Introduction to MATLAB

Functions

Built-In Functions and User-Defined Functions

Multiple Useful Functions

Contents

- Basics of Built-in Functions
- Help Feature
- Elementary Functions
- Data Analysis
- Random Numbers
- Complex Numbers

What is a Built-In Function?

- A computational expression that uses one or more input values to produce an output value.
- MATLAB functions have 3 components: input, output, and name
- For example, for `b = tan(x)`
  - x is the input,
  - b is the output,
  - tan is the name of a built-in function

MATLAB Functions

- Functions take the form:

  variable = function(number or variable)
- MATLAB has many functions stored in its file system.
- To use one of the built-in functions you need to know its name and what the input values are.
- For example, the square root function: **`sqrt( )`**.
- Find the square root of 9 using MATLAB

```
>> a = sqrt(9)
   a = 3
```

HELP Feature

- You may not always recall the name and syntax of a function you want to use since MATLAB has so many built-in functions.
- MATLAB has an indexed library of its built-in functions that you can access through the help feature.
- If you type help in the command window MATLAB provides a list of help topics.

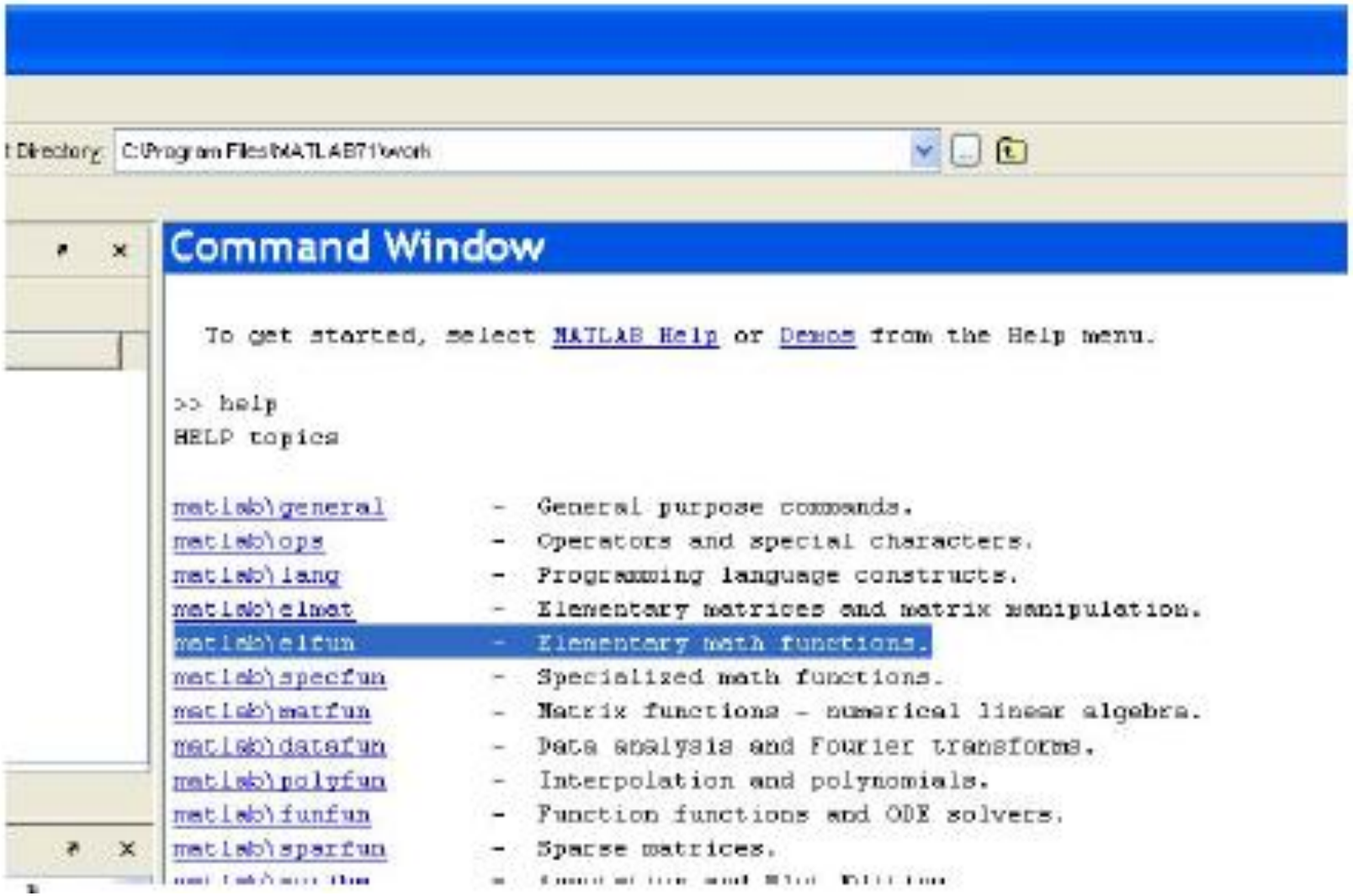
Help

- In MATLAB command window type

```
>> help
```

