# Contents

19.1 Coordinates and Placement ................................................. 2
  19.1.1 Changing Coordinates ........................................... 2
  19.1.2 Placing and Rotating Objects .................................... 4
19.2 Text ................................................................. 5
  19.2.1 Frameboxes ...................................................... 5
  19.2.2 Placing Text .................................................... 7
19.3 Grids and Axes ......................................................... 9
  19.3.1 Drawing Grids .................................................... 9
  19.3.2 Drawing Axes ................................................... 10
19.4 Plotting Data ........................................................... 13
  19.4.1 Plotting Data from a File ....................................... 13
  19.4.2 Plotting Data Directly ......................................... 14
19.5 Diagrams ............................................................... 15
19.1 Coordinates and Placement

We saw in the previous lecture how the \texttt{pspicture} environment sets up a coordinate system and how to use \texttt{psset} to change the default unit of 1 cm.

19.1.1 Changing Coordinates

Here we will look at some more commands for changing coordinates.

Change of Origin

The command
\begin{verbatim}
origin={x,y}
\end{verbatim}
(note the curly braces) translates the coordinate axes so that the origin of the original system is at point \((x, y)\) with respect to the new system. In practical terms, if we want to move the origin to the point \((x, y)\) set \texttt{origin=\{-x, -y\}}. As with other commands of this type, it can be used either with the \texttt{psset} command or as a parameter setting to a command. Note also that successive changes of origin do not accumulate.

\begin{verbatim}
\begin{center}
\begin{pspicture}(0,0)(4,4)
\showgrid
\psset{origin={-2,-2}} % move origin to (2,2)
\psframe[origin={-1,-1}](0,0)(2,2) % origin temporarily at (1,1)
\pscircle(0,0){1} % origin at (2,2)
\end{pspicture}
\end{center}
\end{verbatim}

Polar Coordinates

The command \texttt{SpecialCoor} allows the use of polar coordinates in the form \((r; \theta)\) (note the semicolon) mixed in with the usual Cartesian coordinates.

Here is a way to draw a pentagon with center at \((2, 2)\), a radius of 2 and with vertices at 18°, 90°, 162°, 234° and 306°:
By taking every second vertex of a pentagon we get a star:
Two points to note in the above example:

1. Using \{pspicture\}(-2,-2)(2,2) sets the center of the picture to (0,0).

2. The star forms \texttt{\textbackslash pscircle*} and \texttt{\textbackslash pspolygon*} were used. In this case \texttt{linecolor} is used to determine the color of the fill. Note also that the outline of the shape is not drawn (compare with the examples in the previous lecture).

**Changing Angular Units**

By default angles are measured in degrees. This can be changed with command

\begin{verbatim}
\setlength{\unitlength}{1cm}
\begin{pspicture}(0,0)(2,2)
\SpecialCoor
\degrees[6.28319]
\psgrid
\pspolygon(0;0)(0;1)(0;2)(0;3)(0;4)(0;5)(0;6)
\end{pspicture}
\end{verbatim}

Changing Angular Units

By default angles are measured in degrees. This can be changed with command

\begin{verbatim}
\setlength{\unitlength}{1cm}
\begin{pspicture}(0,0)(2,2)
\SpecialCoor
\degrees[6.28319]
\psgrid
\pspolygon(0;0)(0;1)(0;2)(0;3)(0;4)(0;5)(0;6)
\end{pspicture}
\end{verbatim}

where \texttt{number} is the number of parts into which the circle is divided. For example \texttt{\degrees[6.28319]} would change the angular measurement to radians (there are 2π radians in a circle).

Here is a heptagon drawn without calculating the angles:

\begin{verbatim}
\setlength{\unitlength}{1cm}
\begin{pspicture}(0,0)(2,2)
\SpecialCoor
\degrees[7]
\psgrid
\pspolygon(2;0)(2;1)(2;2)(2;3)(2;4)(2;5)(2;6)
\end{pspicture}
\end{verbatim}

19.1.2 Placing and Rotating Objects

The \texttt{\textbackslash rput} command can be used to place something at specified position and angle. It is used in the form:

\begin{verbatim}
\rput{angle}(x,y){stuff}
\end{verbatim}
which rotates stuff through angle \texttt{angle} and puts it at position \((x,y)\).

