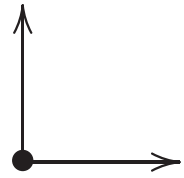


- Not a full manual, but comprehensive enough for quick reference
- Practical applications, rather than fully explaining each package
- Will cover
 - X_Y-pic for general diagrams
 - PSTricks (trees, DAGs, graphs, plots)
 - Circuit diagrams
 - Timing diagrams
- Will finish with a few references to other packages

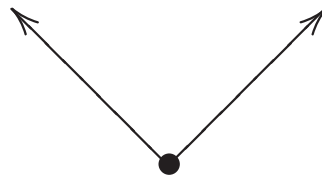


```

1 \xy <1cm,0cm>:           % Define coordinate system
2 \POS (0,0) *{\bullet} % Typeset a \bullet at position (0,0)
3 \POS (0,0) \ar (1,0) % Draw an arrow from (0,0)--(1,0)
4 \POS (0,0) \ar (0,1) % Draw an arrow from (0,0)--(0,1)
5 \endxy

```

Note: only \vec{x} vector (x_0, x_1) specified for coordinate system. \vec{y} is defined orthogonal to \vec{x} , i.e. $(y_0, y_1) = (-x_1, x_0)$.



```

1 \xy <1cm,1cm>:           % Rotated coordinate system
2 \POS (0,0) *{\bullet}
3 \POS (0,0) \ar (1,0)
4 \POS (0,0) \ar (0,1)
5 \endxy

```

^a`\usepackage[all]{xy}`

Xy-pic: Naming and Growing objects



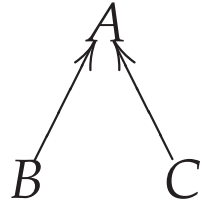
```
1 \xy <1cm,0cm>:  
2 \POS (0,0) *{\bigcirc} ="A" % "Name" the object "A"  
3 \POS (2,0) *{\bigcirc} ="B"  
4 \POS "A" \ar "B"  
5 \endxy
```



```
1 \xy <1cm,0cm>:  
2 \POS (0,0) *+{\bigcirc} ="A" % Add an extra margin  
3 \POS (2,0) *+{\bigcirc} ="B"  
4 \POS "A" \ar "B"  
5 \endxy
```



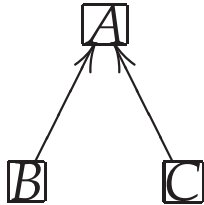
```
1 \xy <1cm,0cm>:  
2 \POS (0,0) ="A" *{\bigcirc} % Note: We name the object  
3 \POS (2,0) ="B" *{\bigcirc} % before adding text  
4 \POS "A" \ar "B"  
5 \endxy
```



```

1 \xy <5mm,0cm> :
2 \POS (0,1) *{A} ="A"
3 \POS (-1,-1) *{B} \ar "A" % Arrow is drawn from
4 \POS (1,-1) *{C} \ar "A" % the "current" object
5 \endxy

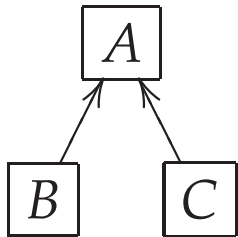
```



```

1 \xy <5mm,0cm> :
2 \POS (0,1) *[F-]{A} ="A" % Add a (rectangular) frame
3 \POS (-1,-1) *[F-]{B} \ar "A"
4 \POS (1,-1) *[F-]{C} \ar "A"
5 \endxy

```

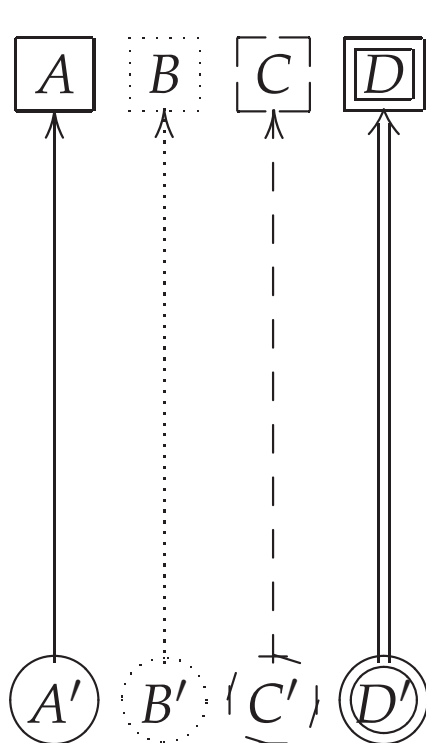


```

1 \xy <5mm,0cm> :
2 \POS (0,1) *+[F-]{A} ="A" % Note again the effect of
3 \POS (-1,-1) *+[F-]{B} \ar "A" % the + operator
4 \POS (1,-1) *+[F-]{C} \ar "A"
5 \endxy

```

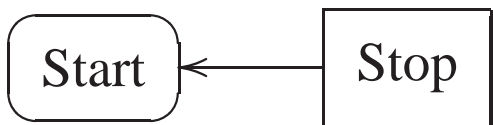
Xy-pic: More Frames and Arrows



```

1 \xy <7mm,0cm>:<0cm,2cm>:: % Manually set y vector
2 \POS (0,1) *+[F-]{A} = "A"
3 \POS (0,-1) *+[Fo]{A'} \ar "A"
4 %
5 \POS (1,1) *+[F.]{B} = "B"
6 \POS (1,-1) *+[F.o]{B'} \ar @{.>} "B"
7 %
8 \POS (2,1) *+[F--]{C} = "C"
9 \POS (2,-1) *+[F-o]{C'} \ar @{-->} "C"
10 %
11 \POS (3,1) *+[F=]{D} = "D"
12 \POS (3,-1) *+[F=]{D'} \ar @{=>} "D"
13 \endxy

