



INTRODUCTION TO SCIENTIFIC & ENGINEERING COMPUTING BIL 108E, CRN24023

Dr. S. Gökhan Karaman

Technical University of Istanbul

March 08, 2010



Tentative Course Schedule, CRN 24023

Week	Date	Topics
1	Feb. 08	Introduction to Scientific and Engineering Computing
2	Feb. 15	Introduction to Program Computing Environment
3	Feb. 22	Variables, Operations and Simple Plot
4	Mar. 01	Algorithms and Logic Operators
5	Mar. 08	Flow Control, Errors and Source of Errors
6	Mar. 15	Functions
6	Mar. 20	Exam 1
7	Mar. 22	Arrays
8	Mar. 29	Solving of Simple Equations
9	Apr. 05	Polynomials Examples
10	Apr. 12	Applications of Curve Fitting
11	Apr. 19	Applications of Interpolation
11	Apr. 24	Exam 2
12	Apr. 26	Applications of Numerical Integration
13	May 03	Symbolic Mathematics
14	May 10	Ordinary Differential Equation (ODE) Solutions with Built-in Functions



LECTURE # 5

LECTURE # 5

- 1 INLINE FUNCTIONS
- 2 M-FILE
 - SCRIPT M-FILES
 - FUNCTION M-FILES
- 3 EXAMPLES
- 4 RECURSIVE FUNCTIONS
- 5 STRUCTURES
- 6 EXAMPLES



INLINE FUNCTIONS

INLINE FUNCTIONS

- Short mathematical functions may be written as one=line inline objects.

- Usage:

```
function_name = inline('function_definition',
                       'argument1', 'argument2',...)
```



INLINE FUNCTIONS

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

```

MATLAB 7.6.0 (R2008a)
File Edit Debug Desktop Window Help
Current Directory: /media
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
All Files
Type
cdrom Folder
cdrom0 Folder
.hal-rtab HAL-MT
>> help inline
INLINE Construct INLINE object.
INLINE(EXPR) constructs an inline function object from the
MATLAB expression contained in the string EXPR. The input
arguments are automatically determined by searching EXPR
for variable names (see SYMVAR). If no variable exists, 'x'
is used.

INLINE(EXPR, ARG1, ARG2, ...) constructs an inline
function whose input arguments are specified by the
strings ARG1, ARG2, ... Multicharacter symbol names may
be used.

INLINE(EXPR, N), where N is a scalar, constructs an
inline function whose input arguments are 'x', 'P1',
'P2', ..., 'PN'.

Examples:
  
```



INLINE FUNCTIONS

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

```

MATLAB 7.6.0 (R2008a)
File Edit Debug Desktop Window Help
Current Directory: /media
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
be used.

INLINE(EXPR, N), where N is a scalar, constructs an
inline function whose input arguments are 'x', 'P1',
'P2', ..., 'PN'.

Examples:
  g = inline('t^2')
  g = inline('sin(2*pi*f + theta)')
  g = inline('sin(2*pi*f + theta)', 'f', 'theta')
  g = inline('x^P1', 1)

See also symvar.

Overloaded methods:
  eml.inline
  
```



INLINE FUNCTIONS

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

EXAMPLE:

- 1 Convert degrees to radians

```
deg2rad=inline(' deg / 180 * pi ', 'deg')
```

- 2 Calculate hypotenus

```
hyp = inline('sqrt(a^2 + b^2 )', 'a', 'b')
```



INLINE FUNCTIONS

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

EXAMPLES;

```

MATLAB 7.6.0 (R2008a)
File Edit Debug Desktop Window Help
Current Directory: /media
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
>> deg2rad=inline(' deg / 180 * pi ', 'deg')
deg2rad =

  Inline function:
  deg2rad(deg) = deg / 180 * pi
  
```



INLINE FUNCTIONS

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

EXAMPLES;

```

MATLAB 7.6.0 (R2008a)
File Edit Debug Desktop Window Help
Current Directory: /media
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
>> deg2rad(180)
ans =
    3.1416
>> deg2rad(45)
ans =
    0.7854
>> |
Command History
deg2rad(180)
clc
deg2rad(180)
deg2rad(45)
Start

```



INLINE FUNCTIONS

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

EXAMPLES;

```

MATLAB 7.6.0 (R2008a)
File Edit Debug Desktop Window Help
Current Directory: /media
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
>> hyp = inline('sqrt(a^2 + b^2)', 'a', 'b')
hyp =
    Inline function:
    hyp(a,b) = sqrt(a^2 + b^2)
>> |
Command History
hyp = inline('sqrt(a^2 + b^2)', 'a', 'b')
clc
hyp = inline('sqrt(a^2 + b^2)', 'a', 'b')
Start

```



INLINE FUNCTIONS

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

EXAMPLES;

```

MATLAB 7.6.0 (R2008a)
File Edit Debug Desktop Window Help
Current Directory: /media
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
>> hyp = inline('sqrt(a^2 + b^2)', 'a', 'b')
hyp =
    Inline function:
    hyp(a,b) = sqrt(a^2 + b^2)
>> hyp(3,4)
ans =
    5
>> |
Command History
hyp = inline('sqrt(a^2 + b^2)', 'a', 'b')
clc
hyp = inline('sqrt(a^2 + b^2)', 'a', 'b')
hyp(3,4)
Start

```



SCRIPT M-FILES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

SCRIPT M-FILES

- Programs are contained in **m-files**. **m-files** are plain text files. Not binary files produced by word processors.
- File must have ".m" extension
- m-file must be in the path Matlab maintains its own internal path
- The path is the list of directories that Matlab will search when looking for an m-file to execute.
- Manually modify the path with the path, addpath, and rmpath built-in functions.

