Welcome back to **CME 292**
Advanced MATLAB for Scientific Computing

WINTER 2023

MATLAB File Exchange

https://www.mathworks.com/matlabcentral/fileexchange/

MATLAB File Exchange is a very useful forum for finding and sharing solutions to many MATLAB-related problems.

It contains many free, open-source MATLAB and Simulink code.

- Data Analysis
- Data Import/Export
- Desktop Tools and Development Environment
- External Interfaces
- GUI Development
- Graphics and Visualization
- Mathematics
- Object-Oriented Programming
- Programming and Data Types
- Clean integration of MATLAB figures in LATEX documents
- Plot formatting and manipulation
- Interfacing to iPhone, iPad, Android, Kinect devices
- Interfacing to Google Earth and Maps
- …

Graphics and Data Visualization

CME 292 LECTURE 2

1/12/2023

Stanford University

Outline

Graphics and Data Visualization

- Graphic handles
- Advanced plotting
- Plots for publications
- Animations

Graphic Handles

Graphics and Data Visualization

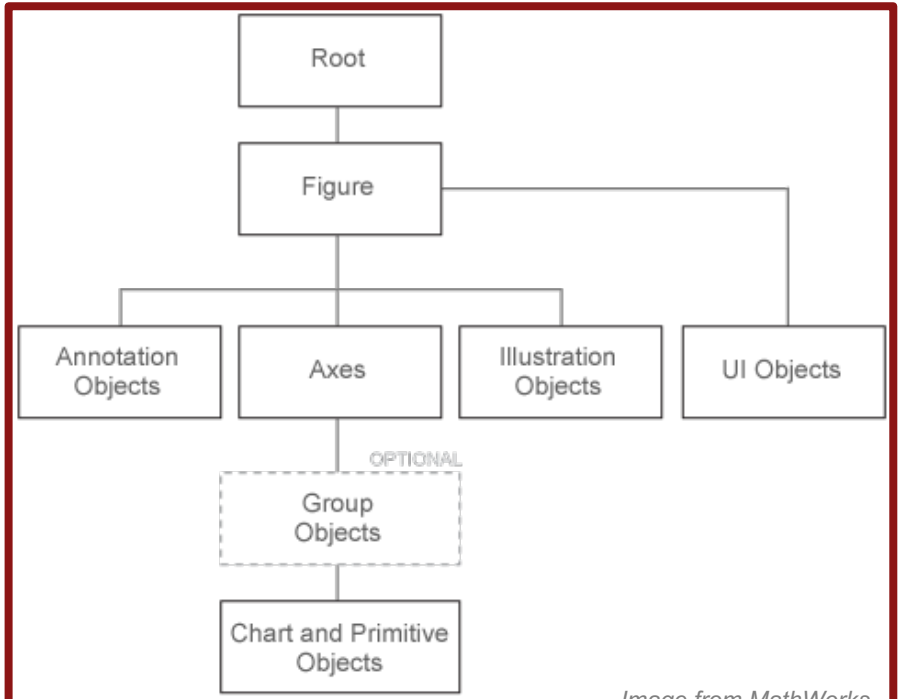


Image from MathWorks

Graphics Objects

Basic drawing elements used by MATLAB to display data

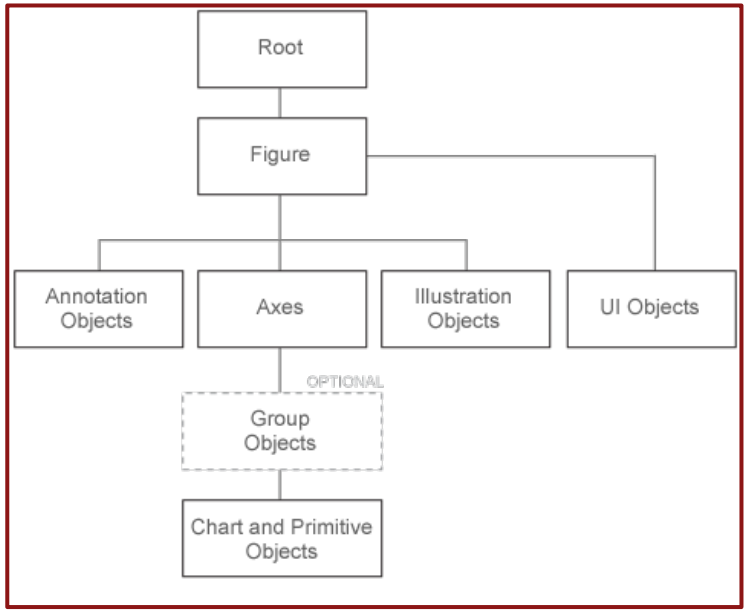
- Each object instance has unique identifier, the *handle.*
- Graphics objects behave like other MATLAB objects.
- Objects are organized in hierarchy.

Parent-Child Relationship

- E.g., the parent of an axes is a figure.

A handle refers to a specific instance of a graphics object used to set and query the values of the object properties.

