatesystem` command.
- The thickness of LaTeX's lines, vectors, circles, and ovals is determined by LaTeX's `\thinlines` and `\thicklines` commands, and bears no relation to PiCTeX's `\linethickness` parameter.
- PiCTeX's dot and dash patterns do not apply to LaTeX's picture objects.
- LaTeX and plain TeX define `\line` differently. The `\line` macro in the file `latexpicobjs.tex` resolves this difficulty as follows. If (as in the PiCture commands given above for Figure 22) the token `\line` is followed by the character '`(`', perhaps with intervening blanks, `\line` has its meaning in LaTeX. Otherwise, `\line` has its meaning in plain TeX.

**Exercise 58.** Give a complete set of instructions to draw Figure 22.

**Exercise 59.** Use LaTeX picture objects to draw




**Exercise 60.** The enclosing box for a LaTeX picture object is not always what you might naively expect. How could you find out what the enclosing box actually is?

### *10.2. USING PiCTeX PiCTURES IN A LaTeX DOCUMENT*

If you're using LaTeX to format your document, you can gain access to PiCTeX by `\input`ting the files[10] `prepictex.tex`, `pictex.tex`, and `postpictex.tex`, in that order. The second file contains the PiCTeX macros proper. The other two files make it possible for PiCTeX to coexist harmoniously with LaTeX's `picture` environment. In particular, pictures constructed solely with LaTeX's picture drawing commands will turn out just as they would if the PiCTeX macros weren't in TeX's memory, and PiCTeX PiCtures will turn out just as they would if the macros defining LaTeX's `picture` environment weren't in TeX's memory.

It's even possible to draw pictures that have both PiCTeX and LaTeX components. The preceding subsection illustrated how to place LaTeX picture objects in a PiCTeX PiCture. In large part that's all you need to know how to do, since with that technique in hand there isn't any picture you could draw in LaTeX's `picture` environment that you couldn't construct with PiCTeX.

[10] The names of these files may vary from system to system; consult your local TeX guru.

Should you so desire, you can `\put` any LaTeX picture into a PiCTeX PiCture, and vice versa.

You need to watch out for the commands

`\frame`, `\linethickness`, `\multiput`, and `\put` ☉☉

which have different meanings (syntax and/or function) in PiCTeX and in LaTeX. The macros in the files `pre/post pictex.tex` set things up so that which meaning is in force at any particular moment depends on which group or environment is active at that time. For example consider the code

```
\newcounter{num}                                  line  1
\beginpicture                                     line  2
  \setcoordinatesystem units <1pt,1pt>            line  3
  \put {%                                         line  4
    \setlength{\unitlength}{1pt}%                 line  5
    \begin{picture}(200,6)%                       line  6
      \multiput(0,0)(20,0){10}{%                  line  7
        \addtocounter{num}{1}\Alph{num}}%         line  8
    \end{picture}} [Bl]  at  0 0                  line  9
\endpicture                                       line 10
```

The `\put` on line 4 falls within the scope of the group begun by the PiCTeX `\beginpicture` command on line 2 and ended by the PiCTeX `\endpicture` command on line 10; this `\put` is therefore interpreted according to PiCTeX's rules. On the other hand, the `\multiput` on line 7 falls within the scope of the (sub-)group begun by the LaTeX `\begin{picture}` command on line 6 and ended by the LaTeX `\end{picture}` command on line 9; this `\multiput` is therefore interpreted according to LaTeX's rules. Any command other than those on the display above with the "watchful eyes" ☉☉ has its usual meaning all the time.

- Both PiCTeX and LaTeX normally allow `\frame` and `\linethickness` to be used outside of pictures. To resolve this ambiguity, the macros in `pre/post pictex.tex` restrict PiCTeX's meanings to PiCtures. Outside of a PiCture you can get the effect of PiCTeX's `\frame` and `\linethickness` commands by typing `\pictexframe` and `\pictexlinethickness` respectively.
- As a LaTeX user you shouldn't `\input` the file `latexpicobjs.tex` mentioned in the preceding subsection, and you should ignore the discussion about `\line` near the end of that subsection.
- The discussion in Section 8.1 concerning PiCtures in displays and inserts used constructs from plain TeX. To achieve the analogous effects in LaTeX, you should use the `displaymath` and `figure` environments.
- ○ As explained in Section 8.2, PiCTeX automatically keeps track of the size of a PiCture so that TeX can position it properly on the printed page. In LaTeX, however, part of the `\begin{picture}` command is used to tell TeX how much room to reserve for a picture. When a LaTeX picture is `\put` into a PiCture,

PICTEX bases its accounting on the reserved size (which is the only information available). If the reserved size is incorrect, i.e., different from the space the picture actually takes up, PICTEX's accounting may be thrown off.

**Exercise 61.** What is produced by the code following the ◐◐ display above?

**Exercise 62.** Give commands to draw the figure '◐◐'.

**Exercise 63.** What commands could you use to place a PICTEX PICture into a LATEX picture with the box enclosing the PICture centered about the LATEX coordinate $(20, 30)$?

# *APPENDICES*

## *A. ANSWERS TO ALL THE EXERCISES*

**1.** Untold hours of frustration.

**2.** The reference point of the first system is at its origin, while the reference point of the second system is 20 $x$-units to the left of its origin. Since 1 $x$-unit in the second system amounts to 10 pts, the origin of the second system is 200 pts to the right of that of the first system.

**3.** (1) He should have omitted the `[`'s and `]`'s. (He presumably skimmed Subsection 1.1 and so misinterpreted the [ ]'s that appear in the statement of the syntax of the `\setcoordinatesystem` command on page 3.) (2) He should have left a space after '`units`'. (3) He should have typed '`1in,2in`' instead of '`1inch,2inches`'. (Page 57 of *The TEXbook* lists the only ways dimensions can be written.) (4) He should have typed '`3 -2`' instead of '`(3,-2)`'.

**4.**
```
\put {$\bullet$} at 1 2
\put {$(1,2)$} [l] <10pt,0pt> at 1 2
```

**5.**
```
\put {\ninepoint$\bullet$} at 1 2
\put {\ninepoint $(1,2)$} [l] <10pt,0pt> at 1 2
```
or (better)
```
\ninepoint
\put {$\bullet$} at 1 2
\put {$(1,2)$} [l] <10pt,0pt> at 1 2
```

**6.** The author used

```
\setcoordinatesystem units <1pt,1pt>
\put {\vrule height .4pt width 300pt} [l] at 0 0
\multiput {\vrule height 18pt} [t] at  0 0  *3 100 0  /
\multiput {\vrule height 14pt} [t] at  0 0  *6  50 0  /
\multiput {\vrule height 10pt} [t] at  0 0  *30 10 0  /
\multiput {\vrule height  6pt} [t] at  5 0  *29 10 0  /
\put {0 pt}   [t] at   0 -24   \put {100 pt} [t] at 100 -24
\put {200 pt} [t] at 200 -24   \put {300 pt} [t] at 300 -24
```

**7.** You get: $(3,2)\ \rightarrow\ \times$ stacking is as simple as ABC

**8.** You get: $(3,2)\ \rightarrow\ \times$ Rows of lines are easy to put into a PiCture

**9.** (a) No. In the first case the word 'lines' sits just above the point $(3,2)$, whereas in the second case the word 'Two' sits just above this point. (b) Yes.

**10.** `\line` produces a single line of length `\hsize`, whereas `\lines` produces an array of lines, the width of the array being the length of the longest line.

