Math 291: Lecture 8

Justin A. James

Minnesota State University Moorhead
http://web.mnstate.edu/jamesju/Spr2017/Content/M291-01Handouts.html
jamesju@mnstate.edu

March 20, 2017

Outline

Including .JPG Graphics Files

- To get things started, you will need to go to the course website and download some files that we will use in our examples today.
- Go to the course webpage and follow the link "Week 8 Additional Materials"
- Download all of the files to the folder that you are using to save your practice files.

Including .JPG Graphics Files

- Open up an example document for this week. Add the normal preamble for a document.
- Next, add the package "graphicx".
- Then add the following command and try compiling.

```
\includegraphics[width=1.5in]{Image1.jpg}
```

- You should see the following:

- **Note:** To successfully include .jpg files, you will need to compile using the profile: LaTeX => PDF

Including .JPG Graphics Files

- You should also note that the command [width=1.5in] is an optional argument that is being used to "size" the image.
- Experiment a bit to see how changing this number alters the output when you compile.
- Another way to "size" an image file is to use the command [scale=#], where # is a decimal number that alters the size of the image.
- for example, [scale=.50] reduces the size of the image by 50%, and [scale=2.0] doubles the size of the image.

Including .JPG Graphics Files

- Next, try adding the file Image2.jpg to your document.
- It should look something like this:

Including .JPG Graphics Files

- You can also rotate an included image using the optional command: `angle`, which sets an angle or rotation clockwise in degrees.
- The option: `keepaspectratio`, which can equal either `true` or `false`, can be used to maintain the aspect ratio of the original image.

Including .JPG Graphics Files

- Start by including the file: Image3.jpg
- Then, add the optional commands `angle` and `keepaspectratio` to alter the previous image, displaying it at the angle shown below:

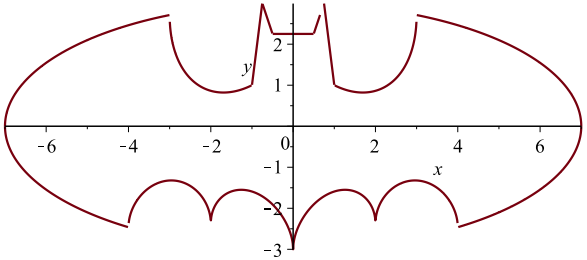
Outline

Including Postscript Graphics Files

- Add the package `pstricks` to the included packages in the preamble of your document.
- Change the Build Option to: Set the build to option to: LaTeX => PS => PDF.
- We are going to include the image file: "Bat.eps" in our document.
  - It will be based on a graph created using Maple
  - Open the file "Bat.mw" , then right-click on the graph.
  - In the context menu, choose "Export"
  - Then choose the file type ".eps" ("encapsulated postscript")
  - Save the resulting file in the same folder as your current document (the compiler will only be able to find graphics files in the same folder)

Including Postscript Graphics Files

- Include this new file in your document (also, go ahead and comment out the .jpg files as they will not compile in this build mode)
- the result should look something like this:



Outline

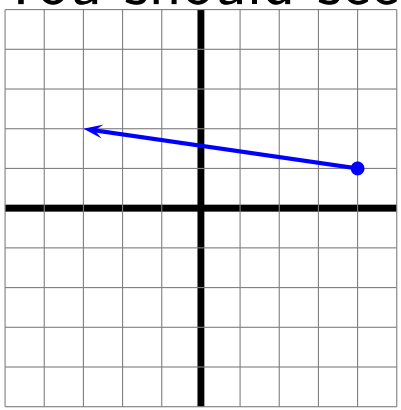
Creating Postscript Graphics: Lines

- Copy the contents from the file `smallgraph.tex` into your practice file.
  - These commands create a square grid going up to 5 units from the origin.
  - The `largegraph.tex` files gives you one that goes 10 units from the origin.
- Just prior to the `\end{pspicture}` command, use the `\psline` command to create a straight line from $(-3, 2)$ to $(4, 1)$.

```
\psline[linewidth=3pt,linecolor=blue,
arrowsize=10pt,dotsize=10pt]{<-*}(-3,2)(4,1)
```

Creating Postscript Graphics: Lines

- You should see:



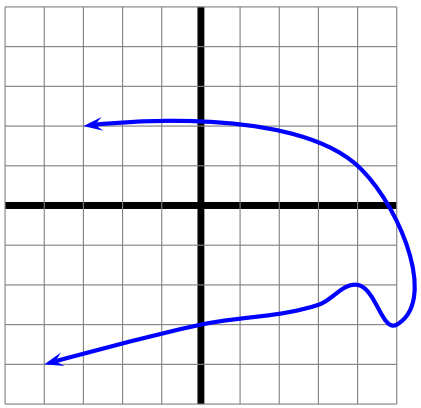
- See what happens if you change the symbols $<$ and $*$ into $<$, $>$, $*$, or o.
- Then, see what happens if you add more ordered pairs to the list in the last command above.

Creating Postscript Graphics: Curves

- To draw a curve, the command syntax is basically the same, including the options available, but the command used is: `\pscurve`.
- In the command above, add points until you have 5 or 6 of them, and change the `\psline` command into `\pscurve`.
- Compile and see what the graphic looks like.
- Notice that this command draws a curve fitting the points you supplied, in the exact order that you supplied them.

Creating Postscript Graphics: Curves

- The resulting curve should look something like this: (depending on which specific points you added, or course)



- I added the points: $(5,-3)(4,-2)(3,-2.5)(0,-3)(-4,-4)$

Creating Postscript Graphics: Curves

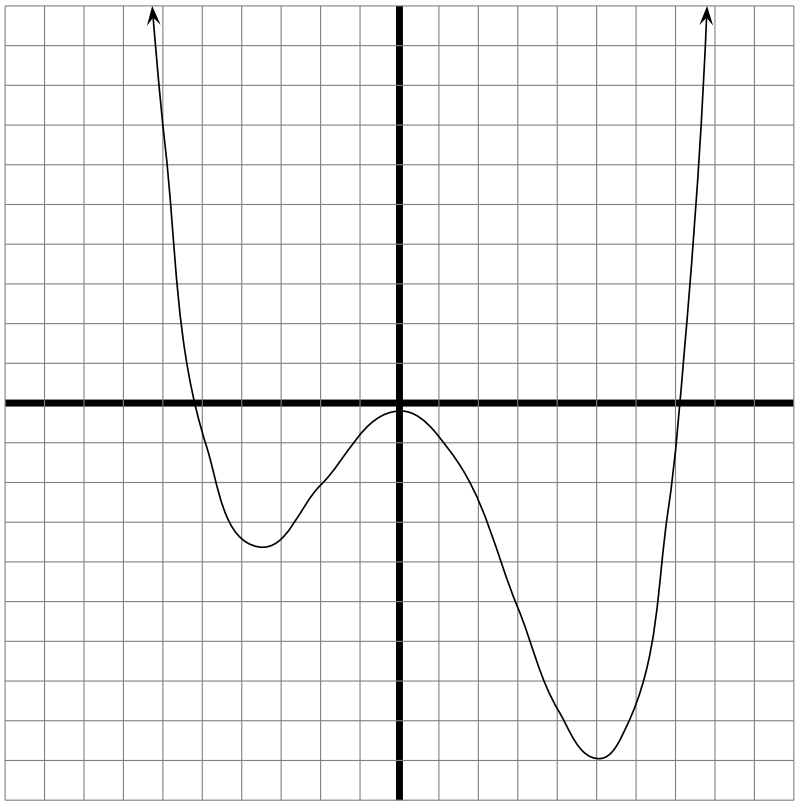
- The following is the command for graphing the function (on the large grid) $f(x) = \frac{1}{12}x^4 - \frac{1}{6}x^3 - 3x^2 - 1$, with the y-scale being 5 units per grid line.

```
\pscurve[linewidth=1.2pt,arrowsize=10pt]{<->}
(-6.27,10)(-5.95,6.547)(-4.89,-1.10)(-3.56,-3.624)
(-2,-2.067)(0,-.2)(1.277,-1.203)(3,-5.15)(4.04,-7.76)
(5.06,-8.954)(5.745,-8.17)(6.81,-2.72)(7.234,1.425)
(7.65,6.99)(7.8,10)
```

- The points used were found by evaluating the function $f(x)$ at various inputs (and rounding the results).

