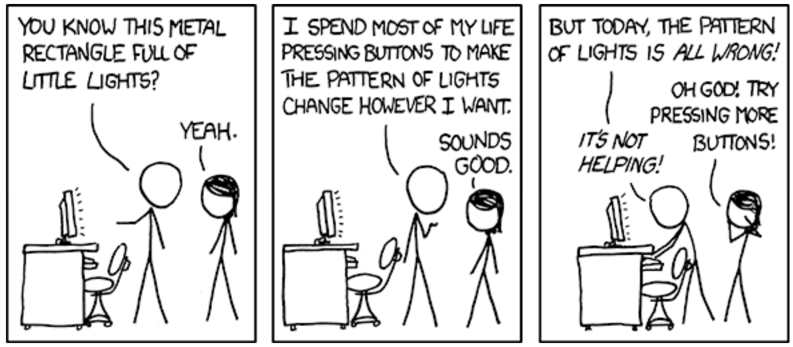


xkcd
722

Data Visualization (Plotting)

ENGR 151, Lecture 23: 1 Dec 14

Announcements

- Project 8 due Wed 10 Dec 11pm
- Final exam: Wed 17 Dec 4pm

MATLAB Type Conversions

- Sometimes automatic (coercion):
  - `'a' + 3` → 100
  - `'a' + 'b'` → 195
  - `90 + true` → 91
  - `~[40 0 -3]` → [0 1 0] (logicals)
- Can also use explicit conversion (cast) functions:
  - `logical([40 0])` → [1 0] (logicals)
  - `char(98)` → b
  - `double('b')` → 98
  - `uint8([38 -5 2.8 666])` → [38 0 3 255]

What is v After these Assignments?

```
v = 97:102; v(3) = 'x';
```

A. `[97 98 'x' 100 101 102]`
B. `[97 98 120 100 101 102]`
C. `'abxdef'`
D. No value (error)
E. None of the above

Array Element Types

- In an array, all elements must be same type
- Element type determined on array initialization
  - `v = 1:5` a vector of numbers
  - `w = 'abcde'` a vector of characters
- Assigning different types to elements will cause a conversion
  - `v(3) = 'x'` → v = [1 2 120 4 5]
  - `w(3) = 120` → w = 'abxde'
- To assign an element to chosen type, initialize that way or perform explicit conversion

Which Produces a Different Vector v ?

A. `v(1:7) = 'x'; v = v + 0;`
B. `v(1) = 'x'; v(2:7) = 'x';`
C. `v = ones(1,7); v = char(v * 120);`
D. `v = char('x' * ones(1,7));`
E. None of the above (all are the same)

- What about:

```
v = [120 120 120 120 'xxx'];
```

Plotting Data in MATLAB

- `plot`: general plotting function
  - Many forms, depending on number and content of arguments
  - Generates a new figure (e.g., for a graph) in the Figure window
    - Opens new window if necessary
- Basic form: `plot(x, y)`
  - x is a vector of x values
  - y is a vector of y values

Plot draws corresponding (x,y) points on graph, connecting adjacent points in vectors by lines

```
x = 1:0.1:10;
y = x .^ 2 - 10 * x + 15;
plot(x, y);
```

Adding Information to a Figure

- `title(title_string)`
  - Adds specified title to figure (or changes existing title)
- `xlabel(label_string)` `ylabel(label_string)`
  - Adds specified axis label to figure (or changes existing label)
- `grid` : sets (or toggles) presence of grid lines

```
title('A Parabola');
xlabel('x value');
ylabel('y value');
grid on;
```

Printing, Saving, etc.