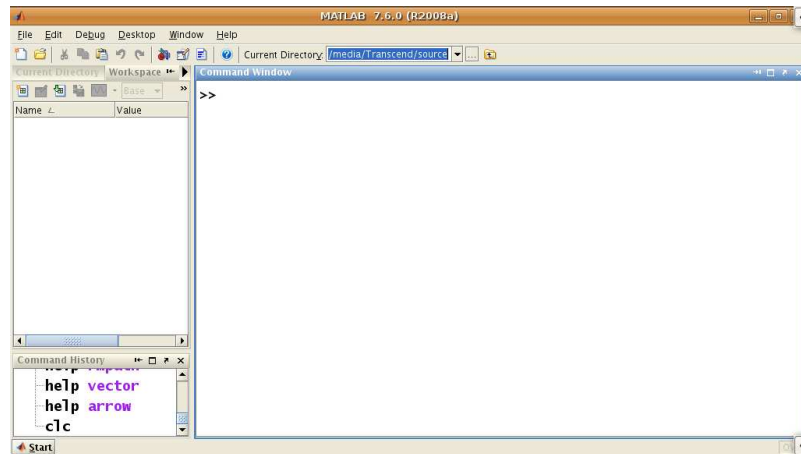


SCRIPT M-FILES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

CURRENT DIRECTORY

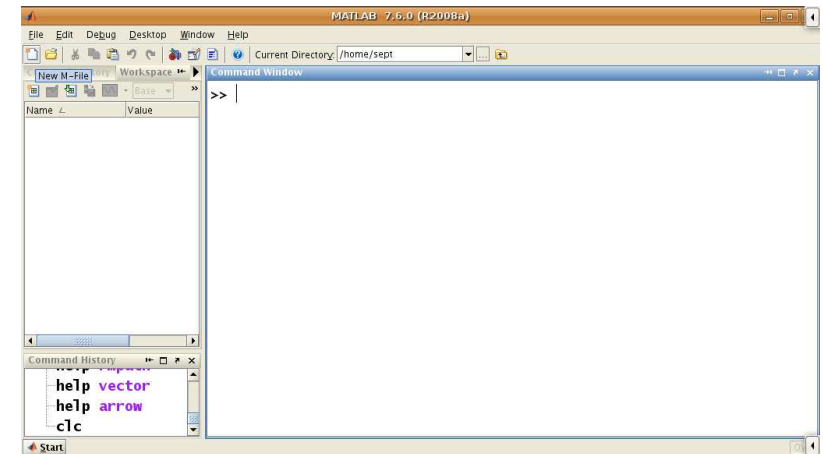


SCRIPT M-FILES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

CREATE M-FILE

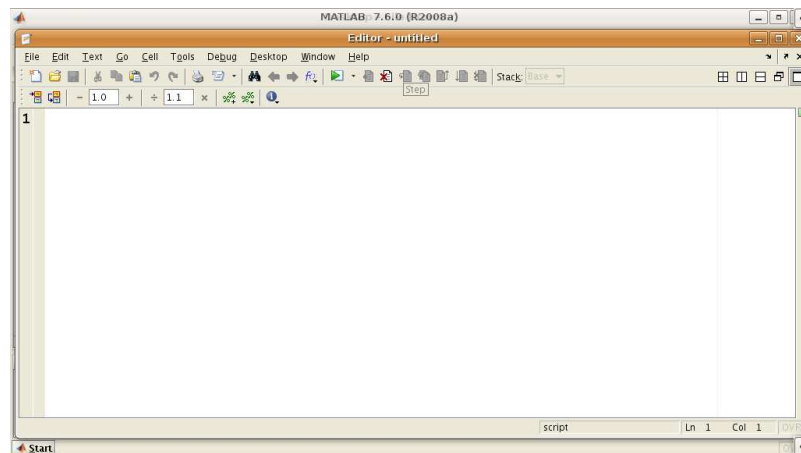


SCRIPT M-FILES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

CREATE M-FILE

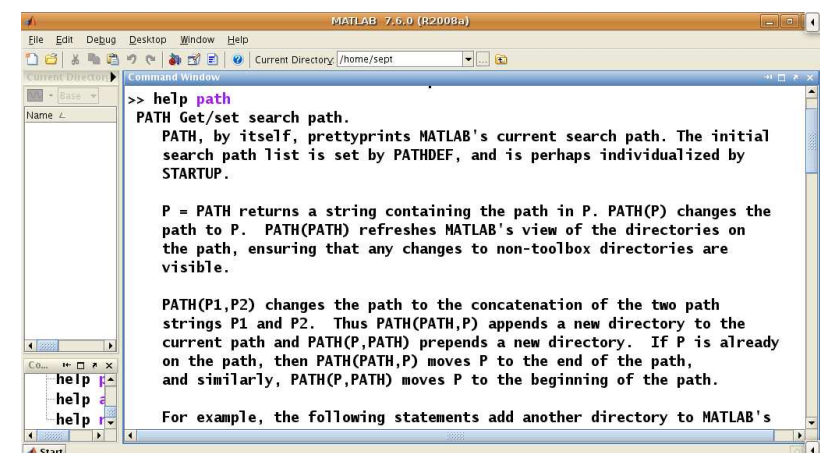


SCRIPT M-FILES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

path





SCRIPT M-FILES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

path

visible.