Here is a simple example:

\begin{center}
\begin{pspicture}(0,0)(4,4)
\showgrid
\psframe(0,0)(2,1)
\rput{30}(2,2){\psframe(0,0)(2,1)}
\end{pspicture}
\end{center}

\section{19.2 Text}

\subsection{19.2.1 Frameboxes}

\textsc{PSTricks} has some commands for dealing with text that can also be useful even outside the \texttt{pspicture} environment. The most used are \texttt{psframebox} and its variants. Here is a simple example:

\begin{center}
\psframebox{Did you know $e^{i \pi} = -1$}
\end{center}

You can nest frameboxes, use colours etc.:

\begin{center}
{\Huge
\psframebox[linecolor=blue]{{\color{red} Did you know} \\
\psframebox[linecolor=red]{{\color{blue} $e^{i \pi} = -1$}}}
}
\end{center}

\textbf{Did you know $e^{i\pi} = -1$}
There are several variations on the \texttt{psframebox} command. Most of the names are self-explanatory:

1. \texttt{\psshadowbox}
2. \texttt{\pscirclebox}
3. \texttt{\psdblframebox}: draws a double frame.
4. \texttt{\pstribbox}: triangular box.
5. \texttt{\psovalbox}

Here are a couple of examples:

\begin{center}
\psshadowbox{Did you know $e^{i \pi} = -1$}
\end{center}

\begin{center}
\pscirclebox[linecolor=red,doubleline=true]{STOP}
\end{center}

**Important Note:**

If you try this example you may find that the DVI viewer doesn’t show the text inside the box, but that when you make the pdf file (via postscript if you are using Windows) it appears as it does in these notes. This seems to be an occasional but very annoying problem: when mixing \LaTeX\ and postscript graphics the DVI viewer sometimes drops or misplaces text. What makes this worse is that constructing graphics is an interactive process where you build up your picture bit by bit, viewing the intermediate results as you go. Besides the above example, all the examples in these notes can be viewed correctly by the DVI viewer.

\begin{center}
\pscirclebox[linecolor=red,doubleline=true]{STOP}
\end{center}
19.2.2 Placing Text

The `rput` command works with text as well as graphics:

\begin{center}
\begin{pspicture}(0,0)(4,4)
\showgrid
\pspolygon(0,0)(2,4)(4,0)
\rput(1,3){Here is some text}
\rput(3,1){\psframebox{Here is more text}}
\end{pspicture}
\end{center}

The coordinates of `rput` correspond to the center of the text.

As noted above there are some problems with the DVI viewer. Two more are relevant here:

1. There is a `psframebox*` command which blots out whatever is behind the box. This is quite useful when the framebox contains text but unfortunately the DVI viewer just shows an empty box.

2. As we saw before the `rput` command allows material to be rotated. This doesn't work with the DVI viewer which either omits or does not rotate text.

As before, these are just problems with the DVI viewer, in the final documents things work as expected.

Placing Text Labels

The `uput` command is an alternative to `rput` when using text to label diagrams. Its first argument gives the position of the text with respect to the point given by the coordinates. You can use an angle, but the following position descriptors are very handy:
The following is a typical example of the use of \put in mathematical diagrams:

\begin{center}
\begin{pspicture}(0,0)(4,3)
\psline(0,0)(3,0)
\psline(0,0)(2.12,2.12) % 2.12 = 3/sqrt(2)
\psarc(0,0){0.6}{0}{45}
\uput[l](0,0){$A$}
\uput[r](3,0){$B$}
\uput[ur](2.12,2.12){$C$}
\uput[r](0.4,0.4){$45^\circ$}
\end{pspicture}
\end{center}

In examples like this is it is often necessary to do some calculations, in this case to find the coordinates of the point $C$ — Scilab is very handy for this sort of thing. In this example it would have been easier to use polar coordinates via SpecialCoor but, again unfortunately, SpecialCoor and text don't interact well with the DVI viewer.