- Print or save your figure by using menu commands
  - choice of many formats
- `figure` : opens a new figure window

Multiple Plots on Same Figure Graph

- Provide additional pairs of x/y vectors as `plot` arguments

```
x = 0:0.1:10;
y1 = x .^ 2 - 10 * x + 15;
y2 = - x .^ 2 + 10 * x;
plot(x, y1, x, y2);
```

Line Specifiers

- Use a line specifier to distinguish among multiple plots, or otherwise achieve preferred style
- Basic form: `plot(x, y, line_spec)`
  - line_spec a string with codes for various options

color
- `y` = yellow
- `m` = magenta
- `c` = cyan
- `r` = red
- `g` = green
- `b` = blue
- `w` = white
- `k` = black

marker style
- `.` = point
- `o` = circle
- `x` = x-mark
- `+` = plus
- `s` = square
- `d` = diamond
- `*` = star
- `p` = pentagram
- `h` = hexagram
- `v` = triangle down
- `^` = triangle up
- `>` = triangle right
- `<` = triangle left

line style
- `-` = solid
- `:` = dotted
- `-.` = dash-dot
- `--` = dashed

Can specify any combination of these, in any order

Multi-Plot Example

- First plot: red and dot-dashed
- Second plot: blue and marked with circles
- `legend`: creates a legend associating text with respective plot styles

```
x = 0:0.1:10;
y1 = x.^2 – 10 * x + 15;
y2 = - x.^2 + 10 * x;
plot(x, y1, 'r.', x, y2, 'bo');
legend('parabola up','parabola down');
```

Multiple Plots: More Methods

- `hold` command
  - `hold on` : changes mode so that subsequent plots use current figure
  - `hold off` : releases hold mode
  - `hold` : toggles
- `line` function: similar to `plot`, but uses current figure

Example: Plot a Circle

```
radius = input('Enter radius: ');
x = linspace(-radius,radius,100);
y1 = sqrt( radius^2 - x.^2 );
plot(x,y1,'b',x,-y1,'r');
```

Which Produces a Vector Equivalent to `low:incr:high` ?

A. `linspace(low,high,incr)`
B. `linspace(low,high,(high-low)/incr)`
C. `linspace(low,high,1+(high-low)/incr)`
D. `linspace(low,high,(high-low+1)/incr)`
E. None of the above

Assume:
- high > low
- incr > 0

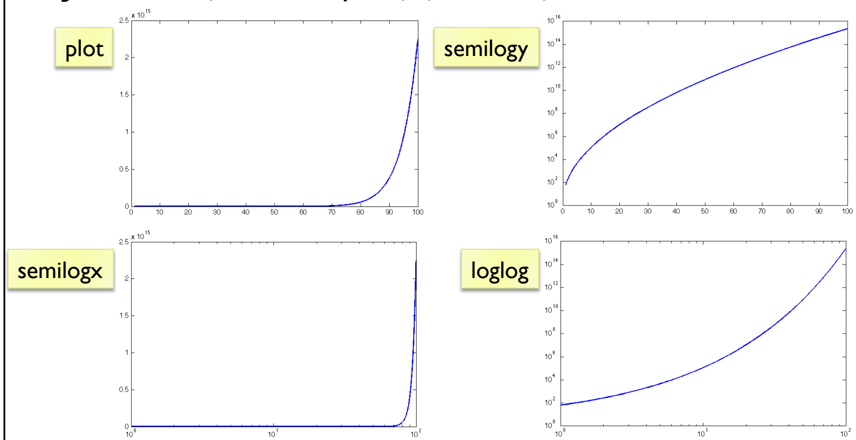
Nonlinear Scales

- In addition to linear scales, data can be plotted on logarithmic and semi-logarithmic scales.

`plot`	both x and y linear
`semilogx`	x logarithmic, y linear
`semilogy`	x linear, y logarithmic
`loglog`	both x and y logarithmic

Nonlinear Scales (Example)

```
x = 1:0.1:100;
y = 2.^( 5 * sqrt(x) + 1 )
```



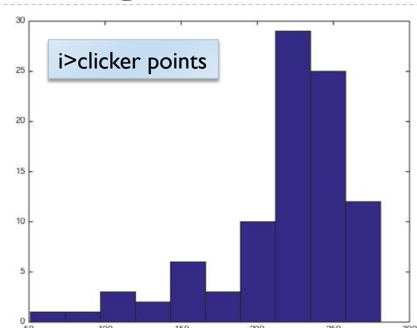
Other Two-Dimensional Plots

`errorbar(x, y, e)`	plot with error bars
`polar(theta, r)`	graph in polar coordinates
`bar(x, y)`	vertical bar chart
`barh(x, y)`	horizontal bar chart (x = labels, y = values)
`pie(x)`	pie chart