PATH(P1,P2) changes the path to the concatenation of the two path strings P1 and P2. Thus PATH(PATH,P) appends a new directory to the current path and PATH(P,PATH) prepends a new directory. If P is already on the path, then PATH(PATH,P) moves P to the end of the path, and similarly, PATH(P,PATH) moves P to the beginning of the path.

For example, the following statements add another directory to MATLAB's search path on various operating systems:

Unix: path(path, '/home/myfriend/goodstuff')
 Windows: path(path, 'c:\tools\goodstuff')

See also [what](#), [cd](#), [dir](#), [addpath](#), [rmpath](#), [genpath](#), [pathtool](#), [savepath](#), [rehash](#).

Reference page in Help browser
[doc path](#)



SCRIPT M-FILES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

addpath

>> help addpath

ADDPATH Add directory to search path.

ADDPATH DIRNAME prepends the specified directory to the current matlabpath. Surround the DIRNAME in quotes if the name contains a space. If DIRNAME is a set of multiple directories separated by path separators, then each of the specified directories will be added.

ADDPATH DIR1 DIR2 DIR3 ... prepends all the specified directories to the path.

ADDPATH ... -END appends the specified directories.
 ADDPATH ... -BEGIN prepends the specified directories.
 ADDPATH ... -FROZEN disables directory change detection for directories being added and thereby conserves Windows change notification resources (Windows only).

Use the functional form of ADDPATH, such as ADDPATH('dir1','dir2',...), when the directory specification is stored in a string.



SCRIPT M-FILES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

addpath

being added and thereby conserves Windows change notification resources (Windows only).

Use the functional form of ADDPATH, such as ADDPATH('dir1','dir2',...), when the directory specification is stored in a string.

P = ADDPATH(...) returns the path prior to adding the specified paths.

Examples

```
addpath c:\matlab\work
addpath /home/user/matlab
addpath /home/user/matlab:/home/user/matlab/test:
addpath /home/user/matlab /home/user/matlab/test
```

See also [rmpath](#), [pathtool](#), [path](#), [savepath](#), [userpath](#), [genpath](#), [rehash](#).

Reference page in Help browser
[doc addpath](#)



SCRIPT M-FILES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

rmpath

>> help rmpath

RMPATH Remove directory from search path.

RMPATH DIRNAME removes the specified directory from the current matlabpath. Surround the DIRNAME in quotes if the name contains a space. If DIRNAME is a set of multiple directories separated by path separators, then each of the specified directories will be removed.

RMPATH DIR1 DIR2 DIR3 removes all the specified directories from the path.

Use the functional form of RMPATH, such as RMPATH('dir1','dir2',...), when the directory specification is stored in a string.

P = RMPATH(...) returns the path prior to removing the specified paths.

Examples

```
rmpath c:\matlab\work
rmpath /home/user/matlab
rmpath /home/user/matlab:/home/user/matlab/test:
```

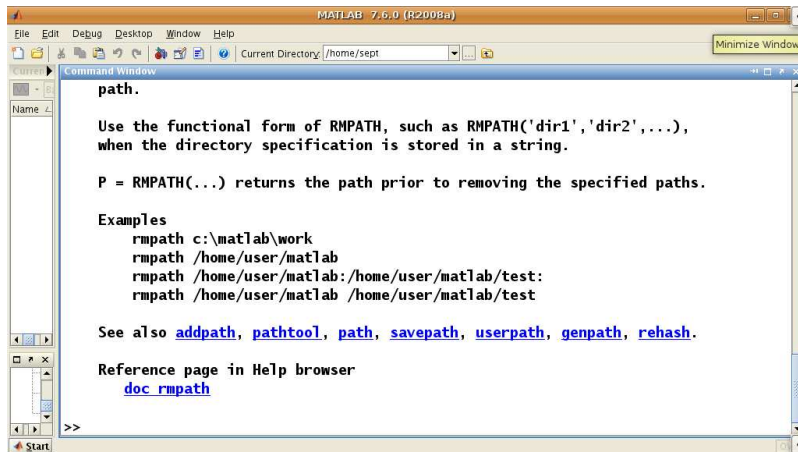


SCRIPT M-FILES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

rmpath



SCRIPT M-FILES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

SCRIPT M-FILES

- Not really programs
- No input / output parameters
- Script variables are part of workspace
- Useful for tasks that never change
- Use a script to run function for specific parameters required by the assignment



SCRIPT M-FILES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

EXAMPLE:

```
% SIMPLE SCRIPT FILE
theta = linspace (1.4 , 4.6);
tandata = tan(theta);
plot (theta, tandata);
xlabel('\theta');
ylabel('tangent');
grid;
% File named as ex_05_03.m and run with
% the name of the filename without the ".m".
```

