

Introduction to Scientific Typesetting

Lesson 13: Changing Defaults and More with `pstricks`

Ryan Higginbottom

January 25, 2011

An Overview

Changing Some
Defaults

Linestyles and Fillstyles

Graphing

Nodes and Connections

Changing Some Defaults

Linestyles and Fillstyles

Graphing

Nodes and Connections

An Overview

**Changing Some
Defaults**

Using `renewcommand`

A List of \LaTeX Defaults

Changing Lengths

A Warning

An Example

Linestyles and Fillstyles

Graphing

Nodes and Connections

Changing Some Defaults

Using `renewcommand`

An Overview

Changing Some Defaults

Using `renewcommand`

A List of \LaTeX Defaults

Changing Lengths

A Warning

An Example

Linestyles and Fillstyles

Graphing

Nodes and Connections

The `renewcommand` can be used to change things that \LaTeX has built in as default.

Some examples:

- The symbol at the end of a proof is \square by default and is stored as `\qedsymbol`. If you wanted to use \dagger instead, type `\renewcommand{\qedsymbol}{\dagger}`.
- \LaTeX stores the names of parts of a document in commands. The name for the bibliography is stored in `\refname`. If you wanted your bibliography to be called “Bibliography” instead of “References”, type `\renewcommand{\refname}{Bibliography}`.

A List of L^AT_EX Defaults

An Overview

Changing Some Defaults

Using `renewcommand`

A List of L^AT_EX Defaults

Changing Lengths

A Warning

An Example

Linestyles and Fillstyles

Graphing

Nodes and Connections

English Default	String to change
Abstract	<code>\abstractname</code>
Contents	<code>\contentsname</code>
References	<code>\refname</code>
Chapter	<code>\chaptername</code>
Figure	<code>\figurename</code>
Proof	<code>\proofname</code>
Table	<code>\tablename</code>

Changing Lengths

An Overview

Changing Some Defaults

Using `renewcommand`

A List of \LaTeX Defaults

Changing Lengths

A Warning

An Example

Linestyles and Fillstyles

Graphing

Nodes and Connections

It is possible to change some of the lengths that \LaTeX uses by default also. For this you will need the `\setlength` command.

Units		
cm	in	pt
mm	em	ex

Example: Typing `\setlength{\parskip}{.5in}` would put one-half inch vertical space between paragraphs in your document. You can change lengths with `\setlength` several times throughout the body of your document.

Length	Default (article)
<code>\parskip</code>	0 inches
<code>\parindent</code>	1.5 em

[An Overview](#)

[Changing Some Defaults](#)

[Using `renewcommand`](#)

[A List of L^AT_EX Defaults](#)

[Changing Lengths](#)

[A Warning](#)

[An Example](#)

[Linestyles and Fillstyles](#)

[Graphing](#)

[Nodes and Connections](#)

Whenever you are typing a length, you must include a unit of measurement.

The most common time for people to make this error is when the value is 0. You must type `0pt`, `0in`, `0cm`, or something. (Of course, these are all the same.)

An Overview

Changing Some Defaults

Using `renewcommand`

A List of \LaTeX Defaults

Changing Lengths

A Warning

An Example

Linestyles and Fillstyles

Graphing

Nodes and Connections

Open the first example file (`.tex`); build and view.

I have included several examples from this section for you.

- changing the name of the abstract
- changing the symbol at the end of a proof
- changing the `parskip` and `parindent` lengths
- changing the name of the bibliography

An Overview

Changing Some
Defaults

Linestyles and Fillstyles

Additional Packages

Linestyles

Other Line Options

Fillstyles

The Gradient Fillstyle

Transparent Filling

Example

Graphing

Nodes and Connections

Linestyles and Fillstyles

Additional Packages

An Overview

Changing Some Defaults

Linestyles and Fillstyles

Additional Packages

Linestyles

Other Line Options

Fillstyles

The Gradient Fillstyle

Transparent Filling

Example

Graphing

Nodes and Connections

There are tons of things to do with `pstricks`, and lots of people have written lots more code to allow you to do lots more things on top of `pstricks`. We'll need to load a few more packages today:

- `pstricks-add`
- `pst-grad`
- `pst-plot`
- `pst-node`

An Overview

Changing Some Defaults

Linestyles and Fillstyles

Additional Packages

Linestyles

Other Line Options

Fillstyles

The Gradient Fillstyle

Transparent Filling

Example

Graphing

Nodes and Connections

There are a lot of options on the way your lines look. The option `linestyle` can have the value `none`, `solid`, `dashed`, or `dotted`.

