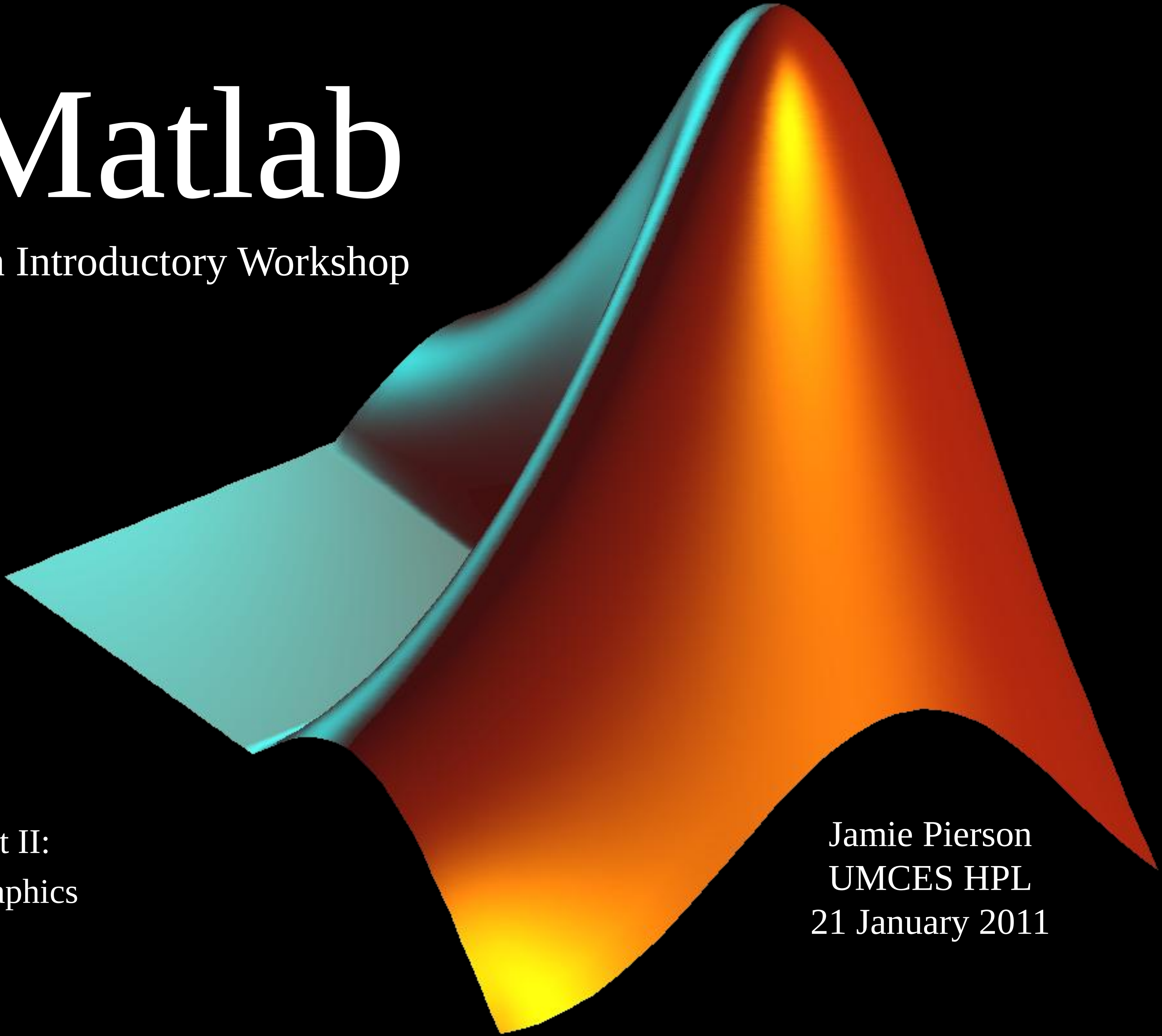
# Matlab

An Introductory Workshop



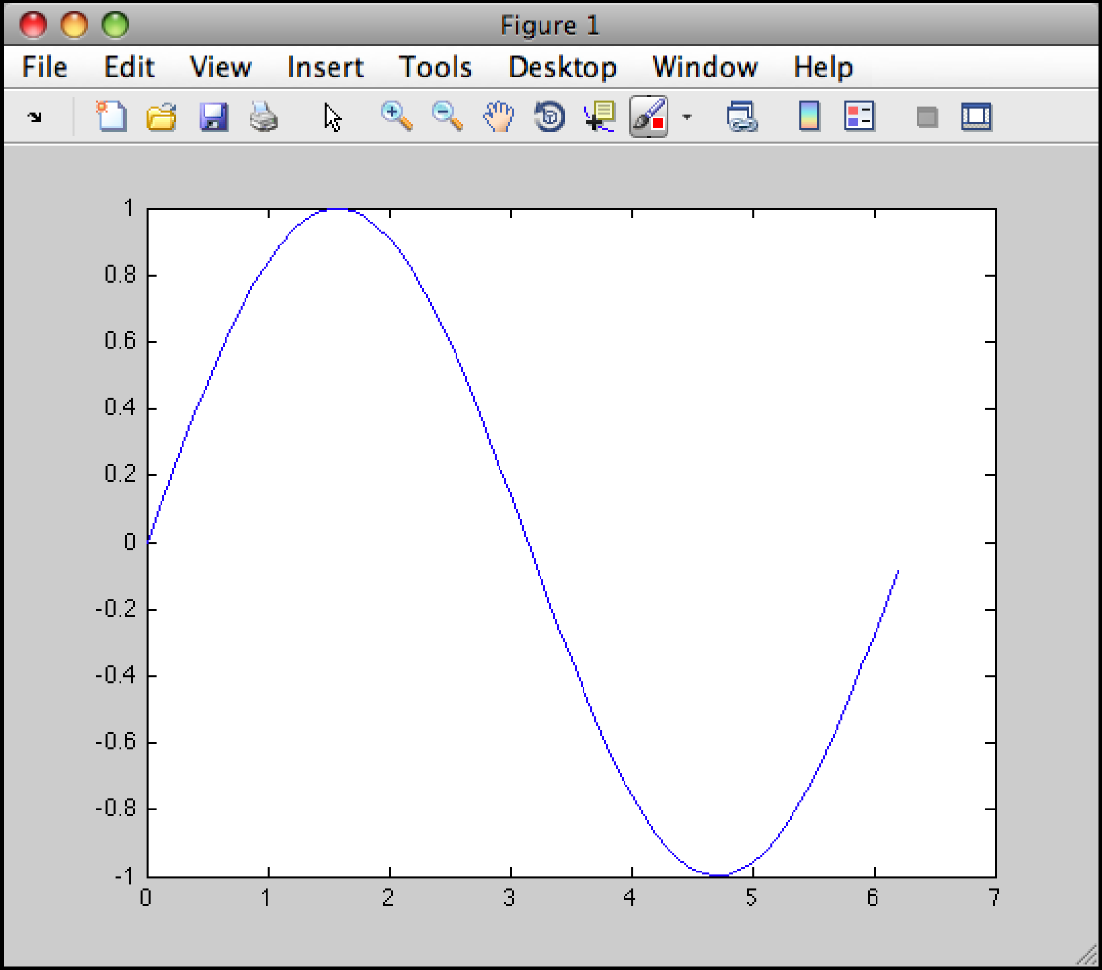
Part II:
Graphics

Jamie Pierson
UMCES HPL
21 January 2011

## GRAPHING BASICS

# Plotting in Matlab

```
>> x=0:0.1:2*pi;
>> y=sin(x);
>> plot(x,y);
```



# Some Line Properties

`‘linestyle’`

| **Specifier** | **Linestyle** |
|---|---|
| `‘-’` | Solid |
| `‘—’` | Dashed |
| `‘:’` | Dotted |
| `‘-.’` | Dash-Dot |
| `‘none’` | None |

## ‘Marker’

| Specifier | Marker Type |
|---|---|
| `'+'` | Plus sign |
| `'o'` | Circle |
| `'*'` | Asterisk |
| `'.'` | Point |
| `'x'` | Cross |
| `'square' or 's'` | Square |
| `'diamond' or 'd'` | Diamond |
| `'^'` | Upward-pointing triangle |
| `'v'` | Downward-pointing triangle |
| `'>'` | Right-pointing triangle |
| `'<'` | Left-pointing triangle |
| `'pentagram' or 'p'` | Five-pointed star (pentagram) |
| `'hexagram' or 'h'''` | Six-pointed star (hexagram) |
| `'none'` | No marker (default) |