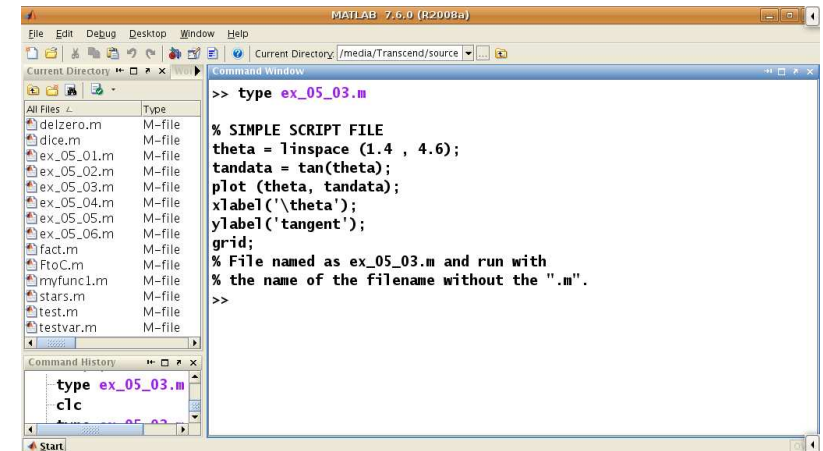


SCRIPT M-FILES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

EXAMPLE:



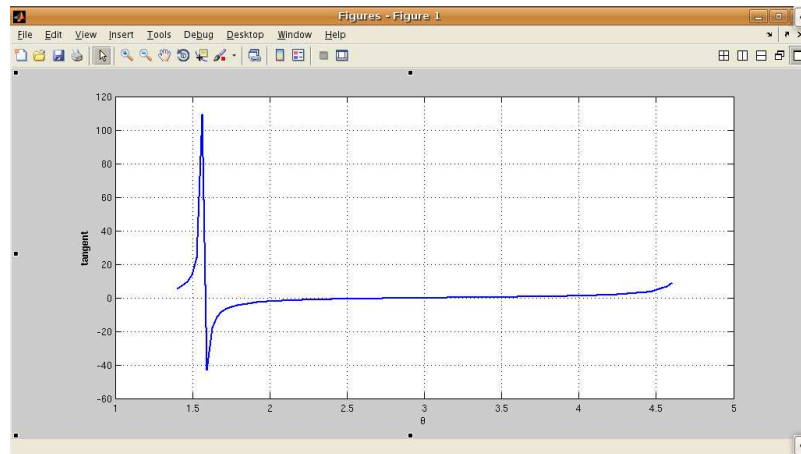


SCRIPT M-FILES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

EXAMPLE:



SCRIPT M-FILES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

SCRIPT M-FILES

- All variables created in a script file are added to the workplace.
- This may have undesirable effects because variables already existing in the workspace may be overwritten



FUNCTION M-FILES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

FUNCTION M-FILES

- Matlab has many built-in functions.
- Use type *functionname* to verify the functions.
- Function m-files differ from a script file in that it communicates with the MATLAB workspace only through specially designated **input** and **output** arguments.



FUNCTION M-FILES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

FUNCTION M-FILES

- Functions use input and output parameters to communicate with other functions and the command window
- Functions use local variables that exist only while the function is executing. Local variables are distinct from variables of the same name in the workspace or in other functions.
- Input parameters allow the same calculation procedure (same algorithm) to be applied to different data. Thus, function m-files are reusable.
- Functions can call other functions.



FUNCTION M-FILES

Introduction to Scientific and Engineering Computing, BIL108E
Karaman

BASIC RULES

General form of a function *filename.m* file

```
function [ outarg1, outarg2, ... ] =
           name(inarg1, inarg2, ...)
% Comments to be displayed with help name
...
outarg1;
...
outarg2;
```



FUNCTION M-FILES

Introduction to Scientific and Engineering Computing, BIL108E
Karaman

EXAMPLE:

- 1 Write a function called FtoC (FtoC.m) to convert Fahrenheit temperatures into Celsius.

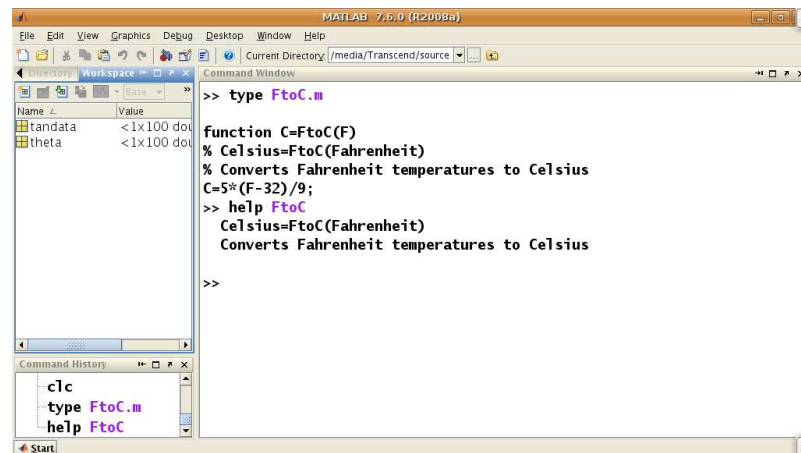
```
function C=FtoC(F)
% Celsius=FtoC(Fahrenheit)
% Converts Fahrenheit temperatures to Celsius
C=5*(F-32)/9;
```