Organization of Graphics Objects

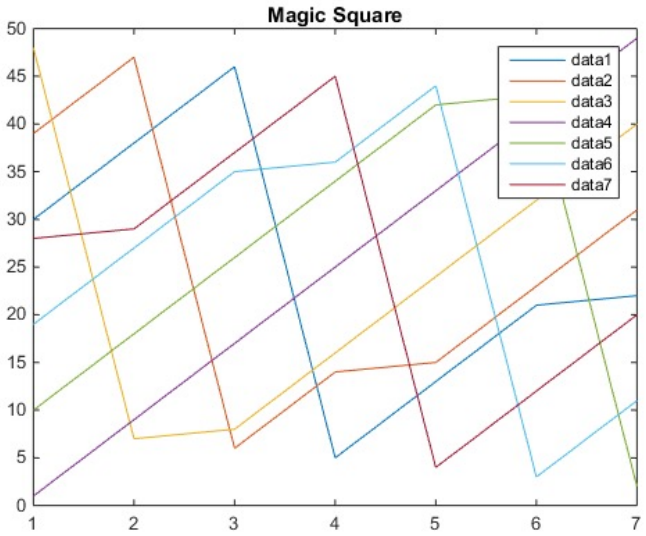


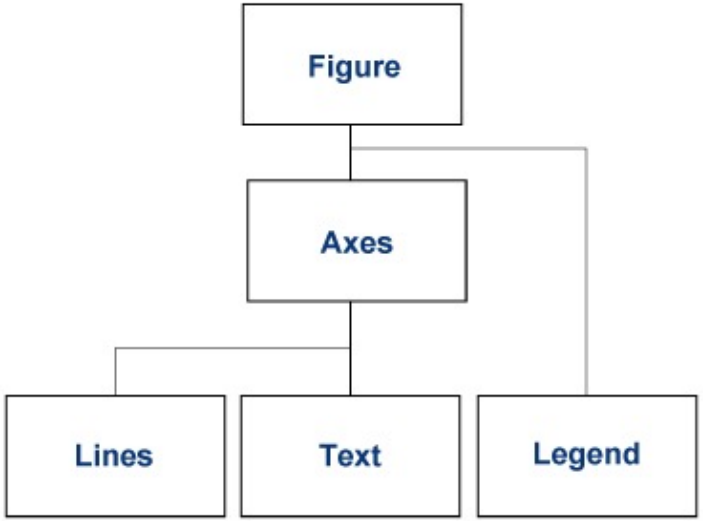
Core graphics object

- `axes, image, line, patch, rectangle, surface`

Composite graphics object

- Plot objects: `areaseries, barseries, contourgroup, errorbarseries, lineseries, quivergroup, scattergroup, staircase, stemseries, surfaceplot`
- Annotation objects: `arrow, doublearrow, ellipse, line, rectangle, textarrow, textbox`
- Illustration objects: `legend, colorbar`
- Group objects: `hggroup, hgtransform`
  - Group objects can contain any of the objects that axes can contain, such as lines, surfaces, text, etc., also other group objects.
- User Interface objects





Images from MathWorks

Quiz

Run the following piece of code.

```
figure; axes(); hold on
x = linspace(0,2*pi,100);
for k = 1:10
    plot(x,sin(k*x));
end
hold off
```

Modify the code such that `sin(k*x)` is plotted versus x for even k's in one single figure, and for odd k's in another figure.

Solution

```
fH(1) = figure; aH(1) = axes(); hold on
fH(2) = figure; aH(2) = axes(); hold on

x = linspace(0,2*pi,100);
p1 = gobjects(5,1);
p2 = gobjects(5,1);
for k = 1:10
    if mod(k,2)==0 % even
        p1(ceil(k/2)) = plot(aH(1),x,sin(k*x));
    else
        p2(ceil(k/2)) = plot(aH(2),x,sin(k*x));
    end
end
% add legend and title
even_k = strcat('k=',string(num2cell(2:2:10)));
legend(p1,even_k);
title(aH(1),"even");
odd_k = strcat('k=',string(num2cell(1:2:9)));
legend(p2,odd_k);
title(aH(2),"odd");
```

```
figure; axes(); hold on
x = linspace(0,2*pi,100);
for k = 1:10
    plot(x,sin(k*x));
end
hold off
```

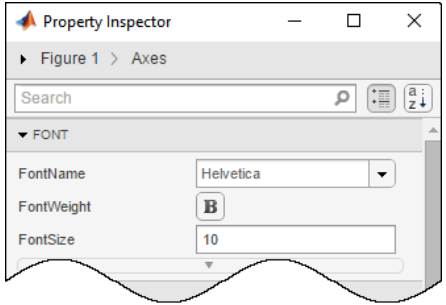
Working with Graphics Objects

There are 2 ways to access/modify object properties.

1. Use dot notation to refer to a particular object and property.
2. Use the `set` and `get` functions to access properties.
   - `s,V: structure; pn, pv: cell array`
   - Set property-value pairs: `set(han,s)`
   - Set value of property `pn{i}` to `pv{i}`: `set(han,pn,pv)`
   - Store all properties-value pairs in a structure: `V = get(han)`
   - Store property value in V: `V = get(han,'Property')`

Property inspector

- Click the Property Inspector icon on Figure toolbar, or
- Use the command prompt `inspect`.



Properties of Figure, Axes, and Plot Objects

Type object name in command to see all properties and defaults.