# 'Color'

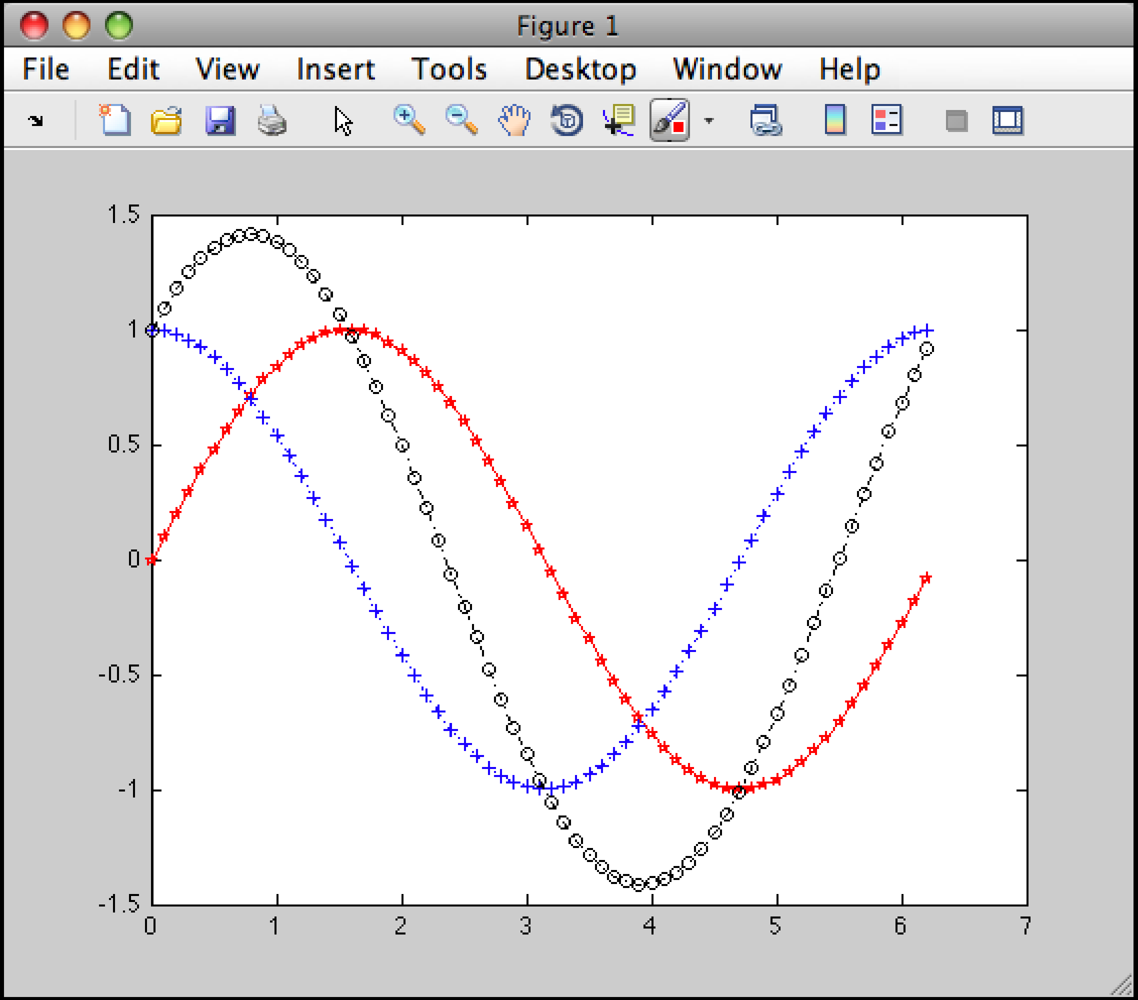
| RGB Value | Short Name | Long Name |
|---|---|---|
| [1 1 0] | y | yellow |
| [1 0 1] | m | magenta |
| [0 1 1] | c | cyan |
| [1 0 0] | r | red |
| [0 1 0] | g | green |
| [0 0 1] | b | blue |
| [1 1 1] | w | white |
| [0 0 0] | k | black |