```

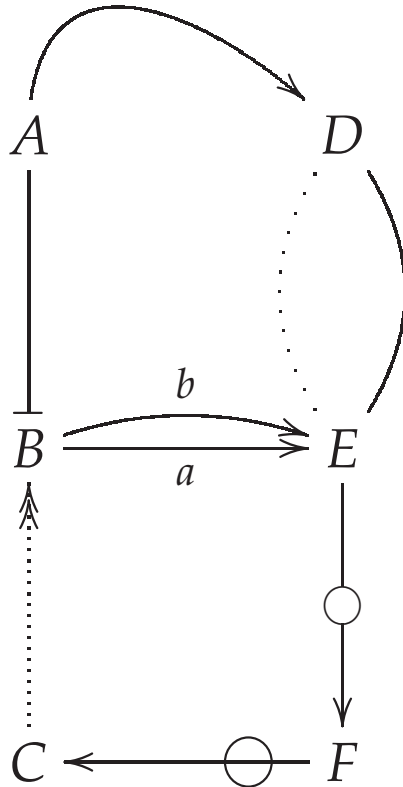


```

1 \xy <1cm,0cm>:
2 % Use \txt command to avoid math mode
3 \POS (0,0) *++[F-:<2mm>]\txt{Start} = "S",
4 \POS (2,0) *++[F-,]\txt{Stop} \ar "S"
5 \endxy

```

Xy-pic: Curved Arrows and Labels

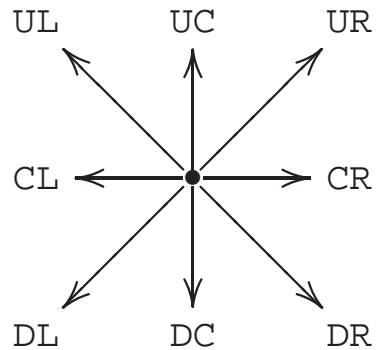


```

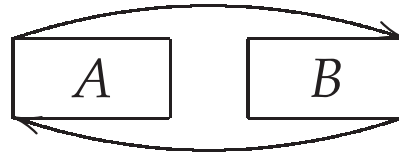
1 \xy <1cm,0cm>:
2 \POS (0,2) *+{A} ="A"
3 \POS -(0,2) *+{B} ="B" % Relative positions
4 \POS -(0,2) *+{C} ="C"
5 \POS +(2,4) *+{D} ="D"
6 \POS -(0,2) *+{E} ="E"
7 \POS -(0,2) *+{F} ="F"
8 %
9 \POS "A" \ar @{-|} "B"
10 \POS "B" \ar @{<<.} "C"
11 \POS "A" \ar @({u,u1}) "D" % See next slide
12 \POS "D" \ar @/_4mm/ @{.} "E" % Curve "down"
13 \POS "D" \ar @/^4mm/ @{-} "E" % Curve "up"
14 \POS "B" \ar _{a} "E" % Label "a" below
15 \POS "B" \ar ^{b} @/^/ "E" % Label "b" above
16 \POS "E" \ar |{\bigcirc} "F" % \bigcirc halfway
17 \POS "F" \ar "C"
18 \POS ?(0.3) *{\bigcirc} % At 30% of the last arrow
19 \endxy

```

Xy-pic: Reference Points



UC, CR, DC and CL are typically equal to U, R, D and L respectively



```

1 \xy <5mm,0cm>:
2 % Use = to specify a fixed size
3 \POS (0,0) *=(2,1)[F-]{A} ="A"
4 \POS (3,0) *=(2,1)[F-]{B} ="B"
5 % Use ! to shift to a corner
6 \POS "B" !DR \ar @/^/ "A" !DL
7 \POS "A" !UL \ar @/^/ "B" !UR
8 \endxy

```

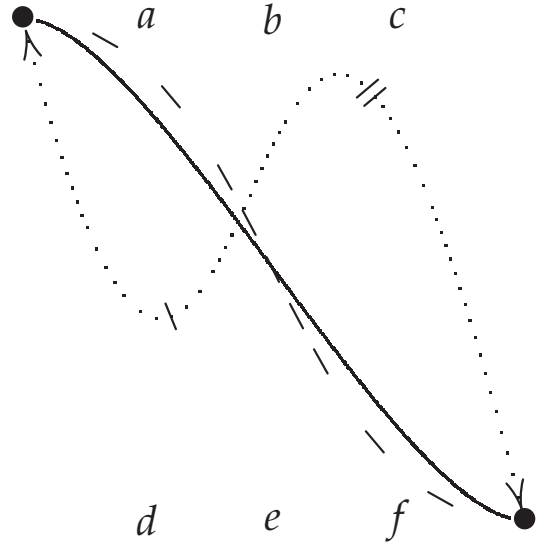


```

1 \xy <5mm,0cm>:
2 (0,1) \ar @(d,d1) (4,1) % @(out angle, in angle)
3 \endxy

```

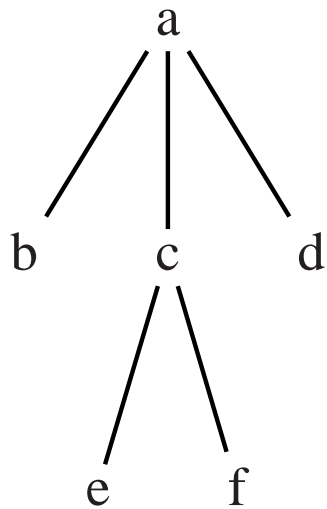
Xy-pic: Arbitrary curves



```
1 \xy <8mm,0cm>:
2 \POS (0, 2) *{\bullet} ="X"
3 \POS (4,-2) *{\bullet} ="Y"
4 \POS (1, 2) *{_a} ="A" -(0,4) *{_d} ="D"
5 \POS (2, 2) *{_b} ="B" -(0,4) *{_e} ="E"
6 \POS (3, 2) *{_c} ="C" -(0,4) *{_f} ="F"
7 % Draw three curves; specify control points
8 % and a dashed (---)/dotted (.) line type
9 \POS "X" ; "Y" **\crv{"A"&"F"}
10 \POS "X" ; "Y" **\crv{~**\dir{--}"B"&"E"}
11 \POS "X" ; "Y" **\crv{~**\dir{.}"D"&"B"&"C"}
12 % Draw an arrowhead at the end (>) and the
13 % start (<) of the last connection; also
14 % vertical strokes (/) at 25% and 75%
15 \POS ?>*\dir{>}
16 \POS ?<*\dir{<}
17 \POS ?(0.25)*\dir{||}
18 \POS ?(0.75)*\dir{|||}
19 \endxy
```