Figure

- `Colormap, Position, PaperPositionMode`

Axes (contain the lines, surfaces, and other objects that represent the data visualized in a graph)

- `XLim, YLim, ZLim, CLim, XGrid, YGrid, ZGrid, XTick, XTickLabel, YTick, YTickLabel, ZTick, ZTickLabel, XScale, YScale, ZScale`

Plot (composite graphics objects of one or more core objects in a group)

- `XData, YData, ZData, Color, LineStyle, LineWidth`

Properties Common to All Objects

- `Parent, Children, BeingDeleted` (on when object's `DeleteFcn` is called), `DeleteFcn` (Callback rountine that excutes when object is deleted), `CreateFcn, Selected, Visible`, etc.

Legend

typical syntax

- `legend('First plotted’, 'Second plotted’, 'Location’, 'Northwest')`

fine-grained control

- `legend(han, 'han(1)label’, 'han(2)label’, 'Location’, 'Northwest')`

legend handle

- get the handle by `leg = legend()`
- use handle to control size/location (more control than `'Location'`), font size/style, line style, etc.

Callback Routines

Function associated with graphics handle that gets called in response to a specific action applied to the associated graphics object

- Object creation, deletion
- Mouse motion, mouse press, mouse release, scroll wheel
- Key press, key release

All callback routines are automatically passed two inputs.

- Handle of component whose callback is being executed
- Event data

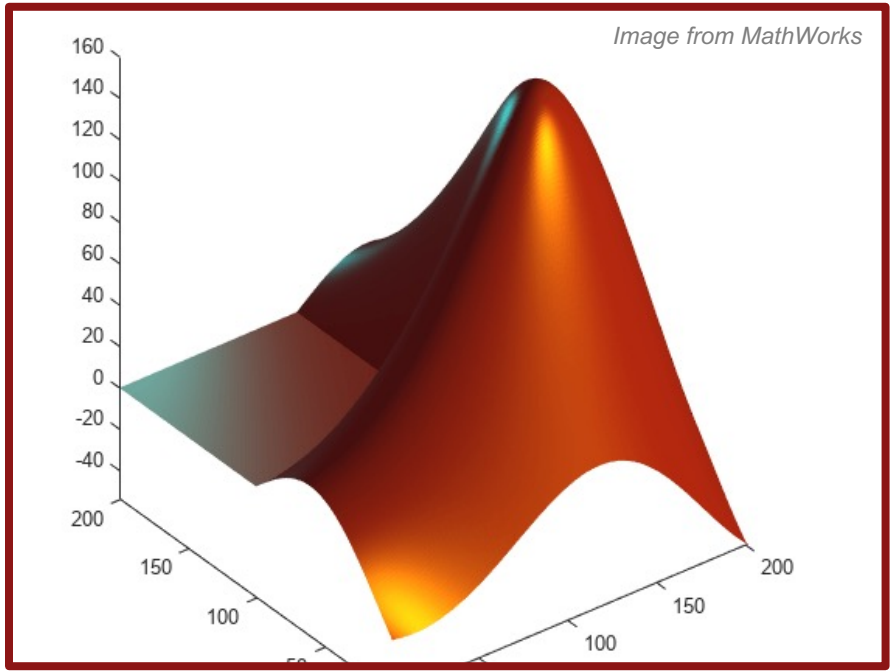
Callback routines can be specified in many possible forms.

- String: expression evaluated in base workspace
- Function handle
- Cell arrays to pass additional arguments to callback routine

Don’t forget to reset root!

Advanced Plotting

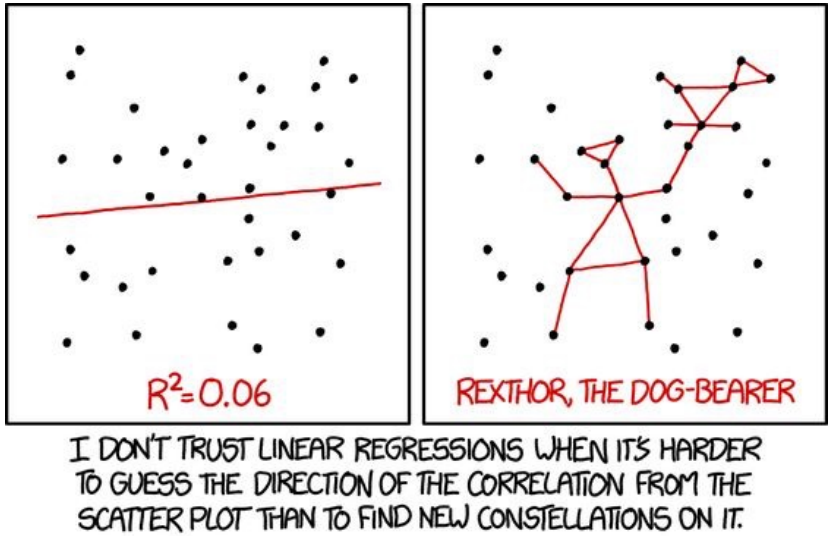
Graphics and Data Visualization



Visualization is everywhere in scientific computing and data analysis,

…

and it is important!