## X,Y Scatter Plots

For simple X,Y scatter plots you can string together line, color, and marker information within the “plot” function

# Example

```
>> x=0:0.1:2*pi;
>> y1=sin(x);
>> y2=cos(x);
>> y3=y1+y2;
>> plot(x,y1,'r-p')
>> plot(x,y2,'b:+')
>> plot(x,y3,'k-.o')
>>
```



## Bar plots

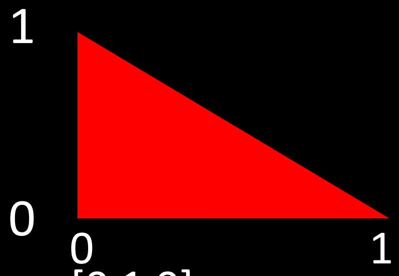
- bar represents a 1D function using 2D objects--rectangles
- the rectangles are represented in Matlab as a patch object
  - Patches are polygons
  - Patches can have complicated colors
  - Patches (or related surface objects) are used by all higher-order functions

## Key properties of patch objects

- edgecolor--color of the edges
- facecolor--color inside the the patch
- Both of these can be set to a specific color (or none)
- Or, we can prescribe another dimension of data at each vertex and let it control the color

## Drawing patches

- Lots of functions produce patches
- patch is the lowest level function (followed closely by fill)
  - patch(x,y,c)--x and y specify vertex coordinates, c controls the color
  - patch(X,Y,C)--Each column of X, Y, and C is a separate patch

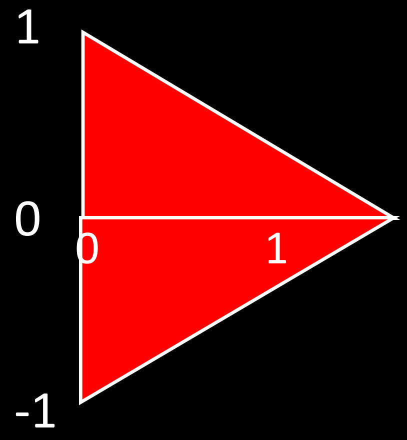


x=[0 1 0]
y=[0 0 1]
c=‘r’

# Drawing patches

- patch(X,Y,C)--Each column of X, Y, and C is a different polygon,
  - but same object!
  - X and Y must be the same size
    - Each polygon must have same number of vertices (rows)

```
X=[0 1 0;
   0 1 0]’
Y=[0 0 1;
   0 0 -1]’
C=‘r’
```



## Key properties of patch objects

- edgecolor--color of the edges
- facecolor--color inside the the patch
- Both of these can be set to a specific color (or none)
- Or, we can prescribe another dimension of data at each vertex and let it control the color