**11.** You get: $(3,2)\ \rightarrow\ \times$ It's easy to put a paragraph into a PiCture.

**12.** (1) He didn't include the mandatory positioning keyword (e.g., `bottom`). (2) He forgot to enclose the axis label in `{}`'s (or maybe he meant to type '`{My First Axis}`'). (3) He should have typed '`<10pt>`'. (All dimensions are specified to PICTEX in `<>`'s). (4) The keyword '`withvalues`' is spelled without a space (as are '`shiftedto`', '`andacross`', and '`butnotacross`'). (5) He forgot the terminating '`/`' for the list of tick values. (6) He should have typed '`15.0`', since the other two numbers in his `from` specification have one digit to the right of the decimal point. (7) He forgot the terminating '`/`' for the `\axis` command.

**13.**
```
\newcount\scratchcount
\def\grid #1 #2 {%
  \scratchcount=#1\advance\scratchcount by 1
  \axis bottom invisible ticks
    length <0pt> andacross quantity {\scratchcount} /
  \scratchcount=#2\advance\scratchcount by 1
  \axis left   invisible ticks
    length <0pt> andacross quantity {\scratchcount} / }
```

**14.** The author used

```
\setcoordinatesystem units <2.5pt,30pt>
\setplotarea x from 0 to 100, y from 0 to 4.301
\ninepoint
\plotheading
  {\sl \lines {Number of words Shakespeare used\cr
  exactly $n$ times, for $n=\hbox{\sl 1}$ to 100 by 1.\cr}}
\axis bottom label {Frequency $n$ of usage} ticks
  numbered from 0 to 100 by 20
  short unlabeled quantity 11 /
\axis left shiftedto x=-5
  label {\stack {N,u,m,b,e,r,,,o,f,,,w,o,r,d,s}}
  ticks logged
    numbered at 1 10 100 1000 10000 /
    unlabeled short from 2 to 9 by 1  from 20 to 90 by 10
      from 200 to 900 by 100  from 2000 to 9000 by 1000
      at 20000 /  /
\def\extralongleftarrow{%          The "%" is mandatory here
  \longleftarrow\joinrel\relbar}%                  and here
\put {$\extralongleftarrow \hskip 6pt
  \vcenter{\hsize=100pt \raggedright \noindent
  \eightpoint%  (See Appendix E of the TeXbook for this)
  Shakespeare used 1043 words exactly 5~times.}$}
  [l] <10pt,0pt> at 5 3.0183
\accountingoff %  (Check the index)
```

```
\multiput {$\circ$} at "Shakespeare.tex"
\accountingon
```

**15.**
```
\setcoordinatesystem units <1pt,1pt>
\setplotarea x from 0 to 300, y from 0 to 0
\tickstovaluesleading=6pt
\axis bottom ticks
  withvalues {0 pt} {100 pt} {200 pt} {300 pt} /
    length <18pt> from 0 to 300 by 100
  unlabeled
    length <14pt> from 0 to 300 by  50
    length <10pt> from 0 to 300 by  10
    length < 6pt> from 5 to 295 by  10  /
```

**16.** The following code constructs the edge of the ruler and the major divisions. The rest of the construction is similar. The novel idea here is one of `\multiput`ting a subPICture into a PICture.

```
\setcoordinatesystem units <1pt,1pt>
\putrule from 0 0 to 300 0
\multiput {\beginpicture \putrule from 0 0 to 0 18 \endpicture}
   [t] at  0 0  *3 100 0  /
```

**17.** The following solution illustrates again how one PICture can be `\multiput`ted into another:

```
\setcoordinatesystem units <10pt,10pt>
\multiput
  {\beginpicture
     \linethickness=1pt
     \putrectangle corners at 0 0 and 4 2
   \endpicture}  [l]  at  3 2  10 2  6.5 -2  20 0  /
\multiput {\beginpicture  \putrule from 0 0 to 3 0  \endpicture}
    [l]  at  -3 0  0 2  7 2  14 2  17 0  24 0  /
\putrule from 0 -2 to 6.5 -2   \putrule from 10.5 -2 to 17 -2
\putrule from 0 -2 to 0    2   \putrule from 17   -2 to 17  2
\ninepoint
\put {$R_1$} at 5    2    \put {$R_2$}  at 12 2
\put {$R_3$} at 8.5 -2    \put {$R_4$}  at 22 0
```

**18.** It gives LEGEND, centered horizontally and vertically about $(100, 20)$.

**19.** You get: ← $(1, 2)$

**20.**
```
\def\rectangle <#1> <#2> {%
  \setbox0=\hbox{}\wd0=#1\ht0=#2\frame {\box0}}
```

**21.** The author used

```
\beginpicture
  \setcoordinatesystem units <.03125in,.25in>
  \ninepoint    \normalgraphs
  \setplotarea x from 0 to 80, y from 0 to 4
  \axis left shiftedto x=-5
    label {\stack {\%,,,p,e,r,,,c,i,g,a,r,e,t,t,e}} ticks
    length <3pt> numbered from 0 to 4 by 1  /
  \axis bottom label {Number of cigarettes} ticks
    length <0pt> numbered at 0 10 20 40 80 /  /
  \plotheading {\lines {\sl Cigarettes smoked per day\cr
    (adult males, current smokers, 1964)\cr}}
  \sethistograms
  \plot 0 0   10 1.5   20 3.5   40 1.5   80 0.5  /
  \eightpoint%  see Appendix E of The TeXbook
  \put {$(1.5)$} [b] <0pt,3pt> at   5  1.5
  \put {$(3.5)$} [b] <0pt,3pt> at  15  3.5
  \put {$(1.5)$} [b] <0pt,3pt> at  30  1.5
  \put {$(0.5)$} [b] <0pt,3pt> at  60  0.5
\endpicture
```

**22.**
```
\setcoordinatesystem units <25pt,25pt>
\putrule from -1 0 to 6 0
\linethickness=.8pt
\sethistograms
\plot   0 0   1 1   2 -1   3 -.5   4 1.5   5 .5  /
```

**23.** The author used

```
\midinsert
  \ninepoint
  \vbox{\narrower\noindent\sl The Binomial distribution for
    20~trials  ...  = e^{-4} 4^k/k!\,$.}
  \bigskip
  \centerline{%
    \beginpicture
      \normalgraphs
      \setcoordinatesystem units <.25in,5in>
      \setplotarea x from -.5 to 12.5, y from 0 to .22
      \longticklength=3pt
      \axis bottom label {$k$} ticks
        numbered from 0 to 12 by 1  /
      \axis left shiftedto x=-1 ticks
          withvalues .00 .05 .10 .15 .20 /
          from .00 to .20 by .05  /
      \setbars <-2pt,0pt> breadth <0pt> baseline at y = 0
      \linethickness=4pt
      \plot "Binomial.tex"
      \setbars < 2pt,0pt> breadth <4pt> baseline at y = 0
```

```
      \linethickness=.25pt
      \plot "Poisson.tex"
    \endpicture}
  }
\endinsert
```

**24.**
```
\setcoordinatesystem units <30pt,30pt>
\setlinear
\plot  0 1  .58779 -.80902  -.95106  .30902
            .95106  .30902  -.58779 -.80902   0 1  /
```