- If we are interested in the elementary mathematics functions, we find it on the list of help topics (5th down) and click it.
- A list of commands appears with the description of what each one does.
- Try a few!

MATLAB Help


Directory: C:\Program Files\MATLAB71\work
Command Window
To get started, select MATLAB Help or Demos from the Help menu.
>> help
HELP topics
matlab\general - General purpose commands.
matlab\ops - Operators and special characters.
matlab\lang - Programming language constructs.
matlab\elmat - Elementary matrices and matrix manipulation.
matlab\elfun - Elementary math functions.
matlab\specfun - Specialized math functions.
matlab\matfun - Matrix functions - numerical linear algebra.
matlab\datafun - Data analysis and Fourier transforms.
matlab\polyfun - Interpolation and polynomials.
matlab\funfun - Function functions and ODE solvers.
matlab\sparfun - Sparse matrices.

More Help

- For more specific help: **help topic**
- Check it out:

```
>> help tan
```

- MATLAB describes what the function **tan** does and provides helpful links to similar or related functions to assist you.

Hands-On

- Use MATLAB help to find the exponential, natural logarithm, and logarithm base 2 functions
- Calculate e^7

```
>> exp(7)
```

- Calculate $\ln(4)$

```
>> log(4)
```

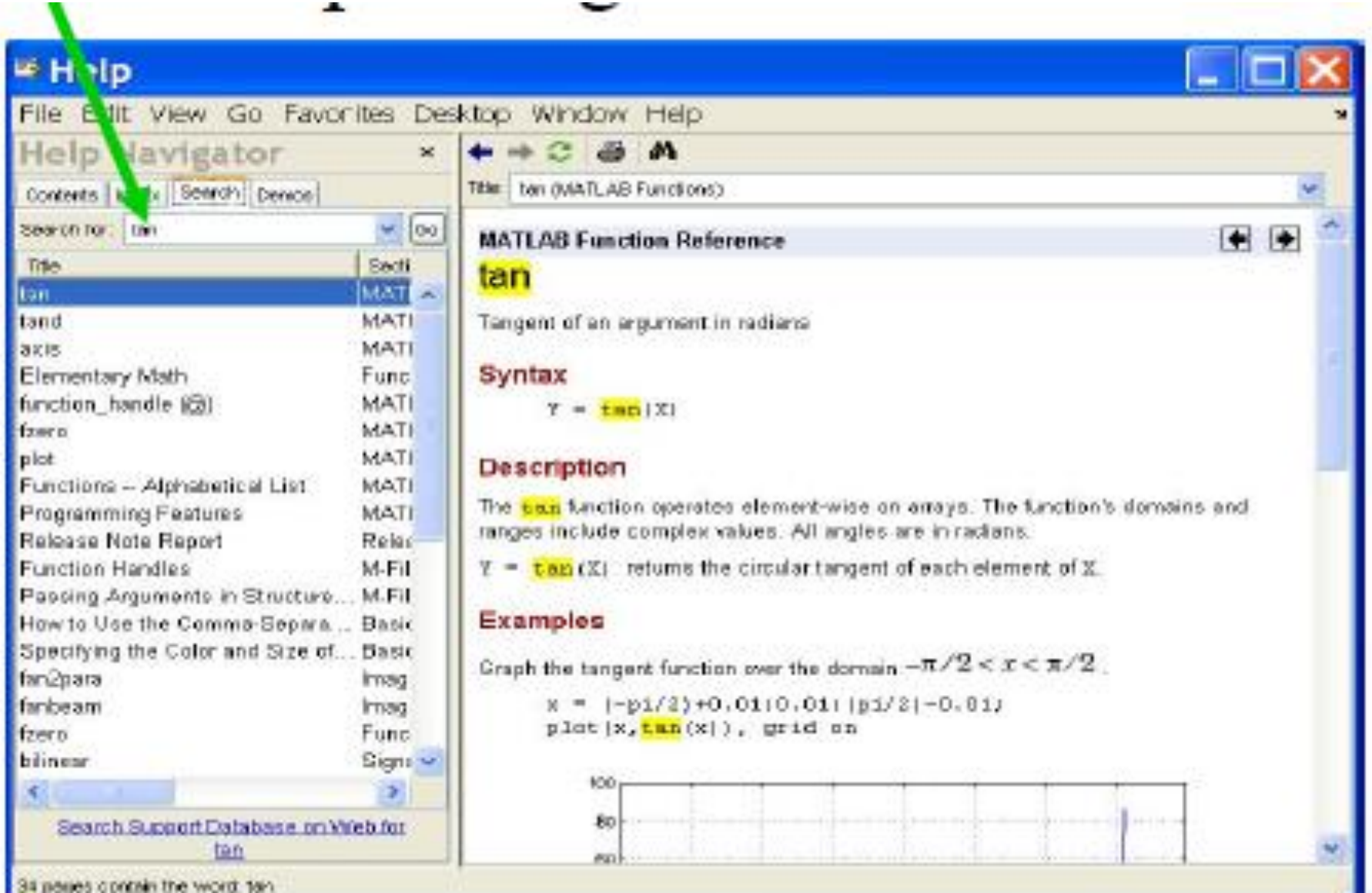
- Calculate $\log_2(12)$

```
>> log2(12)
```