from Internet

Line Plots	Scatter and Bubble Charts	Data Distribution Plots	Discrete Data Plots	Geographic Plots	Polar Plots	Contour Plots	Vector Fields	Surface and Mesh Plots	Volume Visualization	Animation	Images
plot	scatter	histogram	bar	geoplot	polarplot	contour	quiver	surf	streamline	animatedline	image
plot3	scatter3	histogram2	barh	geoscatter	polarhistogram	contourf	quiver3	surfc	streamslice	comet	imagesc
stairs	bubblechart	pie	bar3	geobubble	polarscatter	contour3	feather	surfl	streamparticles	comet3	
errorbar	bubblechart3	pie3	bar3h		polarbubblechart	contourslice		ribbon	streamribbon		
area	swarmchart	scatterhistogram	pareto		compass	fcontour		pcolor	streamtube		
stackedplot	swarmchart3	swarmchart	stem		ezpolar			fsurf	coneplot		
loglog	spy	swarmchart3	stem3					fimplicit3	slice		
semilogx		wordcloud	stairs					mesh			
semilogy		bubblecloud						meshc			
fplot		heatmap						meshz			
fplot3		parallelplot						waterfall			
fimplicit		plotmatrix						fmesh			

University

Line Plots

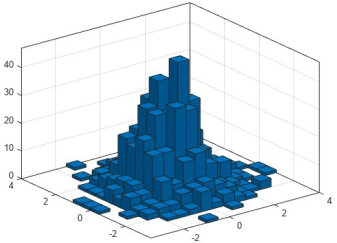
A useful way to compare sets of data or track changes over time.

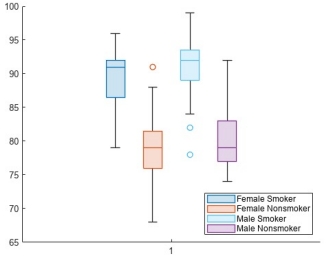
Line Plots	`plot`	2-D line plot
	`plot3`	3-D point or line plot
	`stairs`	Stairstep graph
	`errorbar`	Line plot with error bars
	`area`	Filled area 2-D plot
	`stackedplot`	Stacked plot of several variables with common x-axis
Log Plots	`loglog`	Log-log scale plot
	`semilogx`	Semilog plot (x-axis has log scale)
	`semilogy`	Semilog plot (y-axis has log scale)
Function Plots	`fplot`	Plot expression or function
	`fimplicit`	Plot implicit function
	`fplot3`	3-D parametric curve plotter

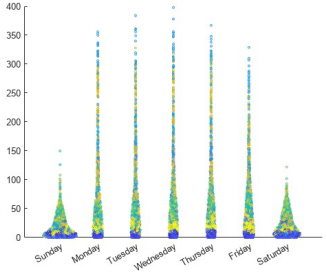
Data Distribution Plots

E.g., histograms, pie charts, word clouds.

Distribution Charts	`histogram`	Histogram plot
	`histogram2`	Bivariate histogram plot
	`morebins`	Increase number of histogram bins
	`fewerbins`	Decrease number of histogram bins
	`histcounts`	Histogram bin counts
	`histcounts2`	Bivariate histogram bin counts
	`boxchart`	Box chart (box plot)
	`swarmchart`	Swarm scatter chart
	`swarmchart3`	3-D swarm scatter chart

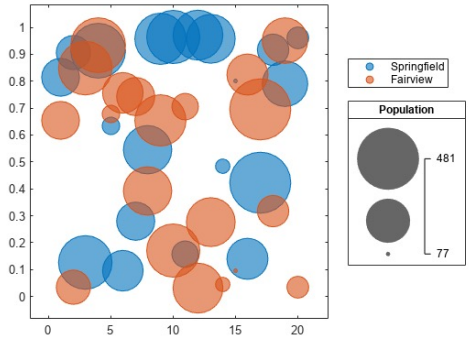


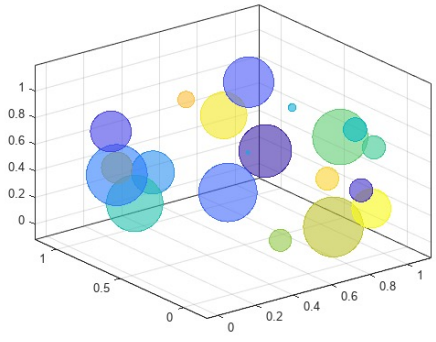




Bubble Charts

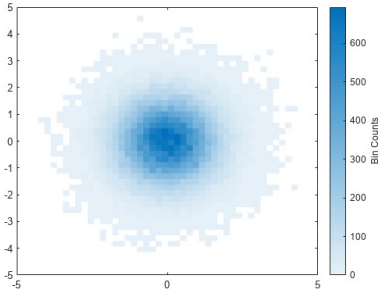
`bubblechart`	Bubble chart
`bubblechart3`	3-D bubble chart
`bubblelim`	Map bubble sizes to data range
`bubblesize`	Set minimum and maximum bubble sizes in points
`bubblelegend`	Create legend for bubble chart