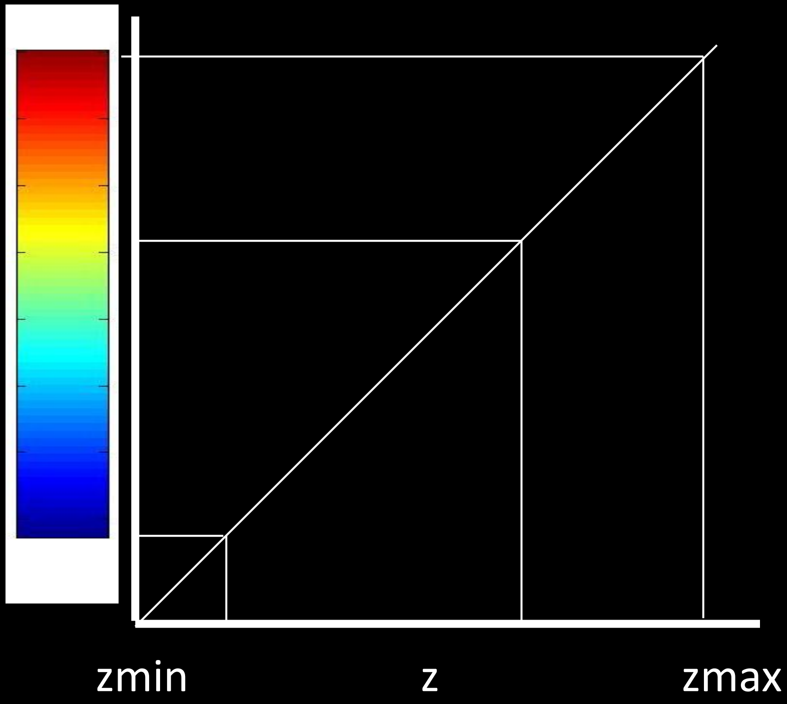
## Visualizing Grids

- Matlab’s surface-based functions want grids:
  - pcolor
  - contour & contourf
  - surf
  - mesh

Adapted from A. Pershing 2010

# Colorizing z

- A standard way of representing 2D data is to make color indicate z

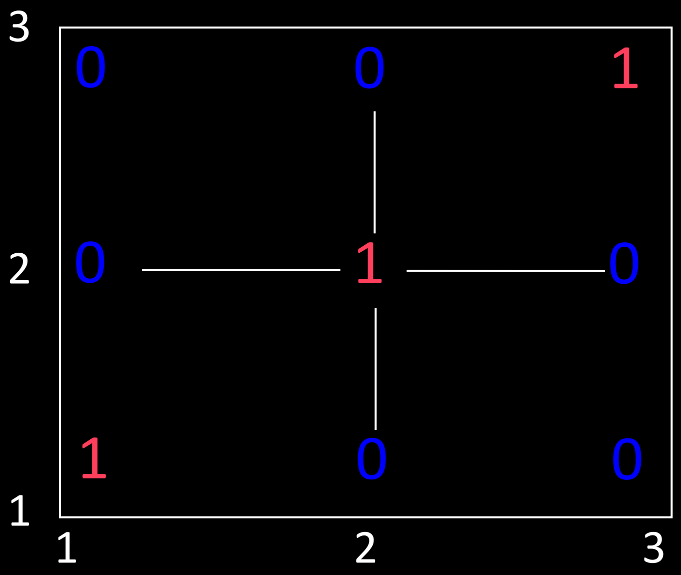


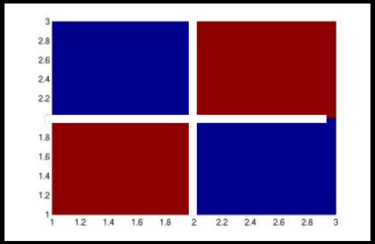
# pcolor

- pcolor(x,y,Z) will colorize Z on grid defined by x and y
  - Z=m-by-n, x=1-by-n, y=m-by-1
- pcolor(X,Y,Z) will colorize Z on an irregular grid
  - X,Y, and Z all m-by-n
- h=pcolor(…) gets the handle.
  - The object is actually a surface object
  - surface objects are nearly identical to patches, but must be constructed from quadrilaterals (a grid)

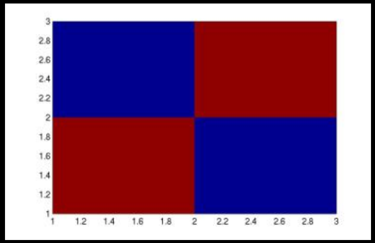
# How it works

- h=pcolor(eye(3));

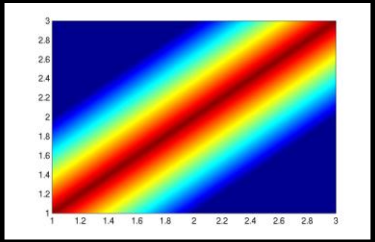




shading('faceted')
color of cell is set by lower left-hand corner



shading('flat')
edgecolor='none'