Sometimes you may need to experiment a bit with text placement; in this example you may need a few tries to get the $45^\circ$ label in the right position.
19.3 Grids and Axes

These are very useful when plotting data (see below) but can be used in any diagrams.

19.3.1 Drawing Grids

Coordinate grids are drawn with the command:

\psgrid(x0,y0)(x1,y1)(x2,y2)

Here (x1,y1) and (x2,y2) are opposing corners as in \texttt{pspicture}. The x-coordinates run along the line with coordinates y0 and the y-coordinates run along the line with coordinates x0.

\begin{center}
\begin{pspicture}(0,0)(4,4)
\psgrid(2,2)(1,1)(3,3)
\psline(0,0)(4,4)
\end{pspicture}
\end{center}

There are various settings to control the way a grid is printed:

1. \texttt{subgriddiv}: The number of subdivisions in the main grid (default = 5).
2. \texttt{gridwidth}: Width of the main grid lines (default = 0.8 pt).
3. \texttt{subgridwidth}: Width of lines in subgrid (default = 0.4 pt).
4. \texttt{griddots}: If positive, main grid lines are dotted with that many dots per division (default 0).
5. \texttt{subgriddots}: If positive, subgrid lines are dotted with this many dots per division (default 0).
6. \texttt{gridlabels}: Font size of grid labels (default 10 pt).
7. **gridcolor**: Colour of main grid lines (default black).

8. **subgridcolor**: Colour of subgrid lines (default gray).


Some of these were used in the `\showgrid` command given in the previous lecture. Here is another example:

```latex
\begin{center}
\begin{pspicture}(0,0)(4,3)
  \psgrid[gridcolor=red,subgriddots=5,
           gridlabels=6pt,gridlabelcolor=blue](0,0)(0,0)(4,3)
\end{pspicture}
\end{center}
```

![](image.png)

### 19.3.2 Drawing Axes

The commands used here and in the rest of this section need:

```latex
\usepackage{pst-plot}
```

in the preamble.

The command

```latex
\psaxes(x0,y0)(x1,y1)(x2,y2)
```

draws x and y axes with the x-axis running from x1 to x2 and the y-axis running from y1 to y2 and with the origin at (x0,y0).

```latex
\begin{center}
\begin{pspicture}(-2,-2)(4,4)
  \showgrid
  \psaxes(0,0)(-1,-1)(3,3)
  \pscurve(-1,3.5)(0,2)(0.9,0.9)(2,0)(3.5,-1)
\end{pspicture}
\end{center}
```
As for lines, you can also use arrows with axes. The labels (numbers) are put next to the axes on the same side as \(x_1\) and \(y_1\). Note that \((x_1,y_1)\) and \((x_2,y_2)\) do not have to correspond to the lower left and upper right corners. Varying this can affect where the labels are placed:

\begin{center}
\begin{pspicture}(-2.5,-0.5)(2.5,3)
\psaxes{<->}(0,0)(-2.5,2.5)(2.5,0)
\end{pspicture}
\hspace{1cm}
\begin{pspicture}(-2.5,-0.5)(2.5,3)
\psaxes{<->}(0,0)(2.5,0)(-2.5,2.5)
\end{pspicture}
\end{center}

As with \texttt{psgrid} there are many ways to control how the axes are printed. The first set determine how ticks and labels are positioned along the axes:

<table>
<thead>
<tr>
<th>Horizontal</th>
<th>Vertical</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x</td>
<td>0y</td>
<td>Label at origin</td>
<td>0</td>
</tr>
<tr>
<td>Dx</td>
<td>Dy</td>
<td>Label increment</td>
<td>1</td>
</tr>
<tr>
<td>dx</td>
<td>dy</td>
<td>Distance between labels</td>
<td>0 pt</td>
</tr>
</tbody>
</table>
Note that when \( dx \) is 0, the built in scale \texttt{xunit} is used.

\begin{center}
\begin{pspicture}(-0.5,-0.5)(4,3.5)
\psaxes[Ox=5,dx=0.5,Dy=0.5](0,0)(0,0)(3,3)
\end{pspicture}
\end{center}

The other settings for \texttt{psaxes} are:

1. \texttt{labels}: Which axes are labelled. Possible values are \texttt{all, x, y} or \texttt{none} (default = \texttt{all}).