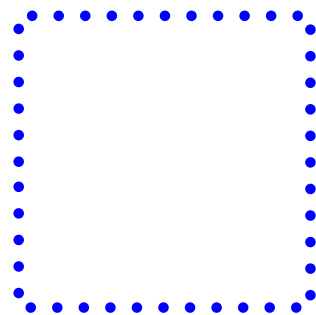
- `dashed` — If you use this option, you'll need to specify the spacing of the dashes too.

Example: `dash=.25in .1in .1in .15in`



- `dotted` — The one option to change here is `dotsep`.

Example: A dotted square and a dashed rectangle!



An Overview

Changing Some Defaults

Linestyles and Fillstyles

Additional Packages

Linestyles

Other Line Options

Fillstyles

The Gradient Fillstyle

Transparent Filling

Example

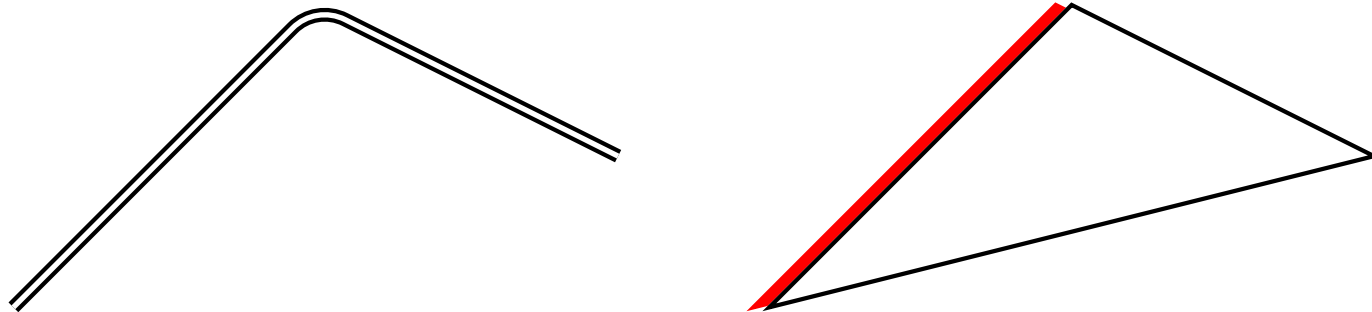
Graphing

Nodes and Connections

What else can we do with lines?

- `doubleline` — true/false; can change some things about this with `doublesep` and `doublecolor`;
- `shadow` — true/false; can change `shadowsize`, `shadowangle`, `shadowcolor`

Example:



An Overview

Changing Some Defaults

Linestyles and Fillstyles

Additional Packages

Linestyles

Other Line Options

Fillstyles

The Gradient Fillstyle

Transparent Filling

Example

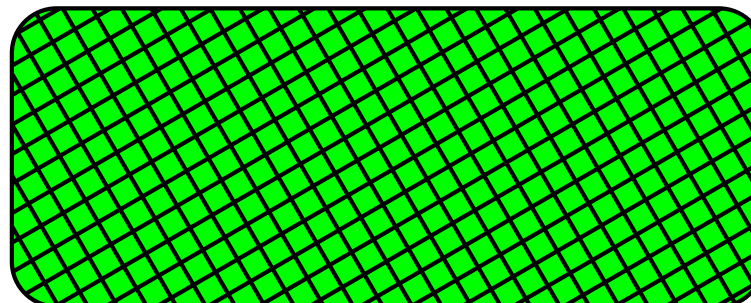
Graphing

Nodes and Connections

The options:

- `hlines`, `hlines*` — fill with lines at an angle (45 degrees is default);
 - change angle with `hatchangle`;
 - `hlines*` has background of `fillcolor`;
- `vlines`, `vlines*` — fill with lines perpendicular to `hlines`;
- `crosshatch`, `crosshatch*` — fill with intersecting lines, half at angle `hatchangle`

Example: `fillstyle=crosshatch*,hatchangle=30,fillcolor=green`



The Gradient Fillstyle

An Overview

Changing Some Defaults

Linestyles and Fillstyles

Additional Packages

Linestyles

Other Line Options

Fillstyles

The Gradient Fillstyle

Transparent Filling

Example

Graphing

Nodes and Connections

An Example:



- `fillstyle=gradient`
- Need to set: `gradbegin` and `gradend`. In above, blue and red. Should be a RGB color.
- `gradangle` — angle of gradient

Transparent Filling

An Overview

Changing Some Defaults

Linestyles and Fillstyles

Additional Packages

Linestyles

Other Line Options

Fillstyles

The Gradient Fillstyle

Transparent Filling

Example

Graphing

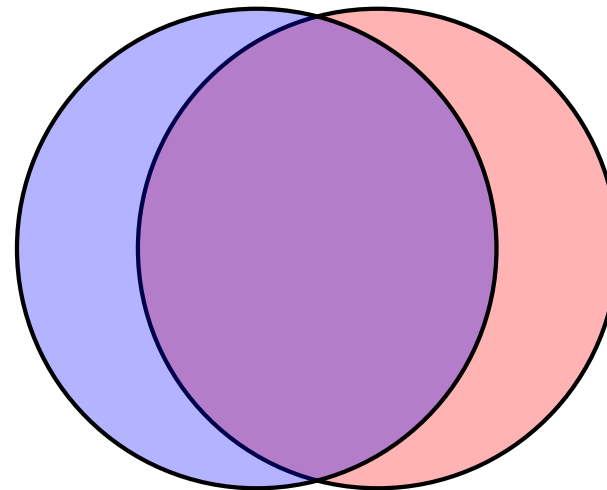
Nodes and Connections

The option `fillstyle=solid` can be modified to have a transparent effect. (This only shows up in the PDF.)

The `opacity=num` option can take on a number between 0 and 1 (opaque).

This option is available for lines as well with the `strokeopacity` option.

The following example has `opacity=.3`.



An Overview

Changing Some
Defaults

Linestyles and Fillstyles

Additional Packages

Linestyles

Other Line Options

Fillstyles

The Gradient Fillstyle

Transparent Filling

Example

Graphing

Nodes and Connections

Open the second example file (`.tex`); build and view.

An Overview

Changing Some
Defaults

Linestyles and Fillstyles

Graphing

Setting the Axes

Labels on the Axes

Other Options for Axes

More with Labels

Drawing Functions

A Graph of $\cos(x)$

Example

A Different Type of
Graph

A Sample Pie Chart

Practice

Nodes and Connections

Graphing

Setting the Axes

An Overview

Changing Some Defaults

Linestyles and Fillstyles

Graphing

Setting the Axes

Labels on the Axes

Other Options for Axes

More with Labels

Drawing Functions

A Graph of $\cos(x)$

Example

A Different Type of Graph

A Sample Pie Chart

Practice

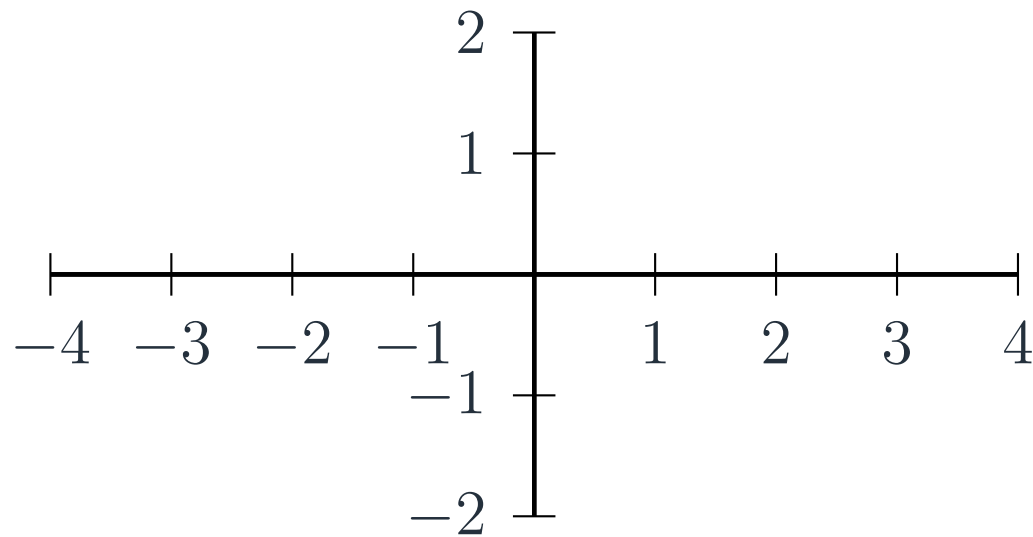
Nodes and Connections

Axes are produced with `\psaxes`. (See help file.)