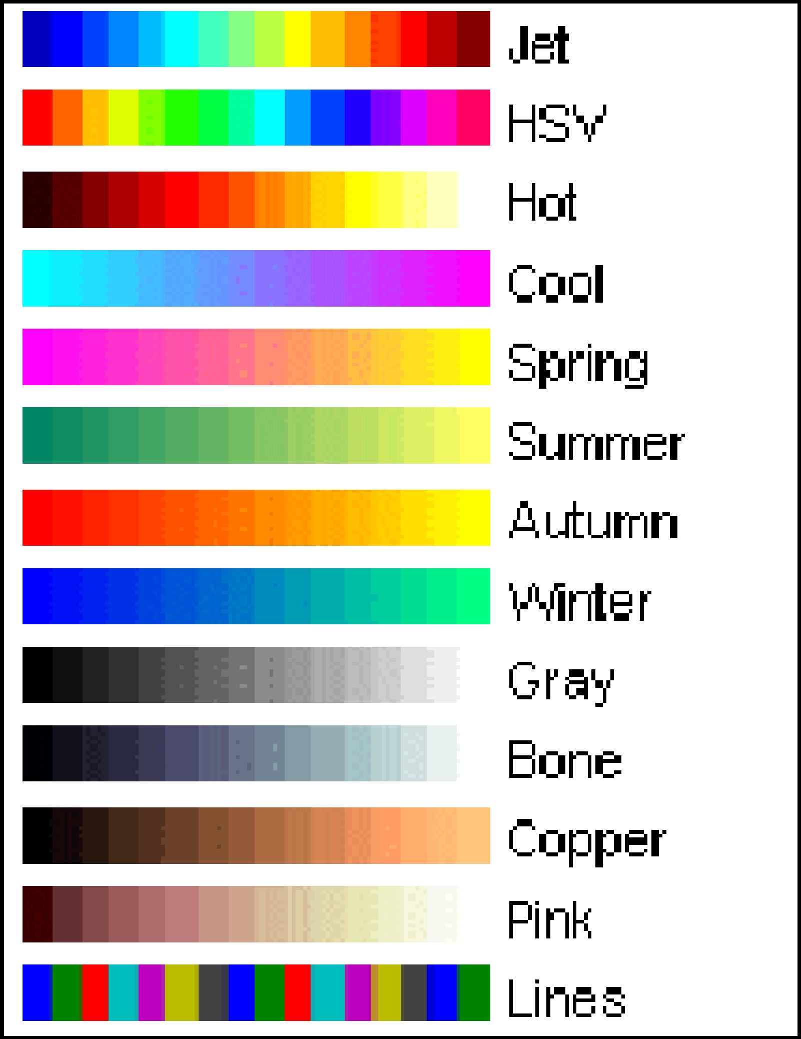
shading('interp')
interpolates between vertices to get color

Adapted from A. Pershing 2010

## Controlling pcolor

- shading(str) sets ‘facecolor’ property to str
  - flat, faceted or interp
- colorbar shows a colorbar
- caxis([zmin, zmax]) controls the color limits
  - same as set(gca,’clim’,[zmin, zmax])
- colormap(cmap)--changes the colors.  help graph3d lists the built in colormaps

# Built-in Colormaps



Adapted from A. Pershing 2010

## HANDLE GRAPHICS

## Handle Graphics

- Handles are just floating point numbers, but
  - they are a unique identifier
  - they function as pointers to Matlab graphics objects
- We can use them to get info about objects and to change the objects' properties
  - everything you see in a figure is a graphic object or part of one
  - every object has a handle
  - every object has a set of properties that can be changed using the handle

## Handle Graphics

- Get properties with “get”
  - get(h)--lists all of the properties of h and their values
  - get(h,property)--returns the value of the property
    - types vary with property (some are text, some are arrays)
- Change properties with “set”
  - set(h)--lists all of the properties and their default values
  - set(h,property,value, property, value,…)--changes the values of the properties
- set is “vectorized” so you can change properties of lots of objects simultaneously

# Handle Graphics

```
get(gca)
```

*get current axis*

Lists all properties of the currently selected axis

```
get(gcf)
```

*get current figure*

Lists all properties of the currently selected figure

## Figures and Axes

- Figures and axes are also objects
- We can get handles to them and change their properties
- These objects are created as needed when graphics routines are called
  - They can also be created explicitly

## Figures

- If no figures are open, Matlab will create one when you call a graphics routine
- If a figure is open, then any subsequent graphics will be placed in that figure
- Figures can be created explicitly by calling figure
  - h=figure;  --creates a new figure, handle saved in h
- Figures can be cleared with clf

## Multiple Figures

- If multiple figures are open and you call plot, where does the new line go?
  - One of the figures is the "current figure"
    - the current figure is the last one you plotted into or the last one created
    - the function gcf returns a handle to the current figure