2. \texttt{showorigin}: Whether origin is labelled. Possible values are \texttt{true} or \texttt{false} (default = \texttt{true}).

3. \texttt{ticks}: Which axes are ticked (same as for \texttt{labels}).

4. \texttt{tickstyle}: Possible values are \texttt{full, top} or \texttt{bottom} (default = \texttt{full}).

5. \texttt{ticksize}: (default 3 pt).

\begin{center}
\begin{pspicture}(-0.5,-0.5)(4,3.5)
\psaxes[dy=0.25,labels=x,tickstyle=bottom]{->}(0,0)(0,0)(4,3)
\end{pspicture}
\end{center}
19.4 Plotting Data

PSTricks can provide be used to generate high-quality graphs and is much more more flexible in this regard than a many graphing programs.

19.4.1 Plotting Data from a File

Suppose we a file `ex1.dat` containing the data

<table>
<thead>
<tr>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1.6</td>
</tr>
<tr>
<td>3</td>
<td>3.2</td>
</tr>
<tr>
<td>4</td>
<td>4.5</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

This data can be plotted with the `fileplot` command:

```latex
\begin{center}
\psset{unit=0.75}
\begin{pspicture}(-1,-1)(6,6)
\psaxes(0,0)(0,0)(6,6)
\fileplot[plotstyle=line]{ex1.dat}
\fileplot[plotstyle=dots]{ex1.dat}
\end{pspicture}
\end{center}
```

A couple of points to note:

1. The data points are measured in whatever units are current. Therefore if your data contains the value 1000, say, you had better do some scaling. This is one situation where you are likely to use different scales for the $x$ and $y$ directions.

2. The example above used two calls to `fileplot`, once to plot the data as a (polygonal) line, the second as points.
More flexibility is obtained by using \texttt{readdata} to read the data from a file and then \texttt{dataplot} to plot the data:

\begin{center}
\psset{unit=0.75}
\begin{pspicture}(-1,-1)(6,6)
\readdata{\mydata}{ex1.dat}
\psaxes(0,0)(0,0)(6,6)
\dataplot[plotstyle=curve,showpoints=true, dotstyle=x]{\mydata}
\end{pspicture}
\end{center}

Some points to note:

1. The name of the array to store the data, \texttt{mydata} in the example, should not contain numerals. I tried the name \texttt{ex1} with nasty results.

2. The extra flexibility of \texttt{dataplot} results from it accepting many of PSTricks drawing settings, while \texttt{fileplot} only accepts \texttt{line} and \texttt{dots} linestyles.

3. The \texttt{readdata} command also allows the data to be reused.

\subsubsection{19.4.2 Plotting Data Directly}

It is also possible to enter data directly with the \texttt{saveddata} command, or plot data directly with the \texttt{listplot} command. Here are two more ways to produce the previous graph:
The commands used here need:
\usepackage{pst-node}
in the preamble.

\begin{center}
\begin{pspicture}(0,0)(5,5)
\showgrid
\rput(1,1){\rnode{A}{\psframebox{Dog}}}
\rput(2,4){\rnode{B}{\psframebox{Cat}}}
\rput(4,2){\rnode{C}{\psovalbox{Mouse}}}
\ncline{->}{A}{B}
\ncarc[linestyle=dashed]{->}{B}{C}
\end{pspicture}
\end{center}

19.5 Diagrams

The\ commands used here need:
\usepackage{pst-node}
in the preamble.

\textsf{PSTricks} has comprehensive commands for drawing diagrams with nodes
and trees. Here is an example:

\begin{center}
\begin{pspicture}(0,0)(5,5)
\showgrid
\rput(1,1){\rnode{A}{\psframebox{Dog}}}
\rput(2,4){\rnode{B}{\psframebox{Cat}}}
\rput(4,2){\rnode{C}{\psovalbox{Mouse}}}
\ncline{->}{A}{B}
\ncarc[linestyle=dashed]{->}{B}{C}
\end{pspicture}
\end{center}
For more on this see the PSTricks documentation.