**25.** The main point here is that you can't have PICTEX draw a curve by simply giving it the formula for that curve — you have to supply explicit coordinate points. Here is the code that produced the figure.

```
\setcoordinatesystem units <.5in, 2.5in>
\setplotarea x from -3 to 3, y from 0 to .4
\ninepoint
\plotheading {\lines {%
  The density $\phi(\zeta) = e^{-\zeta^2\!/2}/\sqrt{2\pi}$ of the\cr
  standard normal distribution.\cr}}
\axis bottom ticks numbered from -3 to 3 by 1
  length <0pt> withvalues $\zeta$ / at 1.5 /  /
\linethickness=.25pt
\putrule from  1.5 0  to  1.5 .12952 %  (.12952 = density at 1.5)
\setbox0=\hbox{$\swarrow$}%
\put {$\swarrow$ \raise \ht0 \hbox{$\phi(\zeta)$}}
  [bl] at 1.5 .12952
\setquadratic  \plot
   0.0     .39894
   0.16667 .39344   0.33333 .37738    0.5 .35207   0.66667 .31945
   0.83333 .28191   1.      .24197   1.25 .18265   1.5     .12952
   1.75    .08628   2.      .05399   2.25 .03174   2.5     .01753
   2.75    .00909   3.0     .00443  /
\plot
   0.0     .39894   ...   -3.0     .00443  /
```

**26.**
```
\setplotsymbol ({\fiverm .})
\plotsymbolspacing=.4pt
```

**27.**
```
\frame <.1in>
  {\beginpicture
     \setcoordinatesystem  units <.25in,.25in>  point at 0 0
     \multiput
       {\beginpicture
          \circulararc 360 degrees from 1 0  center at 0 0
        \endpicture} at  0 0  1 0  /
     \put {$A$} <-6pt,5pt> at -.707  .707
     \put {$B$} <5pt,-5pt> at 1.707 -.707
   \endpicture}
```

**28.**
```
\setcoordinatesystem units <25pt,25pt>
\ellipticalarc  axes ratio 2:1  -315 degrees  from 1.9 .25
  center at 0 0
```

**29.** $c = 2\sqrt{(1.9 \times 25\,\text{pt}/2)^2 + (.25 \times 25\,\text{pt}/1)^2} = 49.1\,\text{pt}$.

**30.**
```
\setcoordinatesystem units <10pt,10pt>
\setplotarea x from -1 to 13, y from 0 to 1
\arrow <10pt> [.1, .4]  from  0 0  to  3 0
\arrow <10pt> [.2, .4]  from  5 0  to  8 0
\arrow <10pt> [.3, .4]  from 10 0  to 13 0
```

**31.** Neither line $A$ nor line $B$ will be drawn — each line lies outside the plot area that is current when it's `\plot`ted.

**32.** He lost all the saves resulting from the first `\savelinesandcurves` command.

**33.** Yes. PICTEX is by default in a non-saving mode. Hence the `\endpicture` command of a (main) PICture automatically returns PICTEX to that mode.

**34.** Line 1 will be saved on file `main.tex`, and Line 2 will be saved on file `sub.tex`. What happens to Line 3 is a bit subtle. The effect of the `\dontsavelinesandcurves` command in the subPICture is local to the subPICture, so when PICTEX starts working on Line 3 it is back in the saving mode established by the save command in the main PICture. However since the file specification of the save command in the subPICture is global, Line 3 will be saved on file `sub.tex`, not file `main.tex`. Since file `sub.tex` has to be `\replot`ted within the subPICture, this is not what's wanted. To remedy the situation a new `\savelinesandcurves` command (with a new file specification) should be be placed just before the `\plot` command for Line 3, or (better) Line 3 should be `\plot`ted before the subPICture so that it will be saved on file `main.tex`.

**35.** First of all, type

```
\newif\ifexpressmode
```

to set up three control sequences: `\ifexpressmode`, for testing an "express mode" switch; `\expressmodetrue`, for making the switch true; and `\expressmodefalse`, for making the switch false. Then for each line and/or curve you want to have the option of skipping, type

```
\ifexpressmode
\else
  code to \plot (or \replot) the line or curve
\fi
```

PICTEX will skip over the optional code if you type `\expressmodetrue` at the start of your document (but after the `\newif` command above), but not if you type `\expressmodefalse`.

**36.** He didn't interpolate enough points to adequately capture the shape of the cubic; what he got was

**37.** In the notation of Subsection 5.7, $\alpha_2 = 2\alpha_1$, so PICTEX thinks the arc length function $A(t)$ is linear, but it isn't. Again, more points need to be interpolated.

**38.** The solution below could be simplified using the commands `\setdotsnear` and `\setdashesnear` presented in Subsection 6.3.

```
\setcoordinatesystem units <.5in,.5in>
\setdashes <.05556in>
\putrule from  0 0  to  0 1   \putrule from  0 0  to  1 0
\dimen0=.5in  \dimen0=1.5708\dimen0
  \advance\dimen0 by -.05pt % To get a dot at the end of the arc
  \divide\dimen0 by 15
\setdots <\dimen0>
\circulararc  90  degrees from  1 0  center at  0 0
```

**39.** `\setdashes` is equivalent to '`\setdashpattern <5pt,5pt>`'; `\setdots` is equivalent to

```
\dimen0=5pt  \advance\dimen0 by -\plotsymbolspacing
\setdashpattern <\plotsymbolspacing, \dimen0>
```

**40.**
```
\dimen0=.25in
\setcoordinatesystem units <\dimen0,\dimen0>
\setplotarea x from 0 to 6, y from 0 to 3
\grid 1 1
\divide \dimen0 by 6
\setdashpattern <\dimen0,\dimen0,2\dimen0,\dimen0,\dimen0>
\linethickness=.25pt
\grid 6 3
```

**41.**
```
\dimen0=.25in
\setcoordinatesystem units <\dimen0,\dimen0>
\setplotarea x from 0 to 6, y from 0 to 3
\grid 1 1
\linethickness=.25pt  \longticklength=0pt
\setdashesnear <4pt> for <3\dimen0>
\axis bottom invisible ticks andacross from 1 to 5 by 1  /
\setdashesnear <4pt> for <6\dimen0>
\axis left   invisible ticks andacross at 1 2 /  /
```

**42.** In the original pattern, the long dash has length 10 pts, and the rest of the pattern takes up 12 pts. As Figure 16 shows, the adjusted pattern should be used in full four times, and the adjusted long dash once more afterwards. The following code does the job:

```
\setcoordinatesystem units <1pt,1pt>
\setplotarea x from 0 to 100, y from 0 to 25
\setquadratic
\findlength {\plot  0 0   50 8   100 25  /}
\dimen0=\totalarclength
\divide\dimen0 by 98 %   98 = 4*(10+12) + 10
\setdashpattern <10\dimen0, 2\dimen0, 3\dimen0, 2\dimen0, %
  3\dimen0, 2\dimen0>
\plot  0 0   50 8   100 25  /
```

**43.** `\setshadesymbol ({\fiverm .})`

**44.** (1) `\setshadesymbol <z,z,z,z> ({$\scriptscriptstyle\times$})`
(2) `\setshadesymbol ({$\scriptscriptstyle\times$})`

**45.** `\setshadegrid span <5pt> point at 0 0`

**46.** The author used

```
\setcoordinatesystem units <1in,1in>
\setplotarea x from 0 to 3, y from 0 to 1
\ninepoint  \normalgraphs
\axis bottom ticks
  numbered from 0 to 3 by 1
  length <0pt> withvalues $x$ /  at .5 /  /
\linethickness=.25pt  \putrule from  .5 0  to  .5 .60653
\putrule from  1 0  to  1 .36788  \putrule from  2 0  to  2 .13534
\put {$\scriptstyle\bullet$}   at .5 .60653
\put {$e^{-x}$} [bl] <4pt,4pt> at .5 .60653
\setquadratic
\plot  0 1  .25 .77880   .50 .60653   .75 .47237  1.00 .36788
          1.25 .28650  1.50 .22313  1.75 .17377  2.00 .13534
          2.25 .10540  2.50 .08208  2.75 .06393  3.00 .04979 /
\setshadegrid span <.025in>
\vshade  1 0 .36788   1.5 0 .22313   2 0 .13534  /
```