Here is the syntax. It's very similar to the syntax for `\psgrid`.
`\psaxes* [pars] (x0,y0) (x1,y1) (x1,y2)`

The axes extend from $(x1, y1)$ to $(x2, y2)$. They intersect at $(x0, y0)$.

Example: `\psaxes(0,0)(-4,-2)(4,2)`



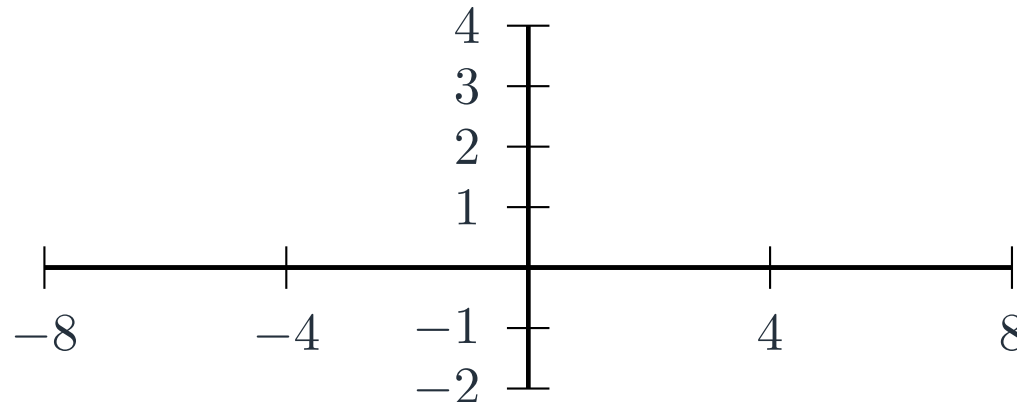
Labels on the Axes

Here are the options for putting labels on the axes.

Horizontal	Vertical	Default	Description
$0x=num$	$0y=num$	0	Label at origin
$Dx=num$	$Dy=num$	1	Label increment
$dx=dim$	$dy=dim$	0pt	Distance between labels

When $dx=0pt$, then horizontal labels are placed every $(Dx) \times (xunit)$.

Example: `\psaxes*[Dx=4] (0,0) (-8,-2) (8,4)`



- An Overview
- Changing Some Defaults
- Linestyles and Fillstyles
- Graphing
 - Setting the Axes
 - Labels on the Axes
 - Other Options for Axes
 - More with Labels
 - Drawing Functions
 - A Graph of $\cos(x)$
 - Example
 - A Different Type of Graph
 - A Sample Pie Chart
 - Practice
- Nodes and Connections

Other Options for Axes

An Overview

Changing Some Defaults

Linestyles and Fillstyles

Graphing

Setting the Axes

Labels on the Axes

Other Options for Axes

More with Labels

Drawing Functions

A Graph of $\cos(x)$

Example

A Different Type of Graph

A Sample Pie Chart

Practice

Nodes and Connections

Other options:

- `labels=all/x/y/none` — determines on which axes you will have labels;
- `ticks=all/x/y/none` — determines on which axes you will have ticks;
- `tickstyle=full/top/bottom` — determines whether the tick will extend above/below the axis; you can also change `tickcolor`;
- `ticksize=dim` — determines how far above/below the axis the tick will extend;
- usual adjustments for lines allowed (`linewidth`, `linecolor`, etc.)

An Overview

Changing Some Defaults

Linestyles and Fillstyles

Graphing

Setting the Axes

Labels on the Axes

Other Options for Axes

More with Labels

Drawing Functions

A Graph of $\cos(x)$

Example

A Different Type of Graph

A Sample Pie Chart

Practice

Nodes and Connections

Labels are automatically set in the current font, so you can precede `\psaxes` by a font size command to adjust this.

You can also get fancy by using `\pshlabel` and `\psvlabel`.

