AMTH142 Lecture 20

Graphics — More Advanced Topics

April 20, 2007

This is the final lecture for the unit.

Contents

20.1 Plotting Functions .................................................. 2
  20.1.1 Reverse Polish Notation ................................. 2
  20.1.2 Plotting Functions ...................................... 3
  20.1.3 Parametric Plots ........................................ 5
20.2 pscustom ......................................................... 6
20.3 Macros ............................................................. 10
  20.3.1 Defining New Styles ................................. 10
  20.3.2 Using \LaTeX \text{ Macros} ............................ 10
  20.3.3 \LaTeX \text{ Macros with Parameters} .................. 12
  20.3.4 Repetition ............................................. 12
20.4 Positioning Graphics .......................................... 13
20.1 Plotting Functions

To draw curves using a mathematical formula we need to know how to work with the reverse polish notation used by Postscript.

20.1.1 Reverse Polish Notation

Reverse polish notation writes the operands before the operator; so, for example,

\[ 3 + 4 \] is written \[ 3 \ 4 \ \text{add} \]

and

\[ (3 + 4) \times 5 \] is written \[ 3 \ 4 \ \text{add} \ 5 \ \text{mul} \]

while

\[ 3 + (4 \times 5) \] is written \[ 3 \ 4 \ 5 \ \text{mul} \ \text{add} \]

This takes a bit of getting used to but it has the advantage, at least for programming language designers, that mathematical expressions can be written unambiguously without brackets and without making any assumptions about the precedence of operators. For this to work each operator, add and mul in the examples above, must take a fixed number of operands. Both add and mul take two operands.

Reverse Polish notation is used in stack based implementations of arithmetic. Think of the stack is a pile of “cells”. Expressions are read, as usual, from left to right:

1. When a number is encountered, it is pushed onto the top of the stack and the numbers below are pushed down one cell.

2. When an operator is encountered, the appropriate count of numbers is popped off the top of the stack, the operator applied to them, and the result pushed back onto the stack.

Here is how \[ 3 \ 4 \ 5 \ \text{mul} \ \text{add} \] is evaluated:

\[
\begin{array}{ccccccc}
3 & \ldots & 4 & \ldots & 5 & \ldots & 20 & \ldots \\
\ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\
\end{array}
\]

\[
\begin{array}{ccccccc}
3 & \rightarrow & \ldots & \rightarrow & 4 & \rightarrow & \ldots & \rightarrow & 5 & \rightarrow & \ldots & \rightarrow & 3 & \rightarrow & \ldots & \rightarrow & \text{mul} & \rightarrow & \ldots & \rightarrow & \text{add} & \rightarrow & \ldots \\
& \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\
\end{array}
\]
Here are some examples of mathematical functions in reverse polish notation:

\[ x \sin x \quad \rightarrow \quad x \times x \sin \text{ mul} \]
\[ x^2 - 3 \quad \rightarrow \quad x \times \text{ mul} \ 3 \ \text{sub} \]
\[ \frac{x}{1 + x^2} \quad \rightarrow \quad x \ 1 \ x \ \text{mul} \ \text{add} \ \text{div} \]

### Postscript Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Result</th>
<th>Operator</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>\textit{x dup}</td>
<td>\textit{x x}</td>
<td>\textit{x y exch}</td>
<td>\textit{y x}</td>
</tr>
<tr>
<td>\textit{x y add}</td>
<td>\textit{x + y}</td>
<td>\textit{x y sub}</td>
<td>\textit{x - y}</td>
</tr>
<tr>
<td>\textit{x y mul}</td>
<td>\textit{x \times y}</td>
<td>\textit{x y div}</td>
<td>\textit{x/y}</td>
</tr>
<tr>
<td>\textit{x abs}</td>
<td>\textit{</td>
<td>x</td>
<td>}</td>
</tr>
<tr>
<td>\textit{x sqrt}</td>
<td>\textit{\sqrt{x}}</td>
<td>\textit{x y exp}</td>
<td>\textit{x^y}</td>
</tr>
<tr>
<td>\textit{x log}</td>
<td>\textit{\log_{10} x}</td>
<td>\textit{x ln}</td>
<td>\textit{\log_e x}</td>
</tr>
<tr>
<td>\textit{x sin}</td>
<td>\textit{\sin x}</td>
<td>\textit{x cos}</td>
<td>\textit{\cos x}</td>
</tr>
<tr>
<td>\textit{x y atan}</td>
<td>\textit{\tan^{-1}(x/y)}</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(Note that angles are measured in degrees)

#### 20.1.2 Plotting Functions

Here too we need:

\texttt{\usepackage{pst-plot}}

in the preamble. The command to plot functions is

\texttt{\psplot[settings] {xmin} {xmax} {function}}

Here settings can be the usual things like \texttt{linecolor=}, \texttt{plotstyle=} etc. An important new parameter is \texttt{plotpoints} which determines the number of points used to draw the graph. The default is 50, but 100 or more points are often needed for best results.