**47.** The author used

```
\def\shadecircle{%
\hshade -.707 -.707 -.707 <,z,,>  0 -1    -.707 .707 -.707 -.707 /
\vshade -.707 -.707  .707 <z,z,,> 0 -1    1     .707 -.707  .707 /
\hshade -.707  .707  .707 <z,,,>  0  .707 1     .707  .707  .707 /}
\frame <.1in>
  {\beginpicture
     \dimen0=.375in
     \setcoordinatesystem units <\dimen0,\dimen0>
     \multiput
       {\beginpicture
          \circulararc  360  degrees from  1 0  center at  0 0
          \divide \dimen0 by 11  \setshadegrid span <\dimen0>
          \setquadratic \shadecircle
        \endpicture}  at  0 0  1 0  /
     \put {$A$}  <-7pt,6pt> at -.707  .707
     \put {$B$}  <6pt,-6pt> at 1.707 -.707
   \endpicture}
```

**48.** Type

```
\setbox0=\hbox{\beginpicture ... \endpicture}
\immediate\write-1{%
  \noexpand\leavespace <\the\ht0,\the\dp0,\the\wd0>}
\box0
```

This will both draw the specified PICture and write a message in your log file of the form '`\leavespace <`*h*`,`*d*`,`*w*`>`', where *h*, *d*, and *w* are respectively the height, depth, and width of the enclosing box of the PICture.

**49.** To leave space for an hbox of height *h*, depth *d*, and width *w*, type

```
\leavespace <h,d,w>
```

where the `\leavespace` command is defined by

```
\def\leavespace <#1,#2,#3>{%
  \setbox0=\null
  \ht0=#1 \dp0=#2 \wd0=#3
  \box0 \ignorespaces}
```

**50.** First of all, type

```
\newif\ifpicturemode
```

to set up three control sequences: `\ifpicturemode`, for testing a "picture mode" switch; `\picturemodetrue`, for making the switch true; and `\picturemodefalse`, for making the switch false. Then for each PICture you want to have the option of skipping, type

```
\ifpicturemode
  code to construct the PICture
\else
  \leavespace <h,d,w>
\fi
```

where the `\leavespace` command is defined as in the answer to Exercise 49, and where *h*, *d*, and *w* are the height, depth, and width of the enclosing box of the PICture, determined during a previous run in the manner suggested in the answer to Exercise 48. If you type `\picturemodefalse` at the start of your input file (but after the `\newif` command above), PICTEX won't draw the optional PICtures but will leave the right amount of space for them. While you're revising things other than the optional PICtures, this will shorten the time TEX takes to process your document without affecting its decisions as to line and page breaks, etc. During production runs specify `\picturemodetrue` at the start of your input file; PICTEX will then draw the optional pictures.

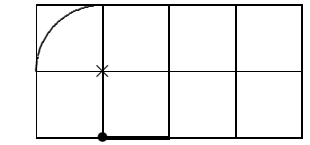
**51.** It would produce '∴'. With '`$\bullet$`' substituted for the period, the effect is more noticeable: '•'.

**52.** The enclosing box for subPiCture A is just what you'd think: the smallest box that the rule fits in. The first `\put` command shifts this box so that it is centered about the point $(0, 0)$, and so the rule ends up in the main PICture going between $(-50, 0)$ and $(50, 0)$. Now the reasoning gets a bit subtle. Since PICTEX's accounting procedure doesn't monitor lines and curves, the enclosing box for subPICture B is a rectangle of zero width and height about the reference point of the subPICture, which defaults to the reference point of the main PICture, namely $(50, 75)$. The second `\put` command shifts this box so that it is centered about $(0, 0)$, and so the line ends up in the main PICture going between $(50, -75)$ and $(150, -75)$.

**53.** The size of the symbol produced by the following code is fixed at the time the `\setbox` command is executed.

```
\newbox\thereforesymbol
\setbox\thereforesymbol=\hbox{%
  \beginpicture
    \setcoordinatesystem units <1ex,1ex> point at 0 0
    \multiput {.} [B] at  -.577 0   0 1   .577 0  /
  \endpicture}
\def\therefore{\copy\thereforesymbol}
```

**54.** You get:




**55.**
```
\setcoordinatesystem units <.25in,.25in>
\setplotarea x from -3 to 3, y from -1.5 to 1.5
\startrotation by .93969 .34202
\ellipticalarc  axes ratio 3:1  360 degrees from 3 0  center at 0 0
```

**56.**
```
\def\ploterrorbars#1 #2 #3 {%
  \puterrorbar at {#1} {#2} with fuzz {#3}
  \futurelet\nextcharacter\ploterrorbars}
\def\ploterrorbars{%
  \if /\nextcharacter
    \def\nextaction{\finish}%
  \else
    \def\nextaction{\ploterrorbars}%
  \fi
  \nextaction}
\def\finish/ {}
```

**57.** Here are a couple of solutions:

```
\def\puterrorbarA at #1 #2 with fuzz #3 {%
  \put {%
    \beginpicture
      \setcoordinatesystem point at 0 0
      \put {$\bullet$} at 0 0
      \putrule from 0 {-#3} to 0 {#3}
      \multiput {\vrule height\linethickness width\crossbarlength}
        at  0 {-#3}  0 {#3}  /
    \endpicturesave <\dimen1,\dimen3>}
      [Bl] <\dimen1,\dimen3> at {#1} {#2} }


\def\puterrorbarB at #1 #2 with fuzz #3 {%
  \put {$\bullet$} at {#1} {#2}
  \dimen0=\Ydistance{#3}
```

```
\put{\vbox{\hsize=\crossbarlength
  \hrule height \linethickness
  \vskip -.5\linethickness
  \centerline{\vrule width \linethickness height 2\dimen0}
  \nointerlineskip
  \vskip -.5\linethickness
  \hrule height \linethickness}} at {#1} {#2} }
```

**58.** See the instructions for Figure 22 in Appendix B.

**59.** The author used

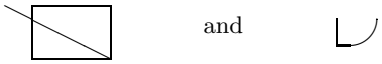
```
\beginpicture
  \setcoordinatesystem units <3pt,3pt>
  \setplotarea x from 0 to 38, y from -3 to 18
  \unitlength=3pt
  \multiput {\circle*{2}} [Bl] at  16  0   32 0  /
  \multiput {\circle{6}}  [Bl] at  16  0   32 0  /
  \put {\oval (28,8)}     [Bl] at  24  8
  \put {\ninerm wagon}         at  24  8
  \put {\line (-2,1){10}} [Bl] at  11 11
  \put {\circle{3}}       [Bl] <-3.9pt, 1.95pt> at   1 16
\endpicture
```

(Compare page 197 of the LaTeX manual.)

**60.** Type '`\frame {`*picture object*`}` '. For example, with `\unitlength=1pt`, the commands

```
\frame{\line(-2,1){40}}
\frame{\oval(30,20)[br]}
```

produce

 and 

respectively. Note how the line sticks out to the left of its enclosing box, and how the quarter-oval lies entirely outside its enclosing box (which has width 0).