Help Navigator

- Click on Help on the tool bar at the top of MATLAB, and select MATLAB Help.
- A HELP window will pop up.
- Under Help Navigator on the left of screen select the Search tab.
- You can Search for specific help topics by typing your topic in the space after the :.
- You can also press F1 on your keyboard to access the windowed help.

Help Navigator window



Elementary Mathematical Functions

- MATLAB can perform all of the operations that your calculator can and more.
- Search for the topic Elementary math in the Help Navigator just described.
- Try the following in MATLAB to continue your exploration of MATLAB capabilities

```
>> sqrt(625)          ans = 25
>> log(7)             ans = 1.9459
```

Of course, try a few others.

Rounding Functions

- Sometimes it is necessary to round numbers. MATLAB provides several functions to do this.

```
>> round(x)  % round to nearest
             % whole number
>> fix(x)    % round towards zero
>> floor(x)  % round down
>> ceil(x)   % round up
```

Trigonometric Functions

- MATLAB can compute numerous trigonometric functions
- The default units for angles is radians.
  - To convert degrees to radians, the conversion is based on the relationship: 180 degrees = pi × radians
  - Note: `pi` is a built-in constant in MATLAB.
- Inverse functions are **`asin( )`**, **`atan( )`**, **`acos( )`**, etc.
- There are trigonometric functions defined for degrees instead of radian such as **`sind( )`**, **`cosd( )`**, etc.
- Type **`help elfun`** in the command window for more functions and information.

Data Analysis Functions

- It is often necessary to analyze data statistically.
- MATLAB has many statistical functions:

```
max( )          sum( )          size( )
min( )          prod( )         length( )
mean( )         sort( )         std( )
median( )       sortrows( )     var( )
                find( )
```

Data Analysis Practice

```
>> x = [5 3 7 10 4]
```

What is the largest number in array x and where is it located?

```
>> [value,position] = max(x)
Value = 10        % highest value is 10
Position = 4      % it is the 4th value
```

What is the median of the above array?

```
>> median (x)
ans = 5
```

What is the sum of the above array?

```
>> sum(x)
ans = 29
```

Hands-On

```
>> v = [2 24 53 7 84 9]
>> y = [2 4 56; 3 6 88]
```

Sort v in descending order

Find the size of y

Find the standard deviation of v

Find the cumulative product of v

Sort the rows of y based on the 3rd column.

Generation of Random Numbers

- **`rand`** produces random number between 0 and 1
- **`rand(n)`** produces n×n matrix of random numbers between 0 to 1.
- **`rand(n,m)`** produces n×m matrix of random numbers between 0 and 1.

To produce a random number between 0 and 40:

```
>> w = 40*rand
```

To produce a random number between -2 and 4:

```
>> w = -2 + (4-(-2))*rand
```

Note: **`rand`** will NEVER give exactly 0 or exactly 1.