Here's the start to (just about) every graph I draw:

```
\begin{center}
\def\pshlabel#1{\scriptsize #1}
\def\psvlabel#1{\scriptsize #1}
\psset{xunit=1cm,yunit=1cm,arrowscale=1.5}
\begin{pspicture}(x,x)(x,x)
\psaxes[linecolor=lightgray,ticks=-4pt 0,
        tickcolor=lightgray,labels=all,
        ticks=all]{<->}(0,0)(x,x)(x,x)
```

Drawing Functions

An Overview

Changing Some Defaults

Linestyles and Fillstyles

Graphing

Setting the Axes

Labels on the Axes

Other Options for Axes

More with Labels

Drawing Functions

A Graph of $\cos(x)$

Example

A Different Type of Graph

A Sample Pie Chart

Practice

Nodes and Connections

Here is the command for producing a graph of a function:

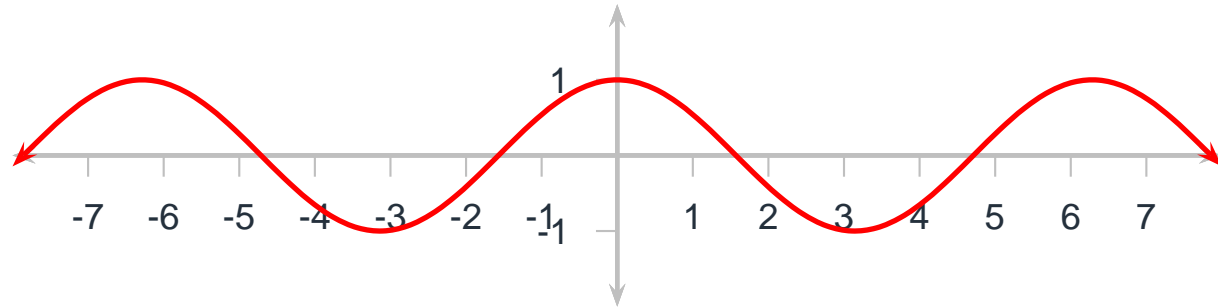
```
\psplot [arrows=<->,algebraic=true,  
        plotstyle=curve] {x1}{x2}{function}
```

This will start graphing the function with beginning x -coordinate $x1$ and end with $x2$.

Typing in the function works like you would guess, with one exception.

- x^2 — exponents;
- $\sin(x)$ — functions;
- $3*x$ — multiplication, **watch this one!**

A Graph of $\cos(x)$



```
\def\pshlabel#1{\scriptsize #1}
\def\psvlabel#1{\scriptsize #1}
\psset{unit=.5cm}
\begin{pspicture}(-8,-2)(8,2)
\psaxes[linecolor=lightgray,ticks=-4pt 0,
tickcolor=lightgray,labels=all,
ticks=all]{<->}(0,0)(-8,-2)(8,2)
\psplot[arrows=<->,algebraic=true,
plotstyle=curve,linewidth=1pt,
linecolor=red]{-8}{8}{cos(x)}
\end{pspicture}
```