**61.** This: A B C D E F G H I J .

**62.** The author used

```
\beginpicture
  \setcoordinatesystem units <1pt,1pt> point at 0 0
  \setplotarea x from -.5 to 14.5, y from 0 to 6
  \unitlength=1pt
  \accountingoff
  \multiput {\circle*{3}} [Bl] at  1.5 3   9.5 3 /
  \multiput {\circle{6}}  [Bl] at  3.0 3  11.0 3 /
\endpicture
```

**63.** This structure will do it:

```
\put(20,30){\makebox(0,0){%
  \beginpicture
    ⋮
  \endpicture}}
```

For other orientations of the PICture, see page 105 of the LaTeX manual.

## *B. HOW SELECTED FIGURES WERE CONSTRUCTED*

Given below is the code that was used to draw various figures in the manual. Some of the `\plot` commands have been abbreviated, omitted coordinates being indicated by '. . .'.

**Figure 1.** (page ii)

```
\newdimen\unit   \unit=1.375in
\newdimen\shadeunit
\def\DF{{\cal A}}%
\beginpicture
  \ninepoint  %   (See Appendix E of the TeXbook.)
  \normalgraphs
  % Density plot
  \setcoordinatesystem  units <\unit,.4\unit>  point at 0 0
  \setplotarea x from 0 to 1, y from 0 to 2.5
  \axis bottom invisible label {\lines {$t$\cr
    shaded area is $\DF(\beta) - \DF(\alpha)$\cr}} ticks
      numbered from 0.0 to 1.0 by 0.5
      unlabeled short quantity 11
      length <0pt> withvalues $\alpha$ $\beta$ / at .65 .85 /  /
  \axis left invisible label {$a(t)$} ticks
    numbered from 0.0 to 2.0 by 1.0
    unlabeled short from 0.5 to 2.5 by 1.0  /
  \plotheading{\lineskiplimit=1pt \lines{%
     Density\cr
     $a(t) = 1\big/\bigl(\pi \sqrt{t(1-t)}\,\bigr)$\cr
     of the arc sine law\cr}}
  \grid 1 1
  \putrule from  .65 0.0  to  .65 .66736
  \putrule from  .85 0.0  to  .85 .89145
  \shadeunit=.2\unit  \divide\shadeunit by 12
  \setshadegrid  span <\shadeunit>  point at .75 0
  \setquadratic
  \vshade .65 0 .66736   <,,,1pt> .75 0 .73511   .85 0 .89145  /
  % Move origin to (.5,0)
  \setcoordinatesystem  point at -.5 0
  \inboundscheckon
  \plot  -.48429 2.55990  -.47553 2.06015  -.46489 1.72936  /
  \inboundscheckoff
  \plot  -.46489 1.72936  -.43815 1.32146  -.40451 1.08308
         -.36448  .92999  -.31871  .82623  -.26791  .75400
         -.21289  .70358  -.12434  .65727   .0      .63662  /
  \inboundscheckon
  \plot   .48429 2.55990   .47553 2.06015   .46489 1.72936  /
  \inboundscheckoff
  \plot   .46489 1.72936         ...        .0      .63662  /
  %
  % Distribution function
```

```
  % Set origin of new coordinate system 1.7*1.375in=2.34in
  %   to the right of the original origin.
  \setcoordinatesystem  units <\unit,\unit>  point at -1.7 0
  \setplotarea x from 0 to 1, y from 0 to 1
  \axis bottom label {$x\vphantom{t}$} ticks
    numbered from 0.0 to 1.0 by 0.5 unlabeled short quantity 11 /
  \axis left label {$\DF(x)$} ticks
    numbered from 0.0 to 1.0 by 0.5 unlabeled short quantity 11 /
  \plotheading{\lines{%
    Distribution function\cr
    $\DF(x) = {2\over \raise1pt\hbox{\seveni ^^Y}}
      \arcsin(\sqrt{x}\,)$\cr
    of the arc sine law\cr}}
  \linethickness=.25pt  \grid {20} {20}
  \linethickness=.4pt   \grid 2 2
  % Now move origin of coordinate system up to (.5,.5)
  \setcoordinatesystem  point at -2.2 -.5
  \plot           -.50000 -.50   -.49901 -.48   -.49606 -.46
    -.49104 -.44  -.48439 -.42   -.47553 -.40   -.46489 -.38
    -.43815 -.34  -.40451 -.30   -.36448 -.26   -.31871 -.22
    -.26791 -.18  -.21289 -.14   -.12434 -.08    .0      .0  /
  \plot            .50000  .50       ...         .0      .0  /
\endpicture
```

**Figure 2.** (page 3)

```
\beginpicture
  \setcoordinatesystem  units <.5in,.25in>  point at 1.5 -2
  \setplotarea x from -2 to 4, y from -3 to 3
  \put {$\times$} at 1.5 -2
  \ninepoint  \normalgraphs  \ticksin
  \axis bottom shiftedto y=0 ticks
    numbered at -2 -1 /  from 1 to 4 by 1  /
  \axis left   shiftedto x=0 ticks
    numbered from -3 to -1 by 1  from 1 to 3 by 1  /
  \setdashes <.0625in>
  \putrule <0in,.0625in>  from  0  3  to   0  4
  \putrule <0in,-.0625in> from  0 -3  to   0 -4
  \putrule <-.0625in,0in> from -2  0  to  -3  0
  \putrule <.0625in,0in>  from  4  0  to   5  0
  \put {$\leftarrow\ \vcenter{\hbox{reference point\strut}}$}
    [l] <8pt,0pt> at 1.5 -2
\endpicture
```

**Figure 3.** (page 4)

```
\font\bigletters=cmssdc10 scaled 4300
\def\bigput{{\bigletters put}}
\beginpicture
  \setbox0=\hbox{\frame {\bigput}}%
```

```
  \dimen0=\wd0  \divide\dimen0 by 12
  \setcoordinatesystem  units <\dimen0,\dimen0>  point at 0 0
  \put {\copy0 } [Br] at 0 0
  \setdashes <\dimen0>
  \putrule from  -18.5 0  to  7.5 0
  \ninepoint
  \put {$\leftarrow\ \vcenter{\hbox{baseline\strut}}$}
    [l] at 8.5 0
\endpicture
```

**Figure 9.** (page 22)

```
\vbox{\ninepoint\lines {%
  \sl SUICIDE RATES IN WESTERN EUROPE\cr
  \sl per 100,000 population per year\cr
  \sl for the years (19xx) indicated\cr
  \noalign{\vskip 9pt}
  \beginpicture
    \setcoordinatesystem units <7pt,11pt>
    \setbars breadth <0pt> baseline at x = 0
      baselabels ([Br] <-5pt,-2pt>)
    \linethickness=2pt  \def\Yr#1{{\sevenrm 7#1}}%
    \plot
      24.1   0 "Austria      \Yr5"  23.8  -1 "Denmark     \Yr3"
      21.0  -2 "West Germany \Yr4"  15.4  -3 "France      \Yr0"
      14.9  -4 "Belgium      \Yr3"  10.6  -5 "Luxembourg  \Yr5"
       9.2  -6 "Netherlands  \Yr4"   8.6  -7 "Portugal    \Yr4"
       7.9  -8 "England      \Yr4"   5.8  -9 "Italy       \Yr2"
       4.0 -10 "Spain        \Yr4"   1.5 -11 "Switzerland \Yr5" /
    \linethickness=.25pt  \eightpoint
    \setplotarea x from 0 to 25, y from 1 to 1
    \axis top ticks numbered from 0 to 25 by 5  /
  \endpicture\cr}}
```