The parameters \texttt{xmax} and \texttt{xmin} are the determine the range of values for which to plot the function. The \texttt{function} must be a Postscript function with variable \texttt{x}.

### Examples

As our example we will plot

\[ f(x) = \frac{\sin x}{x} \quad \text{for} \quad x \in [-30, 30] \]
We need a couple of simple calculations to get the $x$ and $y$ coordinates scaled correctly. Taking $x_{\text{unit}}=0.1$ cm and $y_{\text{unit}}=2$ cm will give us a 6cm by 4cm diagram when we take $x \in [-30, 30]$ and $y \in [-1, 1]$.

We also need to convert radians to degrees as used by Postscript. Since there are $360/2\pi \approx 57.2958$ degree in 1 radian, we need to multiply $x$ by this scale factor when we apply the $\sin$ operator. Hence our Postscript function is:

$$x \times 57.2958 \ \text{mul} \ \sin \ x \ \text{div}$$

\begin{center}
\begin{pspicture}(-30,-1)(30,1)
\showgrid
\psplot{-30}{30}{x 57.2958 mul sin x div}
\end{pspicture}
\end{center}

The blur along the bottom arises from printing all the axes labels from -30 to 30. Otherwise this is not too bad once we remove the grid, but we should use more points to smooth out the graph. We will also double the size, take $y \in [-0.5, 1.5]$ and add axes and some labels:

\begin{center}
\psset{xunit=0.2cm}
\psset{yunit=4cm}
\begin{pspicture}(-30,-0.5)(30,1.5)
\psplot{plotpoints=200}{-30}{30}{x 57.2958 mul sin x div}
\psaxes[Dx=10,Dy=1]{->}(0,0)(-30,-0.3)(30,1.3)
\uput[l](0,1.2){$y$}
\uput[d](28,0){$x$}
\uput[r](2,0.8){\textstyle y = \frac{\sin x}{x}}
\end{pspicture}
\end{center}
20.1.3 Parametric Plots

Parametric plots are graphs represented in the form

\[ x = x(t) \quad y = y(t) \]

For example,

\[ x = \sin t, \quad y = \cos t, \quad t \in [0, 2\pi] \]

represents a circle

The command for parametric plots is

\[
\text{\texttt{\textbackslash parametricplot[settings]{tmin}{tmax}{functions}}}\]

The \texttt{functions} must be a two Postscript function with variable \( t \) (and with no separator between the functions.)

Here is parametric plot of

\[ x = \sin t, \quad y = \sin 2t \]

\begin{verbatim}
\begin{center}
\psset{xunit=2cm}
\psset{yunit=1cm}
\begin{pspicture}(-1,-1)(1,1)
\showgrid
\parametricplot{0}{360}{tsin t 2 mul sin}
\end{pspicture}
\end{center}
\end{verbatim}
Note how the two functions $t \sin$ and $t \ 2 \ \text{mul} \ \sin$ were used.

20.2 \texttt{pscustom}

The \texttt{pscustom} command gives access to many Postscript commands. Here we will use it to perform a common operation in mathematical diagrams — filling the region between curves. We have seen how to draw closed shapes with \texttt{pspolygon} and \texttt{psccurve} and draw lines with \texttt{psline}, \texttt{pscurve} and \texttt{psplot}.

One thing \texttt{pscustom} does is allow us to produce closed shapes by combining any types of lines. Here is an example:

\begin{center}
\begin{pspicture}(0,0)(6,4)
\showgrid
\pscustom[fillstyle=hlines]{
\psline(0.5,0.5)(0.5,3)
\pscurve(0.5,3)(3,2.5)(5.5,3)
\psline(5.5,3)(5.5,0.5)
\pscurve(5.5,0.5)(3,1)(0.5,0.5) }
\uput[l](0.5,1.75){\pscirclebox{1}}
\uput[u](3,2.5){\pscirclebox{2}}
\uput[r](5.5,1.75){\pscirclebox{3}}
\uput[d](3,1){\pscirclebox{4}}
\end{pspicture}
\end{center}
There are two features of this diagram which are unexpected — the slightly sharp angle at the join of curves 3 and 4 and the slightly rounded angle at the join of curves 1 and 2. There is a parameter liftpen which controls what happens between successive line segments. The term current point refers to end point of the previous line:

1. *liftpen* = 0: (the default) the current point is taken as the first point of the new line.