An Overview

Changing Some Defaults

Linestyles and Fillstyles

Graphing

Setting the Axes

Labels on the Axes

Other Options for Axes

More with Labels

Drawing Functions

A Graph of $\cos(x)$

Example

A Different Type of Graph

A Sample Pie Chart

Practice

Nodes and Connections

An Overview

Changing Some
Defaults

Linestyles and Fillstyles

Graphing

Setting the Axes

Labels on the Axes

Other Options for Axes

More with Labels

Drawing Functions

A Graph of $\cos(x)$

Example

A Different Type of
Graph

A Sample Pie Chart

Practice

Nodes and Connections

Open the third example file (.tex); build and view

A Different Type of Graph

An Overview

Changing Some Defaults

Linestyles and Fillstyles

Graphing

Setting the Axes

Labels on the Axes

Other Options for Axes

More with Labels

Drawing Functions

A Graph of $\cos(x)$

Example

A Different Type of Graph

A Sample Pie Chart

Practice

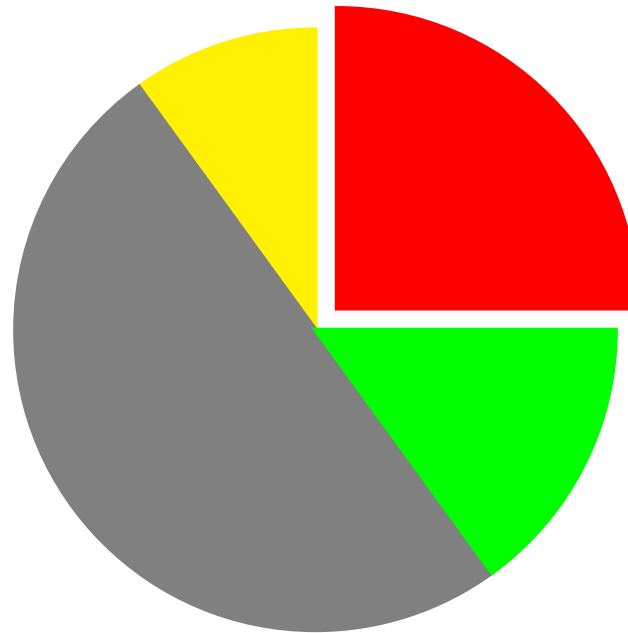
Nodes and Connections

There's a command for pie charts!

```
\psChart [options]{list1}{list2}{radius}
```

- `list1` — list of percentages, adding up to 100;
- `list2` — list of pieces to offset;
- Options
 - `chartSep` — distance to offset pieces;
 - `chartColor` — gray or color;
 - `userColor` — a comma-separated list of colors to use; if not included, will be chosen for you

A Sample Pie Chart



```
\psset{unit=2cm}  
\begin{pspicture}(-1,-1)(1,1)  
\rput(0,0){  
  \psChart[userColor={red,yellow,gray,green},  
chartSep=.2cm]{25,10,50,15}{1}{1}  
}
```

An Overview

Changing Some
Defaults

Linestyles and Fillstyles

Graphing

Setting the Axes

Labels on the Axes

Other Options for Axes

More with Labels

Drawing Functions

A Graph of $\cos(x)$

Example

A Different Type of
Graph

A Sample Pie Chart

Practice

Nodes and Connections

An Overview

Changing Some
Defaults

Linestyles and Fillstyles

Graphing

Setting the Axes

Labels on the Axes

Other Options for Axes

More with Labels

Drawing Functions

A Graph of $\cos(x)$

Example

A Different Type of
Graph

A Sample Pie Chart

Practice

Nodes and Connections

Let's practice!

Open the fourth example file (.pdf) and reproduce it.

An Overview

Changing Some
Defaults

Linestyles and Fillstyles

Graphing

Nodes and Connections

Getting Started

Nodes

Node Connections

Example

Node Connection

Parameters

More Node

Connections

Labeling Node

Connections

Practice

Nodes and Connections

An Overview

Changing Some Defaults

Linestyles and Fillstyles

Graphing

Nodes and Connections

Getting Started

Nodes

Node Connections

Example

Node Connection Parameters

More Node Connections

Labeling Node Connections

Practice

There are lots of occasions where connecting two things with a line is handy. This can be done with the basic drawing skills we have already. However, if you want to move things around, then you have to recalculate everything. The idea of *nodes* fixes this problem!

- You place objects in boxes and give them names. These are called your *nodes*.
- Then you connect these to each other as desired by simply referring to their names.

Warning: There is a ton of stuff you can do with this, and we'll just scratch the surface. See the handout for more information.

An Overview

Changing Some Defaults

Linestyles and Fillstyles

Graphing

Nodes and Connections

Getting Started

Nodes

Node Connections

Example

Node Connection Parameters

More Node Connections

Labeling Node Connections

Practice

Here is the syntax for creating nodes.

- `\rnode [ref] {name}{stuff}`
- `\cnode* [par] (x,y){radius}{name}` — this simply draws a circle
- `\circnode* [par] {name}{stuff}` — like `\pscirclebox`
- `\ovalnode* [par] {name}{stuff}` — like `\psovalbox`

An Overview

Changing Some Defaults

Linestyles and Fillstyles

Graphing

Nodes and Connections

Getting Started

Nodes

Node Connections

Example

Node Connection Parameters

More Node Connections

Labeling Node Connections

Practice

Two initial ways to connect nodes to each other.

- `\ncline* [par] {arrows}{nodeA}{nodeB}` — connects nodes with a line
- `\ncarc* [par] {arrows}{nodeA}{nodeB}` — connects nodes with an arc

One option that applies to all node connections: `nodesep`. This should be a length, and it defaults to 0pt.

`nodesep` is the distance between your node and where you want the node connection to stop.

An Overview

Changing Some
Defaults

Linestyles and Fillstyles

Graphing

Nodes and Connections

Getting Started

Nodes

Node Connections

Example

Node Connection

Parameters

More Node

Connections

Labeling Node

Connections

Practice

Open the fifth example file (`.tex`); build and view.