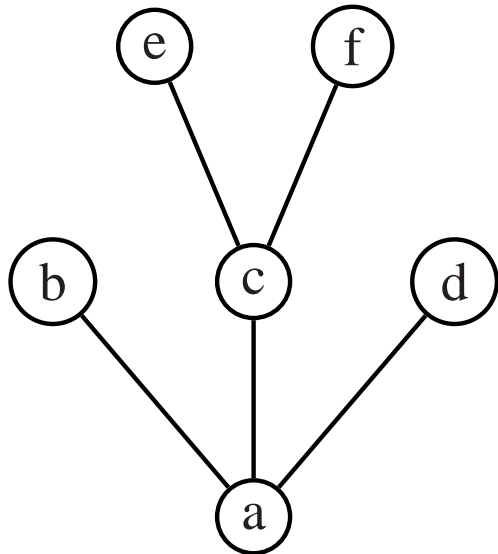
- PSTricks is a very large and very powerful package
- Downside: it works by inserting “PS (= PostScript) specials”—does not work with `pdflatex` (use `dvips` → `ps2pdf` instead)
- Note that the package to typeset Circuit Macros (discussed later) uses PSTricks internally, so the same remark applies
- Here we will cover only a tiny subset of PSTricks: trees, “tree-like graphs”, graphs and plots

^a`\usepackage{pst-all}`



```

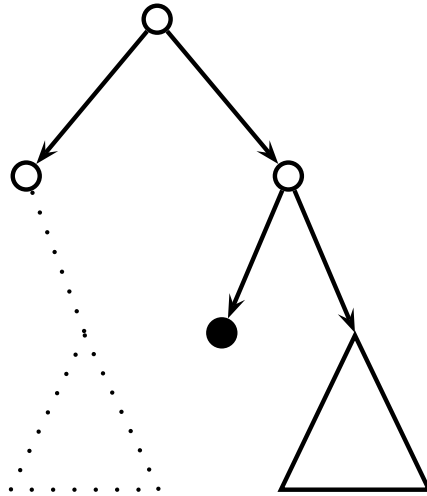
1 \psset {levelsep=15mm,nodesep=1mm}
2 \pstree{\TR{a}} {
3   \TR{b}
4   \pstree{\TR{c}} {
5     \TR{e}
6     \TR{f}
7   }
8   \TR{d}
9 }
  
```



```

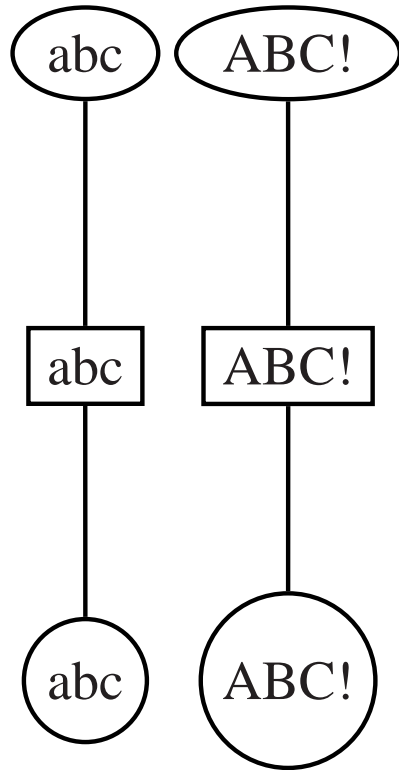
1 % Grow (U)p, (D)own, (R)ight or (L)eft
2 \psset {levelsep=15mm,nodesep=0mm,treemode=U}
3 \pstree{\Tcircle{a}} {
4   \Tcircle{b}
5   \pstree{\Tcircle{c}} {
6     \Tcircle{e}
7     \Tcircle{f}
8   }
9   \Tcircle{d}
10 }
  
```

PSTricks: More Trees



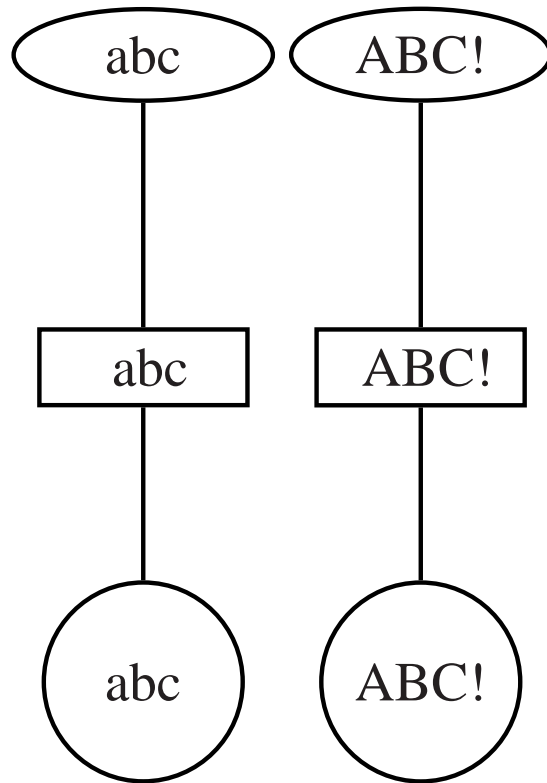
```
1 \psset{radius=1mm,levelsep=10mm}
2 \pstree[arrows={->}]{\TC} { % \TC: Open circle
3   \pstree[linestyle=dotted,arrows={-}]{\TC} {
4     \Tn % "Null" node
5     \pstree{\Tp} { % "Point" node
6       \Tfan % Subtree
7     }
8   }
9 \pstree{\TC} {
10  \TC* % Closed circle
11  \pstree{\Tp}{
12    \Tfan
13  }
14 }
15 }
```

PSTricks: More Nodes (1)