## Multiple Figures

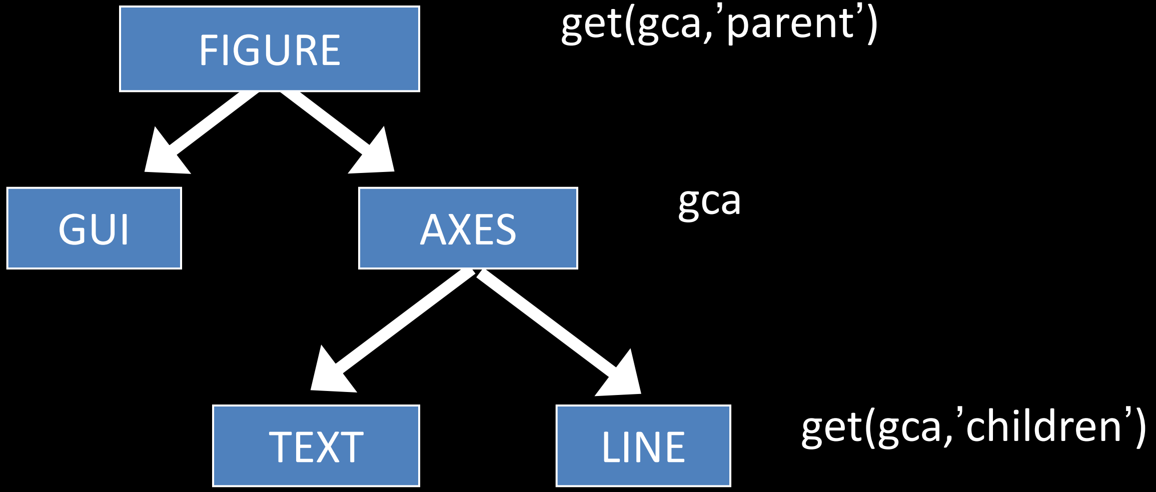
- More ways to use figure
  - figure(n)
    - if figure number n  doesn’t exist, then it is created
    - if it exists, then it becomes the current figure
    - regardless, it will be the current figure
  - figure(h)--changes current figure to h (a figure handle)
- Delete figures with close
  - close(h)--closes figure with handle h
  - close(n)--closes figure number n
  - close all closes all figures

## Handle Properties--ALL objects

- The last 18 properties from get(h) are properties that all objects have
- Most important:
  - Parent--handle to parent object
  - Children--handles to child objects
  - Type--tells what it is (e.g. line)
  - Visible--(on/off) can hide objects
- A few other general properties are used for GUI’s

## Handle Tree

- Matlab organizes graphics like a tree
- The parent and children fields allow you to traverse the tree



## MULTIPLE AXES

## Multiple Axes

- In many ways, axes and figures are managed the same way, but…
  - axes are not numbered in any intelligible way, so axes(1) is meaningless
  - If you have multiple axes, you must save their handles and switch axes using axes(h)
  - Matlab's subplot command returns some of this functionality (example in a minute)

## Axes Properties

- Box--on/off --switches box around axes on and off
- Camera stuff--controls how the objects in axes are viewed
- Clim--limits for color mapping
- Color--color of the axes (usually white)
- Font stuff--controls fonts on labels
- Line stuff--properties of the axes lines (options for grid lines)

## Axes Properties

- Position--controls where the axes goes in figure
- Tick stuff--controls properties of tick marks
- Title--handle of text object with axes title
  - title('axes title') will title the axes
- Units--several options, default is normalized
- Etc.