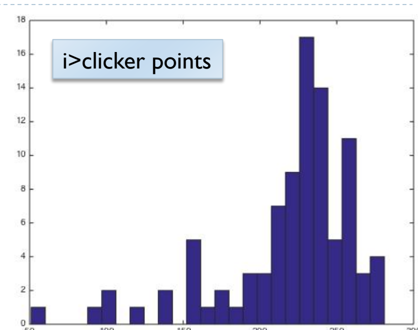
i>clicker Leader Board

leeperry	281	tiberiu	257
nrfenner	279	eriwang	257
ianor	274	colinwa	256
mjschoen	273	cschmotz	255
kishbrao	266	rumacsam	254
lawtondy	264	mtmyers	251
nitinram	264	karnson	249
eagattas	262	steica	248
aarondc	262	tongyliu	247
cwfenner	262	tanvirs	245
tsaoa	262	jmpat	243
amorgese	261	sthmprrn	243
snavraj	258	yergicol	243

Histograms



hist(iclick)



hist(iclick,25)

```
n = hist(iclick)
n =
     1     1     3     2     6     3    10    29    25    12
```

Time-Dependent Data

- Suppose the location of an object at time t is described by the equations:

$$x(t) = e^{-0.2t}\cos(2t)$$

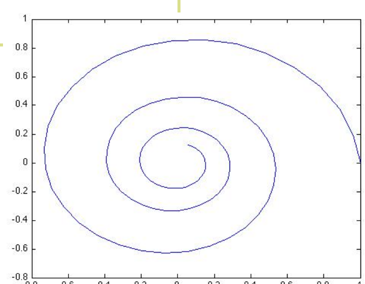
$$y(t) = e^{-0.2t}\sin(2t)$$

- We know how to plot this in two dimensions

```
t = 0:0.1:10;
x = exp(-0.2*t) .* cos(2*t);
y = exp(-0.2*t) .* sin(2*t);
plot(x,y);
```

Plot of (x,y) over Time

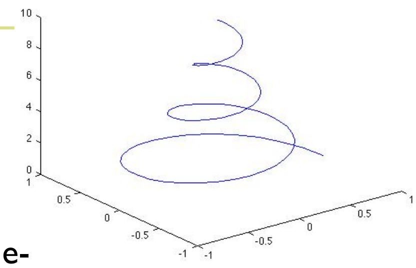
```
t = 0:0.1:10;
x = exp(-0.2*t) .* cos(2*t);
y = exp(-0.2*t) .* sin(2*t);
plot(x,y);
```



- Useful, but does not really show us where object is at particular times

A 3-Dimensional Plot

```
t = 0:0.1:10;
x = exp(-0.2*t) .* cos(2*t);
y = exp(-0.2*t) .* sin(2*t);
plot3(x,y,t);
```



- Now we can really see the time-space trajectory of the object

Visualizing Three-Dimensional Surfaces

- What does the following function look like?

$$z = \left(x^2 - y^2\right) e^{-\left(x^2+y^2\right)}$$

Cannot use line plot, because there is a z for every (x,y) pair

- Set up a mesh of (x,y) coordinates:

```
[x,y]= meshgrid(xstart:xinc:xend, ystart:yinc:yend)
```

- Once the grid is set up, can compute z values:

```
z = (x.^2 - y.^2) .* exp(-(x.^2 + y.^2));
```

Meshgrid Revisited

```
[x,y] = meshgrid(xvec,yvec)
```

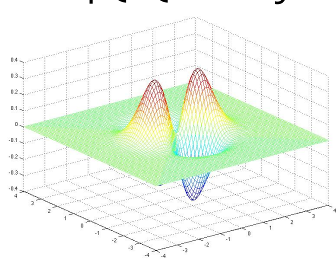
- Let xlen be length of `xvec`, and ylen length of `yvec`.
- What are dimensions of x ?