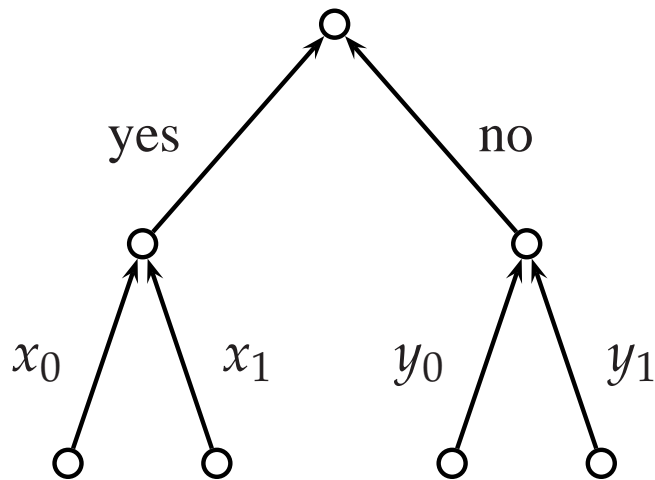


```
1 \pstree{\Toval{abc}} {  
2   \pstree{\TR{\psframebox{abc}}}} {  
3     \Tcircle{abc}  
4 }}  
5 \pstree{\Toval{ABC!}} {  
6   \pstree{\TR{\psframebox{ABC!}}}} {  
7     \Tcircle{ABC!}  
8 }}
```

PSTricks: More Nodes (2)



```
1 % Make sure all nodes are exactly 1cm
2 % wide. We use Latex' makebox command.
3 \newcommand{\boxsize}{1cm}
4 \newcommand{\fTR}[1] {
5   \TR{\psframebox{
6     \makebox[\boxsize][c]{#1}}} }
7 \newcommand{\fToval}[1] {
8   \Toval{\makebox[\boxsize][c]{#1}} }
9 \newcommand{\fTcircle}[1] {
10  \Tcircle{\makebox[\boxsize][c]{#1}} }
11
12 \pstree{\fToval{abc}} {
13   \pstree{\fTR{abc}} {
14     \fTcircle{abc}
15  }}
16 \pstree{\fToval{ABC!}} {
17   \pstree{\fTR{ABC!}} {
18     \fTcircle{ABC!}
19  }}
```

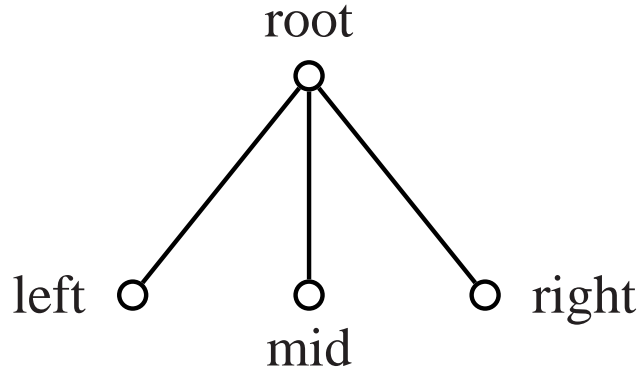


```

1 \psset{radius=1mm,levelsep=14mm}
2 \pstree[arrows={<-}]{\TC} {
3   \psset{tpos=.5} % Labels halfway
4   \pstree{\TC ^{yes}} {
5     \TC ^{$x_0$} % Uses math mode
6     \TC _{$x_1$}
7   }
8   \Tn \Tn % Add a bit of extra space
9   \pstree{\TC _{no}} {
10    \TC ^{$y_0$}
11    \TC _{$y_1$}
12  }
13 }

```

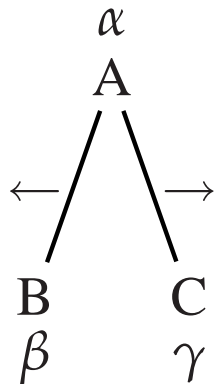
- `xdvi` might display this incorrectly (ps or pdf output *is* correct)
- `tpos` parameter must be set *within* the tree



```

1 \psset{radius=1mm,levelsep=14mm,
2     labelsep=0mm}
3 \pstree{\TC ~[tnpos=a]{root}} {
4   \TC ~[tnpos=1,labelsep=2mm]{left}
5   \TC ~{mid} % Default "tnpos"
6   \TC ~[tnpos=r,labelsep=2mm]{right}
7 }

```

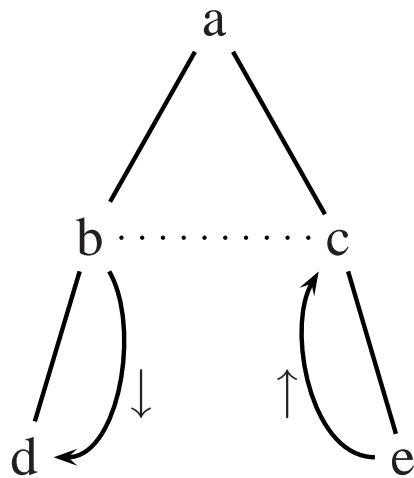


```

1 \psset{levelsep=14mm,nodesep=1mm,labelsep=0mm}
2 \pstree{\TR{A} ~[tnpos=a]{\alpha}} {
3   \psset{tpos=.5}
4   \TR{B} ~{\beta} ^{\leftarrow}
5   \TR{C} ~{\gamma} _{\rightarrow}
6 }

```

(Or trees with extra connections)



```

1 \psset{nodesep=1mm,levelsep=14mm,
2   labelsep=0mm}
3 \pstree{\TR{a}} {
4   \pstree{\TR[name=b]{b}} {
5     \TR[name=d]{d}
6     \Tn
7   }
8   \pstree{\TR[name=c]{c}} {
9     \Tn
10    \TR[name=e]{e}
11  }
12 }
13
14 \ncline[linestyle=dotted]{b}{c}
15 \nccurve[angleA=300,angleB=0,arrows={->}] {b}{d}
16 \Aput{$\downarrow$} % Label above the curve
17 \nccurve[angleA=240,angleB=180,arrows={<-}] {c}{e}
18 \Bput{$\uparrow$} % Label below the curve

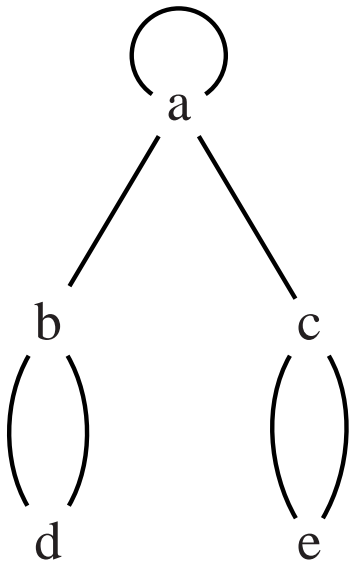
```