Node Connection Parameters

An Overview

Changing Some Defaults

Linestyles and Fillstyles

Graphing

Nodes and Connections

Getting Started

Nodes

Node Connections

Example

Node Connection Parameters

More Node Connections

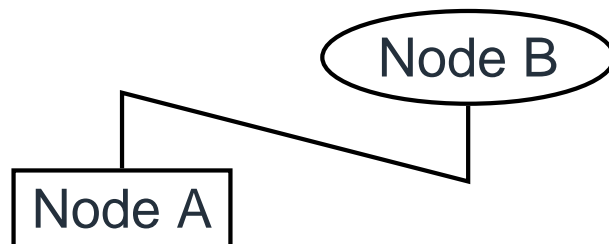
Labeling Node Connections

Practice

Two parameters that affect how the other node connections work:

- `angle` — specifies the angle at which the node connections join (default is 0)
 - You can set `angleA` and `angleB` as well.
- `arm` — specifies the length of the line segment at which the node connection joins (default is 10pt)
 - You can set `armA` and `armB` as well.

Example: `angleA=90, angleB=-90, arm=.5`



More Node Connections

An Overview

Changing Some Defaults

Linestyles and Fillstyles

Graphing

Nodes and Connections

Getting Started

Nodes

Node Connections

Example

Node Connection Parameters

More Node Connections

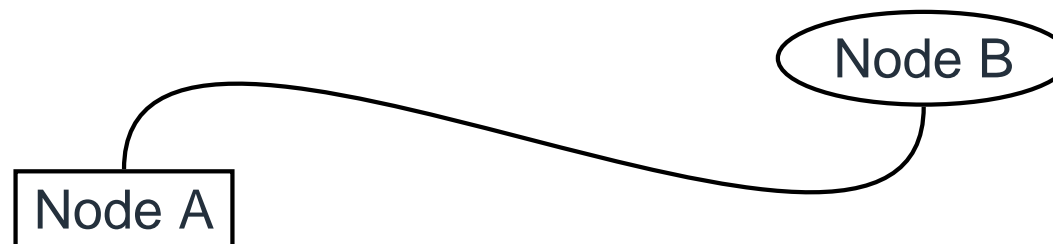
Labeling Node Connections

Practice

More ways to connect nodes to each other.

- `\ncdiag* [par] {arrows}{nodeA}{nodeB}` — three line segments make up connection (see previous slide for example); you can modify this with line options
- `\nccurve* [par] {arrows}{nodeA}{nodeB}` — draws a *Bezier curve* connecting the nodes; should specify angles
- see others on handout

Example: `angleA=90, angleB=-90`



Labeling Node Connections

An Overview

Changing Some Defaults

Linestyles and Fillstyles

Graphing

Nodes and Connections

Getting Started

Nodes

Node Connections

Example

Node Connection Parameters

More Node Connections

Labeling Node Connections

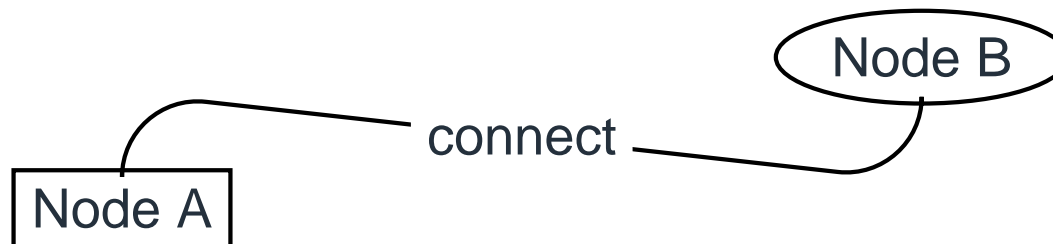
Practice

There are a lot of these, we'll just discuss three.

All of these need to come *immediately after* the node connection they will label.

- `\ncput* [par] {stuff}` — places label on the line
- `\naput* [par] {stuff}` — places label above the line
- `\nbput* [par] {stuff}` — places label below the line

Example: `\ncput*{connect}`



An Overview

Changing Some
Defaults

Linestyles and Fillstyles

Graphing

Nodes and Connections

Getting Started

Nodes

Node Connections

Example

Node Connection

Parameters

More Node

Connections

Labeling Node

Connections

Practice

Let's practice!

Open the sixth example file (.pdf) and reproduce it.