Complex Numbers

- Complex numbers take the form of a+b*i:
  - a is the real part
  - b is the imaginary part
  - and $i = \sqrt{-1}$
- Complex numbers can be created as follows:

```
>> a = 2; b = 3;
>> c = a+b*i          %method 1
>> c = complex(a,b)   %method 2

   c = 2.0000 + 3.0000i
```

- Note: both `i` and `j` are built-in MATLAB constants that equal $\sqrt{-1}$

Complex Numbers Continued

- Function to extract the real and imaginary components of a complex number:

```
real(c)
imag(c)
```

- Function to find the absolute value or modulus of a complex number:

```
abs(c)  % = sqrt(real(c)^2 + imag(c)^2)
```

- Function to find the angle or argument (in radians) of a complex number:

```
angle(c)  % = atan(imag(c)/real(c))
```

Other Useful Functions

- `clock %Outputs 1x6 array containing year, month, day, hour, min, sec.`
- `date %Outputs date as string`
- `tic %Start timer`
- `toc %Output time elapsed`
- `pause(XX) %pause for XX seconds`

Exercises

- Find the modulus (magnitude, abs) and angle (argument) of the complex number 3+4i.
- Generate a 4x4 array of random numbers between 0 and 10. Sort each column of the array.
- Use MATLAB's help function to find built-in functions to determine:
  - The base 10 logarithm of 5
  - The secant of pi

User-Defined Functions in Matlab

- MATLAB permits us to create our own functions
- These are scripts that take in certain inputs and return a value or set of values
- We will need these as we use built-in functions for problem solving

Format of Function Declaration

```
function [output arguments]
  =function_name(input arguments)
```

User-Defined Functions

- Suppose we want to plot

```
sin(3*x)+sin(3.1*x)
```

- Create user-defined function

```
function r=f(x)
  r=sin(3*x)+sin(3.1*x)
```

- Save as f.m

User-Defined Functions (cont)

- Now just call it:

```
x=0:0.1:50;
y=f(x);
plot(x,y)
```

Practice

- Create an m-file that calculates the function

  $g(x)=\cos(x)+\cos(1.1*x)$

- Use it to plot g(x) from x=0 to 100
- Note: previous function was

```
function r=f(x)
  r=sin(3*x)+sin(3.1*x)
```

- … and plot commands were

```
x=0:0.1:50;
y=f(x);
plot(x,y)
```

Practice

- Create an m-file that calculates the function $g(x, \delta)=\cos(x)+\cos((1+\delta)x)$ for a given value of δ
- Use it to plot $g(x, \delta)$ from x=0 to 100 for $\delta = 0.1$ and 0.2

Flow Control

```
if x<10 then
   x=x+1
else
   x=x^2
end
```

Flow Control (cont)

```
for i=1:10
   z=z*i
end
```

Flow Control (cont)

```
A=0
sum=0
while A < 10,
    sum=sum+A;
    A=A+1;
end
```

Practice

- The following function calculates the sum of cubes of the first N integers

```
function r=sumofcubes(N)
ans=0;
for i=1:N
    ans=ans+i^3;
end
r=ans;
```

- Run the code and answer the following questions:

  What is the result for N=20?

  Modify the script to do the same calculation with a “while” loop

Practice

- Now modify this script to add up the cubes of the even integers.
- Note that mod(I,2)=0 when i is an even number

Inline Functions

- One downside to Matlab functions in m-files is the proliferation of files resulting from having each function in its own file
- For simple functions, this can be avoided with an inline function.