PSTricks: More on “Graphs”

```

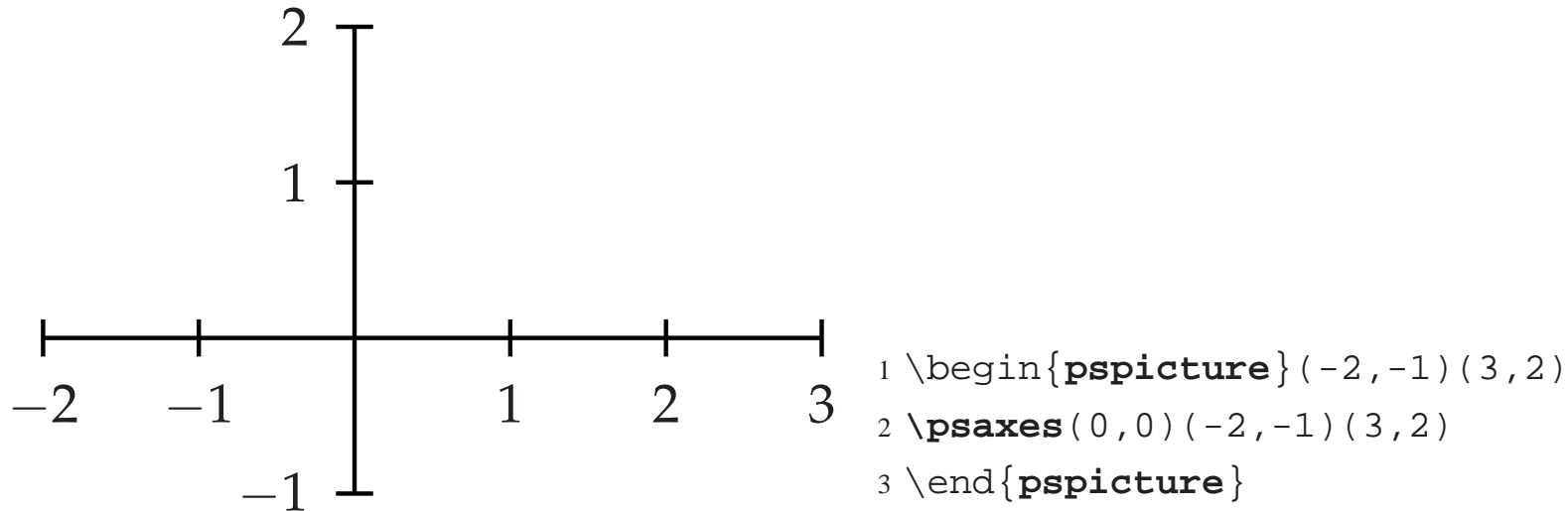
1 \psset{nodesep=1mm,levelsep=14mm}
2 \pstree{\TR[name=a]{a}} {
3   \pstree[edge=none]{\TR[name=b]{b}} {
4     \TR[name=d]{d}
5   }
6   \Tn
7   \pstree[edge=none]{\TR[name=c]{c}} {
8     \TR[name=e]{e}
9   }
10 }
11
12 \nccurve[angleA=240,angleB=120]{b}{d}
13 \nccurve[angleA=300,angleB=60]{b}{d}
14
15 \nccurve[angleA=240,angleB=120]{c}{e}
16 \nccurve[angleA=300,angleB=60]{c}{e}
17
18 \nccircle[angleA=0]{a}{3mm}

```



PSTricks: Setting coordinate system

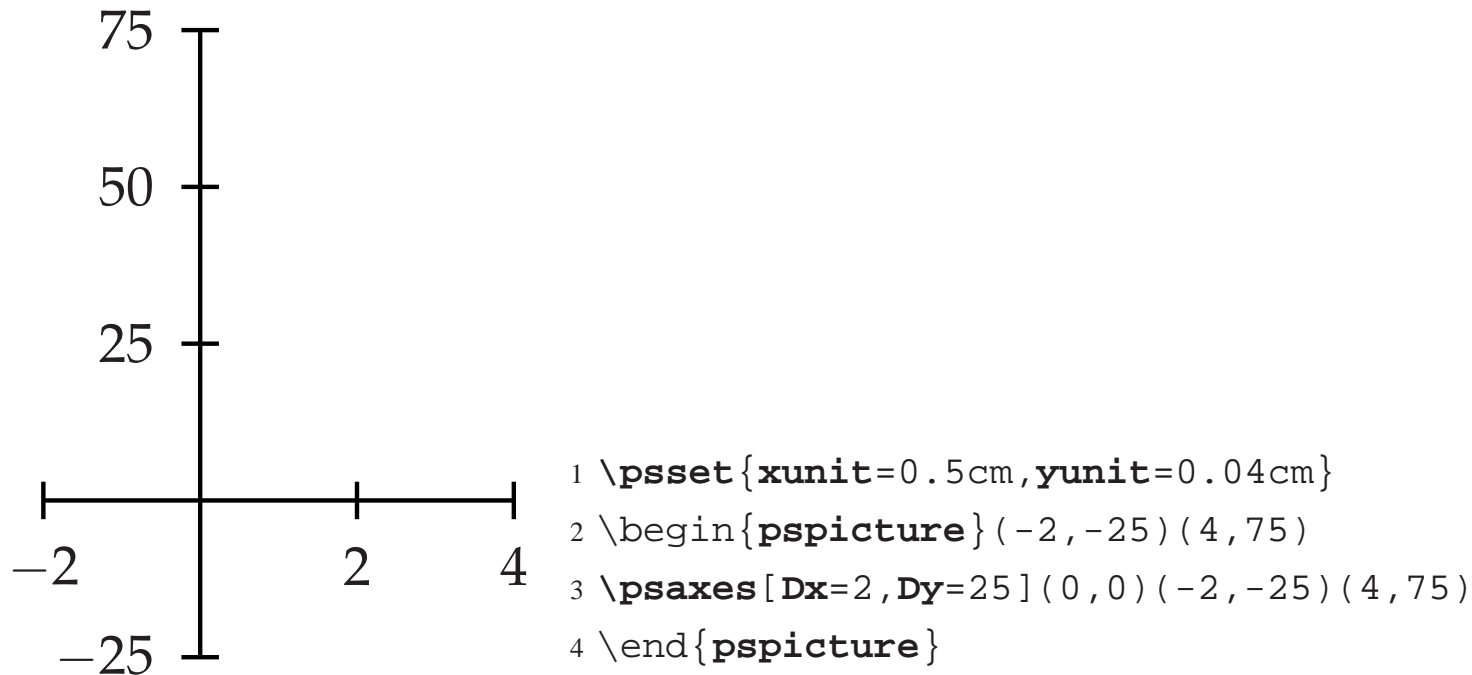
Suppose we want to draw a graph of a function $f(x)$ from $x = -2$ up to $x = +3$, and suppose that $f(x)$ ranges from -1 to $+2$ within that domain. We need a coordinate system such as



By default, the unit is 1cm, so this graph is 5 cm wide and 3 cm high.