FUNCTION M-FILES

Introduction to Scientific and Engineering Computing, BIL108E
Karaman

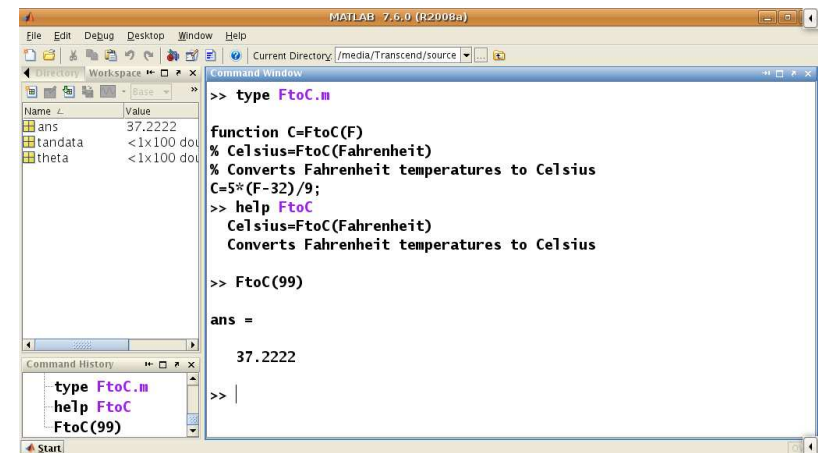
EXAMPLE:



FUNCTION M-FILES

Introduction to Scientific and Engineering Computing, BIL108E
Karaman

EXAMPLE:





FUNCTION M-FILES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

SUMMARY OF INPUT AND OUTPUT ARGUMENTS

- Values are communicated through input arguments and output arguments.
- Variables defined inside a function are local to that function. Local variables are invisible to other functions and to the command environment.
- The number of return variables should match the number of output variables provided by the function.



FUNCTION M-FILES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

SCOPE

LOCAL VARIABLES

- Any variable defined inside a function is inaccessible outside it.
- Such variables are referred to as local. They exist only inside the function, which has its own workspace separate from the base workspace of variables defined in the Command Window.



FUNCTION M-FILES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

SCOPE

GLOBAL VARIABLES

- Variables defined in the base workspace are not normally accessible inside functions.
- Their scope is restricted to the workspace itself unless they have been declared global: `global VARIABLENAME`
- If several functions, and possibly the base workspace, declare particular variables as global, then they all share single copies of them.



FUNCTION M-FILES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

SCOPE

GLOBAL VARIABLES cont'd.

- Matlab recommends that global variables be typed in capital letters to remind you that they are global.
- The function `isglobal(VARNAME)` returns 1 if A is global, and 0 otherwise.
- The command `who global` gives a list of global variables.
- The command `clear global` makes all variables nonglobal. Example: `clear VARNAME` makes VARNAME nonglobal.



FUNCTION M-FILES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

PERSISTENT VARIABLES

Persistent variables remain in existence between function calls.

Example:

```
function test
persistent count
if isempty(count)
count = 1
else
count = count + 1
end
```

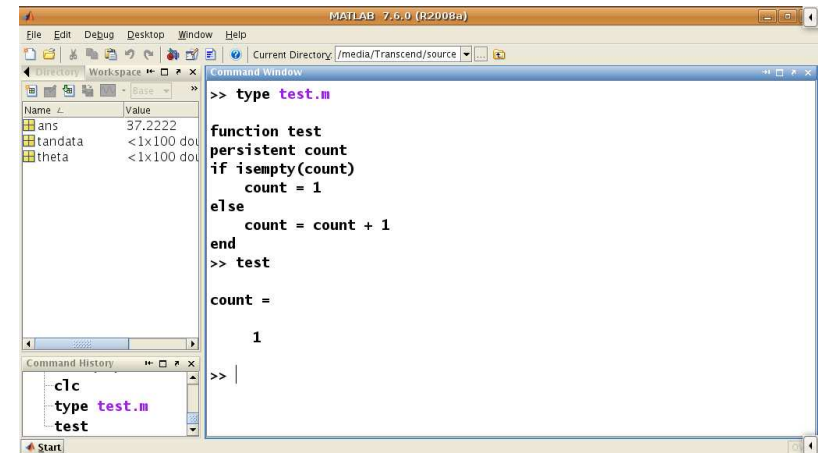


FUNCTION M-FILES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

EXAMPLE:

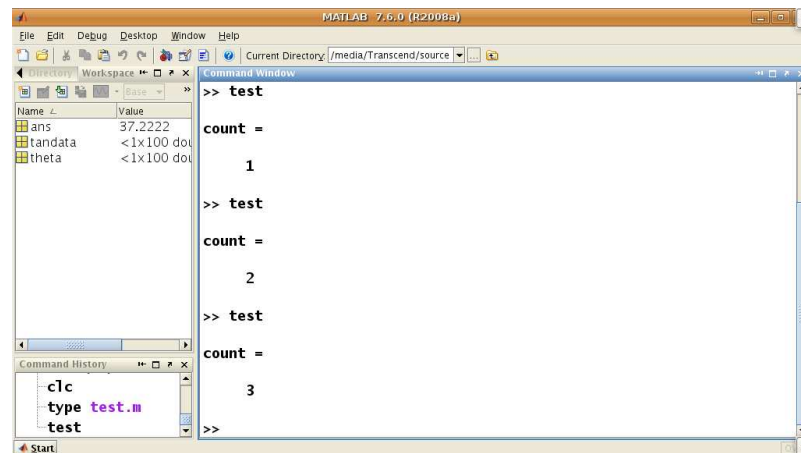


FUNCTION M-FILES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

EXAMPLE:



FUNCTION M-FILES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

FUNCTIONS WITHOUT RETURN VALUE

Omit the equal sign and output arguments in the function definition line.

EXAMPLE:

```
function stars(n)
asterisk = char(abs('**')*ones(1,n));
disp(asterisk)
```



FUNCTION M-FILES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

EXAMPLE:

```

MATLAB 7.6.0 (R2008a)
File Edit Debug Desktop Window Help
Current Directory: /media/Transcend/source
Directory: Workspace
Name Value
ans 37.2222
tandata <1x100 dou
theta <1x100 dou
>> type stars
function stars(n)
asterisk = char(abs('*')*ones(1,n));
disp(asterisk)
>> stars(9)
*****
>>
Command History
clc
type stars
stars(9)
Start

```



FUNCTION M-FILES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

EXAMPLE:

```

MATLAB 7.6.0 (R2008a)
File Edit Debug Desktop Window Help
Current Directory: /media/Transcend/source
Directory: Workspace
Name Value
ans 42
tandata <1x100 dou
theta <1x100 dou
>> type stars
function stars(n)
asterisk = char(abs('*')*ones(1,n));
disp(asterisk)
>> stars(9)
*****
>> abs('*')
ans =
42
>>
Command History
type stars
stars(9)
abs('*')
Start

```



FUNCTION M-FILES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

EXAMPLE:

```

MATLAB 7.6.0 (R2008a)
File Edit Debug Desktop Window Help
Current Directory: /media/Transcend/source
Directory: Workspace
Name Value
ans **
tandata <1x100 dou
theta <1x100 dou
>> function stars(n)
>> asterisk = char(abs('*')*ones(1,n));
>> disp(asterisk)
>> stars(9)
*****
>> abs('*')
ans =
42
>> char(ans)
ans =
*
>>
Command History
stars(9)
abs('*')
char(ans)
Start

```



FUNCTION M-FILES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

VECTOR ARGUMENTS

- Input and output arguments can be defined as vectors.
- A vector is initialized each time the function is called.

EXAMPLE:

dice function generates a vector of n random rolls of a die.

```

function d= dice(n)
d = floor(6 * rand(1, n) + 1);

```



FUNCTION M-FILES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

EXAMPLE:

```

MATLAB 7.6.0 (R2008a)
File Edit Debug Desktop Window Help
Current Directory: /media/Transcend/source
Workspace: ans, tandata, theta
Command Window:
>> type dice.m

function d= dice(n)
d = floor(6 * rand(1, n) + 1);
>> |

Command History:
char(ans)
clc
type dice.m

```



FUNCTION M-FILES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

EXAMPLE:

```

MATLAB 7.6.0 (R2008a)
File Edit Debug Desktop Window Help
Current Directory: /media/Transcend/source
Workspace: ans [5,6,1], tandata, theta
Command Window:
>> type dice.m

function d= dice(n)
d = floor(6 * rand(1, n) + 1);
>> dice(3)

ans =

    5    6    1

>> |

Command History:
clc
type dice.m
dice(3)

```



FUNCTION M-FILES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

EXAMPLE:

```

MATLAB 7.6.0 (R2008a)
File Edit Debug Desktop Window Help
Current Directory: /media/Transcend/source
Workspace: ans 4, tandata, theta
Command Window:
>> type dice.m

function d= dice(n)
d = floor(6 * rand(1, n) + 1);
>> dice(3)

ans =

    5    6    1

>> floor(4.7)

ans =

    4

>> |

Command History:
type dice.m
dice(3)
floor(4.7)

```



FUNCTION M-FILES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

EXAMPLE:

```

MATLAB 7.6.0 (R2008a)
File Edit Debug Desktop Window Help
Current Directory: /media/Transcend/source
Workspace: ans 4, tandata, theta
Command Window:
>> dice(3)

ans =

    5    6    1

>> floor(4.7)

ans =

    4

>> floor(4.2)

ans =

    4

>> |

Command History:
dice(3)
floor(4.7)
floor(4.2)

```



FUNCTION M-FILES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

PASSING AN ARGUMENT

- Passing an argument by value:
An input argument is passed by value only if a function modifies it.
- Passing an argument by reference

```
function y = delzero(x)
y = x(x ~= 0);

% x = delzero(x)
```