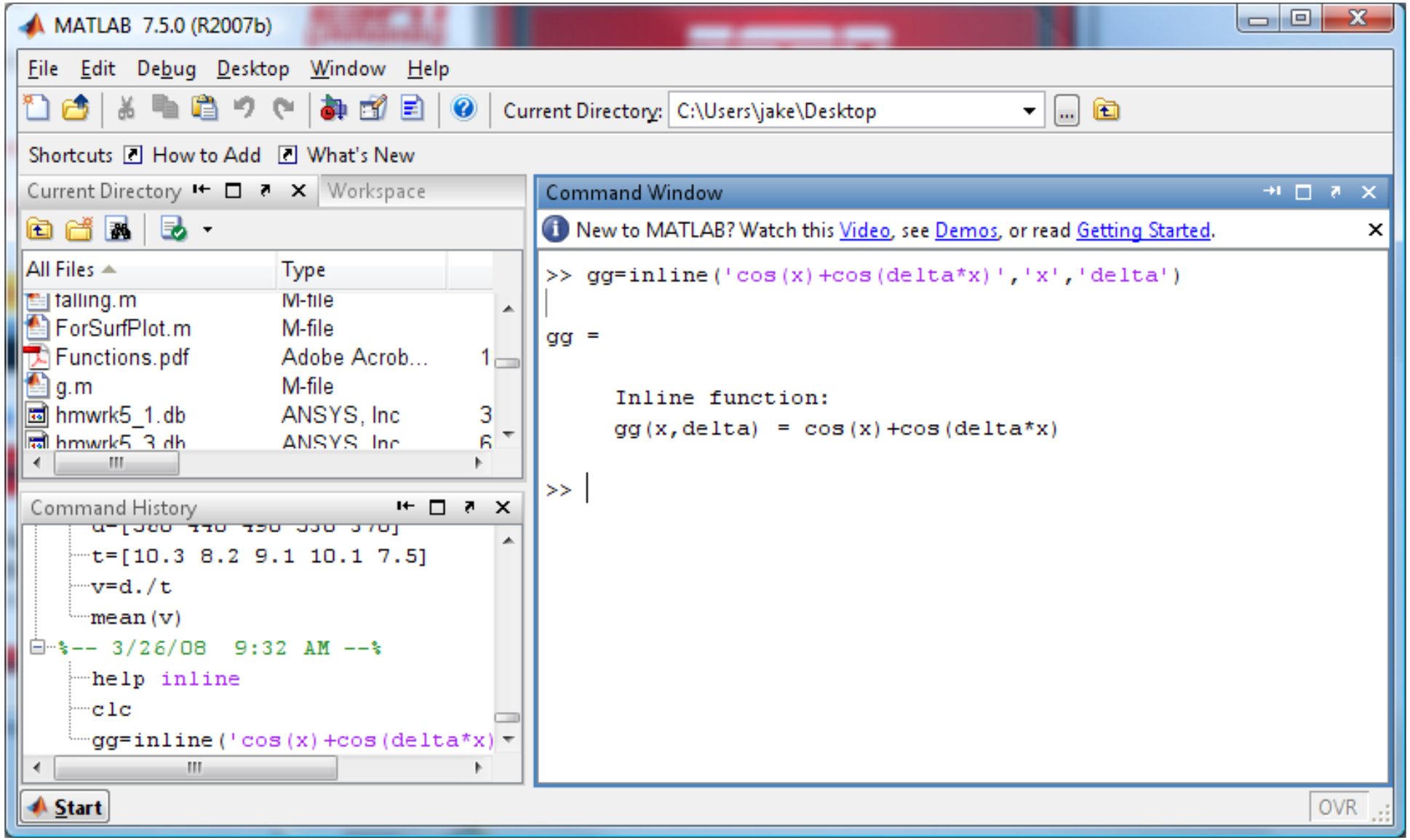
Example

```
g=inline('cos(x)+cos(1.1*x)')
x=0:0.01:100;
y=g(x);
plot(x,y)
```

Parameters

```
x=0:0.01:100;
gg=inline('cos(x)+cos(delta*x)','x','delta')
delta=1.05
y=gg(x,delta);
plot(x,y)
```

Command Window Shows Form of Function



An Alternative Form (Anonymous Functions)

```
x=0:0.01:100;
delta=1.05
gg=@(x, delta) cos(x)+cos(delta*x)
y=gg(x, delta);
plot(x,y)
```

Practice

- Consider the function

$$f(x) = \exp(-a * x) * \sin(x)$$

- Plot using an inline function
- Use 0<x<10 and a=0.25
- Note: syntax can be taken from:

```
gg=inline('cos(x)+cos(delta*x)','x','delta')
gg=@(x, delta) cos(x)+cos(delta*x)
```

Subfunctions

- Subfunctions allow us to put two functions in one file
- The second function will not be available to other functions

```
function example
clear all
r=sumofcubes(20);
fprintf('The sum of the first 20 cubes is %i\n',r)
%
function r=sumofcubes(N)
ans=0;
for i=1:N
   ans=ans+i^3;
end
r=ans;
```