PSTricks: Setting the coordinate system (2)

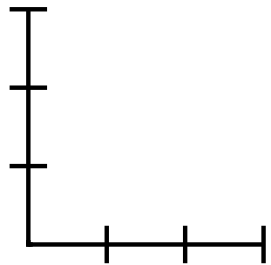
Suppose we want to plot another function $g(x)$ from $x = -2$ up to $x = +4$, and suppose that $g(x)$ ranges from -25 to $+75$ within that domain. We will want to change the default unit!



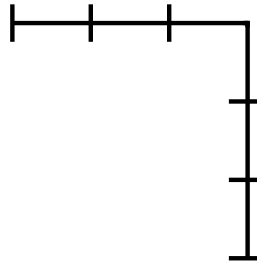
This graph is $(6 \times 0.5) = 3$ cm wide and $(100 \times 0.04) = 4$ cm high.

PSTricks: Setting the coordinate system (3)

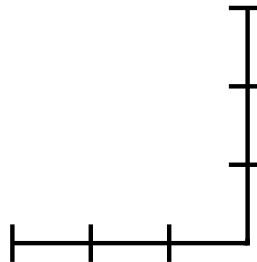
A few more examples.



```
1 \begin{pspicture}(0,0)(3,3)
2 \psaxes[labels=none](0,0)(0,0)(3,3)
3 \end{pspicture}
```



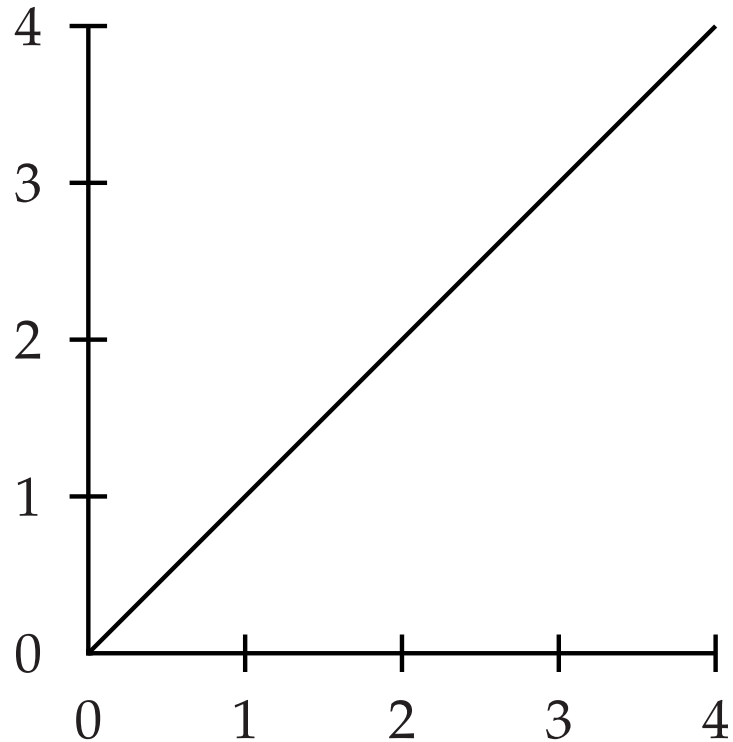
```
1 \begin{pspicture}(-3,-3)(0,0)
2 \psaxes[labels=none](0,0)(-3,-3)(0,0)
3 \end{pspicture}
```



```
1 \begin{pspicture}(-3,0)(0,3)
2 \psaxes[labels=none](0,0)(-3,0)(0,3)
3 \end{pspicture}
```

For our purposes the first argument to `\psaxes` is always $(0,0)$.

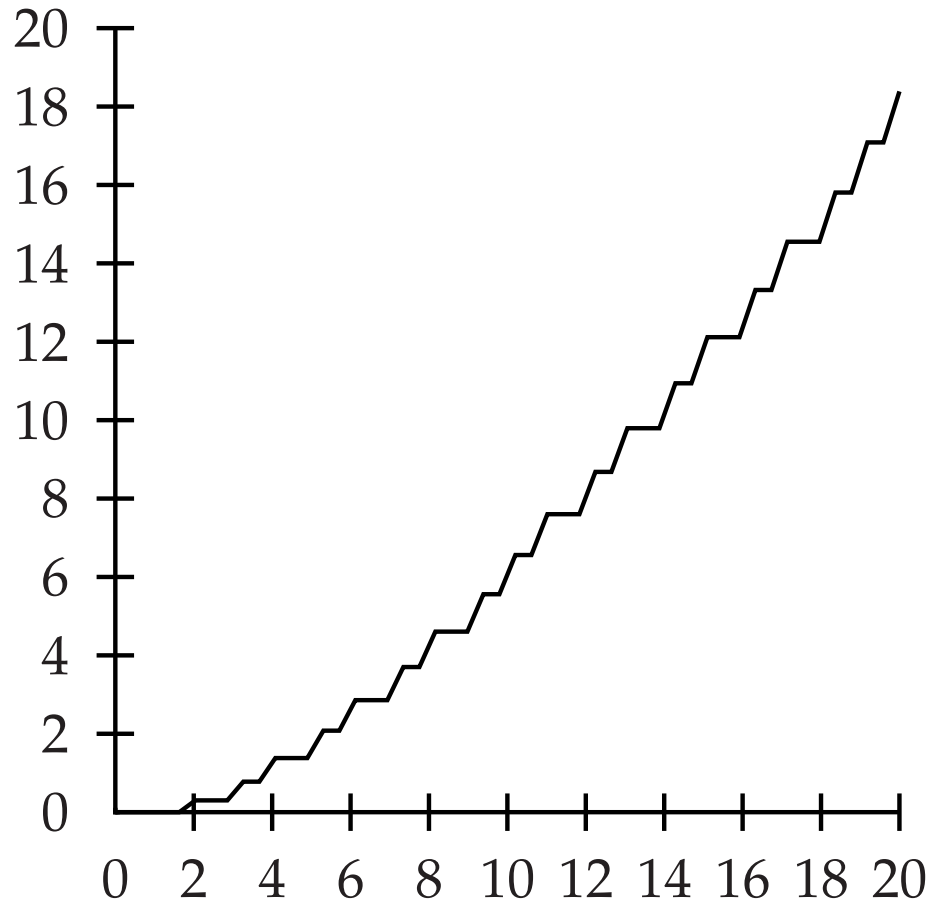
How do we actually plot a function?



```
1 \begin{pspicture}(0,0)(4,4)
2 \psaxes(0,0)(0,0)(4,4)
3 \psplot{0}{4}{x} % Plot f(x) = x
4 \end{pspicture}
```