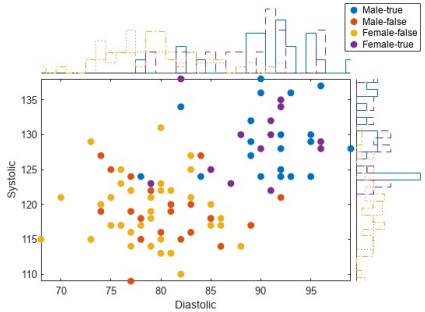


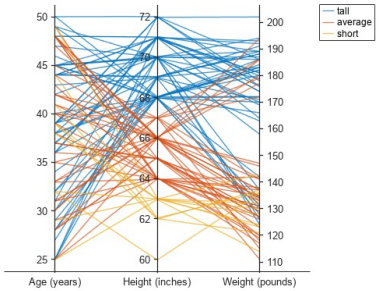


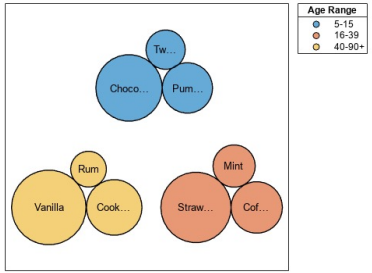
Scatter Plots

`scatter`	Scatter plot
`scatter3`	3-D scatter plot
`binscatter`	Binned scatter plot
`scatterhistogram`	Create scatter plot with histograms
`spy`	Visualize sparsity pattern of matrix
`plotmatrix`	Scatter plot matrix
`parallelplot`	Create parallel coordinates plot

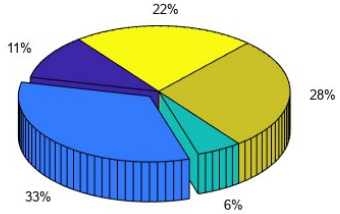


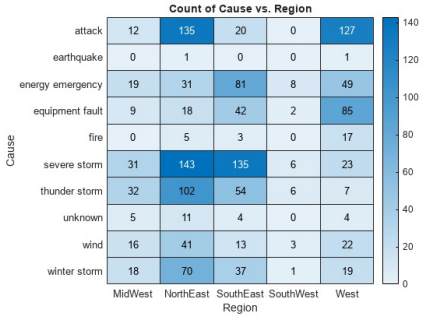






`bubblecloud`	Create bubble cloud chart
`wordcloud`	Create word cloud chart from text data
`pie`	Pie chart
`pie3`	3-D pie chart
`heatmap`	Create heatmap chart
`sortx`	Sort elements in heatmap row
`sorty`	Sort elements in heatmap column

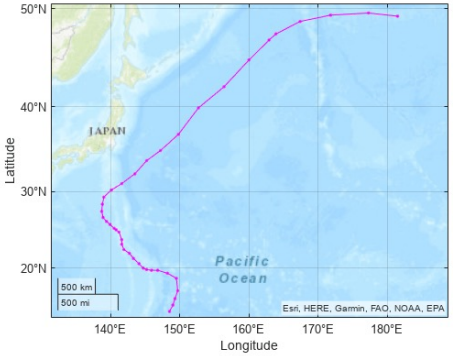


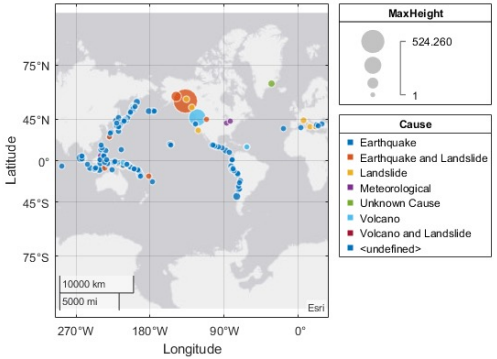


Geographic Plots

Visualize latitude and longitude data over interactive maps.

`geoplot`	Plot line in geographic coordinates
`geoscatter`	Scatter chart in geographic coordinates
`geobubble`	Visualize data values at specific geographic locations
`geodensityplot`	Geographic density plot

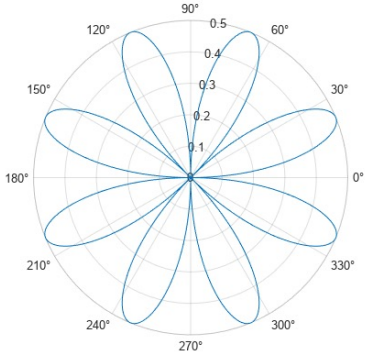


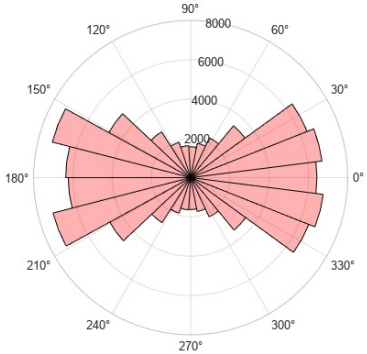


Polar Plots

Plots in polar coordinates

`polarplot`	Plot line in polar coordinates
`polarscatter`	Scatter chart in polar coordinates
`polarbubblechart`	Polar bubble chart
`polarhistogram`	Histogram chart in polar coordinates
`compass`	Arrows emanating from origin
`ezpolar`	Easy-to-use polar coordinate plotter