An Example – with Numerics

- Suppose we're looking for a $100k, 30-year mortgage. What interest rate do I need to keep payments below $700 per month?

$$100000 - 700\left[\frac{(1+i)^{360}-1}{i(1+i)^{360}}\right] = 0$$

- Solve for i

Create the function

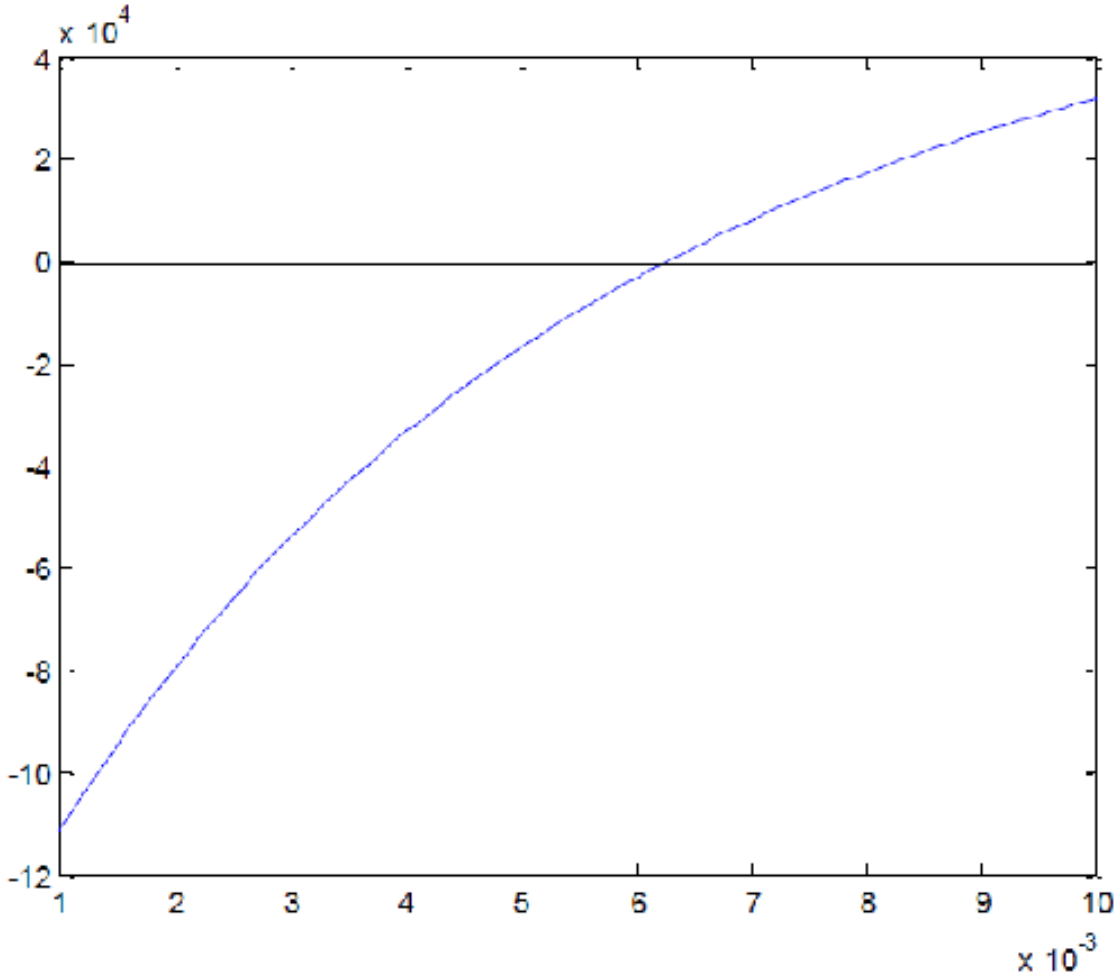
```
function s=f(i)
p=100000;
n=360;
a=700;
s=p-a*((1+i).^n-1)./(i.*(1+i).^n);
```

Plot the Function

- First save file as f.m
- Now enter the following

```
i=0.001:0.0001:0.01;
y=f(i);
plot(i,y)
```

The Plot



Result

- Zero=crossing is around i=0.006
- Annual interest rate is 12*i, or about 7%
- Try more accurate solution

```
12*fzero('f',0.006)
```

- This gives about 7.5%

Approach

- Create user-defined function
- Plot the function
- Find point where function is zero