`\psplot{x0}{x1}{f(x)}` plots $f(x)$ from x_0 up to x_1 .

Unfortunately, the way functions must be specified is by coding the function directly in PostScript. Here is a plot of $\log_{10}(\lfloor x \rfloor!)$



Here is the code (!):

```
1 \psset{xunit=0.25cm,yunit=0.25cm}
2 \begin{pspicture}(0,0)(20,20)
3 \psaxes[Dx=2,Dy=2](0,0)(0,0)(20,20)
4 \psplot{0}{20}{
5   /fac {
6     1 dict
7     begin
8     /n exch def
9     n 0 eq {1 } {n n 1 sub fac mul } ifelse
10    end
11   } def
12   /main {
13     x floor fac log
14   } def
15   main
16 }
17 \end{pspicture}
```

Last year in the DUCSS talk (see edsko.net), I explained the basics of PostScript programming. This year, I won't (see the DUCSS slides if you are interested).

Instead, I wrote a small compiler, called `psfc` (PostScript formula compiler). This compiler is available from my website, and you can even use it directly from my website without installing it locally.

Now, `psfc` is the result of a day's work, so it might not be completely bug-free, and it certainly does not cover all of PostScript, but for the current purposes I think it will suffice.

With `psfc`, $\log_{10}(\lfloor x \rfloor!)$ can be written as

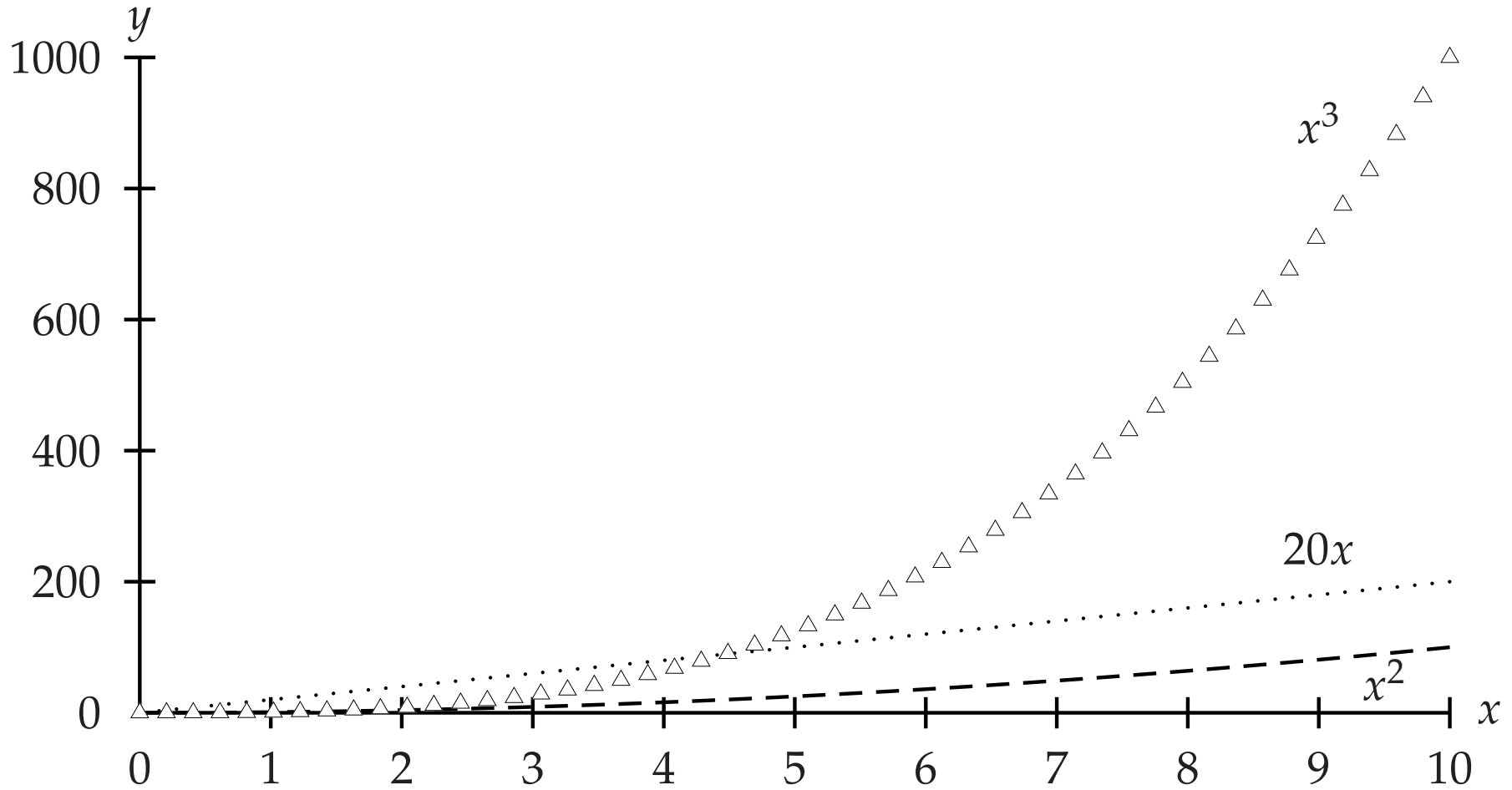
```
1 fac (n) = if n == 0 then 1 else n * fac(n - 1) endif  
2 main = log(fac(floor(x)))
```

Much easier! Compile this code using `psfc` to get the nasty PostScript code that you can then cut&paste into your Latex document.