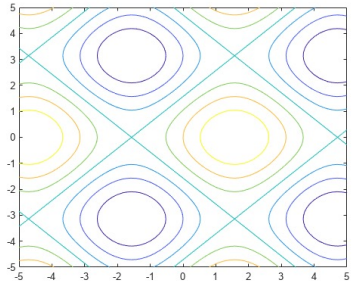




Contour, Surface, and Mesh Plots

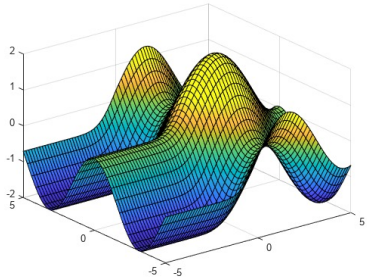
Contour Plots

`contour`	Contour plot of matrix
`contourf`	Filled 2-D contour plot
`contourc`	Low-level contour matrix computation
`contour3`	3-D contour plot
`contourslice`	Draw contours in volume slice planes
`clabel`	Label contour plot elevation
`fcontour`	Plot contours



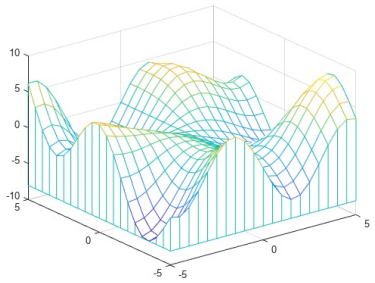
Surface Plots

`surf`	Surface plot
`surfc`	Contour plot under surface plot
`surface`	Primitive surface plot
`surfl`	Surface plot with colormap-based lighting
`surfnorm`	Surface normals
`hidden`	Remove hidden lines from mesh plot
`fsurf`	Plot 3-D surface

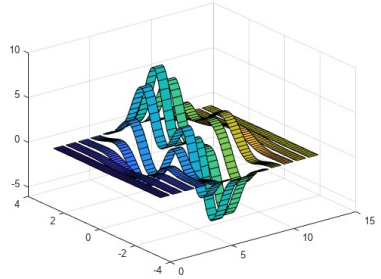


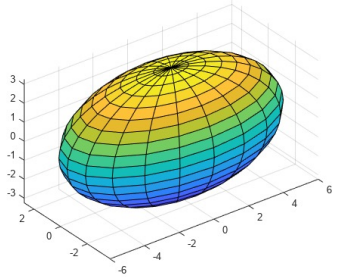
Mesh Plots

`mesh`	Mesh surface plot
`meshc`	Contour plot under mesh surface plot
`meshz`	Mesh surface plot with curtain
`fmesh`	Plot 3-D mesh
`fimplicit3`	Plot 3-D implicit function



`waterfall`	Waterfall plot
`ribbon`	Ribbon plot
`peaks`	Peaks function
`cylinder`	Create cylinder
`ellipsoid`	Create ellipsoid
`sphere`	Create sphere
`pcolor`	Pseudocolor plot
`surf2patch`	Convert surface data to patch data

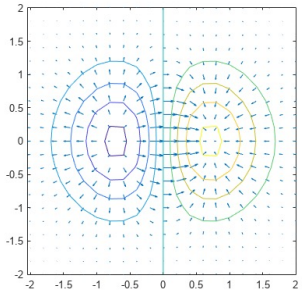


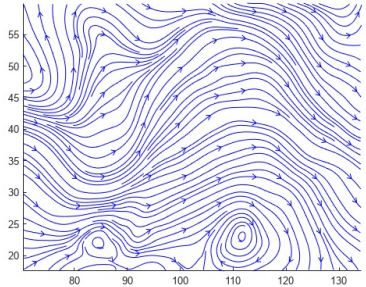


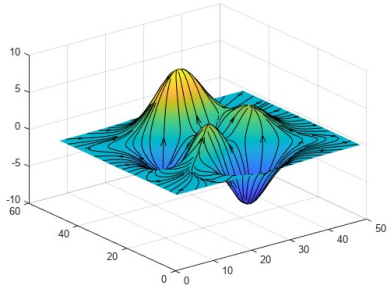
Vector Fields

Vector fields are useful for modeling velocity, magnetic force, fluid motion, and gradients.

`quiver`	Quiver or vector plot
`quiver3`	3-D quiver or vector plot
`compass`	Arrows emanating from origin
`feather`	Arrows from x-axis
`streamline`	Plot streamlines from 2-D or 3-D vector data
`streamslice`	Plot streamlines in slice planes







Quiz

Start with the following piece of code.

```
s = linspace(0,2*pi,20);
[X,Y] = meshgrid(s,s);
F = sin(X.*Y);
v = -1:0.25:1; % levels
```

- Plot the filled contour plot using X, Y, F, v. Use a colormap of your choice, e.g., spring or autumn.
- Compute the gradient and plot the vector field on the same figure.
- Add streamlines starting at x=1 to 5 and y=0. Set line color to white, with line width = 3.

Solution

```
s = linspace(0,2*pi,20);
[X,Y] = meshgrid(s,s);
F = sin(X.*Y);
v = -1:0.25:1;

figure; colormap(spring)
[~,hc] = contourf(X,Y,F,v); colorbar; hold on
[Fx,Fy] = gradient(F);
qh = quiver(X(1:5:end,1:5:end),Y(1:5:end,1:5:end),...
    Fx(1:5:end,1:5:end),Fy(1:5:end,1:5:end),...
    'Color’,Blue', 'LineWidth’,1);
startx = 1:5;
sh = streamline(X,Y,Fx,Fy,startx,zeros(size(startx)));
set(sh,'Color','white', 'LineWidth',3);
```

Volume Visualization of Data

Visualize data defined on three-dimensional grids.