**Figure 12.** (page 29)

```
\beginpicture
  \ninepoint
  \dimen0=1.5708in \dimen2=.25in
  \setcoordinatesystem  units <\dimen0,\dimen2>
  \setplotarea x from 0 to 1, y from 0 to 4
  \longticklength=0pt  \tickstovaluesleading=.5\baselineskip
  \axis bottom invisible ticks
    withvalues 0 $\theta$ $\pi/2$ /  at 0 .75 1 /  /
  \axis left invisible ticks
    withvalues 0 $x$ 4 /  at 0 2.4142 4 /  /
  \grid 1 1
  \headingtoplotskip=1.25\baselineskip
  \plotheading {Graph of $x=\tan(\theta)$}
```

```
  \setquadratic  \plot  0 0
    .15  .24008   .30  .50953   .43  .80115   .55 1.17085
    .63 1.52235   .70 1.96261   .75 2.41421   .80 3.07768  /
  \inboundscheckon
  \plot  .80 3.07768   .825 3.54573   .85 4.16530  /
  \linethickness=.25pt
  \setdashesnear <5pt> for <2.41421\dimen2>
  \putrule from  .75 0  to  .75 2.41421
  \setdashesnear <5pt> for <.75\dimen0>
  \putrule from  0 2.41421  to  .75 2.41421
\endpicture
```

**Figure 13.** (page 32)

```
\beginpicture
  \ninepoint
  \setcoordinatesystem units <1pt,1pt>
  \setplotarea x from 0 to 150, y from -5 to 75
  \axis bottom ticks
    withvalues 0 1 2 / quantity 3
    length <0pt> withvalues $t$ /  at 40 /  /
  \setquadratic
  \plot  0 10   75 32   150 58  /
  \plot  0 75   75 53   150 15  /
  \multiput {$\scriptstyle\times$} at
    0 10   75 32   150 58   0 75   75 53   150 15   /
  \put {$x_0$} [t] <0pt,-5pt> at   0 10
  \put {$x_1$} [t] <0pt,-5pt> at  75 32
  \put {$x_2$} [t] <0pt,-5pt> at 150 58
  \put {$y_0$} [b] <0pt, 5pt> at   0 75
  \put {$y_1$} [b] <0pt, 5pt> at  75 53
  \put {$y_2$} [b] <0pt, 6pt> at 150 15
  \put {$\scriptscriptstyle\bullet$} at  40 21.236
  \put {$x(t)$} [t] <0pt,-5pt>       at  40 21.236
  \put {$\scriptscriptstyle\bullet$} at  40 65.258
  \put {$y(t)$} [b] <0pt, 5pt>       at  40 65.258
\endpicture
```

**Figure 14.** (page 33)

```
\beginpicture
  \setcoordinatesystem units <1.5pt,1.5pt>
  \setplotarea x from 10 to 58, y from 15 to 75
  \setquadratic
%  Of the remaining instructions, those that are preceded by
%    a '%' were used in the original construction of the figure.
%    See Subsection 5.6 on replotting.
%  \savelinesandcurves on "Equalspacing1.tex"
%  \writesavefile {Arc for Figure 14}
%  \plot  10 75   32 53   58 15  /
```

```
  \multiput {$\times$}  at  10 75   32 53   58 15  /
%  \arrow <3pt> [.2,.8] <-5pt,0pt> from   5 75  to  10 75
%  \arrow <3pt> [.2,.8] <5pt,0pt>  from  37 53  to  32 53
%  \arrow <3pt> [.2,.8] <5pt,0pt>  from  63 15  to  58 15
  \replot "Equalspacing1.tex" % Contains saved locations from
                              % the preceding \plot and \arrow's
  \ninepoint
  \put {$(x_0,y_0)$} [r] <-16pt,0pt> at 10 75
  \put {$(x_1,y_1)$} [l]  <16pt,0pt> at 32 53
  \put {$(x_2,y_2)$} [l]  <16pt,0pt> at 58 15
  \put
    {\beginpicture
       \setcoordinatesystem  units <1pt,1pt>  point at 0 0
       \startrotation by .69486 .71915 about 0 0
%       \savelinesandcurves on "Equalspacing2.tex"
%       \writesavefile {Arrow for "C" in Figure 14}
%       \arrow <3pt> [.2,.8] from  -7.5 0  to  0 0
       \replot "Equalspacing2.tex" %  The preceding \arrow
       \put {$C$} at -15 0
     \endpicturesave <\dimen1,\dimen3>}
       [Bl] <\dimen1,\dimen3> at 22.05 64.42
  \put
    {\beginpicture
       \setcoordinatesystem  units <1pt,1pt>  point at 0 0
       \startrotation by .83340 .55267 about 0 0
%       \savelinesandcurves on "Equalspacing3.tex"
%       \writesavefile {"Delta configuration" for Figure 14}
%       \setdashes <4pt>
%       \setlinear
%       \plot  0 0   -20 0  /   \plot 0 10   -20 10  /
%       \setsolid
%       \arrow <3pt> [.2,.8] from  -18 -7.5  to  -18  0
%       \arrow <3pt> [.2,.8] from  -18 17.5  to  -18 10
       \replot "Equalspacing3.tex" %  The preceding  2 \plot's
                                   %    and  2 \arrow's
       \put {$\delta$} at -18 5
     \endpicturesave <\dimen1,\dimen3>}
       [Bl] <\dimen1,\dimen3> at 49.547 28.432
  \put {\rlap{$\smash{p_i}$}$\phantom{p}$}
    <5pt,3.3pt> at 49.547 28.432
  \dontsavelinesandcurves
  \setplotsymbol ({$\scriptscriptstyle\bullet$})
  \setdots <10pt>
  \plot  10 75   32 53   58 15  /
\endpicture
```

**Figure 15.** (page 33)

```
\beginpicture
  \ninepoint
  \setcoordinatesystem units <75pt,100pt>
  \setplotarea x from -1 to 1, y from -.4 to .4
  \setquadratic   \plot  0 0
    .15 -.14663   .3 -.27300   .45 -.35888    .57735 -.38490
    .70 -.35700   .8 -.28800   .90 -.17100   1.0     0.0  /
  \plot  0 0               ...              -1.0     0 0  /
  \longticklength=0pt  \tickstovaluesleading=6pt
  \axis bottom shiftedto y=0 ticks
    numbered at -1 0 /  withvalues $x$ / at -.57735 /  /
  \axis top shiftedto y=0 invisible ticks numbered at 1 /  /
  \put {$y=x(x^2-1)$} [b] <0pt,5pt> at -.57735 .38490
  \setdashesnear <5pt> for <38.49pt>
  \putrule from  -.57735 0 to  -.57735 .38490
\endpicture
```

**Figure 17.** (page 39)

```
\beginpicture
  \setcoordinatesystem units <1pt,1pt>
  \setplotarea x from 0 to 90, y from 0 to 90
  \circulararc  360  degrees from  45 90  center at  45 45
  \ninepoint
  \put {$\bullet$} at 35 40
  \arrow <6pt> [.15,.6] from  35 40  to  17 40
  \arrow <6pt> [.15,.6] from  35 40  to  35 20
  \arrow <6pt> [.15,.6] from  35 40  to  63 40
  \arrow <6pt> [.15,.6] from  35 40  to  35 62
  \put {\rlap{$\smash{\epsilon_l}$}%
    $\vphantom{\epsilon}\hphantom{\epsilon_l}$}
    [r] <-5pt,0pt> at 17 40
  \put {\rlap{$\epsilon_d$}$\phantom{\epsilon}$}
    [t] <0pt,-5pt> at  35 20
  \put {\rlap{$\smash{\epsilon_r}$}$\phantom{\epsilon}$}
    [l] <5pt,0pt>  at  63 40
  \put {\rlap{$\epsilon_u$}$\phantom{\epsilon}$}
    [b] <0pt,5pt>  at  35 62
  \put {$R$} at  70 70
\endpicture
```