A. xlen × ylen
B. ylen × xlen
C. xlen × xlen
D. min(xlen,ylen) × min(xlen,ylen)
E. None of the above

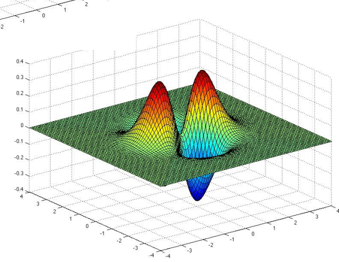
Mesh and Surface Plots

```
[x, y] = meshgrid(-4:0.1:4);
z = (x.^2-y.^2).*exp(-(x.^2+y.^2));

mesh(x, y, z);
```



```
surf(x, y, z);
```



More 3-D Plots

- Pseudocolor (just show z values as colors)
```
pcolor(x, y, z);
```
- Contour plots
```
contour(x, y, z);
contourf(x, y, z);      % filled contour
```
- Mesh and surface
```
mesh(x, y, z);
meshc(x, y, z);         % with contour
surf(x, y, z);
surfc(x, y, z);         % with contour
surfl(x, y, z);         % with lighting
```

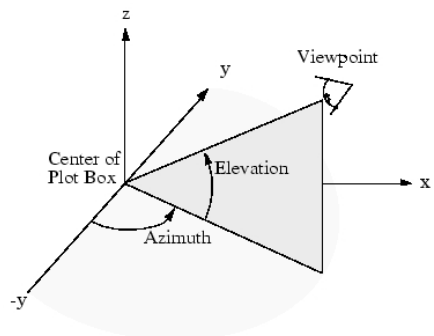
Options with 3D Plots

- In shaded plots (e.g., pcolor, surf) you may choose different shading options:
```
shading faceted
shading flat
shading interp          % can be slow
```
- You can also choose different color maps
```
colormap pink
colormap copper
```

See the zyBook or on-line help for more options

Figure Window Tools

- The tools in the figure window can also be used to manipulate your image.
- For example:
  - Zoom in/out
  - Rotate, Pan
  - Change camera viewpoint
    - Azimuth
    - Elevation



Making Movies in MATLAB

- Use the getframe and movie commands to create and play a movie.
- Make a motion picture, *featuring*:

$$\cos^2(t/2)\left\{\left(x^2-y^2\right)\cos(t)+2xy\sin(t)\right\}\exp\left[-\left(x^2+y^2\right)\right]$$

```
[x, y] = meshgrid(-4:0.1:4);
f1 = exp(-(x.^2 + y.^2));
f2 = x.^2 -y.^2;
f3 = 2*x.*y;
...
```

Scripting the MATLAB Movie (cont.)

```
[x, y] = meshgrid(-4:0.1:4);
f1 = exp(-(x.^2 + y.^2));
f2 = x.^2 -y.^2;
f3 = 2*x.*y;

for t = 0:pi/20:4*pi
  z = (cos(0.5*t).^2) .* ...
          (f2.*cos(t)+f3.*sin(t)).*f1;
end
```

Rotate twice

$$\cos^2(t/2)\left\{\left(x^2-y^2\right)\cos(t)+2xy\sin(t)\right\}\exp\left[-\left(x^2+y^2\right)\right]$$

Scripting the MATLAB Movie (cont.)

```
[x, y] = meshgrid(-4:0.1:4);
f1 = exp(-(x.^2 + y.^2));
f2 = x.^2 -y.^2;
f3 = 2*x.*y;
j = 1;
colormap copper
for t = 0:pi/20:4*pi
  z = (cos(0.5*t).^2) .* ...
         (f2.*cos(t)+f3.*sin(t)).*f1;
  surfl(x,y,z);
  axis([-4 4 -4 4 -0.5 0.5]); % axes always same
  shading interp;
  mov(j) = getframe;
  j = j+1;
end
movie(mov); % play the movie
```