2. *liftpen* = 1: a straight line is drawn from the current point to the start of the new line.

3. *liftpen* = 2: The two curves are not joined.

As expected, judicious use of the *liftpen* parameter changes the shape of the previous diagram to what we intended. In our example, the default *liftpen*=0, starts the each of the \pscurve commands with what in effect amounts to two identical points. Changing to *liftpen*=1 does the trick:

\begin{center}
\begin{pspicture}(0,0)(6,4)
\showgrid
\pscustom[fillstyle=hlines]{
\psline(0.5,0.5)(0.5,3)
\pscurve[liftpen=1](0.5,3)(3,2.5)(5.5,3)
\psline(5.5,3)(5.5,0.5)
\pscurve[liftpen=1](5.5,0.5)(3,1)(0.5,0.5) }
\uput[l](0.5,1.75){\pscirclebox{1}}
\uput[u](3,2.5){\pscirclebox{2}}
\uput[r](5.5,1.75){\pscirclebox{3}}
\uput[d](3,1){\pscirclebox{4}}
\end{pspicture}
\end{center}
Here is another example which further illustrates the use of \texttt{liftpen}:

\begin{center}
\psset{unit=1.5cm}
\begin{pspicture}(0,0)(2,2)
\showgrid
\pscustom[fillstyle=solid,fillcolor=lightgray]{
\pscurve(0.2,1.5)(1,1.8)(1.2,1.2)
\pscurve[liftpen=0](1.8,0.5)(1,0.2)(0.2,0.5)}
\end{pspicture}
\hspace{1cm}
\begin{pspicture}(0,0)(2,2)
\showgrid
\pscustom[fillstyle=solid,fillcolor=lightgray]{
\pscurve(0.2,1.5)(1,1.8)(1.2,1.2)
\pscurve[liftpen=1](1.8,0.5)(1,0.2)(0.2,0.5)}
\end{pspicture}
\hspace{1cm}
\begin{pspicture}(0,0)(2,2)
\showgrid
\pscustom[fillstyle=solid,fillcolor=lightgray]{
\pscurve(0.2,1.5)(1,1.8)(1.2,1.2)
\pscurve[liftpen=2](1.8,0.5)(1,0.2)(0.2,0.5)}
\end{pspicture}
\end{center}
By the way, this example shows that we can fill open curves. However it good style to always used a closed curve when you intend to fill a shape. The Postscript command `\closepath` can be used to do this. (It is also good style to use `\closepath` even when the first and last points of a compound curve coincide.)

\begin{center}
\psset{unit=1.5cm}
\begin{pspicture}(0,0)(2,2)
  \showgrid
  \pscustom[fillstyle=solid,fillcolor=lightgray]{
    \pscurve(0.2,1.5)(1,1.8)(1.2,1.2)
    \pscurve[liftpen=1](1.8,0.5)(1,0.2)(0.2,0.5)
    \closepath
  }
\end{pspicture}
\hspace{2cm}
\begin{pspicture}(0,0)(2,2)
  \showgrid
  \pscustom[linestyle=none,fillstyle=solid,fillcolor=lightgray]{
    \pscurve(0.2,1.5)(1,1.8)(1.2,1.2)
    \pscurve[liftpen=1](1.8,0.5)(1,0.2)(0.2,0.5)
    \closepath
  }
\end{pspicture}
\end{center}

In the second figure, `\linestyle=\text{none}` was used to prevent the outline being shown.
20.3 Macros

20.3.1 Defining New Styles

We have already seen an example where we used \texttt{\newpsobject} to define the \texttt{\showgrid} command:

\begin{verbatim}
\newpsobject{showgrid}{psgrid}{subgriddiv=1,griddots=10,gridlabels=6pt}
\end{verbatim}

The general form of the \texttt{\newpsobject} command is

\begin{verbatim}
\newpsobject{name}{object}{parametervalues}
\end{verbatim}

\texttt{\newpsobject} is quite limited as it just associates a set of parameter values to an existing object. Even more limited is

\begin{verbatim}
\newpsstyle{name}{parametervalues}
\end{verbatim}

which just gives a name to some parameter values. For example

\begin{verbatim}
\newpsstyle{newdots}{linestyle=dotted,dotsep=1pt}
\end{verbatim}

\begin{center}
\begin{pspicture}(-1,-1)(1,1)
\showgrid
\pscircle[style=newdots,linecolor=red](0,0){1}
\end{pspicture}
\end{center}