Creating postscript graphics — curves

- The following is the result of adding the previous pscurve command at the end of the "largegraph" command file:



Other Shapes Available for PS Graphics

The following is a list of some other commands.

- `\psdots[...](x,y)` [or just `\psdot` for just one].
- `\psframe[...](x0,y0)(x1,y1)`
- `\psdiamond[...](h,k)(x-length,y-length)`
- `\pscircle[...](h,k){r}`
- `\psellipse[...](h,k)(x-length,y-length)`
- `\pswedge[...](h,k){r}{start-angle}{end-angle}`
- `\parabola[...](x0,y0)(h,k)`
- `\psarc[...](h,k){r}{start-angle}{end-angle}`
- `\psbezier[...](x1,y1)(x2,y2)(x3,y3)(x4,y4)`
- `\pspolygon[...](supply n ordered pairs)`
- `\pstriangle[...](x0,y0)(x1,y1)`
  base centered at $(x0, y0)$ base length $x1$ height $y1$

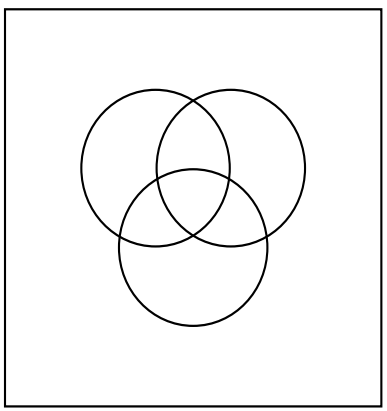
Postscript Graphics Options

Some options that can be used to modify PS Graphics:

- linewidth=(number), linecolor=color, arrowsize=(number), dotsize=(number), arrowlength=(number)
- Colors: black, darkgray, gray, lightgray, white, red, green, blue, cyan, magenta, yellow, or
  - `\definecolor{name}{rgb}{# # #}`, where each $\# \in [0, 1]$
- Line terminators: $-$, $<$, $>$, $<<$, $>>$, (,), $|$, [,], o, $*$, $|<$, $>|$, c
- doubleline=true (and then doublesep=(number))
- linestyle=solid, dashed, dotted, none
- dotstyle=$*$, $+$, $|$, o, x, asterisk, diamond*, diamond, oplus, otimes, pentagon*, pentagon, square*, square, triangle*, triangle
- linearc=(number) showpoints=true/false
- fillstyle=none, solid, vlines, vlines*, hlines, hlines*, crosshatch, crosshatch*, (and gradient, with pst-grad package)
- fillcolor=

Postscript Graphics Options - Example

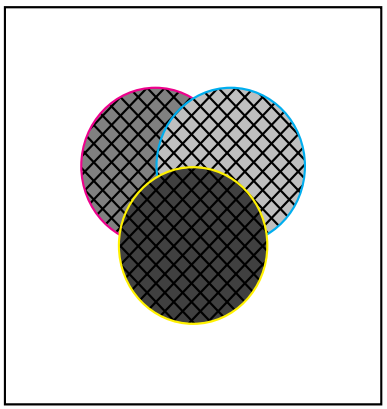
- To see an example of how to use some of these commands, we will build a Venn diagram:
- The following commands create an empty 3-circle Venn diagram:



```
\begin{pspicture}(0,0)(5,5)
\pspolygon(0,0)(5,0)(5,5)(0,5)(0,0)
\pscircle(2,3){1}
\pscircle(3,3){1}
\pscircle(2.5,2){1}
\end{pspicture}
```

Postscript Graphics Options - Example

- A few alterations give the following:



```
\begin{pspicture}(0,0)(5,5) \resizebox{1in}{1in}{
\begin{pspicture}(0,0)(5,5) \pspolygon(0,0)(5,0)(5,5)(0,5)(0,0)
\pscircle[linecolor=magenta, fillcolor=gray,
fillstyle=crosshatch*](2,3){1} \pscircle[linecolor=cyan,
fillcolor=lightgray, fillstyle=crosshatch*](3,3){1}
\pscircle[linecolor=yellow, fillcolor=darkgray,
fillstyle=crosshatch*](2.5,2){1} \end{pspicture} }
```