**Figure 18.** (page 40)

```
\beginpicture
  \setcoordinatesystem units <.375in,.375in>
  \setplotarea x from -2.5 to 2.5, y from -1.5 to 2.5
  \linethickness=.25pt  \longticklength=0pt
  \axis bottom invisible ticks andacross from -2 to 2 by 1  /
```

```
  \axis left   invisible ticks andacross from -1 to 2 by 1  /
  \ninepoint
  \setshadesymbol ({$\bullet$})  \setshadegrid span <.375in>
  \setlinear \hshade -1.5 -2.5 2.5  2.5 -2.5 2.5  /
  \put {\hbox to .375in{$\mathord\leftarrow\hfil
    \vcenter{\hbox{$\sigma$\strut}}\hfil\mathord\rightarrow$}}
    [l] at 1 -1.25
  \put {$\downarrow$}  [b] at 2.25 -1
  \put {$\sigma$}          at 2.25  -.5
  \put {$\uparrow$}    [t] at 2.25  0
  \put {$(x,y)$} [br] <-2pt,3pt> at 0 0
  \put {A shading lattice} [B] <0pt,10pt> at 0 2.5
\endpicture
```

**Figure 19.** (page 41)

```
\def\dottedline #1 #2 #3 #4 {%
  \findlength {\plot #1 #2 #3 #4 /}
  \setdotsnear <5pt> for <\totalarclength>
  \plot #1 #2 #3 #4 /}
\def\xylabel#1#2{(x_{#1},y_{#1}^{(#2)})}
\beginpicture
  \setcoordinatesystem units <.75pt,.75pt>
  \ninepoint
  \multiput {$\scriptstyle\bullet$} at
     0 0   80 -15   160 -5   0 40   80 50   160 45  /
  \put {$R_1$} at 40 20     \put {$R_2$} at 120 20
  \setlinear
  \dottedline 0 0 0 40     \dottedline 80 -15 80 50
    \dottedline 160 -5 160 45
  \dottedline 0 0 80 -15   \dottedline 80 -15 160 -5
    \dottedline 160 -5 200 -8
  \dottedline 0 40 80 50   \dottedline 80 50 160 45
    \dottedline 160 45 200 48
  \put {$\xylabel0b$} [t] <0pt,-7pt> at   0   0
  \put {$\xylabel1b$} [t] <0pt,-7pt> at  80 -15
  \put {$\xylabel2b$} [t] <0pt,-7pt> at 160  -5
  \put {$\xylabel0t$} [b] <0pt, 7pt> at   0  40
  \put {$\xylabel1t$} [b] <0pt, 7pt> at  80  50
  \put {$\xylabel2t$} [b] <0pt, 7pt> at 160  45
\endpicture
```

**Figure 20.** (page 45)

```
\beginpicture
  \setcoordinatesystem units <1pt,1pt>
  \putrectangle corners at -20 -30  and   50 30
  \putrectangle corners at  20  25  and   70 70
  \putrectangle corners at  55  10  and  100 50
  \putrectangle corners at  35 -10  and   85 17
```

```
  \putrectangle corners at   0   8  and   10 21
  \linethickness=1pt
  \putrectangle corners at -20 -30  and  100 70
  \linethickness=.4pt
  \ninepoint
  \setbox0=\hbox{$\swarrow$}%
  \put {$\swarrow$ \raise\ht0\hbox{$B$}} [bl] at 100 70
  \setdashes
  \putrule from  -72.5 0  to  115 0
  \put {$\times$} at -60 0
  \put {\Lines {reference\cr point\cr}}
    [B] <0pt,-13pt> at -60 0
  \put {\strut baseline} [l] at 120 0
\endpicture
```

**Figure 21.** (page 49)

```
\beginpicture
  \ninepoint
  \def\basicfigure{%
    \put {$\bullet$} at 0 1  \put {$F=(0,1)$} [b] <0pt,5pt> at 0 1
    \setquadratic  \setsolid
    \plot  0 0   .5 .0625   1 .25   1.75 .76563   2.5 1.56250  /
    \plot  0 0  -.5 .0625  -1 .25  -1.75 .76563  -2.5 1.56250  /
    \arrow <6pt> [.15,.6] from  1.5 1.6  to  1.5 1.1
    \setlinear
    \plot  1.5 1.1   1.5 .56250  /
    \arrow <6pt> [.15,.6] from  1.5 .56250  to  .75 .78125
    \plot  .75 .78125   0 1  /
    \setdashesnear <5pt> for <28pt>
    \plot  0 0  0 1  /
    \put {} at -2.5 1.56250   \put {} at 2.5 1.56250 }
  %
  % "before rotation"
  \setcoordinatesystem units <28pt,28pt> point at 0 0
  \put {\Lines {The parabola $y=x^2\!/4$\cr  before rotation\cr}}
    [B] <0pt,-18pt> at 0 0
  \basicfigure
  %
  % "after rotation"
  \setcoordinatesystem point at -5.5 0
  \put {\Lines {After rotation about the\cr
    focus $F$ by 15 degrees\cr}}
    [B] <0pt,-18pt> at 0 0
  \startrotation  by .96593 -.25882  about  0 1
  \basicfigure
\endpicture
```

**Figure 22.** (page 54)

```
\vbox\bgroup
  \ninepoint
  \line{\hfil\sl SOME \LaTeX\ PICTURE OBJECTS\hfil}  % (Notice
        % the different uses of  \line  here and below.)
  \medskip
  \hbox to \hsize\bgroup\hss
    \beginpicture
      \setcoordinatesystem units <50pt,5pt>
      \setplotarea x from 0 to 6, y from -13 to 11
      \unitlength=1pt
      \thinlines
      \put {Lines}               [B]  at 1 -10
      \put {\line (3, 1){30}}    [Bl] at 1 0
      \put {\line(-3, 4){30}}    [Bl] at 1 0
      \thicklines
      \put {\line(-2,-5){20}}    [Bl] at 1 0
      \put {Vectors}             [B]  at 3 -10
      \put {\vector( 1, 0){25}} [Bl] at 3 0
      \put {\vector(-3, 4){30}} [Bl] at 3 0
      \thinlines
      \put {\vector(-2,-3){20}} [Bl] at 3 0
      \put {Circles}             [B]  at 5 -10
      \put {\circle*{15}}        [Bl] at 5 0
      \put {\circle{24}}         [Bl] at 5 0
      \put {\circle{32}}         [Bl] at 5 0
      \thicklines
      \put {\circle{40}}         [Bl] at 5 0
      \put {\oval(300,120)}      [Bl] <0pt,-5pt> at 3 0
    \endpicture
  \hss\egroup
\egroup
```

## C. QUICK REFERENCE GUIDE

This appendix lists PICTEX's commands and parameters alphabetically, so you can easily check the spelling and syntax of the PICTEX instructions you're writing.

- In typing a command, you must use at least one blank wherever the prototype command has a blank.
- Quantities in <>'s must be specified as explicit dimensions (e.g., 1 in, or 0 pt), or in terms of TEX's dimension registers. If you use any dimension registers other than `\dimen0` ... `\dimen9`, be sure to allocate them with TEX's `\newdimen` command.
- *coord*, *xcoord*, *ycoord*, $x$, and $y$, with or without subscripts or superscripts, denote coordinates with respect to the current coordinate system. In particular, they are dimensionless quantities. Values must be expressed in fixed point notation, with at most 5 digits to the right of the decimal point.
- Parts of a command enclosed in [ ]'s may be omitted.