## Axes Properties

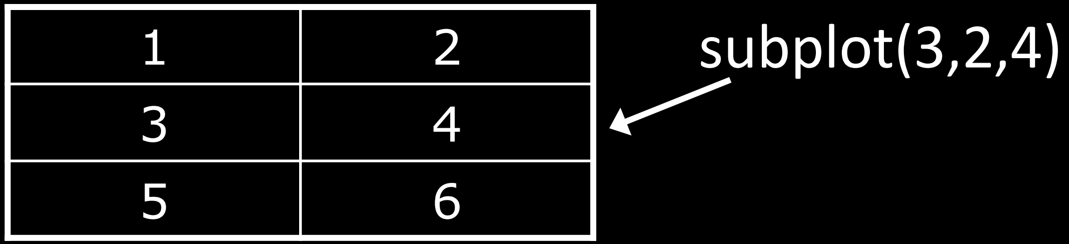
- Axes have 3 axes: X (horizontal), Y (vertical),  Z (height)
- We can control the range and appearance of each
  - XColor--color of the axis lines
  - XGrid--on/off turns grid lines on or off
  - XLabel--handle of text object with x axis label
    - xlabel('x label') will label the x axis
  - XLim--range of the x axis
    - cas set xlim and ylim togther with axis command
  - XScale--linear/log --can plot on a log10 scale

## Axes Properties

- Xtick--where the tick marks (and labels) occur
- XTickLabel--the labels
  - Matlab works hard to pick “good” labels (base 10)
  - Can change labels by setting ticklabel
    - set(gca, ‘xticklabel’, ‘first|second|third’)
- Setting Xtick or XTickLabel will change XTickMode or XLabelModes to ‘manual’--may give problems if figure is resized

# subplot

- You can produce multiple axes laid out in a regular fashion using subplot
  - subplot(m,n,j) produces the jth axes from an m-by-n grid of axes



  - if subplot(m,n,j) exists, then calling it will set gca to this axes
  - h=subplot(m,n,j) returns the handle to the jth subplot

## PRINTING AND SAVING FIGURES

## Printing

- Print through GUI or command line
  - print -depsc fname.eps will save gcf to an EPS file
  - print -djpeg fname.jpg will save gcf to a JPEG
  - Can also save figure to a .fig file from the GUI
    - Opening the file (from GUI) will recreate the figure

*See help for "print" to find more properties*

# Printing

| Graphics Format | Bitmap or Vector | Driver |
|---|---|---|
| **BMP** | Bitmap | Ghostscript |
| **EMF** | Vector | MATLAB |
| **EPS** | Vector | MATLAB |
| **HDF** | Bitmap | MATLAB |
| **ILL** | Vector | MATLAB |
| **JPEG** | Bitmap | MATLAB |
| **PBM** | Bitmap | Ghostscript |
| **PCX** | Bitmap | Ghostscript |
| **PDF** | Vector | Ghostscript |
| **PGM** | Bitmap | Ghostscript |
| **PNG** | Bitmap | MATLAB |
| **PPM** | Bitmap | Ghostscript |
| **SVG** | Vector | MATLAB |
| **TIFF** | Bitmap | MATLAB |

## Printing

*If you are familiar with Adobe Illustrator or another “vector graphics” program I suggest using “pdf” output with “painters” rendering*

```
>> print -dpdf -painters [filename]
```

## Printing and Saving

- Lots of matlab functions (print, save, load), allow you to type your input outside parentheses
  - Ex: print -djpeg foo.jpg
- However, Matlab is hiding the real function call (and function) from you.
- Inputs typed after a command, without parentheses are passed as strings to the function
  - print('-djpeg', 'foo.jpg');
  - Useful in your own functions

  ***You must use this method () if you are specifying a path or filename with spaces in it***

## ANIMATIONS AND MOVIES

## Animations

- Animations are extremely easy:
  1. Make an image
  2. Change it
  3. Repeat

# Animations in Matlab

- You can do this with a for-loop
  - for j=1:n
    - Make image n
  - end
- Problem:  Matlab does this too fast
  - Solution: insert pause command
    - pause; %waits until user hits a key
    - pause(t); %pauses for t seconds

## Creating AVI files

- Problems with previous scheme
  - Not portable (only in Matlab)
  - Not efficient: must render each image every time
- Solution: save to a standard movie format
  - AVI is a simple video format which is easy to create with Matlab

# Creating AVI files

- Procedure is similar to before:
  - First, open a file:
    - mov = avifile(name); %opens file called name
  - Set any options
    - mov.Quality=100;%quality of images
    - mov.Compression='None'; %compression
    - mov.Fps=fps;%frames per second
  - Create an image as before
  - Then, capture it:
    - F = getframe(gcf);%capture the frame
    - mov = addframe(mov,F);%add it to the movie
  - Repeat
  - Close the movie
    - mov=close(mov);