Day 2: More Stuff
Scientific Functions

Trig: sin, cos, tan, asin, acos, atan
      sinh, cosh, tanh, asinh, acosh, ...

Rounding: floor, ceil, round, fix

Modular: rem, mod

Exponential: exp, log, log2, log10, sqrt

Primes: factor, primes

Polynomials: roots, polyfit, polyval
Matrix Functions

Determinant: \text{det}

Inverse: \text{inv, pinv}

Eigenvalues: \text{eig, svd}

Fourrier: \text{fft}

And many, many more...
Inf and NaN

3/0 returns Inf

0/0 returns NaN

3+Inf

Inf/Inf

-Inf, -NaN
Complex Numbers

\[ \sqrt{-16} \]

3.5i

2 - 3.5i

\((2+3i) \times (4+5i)\)
Predicates

isreal(3)
isprime(1 : 13)
isnumeric([2 3 5])
isempty([ ])
isinf(Inf)
isnan(NaN)
islogical(1 == 1)
ischar('a')
isequal('foo', 'aardvark')

What percentage of the first 1000 integers is prime?
mean(isprime(1:1000))
Return Values

Functions can return multiple values:

\[
A = \text{rand}(5, 3);
\]

\[
s = \text{size}(A)
\]

\[
[\text{rows}, \text{cols}] = \text{size}(A)
\]
Optional Return Values

Functions can choose whether to return values, depending on if the user is asking for values.

```matlab
plot([1 2 3], [3 1 2])
```
no return value

```matlab
h = plot([1 2 3], [3 1 2])
set(h, 'LineStyle', '--')
set(h, 'LineWidth', 8)
```
single return value
Variable Number of Arguments

Some functions accept a variable number of arguments:

peaks

peaks(10)
Variable In and Out

hist(randn(2000,1))

hist(randn(2000,1), 50)

counts = hist(randn(2000,1), 5)

[counts, centers] = hist(randn(2000,1), 5)
nargin and nargout

Inside a function, **nargin** is the number of input arguments supplied with the call.

**nargout** is the number of output arguments requested with the call.
Testing nargin/nargout

function [x,y,z] = nargtest(p,q,r,s,t)
    if nargin >= 1
        x = 50;
        if nargin >= 2
            y = 'foo';
            if nargin >= 3
                z = 3:7;
            end
        end
    end
end
end
whos % show the local workspace
end

Try:
a = nargtest(5,6,7)
[a, b] = nargtest(3)
[a, ~, c] = nargtest(9,8)
Name Spaces

- **Base workspace**: variables created outside of any function exist in the base workspace.

- **Local workspaces**: each function executes in a separate local workspace holding the arguments, return variables, and any local variables created by the function.

Functions cannot access variables of the base workspace.
Name Spaces (cont.)

- **Global workspace**: variables declared global by a function are accessed in the global workspace.

  It's a good idea to also declare the variable global in the base workspace.
Global Variables

global pts
pts = 0 : pi/20 : 2*pi ;

function h = circ(x,y)
    % draws a circle centered on (x,y)
    global pts
    hh = plot(x+cos(pts), y+sin(pts));
    if nargout > 0
        h = hh;    % return h only if requested
    end
end
Scripts Called By Functions

- Scripts do not have their own workspaces.

- A script called from the keyboard executes in the base workspace.

- A script called from within a function executes in the function's local workspace.
Resetting Variables

clear x  removes variable x and
         undoes any global declaration

You can also click on a variable in the workspace
pane and hit the Delete key, or right-click on the
variable and choose from the menu.

clear all  clears everything

clear global  clears global declarations

whos global  shows all global variables
Handle Graphics

Root = 0

Figure = 1, 2, ...

Axes

Line  Text  Image  Surface
Taking Apart A Figure

clf, plot(rand(5, 3))

ax = get(gcf, 'Children')
get(ax)

lines = get(gca, 'Children')
get(lines(1))
Multiple Axes: Subplot

clf

subplot(2,2,1),  plot(rand(5, 5))
subplot(2,2,2),  bar3(rand(5, 3))
subplot(2,2,3),  a=rand(15, 1);  pie(a, a > 0.7)
subplot(2,2,4),  polar(cos(0:150))

set(gca, 'Position', [0.32 0.1 0.4 0.4])
Exploring Graphics Objects

set(gca,'Units')
set(gca)
propedit(gca) \textit{click on “More Properties”}

Matlab online documentation:
Help pulldown menu or ‘?’ icon:
\textgreater Documentation
\textgreater MATLAB
\textgreater Graphics
\textgreater Graphics Objects
3D Graphics

peaks

rotate3d on

or click on the rotation arrow in the toolbar

set(gca, 'CameraViewAngleMode', 'manual')

or right-click in the figure,
select Rotate Options, then
select Fixed Aspect Ratio Axes
Plotting Surfaces