*COMMANDS*

`\accountingoff`

`\accountingon`

`\arrow <`$\ell$`> [`$\beta$`,`$\gamma$`] [<`*xshift*`,`*yshift*`>] from` $xcoord_s$ $ycoord_s$ `to` $xcoord_e$ $ycoord_e$

`\axis [bottom] [top] [left] [right]`
`[shiftedto y=`*ycoord*`] [shiftedto x=`*xcoord*`]`
`[visible] [invisible]`
`[label {`*axis label*`}]`
`[ticks]`
`[out] [in]`
`[long] [short] [length <`*length*`>]`
`[width <`*width*`>]`
`[butnotacross] [andacross]`
`[unlabeled] [numbered] [withvalues` $value_1$ $value_2$ ... `/]`
`[unlogged] [logged]`
`[quantity` $q$`] [from` $coord_s$ `to` $coord_e$ `by` *dcoord*`]`
`[at` $coord_1$ $coord_2$ ... `/]`
`/`

`\beginpicture`

`\betweenarrows {`*text*`} [[`$o_x$`][`$o_y$`]] [<`*xshift*`,`*yshift*`>] from` $xcoord_s$ $ycoord_s$
`to` $xcoord_e$ $ycoord_e$

`\circulararc` $\theta$ `degrees from` $xcoord_s$ $ycoord_s$ `center at` $xcoord_c$ $ycoord_c$

`\Divide <`*dividend*`> by <`*divisor*`> forming <`*quotient*`>`

`\dontsavelinesandcurves`

`\ellipticalarc axes ratio` $\xi$`:`$\eta$ $\theta$ `degrees from` $xcoord_c$ $ycoord_s$
`center at` $xcoord_c$ $ycoord_c$

`\endpicture`

`\endpicturesave <`*xreg*`,`*yreg*`>`

`\findlength {`*curve commands*`}`

`\frame [<`*separation*`>] {`*text*`}`

`\grid {`$c$`} {`$r$`}`

\gridlines
\hshade $y_0\ x_0^{(l)}\ x_0^{(r)}\ \ldots\ [\langle\epsilon_{l;i},\epsilon_{r;i},\epsilon_{d;i},\epsilon_{u;i}\rangle]\ y_i\ x_i^{(l)}\ x_i^{(r)}\ \ldots$ /
\hshade $y_0\ x_0^{(l)}\ x_0^{(r)}\ \ldots\ [\langle\epsilon_{l;i},\epsilon_{r;i},\epsilon_{d;i},\epsilon_{u;i}\rangle]\ y_{2i-1}\ x_{2i-1}^{(l)}\ x_{2i-1}^{(r)}\ y_{2i}\ x_{2i}^{(l)}\ x_{2i}^{(r)}$
... /
\inboundscheckoff
\inboundscheckon
\invisibleaxes
\lines [[*o*]] {*line*$_1$\cr *line*$_2$\cr ... }
\Lines [[*o*]] {*line*$_1$\cr *line*$_2$\cr ... }
\loggedticks
\multiput {*text*} [[[$o_x$][$o_y$]]] [<*xshift*,*yshift*>] at "*file name*"
\multiput {*text*} [[[$o_x$][$o_y$]]] [<*xshift*,*yshift*>] at ... *xcoord ycoord* ...
*n *dxcoord dycoord* ... /
\nogridlines
\normalgraphs
\placehypotenuse for <$\xi$> and <$\eta$> in <$\zeta$>
\placevalueinpts of <*dimension register*> in {*control sequence*}
\plot "*file name*"
\plot $xcoord_0\ ycoord_0\ \ xcoord_1\ ycoord_1\ \ xcoord_2\ ycoord_2\ \ \ldots$ /
\plotheading {*heading*}
\put {*text*} [[[$o_x$][$o_y$]]] [<*xshift*,*yshift*>] at *xcoord ycoord*
\putbar [<*xshift*,*yshift*>] breadth <$\beta$> from $xcoord_s\ ycoord_s$
to $xcoord_e\ ycoord_e$
\putrectangle [<*xshift*,*yshift*>] corners at $xcoord_s\ ycoord_s$
and $xcoord_e\ ycoord_e$
\putrule [<*xshift*,*yshift*>] from $xcoord_s\ ycoord_s$ to $xcoord_e\ ycoord_e$
\rectangle <*w*> <*h*>
\replot "*file name*"
\savelinesandcurves on "*file name*"
\setbars [<*xshift*,*yshift*>] breadth <$\beta$> baseline at $z$ = *zcoord*
[baselabels ([[[$o_x$][$o_y$]]] [<*xshift*,*yshift*>])]
[endlabels ([[[$o_x$][$o_y$]]] [<*xshift*,*yshift*>])]
\setcoordinatemode
\setcoordinatesystem [units <*xunit*,*yunit*>] [point at *xcoord ycoord*]
\setdashes [<$\ell$>]
\setdashesnear <$\ell$> for <$\lambda$>
\setdashpattern <$d_1,g_1,d_2,g_2,\ldots$>
\setdimensionmode
\setdots [<$\ell$>]
\setdotsnear <$\ell$> for <$\lambda$>
\sethistograms
\setlinear
\setplotarea x from $xcoord_l$ to $xcoord_r$, y from $ycoord_b$ to $ycoord_t$
\setplotsymbol ({*plot symbol*} [[[$o_x$][$o_y$]]] [<*xshift*,*yshift*>])
\setquadratic

\setshadegrid [span <*s*>] [point at *xcoord ycoord*]
\setshadesymbol [<$\epsilon_l$,$\epsilon_r$,$\epsilon_d$,$\epsilon_u$>] ({*shade symbol*} [[$o_x$][$o_y$]]
[<*xshift*,*yshift*>])
\setsolid
\shaderectanglesoff
\shaderectangleson
\stack [[*o*]] [<*leading*>] {*list*}
\startrotation [by $\cos(\theta)$ $\sin(\theta)$] [about $x_p$ $y_p$]
\stoprotation
\ticksin
\ticksout
\unloggedticks
\visibleaxes
\vshade $x_0$ $y_0^{(b)}$ $y_0^{(t)}$ ... [<$\epsilon_{l;i}$,$\epsilon_{r;i}$,$\epsilon_{d;i}$,$\epsilon_{u;i}$>] $x_i$ $y_i^{(b)}$ $y_i^{(t)}$ ... /
\vshade $x_0$ $y_0^{(b)}$ $y_0^{(t)}$ ... [<$\epsilon_{l;i}$,$\epsilon_{r;i}$,$\epsilon_{d;i}$,$\epsilon_{u;i}$>] $x_{2i-1}$ $y_{2i-1}^{(b)}$ $y_{2i-1}^{(t)}$ $x_{2i}$ $y_{2i}^{(b)}$ $y_{2i}^{(t)}$
... /
\writesavefile {*message*}
\Xdistance{*xcoord*}
\Ydistance{*ycoord*}

*PARAMETERS*

\headingtoplotskip
\linethickness
\longticklength
\plotsymbolspacing
\shortticklength
\stackleading
\tickstovaluesleading
\totalarclength
\valuestolabelleading

*MISCELLANEOUS*

\PiC
\PiCTeX

*D. BIBLIOGRAPHY*

Knuth, Donald E. *The TEXbook*. Reading: Addison-Wesley, 1984.

Lamport, Leslie *LATEX: A Document Preparation System*. Reading: Addison-Wesley, 1986.

*E. INDEX*

Underlined page numbers give the main source of information about whatever is being indexed. Pages with *slanted* numbers contain example(s) of the use of the concept in question.