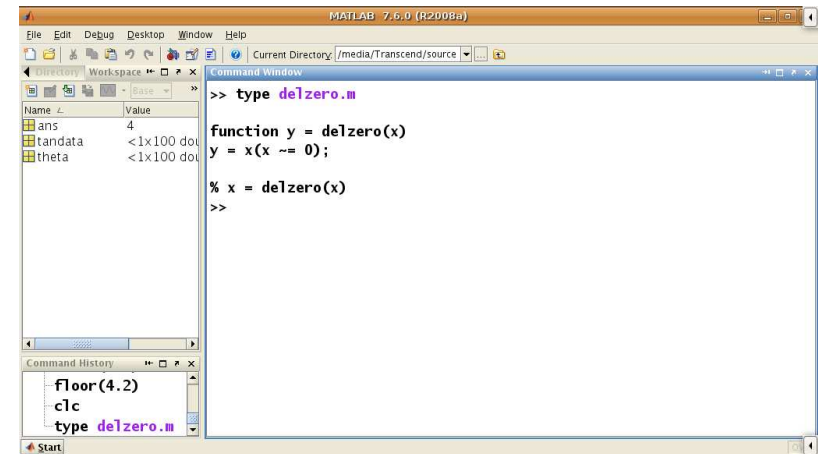


FUNCTION M-FILES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

EXAMPLE:

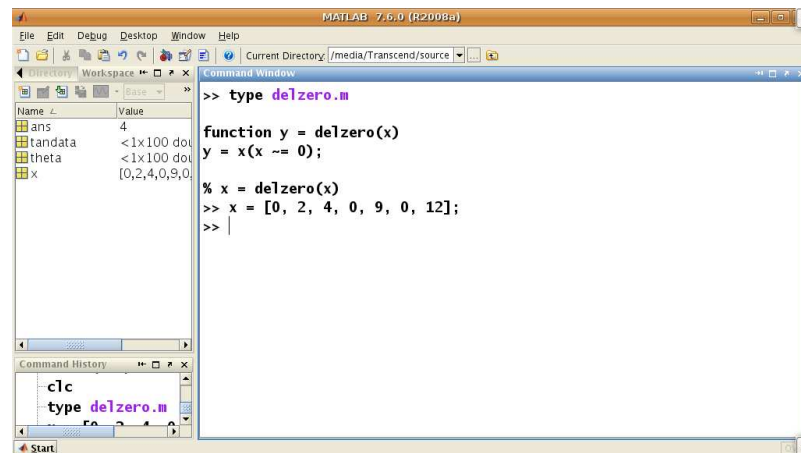


FUNCTION M-FILES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

EXAMPLE:

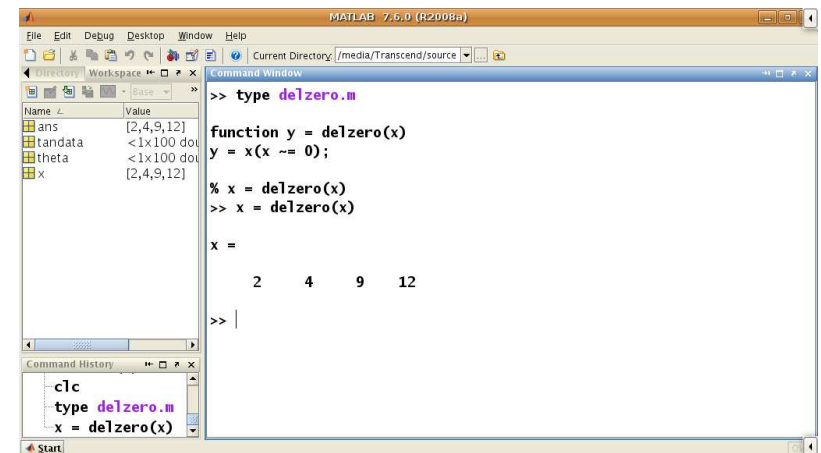


FUNCTION M-FILES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

EXAMPLE:





FUNCTION M-FILES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

NUMBER OF ARGUMENTS OF A FUNCTION

Use `nargin` and `nargout` functions to display the number of input and output arguments.

EXAMPLE:

```
function y = myfunc1(a, b, c)
disp(nargin)
y = nargin;
```



FUNCTION M-FILES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

EXAMPLE:

```

MATLAB 7.6.0 (R2008a)
File Edit Debug Desktop Window Help
Current Directory: /media/Transcend/source
Workspace:
Name Value
ans 2
tandata < 1x100 dot
theta < 1x100 dot
x [2,4,9,12]

Command Window:
>> type myfunc1.m
function y = myfunc1(a, b, c)
disp(nargin)
y = nargin;
>> myfunc1(1,2)
2

ans =
2

>>

Command History:
clc
type myfunc1.m
myfunc1(1,2)

```



FUNCTION M-FILES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

SUBFUNCTIONS

- An M-file may contain the code for more than one function.
- The first one in the file is the **primary function** and is invoked with the M-file name.
- Additional functions are called **subfunctions** and are visible only to the primary function and to other subfunctions.
- Each subfunction begins with its own function definition line.
- Subfunctions follow each other in any order after the primary function.



RECURSIVE FUNCTIONS

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

RECURSIVE FUNCTIONS

MATLAB allows functions to call themselves in a process called recursion.