Note: If you don't understand this definition of factorial, compare it with the mathematical definition of factorial:

$$\text{fac } n = \begin{cases} 1 & \text{if } n = 0 \\ n \times \text{fac}(n - 1) & \text{otherwise} \end{cases}$$

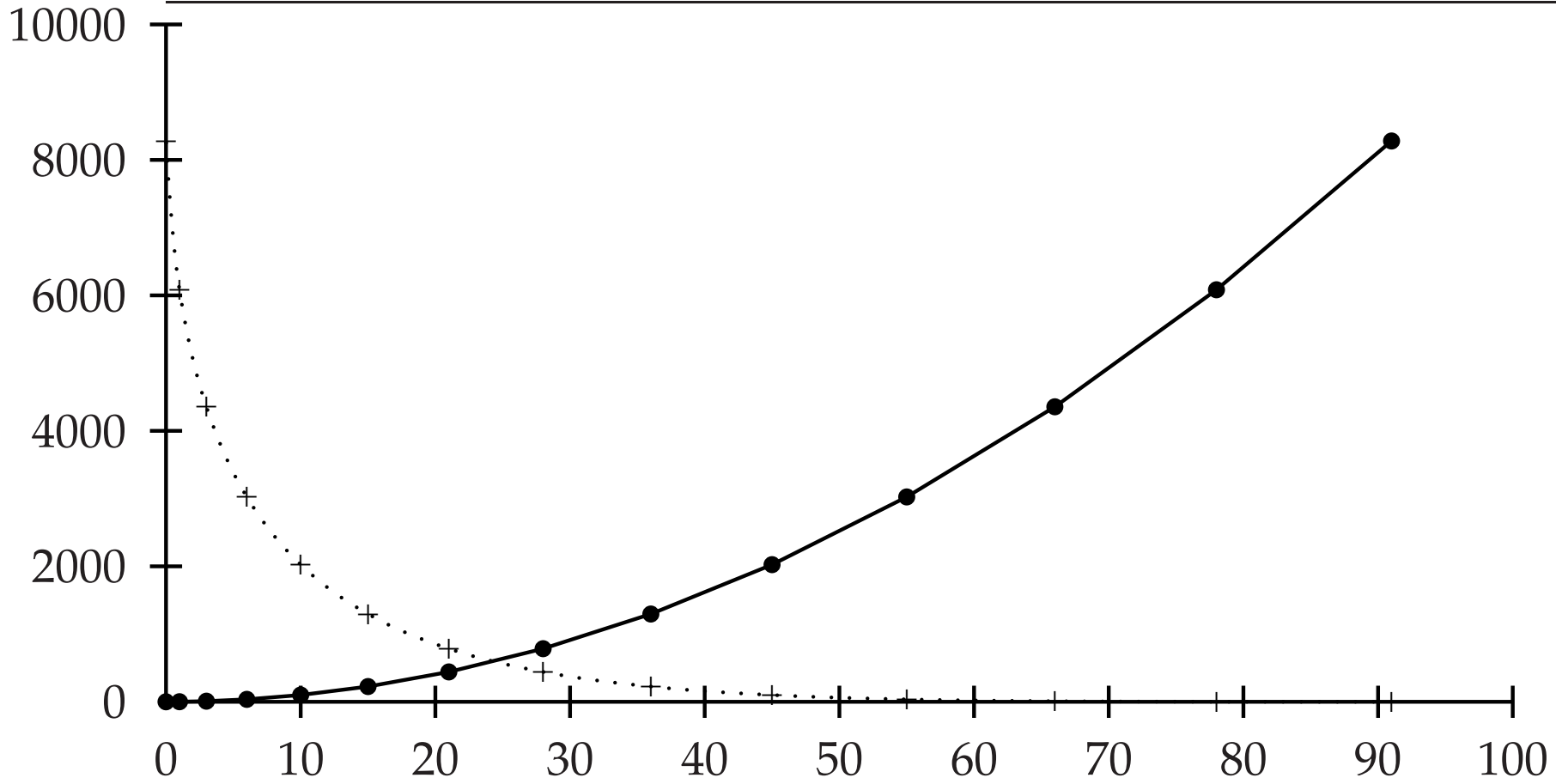
PSTricks: More example graphs (1)



PSTricks: More example graphs (2)

The code...

```
1 \psset{xunit=1cm,yunit=0.005cm}
2 \begin{pspicture}(0,0)(10,1000)
3 \psaxes[Dx=1,Dy=200](0,0)(0,0)(10,1000)
4 \psplot[linestyle=dotted]{0}{10}{20 x mul} % 20 * x
5 \psplot[linestyle=dashed]{0}{10}{x x mul} % x^2
6 \psplot[plotstyle=dots,dotstyle=triangle]{0}{10}{x x x mul mul} % x^3
7 \rput(10.3,0){$x$}
8 \rput(0,1050){$y$}
9 \rput(9,900){$x^3$}
10 \rput(9,250){$20x$}
11 \rput(9.5,50){$x^2$}
12 \end{pspicture}
```



The code:

```
1 \psset{xunit=0.1cm,yunit=0.0005cm}
2 \readdata{\data}{plotdata1}
3 \readdata{\datb}{plotdata2}
4 \begin{pspicture}(0,0)(100,10000)
5 \psaxes[Dx=10,Dy=2000](0,0)(0,0)(100,10000)
6 \dataplot[plotstyle=line]{\data}
7 \dataplot[plotstyle=dots,dotstyle=*]{\data}
8 \dataplot[plotstyle=curve,linestyle=dotted]{\datb}
9 \dataplot[plotstyle=dots,dotstyle=+]{\datb}
10 \end{pspicture}
```

Dotstyles: *, o, +, triangle, triangle*, square, square*,
pentagon, pentagon*, |

Plotstyles: dots, line, polygon, curve, ecurve, ccurve

The contents of `plotdata1` (normal text file):

```
0 0
1 1
3 9
6 36
10 100
15 225
21 441
28 784
36 1296
45 2025
55 3025
66 4356
78 6084
91 8281
```

Simply a list of (x, y) pairs.