[x, y, z] = peaks;

surf(x, y, z, z)

surf(x, y, z, x)

surf(x, y, z, rand(length(x)))
Plotting in 3D

Don't type all this in! Download this file:

```plaintext
www.cs.cmu.edu/~dst/Tutorials/Matlab/helix.m
```

or
```plaintext
cd /afs/andrew/usr/dst/matlab
```

```plaintext
function helix
    pts = 0 : pi/20 : 4*pi;
    x1 = cos(pts);  y1 = sin(pts);
    x2 = cos(pts+pi);  y2 = sin(pts+pi);
    z = pts/(2*pi);
    clf, whitebg(gcf, [0 0 0]), hold on
    plot3(x1, y1, z, 'y')
    plot3(x2, y2, z, 'w')
    axis([-3 3 -3 3 0 2])
    view(95, 9)
end
```
colors = 'rgbm';

for i = 4 : 4 : length(pts)-4
    plot3([x1(i) x2(i)], [y1(i) y2(i)], z([i i]), ...
          colors(ceil(rand(1)*length(colors))), 'LineWidth', 3)
end

axis off
set(gcf, 'Color', 'k')
set(gca, 'CameraViewAngleMode', 'manual')

az = -180 ;

while true
    view(az, 9), pause(0.05)
    az = az + 5 ;
end
clf reset, peaks, colorbar
m = colormap;
whos m
colormap(spring)
brighten(0.5)
colormap(jet)
colormap(parula)
colormap(bone)
colormap(hot)
colormapeditor

Northern parula
2D Data

\[
[x, y] = \text{meshgrid}(-2 : 0.05 : 2) ; \\
z = \sin(x) \cdot \cos(y) ; \\
\text{contour}(z, 20) \\
\text{imagesc}(z) \\
\text{colormap(hot)} \\
\text{imagesc}(x(:), y(:), z) \\
\text{surf}(z), \text{colormap(jet)} \\
\text{surfc}(z)
\]
Surface Objects

sphere

[x,y,z] = sphere(20);
x(1 : 5 : 21*21) = NaN;
surf(x, y, z)
alpha(0.7)

Use the rotate tool to rotate the sphere; set Fixed Aspect Ratio Axes first.

surf(x, y, z, rand(size(x)))
shading interp, grid off, axis off
set(gcf, 'Color', 'w')
Create a file temps.txt:
  Use the “New Script” button.

Enter this data:
  38   50
  42   53
  33   57
  45   56
  44   46
  41   40

*Save the file as temps.txt*

load temps.txt

plot(temps)
Importing Data From Files

- You can import data from Excel (and many other file formats) using the Import Data button.

Select the file you want to import; the wizard will guide you through the rest.

- There are also built-in functions specifically for dealing with Excel files:
  - doc xlsread
  - doc xlswrite
Curve Fitting for Extrapolation

x = randn(1, 2000);
y = sin(x) + 0.2 * randn(1, 2000) ;
clf, hold on, plot(x, y, '.');
c = polyfit(x, y, 3)
Example polynomial representation:
\[ c = [5 \ -1 \ 4 \ 3] \]
\[ 5x^3 - x^2 + 4x + 3 \]
pts = min(x) : range(x)/100 : max(x);
plot(pts, polyval(c, pts), 'r', 'LineWidth', 3)
Saving Variables

clear all
a = 'aardvark'
[x, y, z] = sphere(5);
save stuff.mat
clear all
whos -file stuff.mat
load stuff.mat

save junk.dat x y -ascii
type junk.dat
General Operating System Stuff

pwd
cd
dir
ls *.m
delete stuff.mat
!ps -a
Debugging

Poor man's debugger:
Remove semicolons from assignments.
Add 'quoted strings' in appropriate places.
Add a call to keyboard. (Use return to return from keyboard input mode.)

```matlab
function y = buggy(vec)
    p = vec > 5
    'got this far'
    keyboard
    z = p * vec
    y = sin(z) ;
end
```

Try: buggy([4 6])
Type 'return' to exit keyboard mode and continue.
The Matlab Debugger

dbtype helix
dbstop helix 5
helix
dbstep
dbstep 7
whos

Look at the Stack pulldown menu in the toolbar.
dbstep 30
dbquit
dbclear helix
doc debug
for i = 1 : 10
    fprintf('The square root of %2d is %f \n', ...
        i, sqrt(i))
end

doc fprintf

title(sprintf('f(x) over range %g to %g', ... -3.5, 5.125))