Scalar Data

`contourslice`	Draw contours in volume slice planes
`flow`	Simple function of three variables
`isocaps`	Compute isosurface end-cap geometry
`isocolors`	Calculate isosurface and patch colors
`isonormals`	Compute normals of isosurface vertices
`isosurface`	Extract isosurface data from volume data
`reducepatch`	Reduce number of patch faces
`reducevolume`	Reduce number of elements in volume data set
`shrinkfaces`	Reduce size of patch faces
`slice`	Volume slice planes
`smooth3`	Smooth 3-D data
`subvolume`	Extract subset of volume data set
`volumebounds`	Coordinate and color limits for volume data

Volume Data

`coneplot`	Plot velocity vectors as cones in 3-D vector field
`curl`	Curl and angular velocity of vector field
`divergence`	Compute divergence of vector field
`interpstreamspeed`	Interpolate stream-line vertices from flow speed
`stream2`	Compute 2-D streamline data
`stream3`	Compute 3-D streamline data
`streamline`	Plot streamlines from 2-D or 3-D vector data
`streamparticles`	Plot stream particles
`streamribbon`	3-D stream ribbon plot from vector volume data
`streamslice`	Plot streamlines in slice planes
`streamtube`	Create 3-D stream tube plot

Generate Plots for Publications

Graphics and Data Visualization

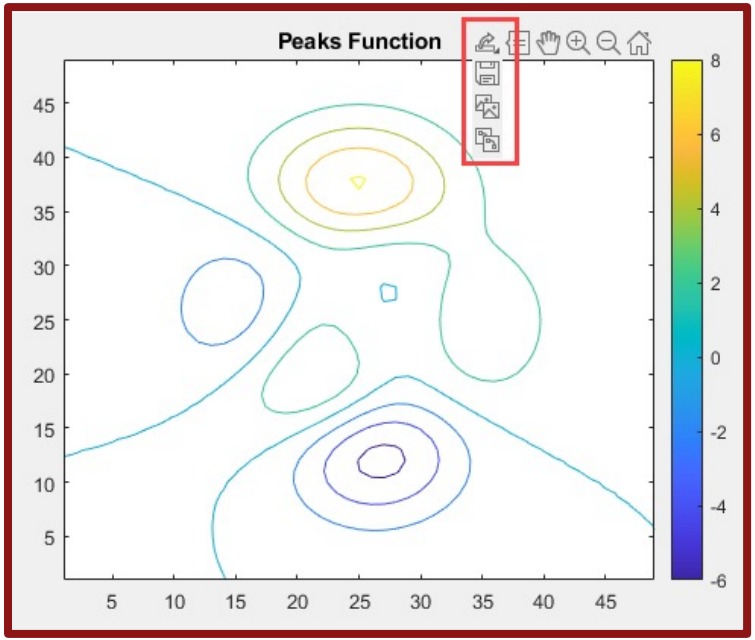


Image from MathWorks

- Use graphics handles to deal with aesthetics
- Generate data in MATLAB and plot in document
- Save to a proper format
  - e.g., PNG, SVG, PDF, TikZ/PGF (popular choices for LATEX)
- `Matlab2tikz` (a popular script available via File Exchange)
  - Convert MATLAB figures into native TikZ/Pgfplots figures
- `exportgraphics(ax,filename)` (R2020a or later)
  - The graphics object can be any type of axes, a figure, a standalone visualization, a tiled chart layout, or a container within the figure. The resulting graphic is tightly cropped to a thin margin surrounding the content.
  - E.g., `exportgraphics(gca,"myplot.jpg","Resolution",300)`

Generate plot with all lines/labels/annotations/legends/etc

Set properties (graphics handles or interactively)

- Figure width/height
- Axes line width, object line width, marker size
- Font sizes
- Adjust white space if necessary

Save figure to file

- WYSIWYG: 'PaperPositionMode' is 'auto' by default.
  You may also set it by `f.PaperPositionMode = 'auto';`
- Print to file for inclusion in document:

```
print(gcf,'-depsc2',filename);
saveas(gcf, filename);
matlab2tikz(filename);
```

Publish MATLAB Code Files (.m)