20.3.2 Using \LaTeX\ Macros

Recall from Lecture 5 that \LaTeX\ macros:

\begin{verbatim}
\newcommand{name}{definition}
\end{verbatim}

can be used to define new \LaTeX\ commands. These can be used within \texttt{PSTricks}, but you need to remember that commands defined this way are \LaTeX\ commands and will not accept the usual sort of \texttt{PSTricks} settings and parameters. As an example we can define a command to draw a solid star (see Lecture 19):

\begin{verbatim}
\newcommand{\psstar}{\pspolygon*(1;18)(1;162)(1;306)(1;90)(1;234)}
\end{verbatim}

We can use this within a \texttt{PSTricks} picture (because we have used polar coordinates in the definition of \texttt{\psstar}, we also need to use \texttt{SpecialCoor}):
With \texttt{rput} and change of units the \texttt{psstar} command can be used more flexibly:

\begin{center}
\begin{pspicture}(0,0)(4,4)
\SpecialCoor
\showgrid
\rput(2,2){\psstar}
\psset{unit=0.5cm}
\rput(1,1){\psstar} \hspace{1cm} \% really at (0.5,0.5)
\rput(1,7){\psstar} \hspace{1cm} \% really at (0.5,3.5) etc
\rput(7,7){\psstar}
\rput(7,1){\psstar}
\end{pspicture}
\end{center}
20.3.3 \LaTeX\ Macros with Parameters

\LaTeX\ macros can take parameters which allows more flexibility. We can define a new version of \texttt{\textbackslash psstar}, called \texttt{\textbackslash psvstar}, which will allow us to vary the size and color of the star:

\begin{verbatim}
\newcommand{\psvstar}[2]{\pspolygon*[linecolor=#2](#1;18)
  (\#1;162)(\#1;306)(\#1;90)(\#1;234)}
\end{verbatim}

The [2] in the \texttt{\newcommand} indicates that it takes two parameters. Within the \texttt{\newcommand} the parameters are then referred to as \texttt{#1} and \texttt{#2}. When the new command is applied the actual arguments to the command are substituted for \texttt{#1} and \texttt{#2} as illustrated below:

```
\begin{center}
\begin{pspicture}(0,0)(4,4)
\SpecialCoor
\showgrid
\rput(2,2){\psvstar{1}{red}}
\rput(0.5,0.5){\psvstar{0.5}{blue}}
\rput(3.5,0.5){\psvstar{0.5}{blue}}
\rput(0.5,3.5){\psvstar{0.5}{blue}}
\rput(3.5,3.5){\psvstar{0.5}{blue}}
\end{pspicture}
\end{center}
```

We might define an even more versatile version of \texttt{\textbackslash psstar} by including the position as a parameter to the macro.

20.3.4 Repetition

PSticks has some facilities for repeating commands.

```
\multirput(x0,y0)(x1,y1){count}{stuff}
```

Is a variant of \texttt{\rput} which puts down \texttt{count} copies of \texttt{stuff} beginning at \texttt{(x0,y0)} and then at increments of \texttt{(x1,y1)}.  

12
20.4 Positioning Graphics

Sometimes we want to position text and graphics side by side as below:

We can draw a star using the \texttt{psstar} macro as follows:

\begin{pspicture}(-1,-1)(1,1)
\SpecialCoor
\psstar
\end{pspicture}

This is achieved using the \texttt{minipage} environment. Its most general form is:

\begin{minipage}[pos][height][inner_posn]{width}
\text{text}
\end{minipage}

Here \texttt{pos} controls how the box containing the minipage is aligned with respect to the surrounding text. It can be

1. \texttt{b} for bottom
2. \texttt{c} for centre
3. \texttt{t} for top.

The \texttt{width} determines the width of the minipage.

The other two parameters are optional; \texttt{height} specifies the height of the minipage, \texttt{inner_posn} determines where the text or graphics is positioned within the minipage. These are mainly used for fine-tuning positioning.
The example above was created using two \texttt{minipages}, one for the text and one for the graphics by:

\begin{verbatim}
\noindent
\begin{minipage}[b]{7cm}
Wecan draw a star using the \texttt{\verb+\psstar+} macro as follows:
\begin{verbatim}
\begin{pspicture}(-1,-1)(1,1)
\SpecialCoor
\psstar
\end{pspicture}
\end{verbatim}
\end{minipage}\hfill
\begin{minipage}[b][3cm][c]{6cm} %start of second minipage
\begin{center}
\begin{pspicture}(-1,-1)(1,1)
\SpecialCoor
\psstar
\end{pspicture}
\end{center}
\end{minipage}
\end{verbatim}
\end{verbatim}