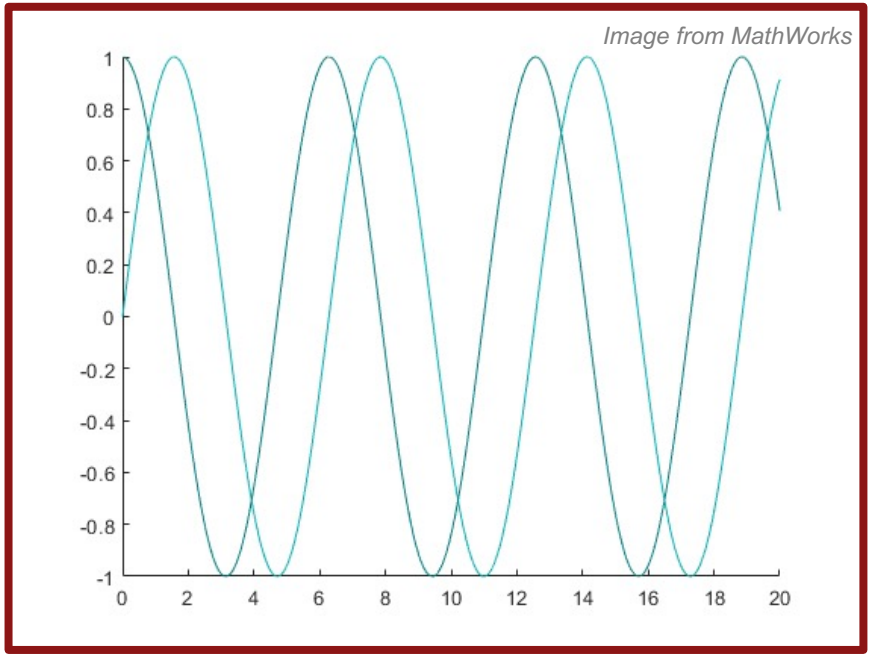
Publishing a MATLAB Code file (.m) creates a formatted document that includes the code, comments, and output.

To publish your code:

1. Create a MATLAB script or function. Divide the code into steps or sections by inserting two percent signs (%%) at the beginning of each section.
2. Document the code by adding explanatory comments at the beginning of the file and within each section.
3. Publish the code. On the **Publish** tab, click **Publish**, or
   Use `publish(file,format)`

Animation

Graphics and Data Visualization



3 basic techniques for creating animations in MATLAB

1. Update the properties of a graphics object and display the updates on the screen.
   - Useful for creating animations when most of the graph remains the same.
2. Apply transforms to objects.
   - Useful when operating on the position and orientation of a group of objects together.
   - Group the objects as children under a transform object. Create the transform object using `hgtransform`. Setting the Matrix property of the transform object adjusts the position of all its children.
3. Create a movie.
   - Useful for complex animation that does not draw quickly in real time, and for replay.
   - Use the `getframe` and `movie` functions to create a movie.

How to Create an Animation

Before entering loop

- Create figure and axes
- Modify object using handles to achieve desired appearance
- Use command `set(gca,'nextplot','replacechildren')` to ensure only children of axes object will be replaced upon next plot command (will not modify axes properties)

During loop

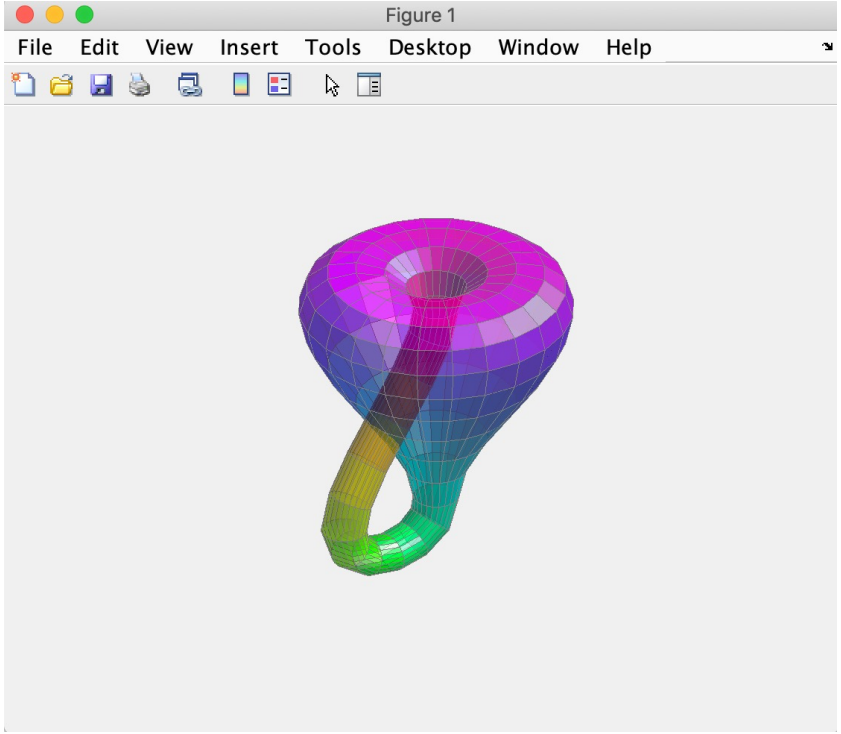
- Plot command to generate data on plot
- Modify object using handle to achieve desired appearance
- Use command `drawnow` to draw object, otherwise it will not be drawn until execution is complete (MATLAB optimization, as plotting is expensive)

Alternatively, modify XData, YData, ZData properties of initial plot object

Additionally, save sequence of plotting command as frames (`getframe`) and play back from MATLAB window (`movie`)

Fun with MATLAB

Type `xpklein` in Command Window.



Next Lecture

File Manipulation and System Interaction

- Search path
- Data import/export & file commands
- Operating system

Handling Big Data

- Efficient use of memory
- Datastore and tall arrays
- MapReduce

Integration with Other Languages