Recursive function can cost too much time and memory.

Example:

Factorial function can be written in a recursive ".m" file.

$$n! = n \times (n - 1)!$$



RECURSIVE FUNCTIONS

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

EXAMPLE:

```
function y = fact(n)
% Factorial Recursive definition of n!
disp(n)
if n > 1
    y = n * fact(n-1);
else
    y = 1;
end
```

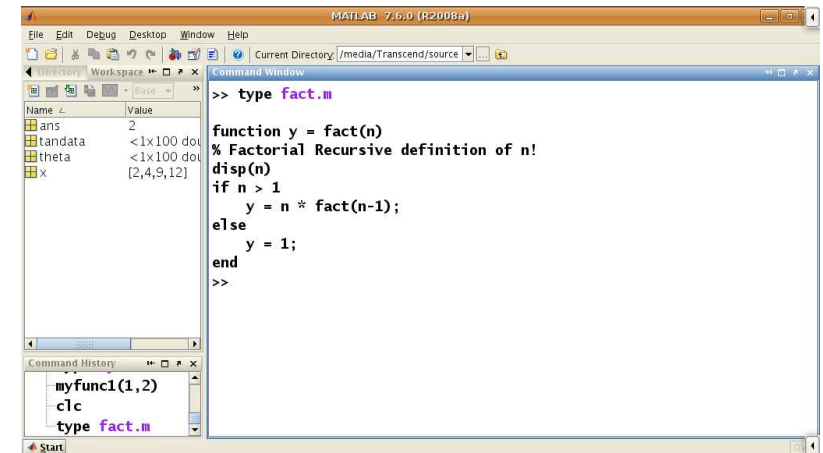


RECURSIVE FUNCTIONS

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

EXAMPLE:

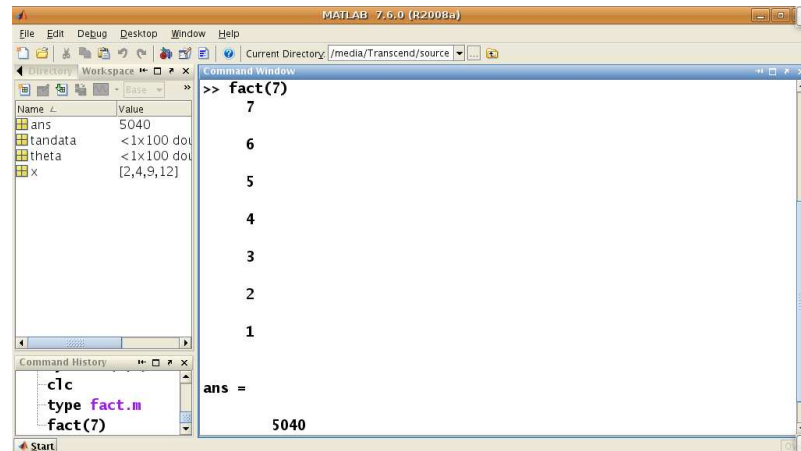


RECURSIVE FUNCTIONS

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

EXAMPLE:



FUNCTION HANDLES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

FUNCTION HANDLES

Function handles points the defined function.

- A handle for a function is created with @. A function may be represented by its handle. In particular, the handle may be passed as an argument to another function.
- feval evaluates a function whose handle is passed to it as an argument.



FUNCTION HANDLES

Introduction to Scientific and Engineering Computing, BIL108E
Karaman

EXAMPLE:

```
function myplotfunc(funchand, limit1, sample)
% plot the given function
% with the given limits
x = linspace(limit1(1), limit1(2),sample);
y = feval(funchand, x);
plot(x,y)

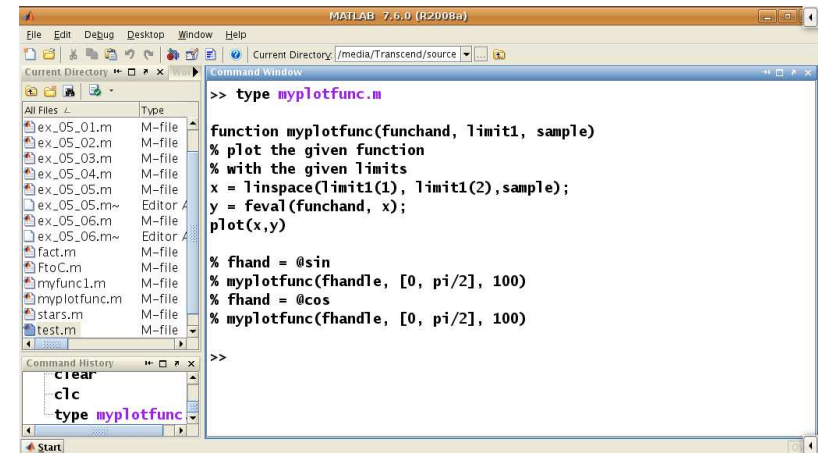
% fhand = @sin
% myplotfunc(fhandle, [0, pi/2], 100)
% fhand = @cos
% myplotfunc(fhandle, [0, pi/2], 100)
```



FUNCTION HANDLES

Introduction to Scientific and Engineering Computing, BIL108E
Karaman

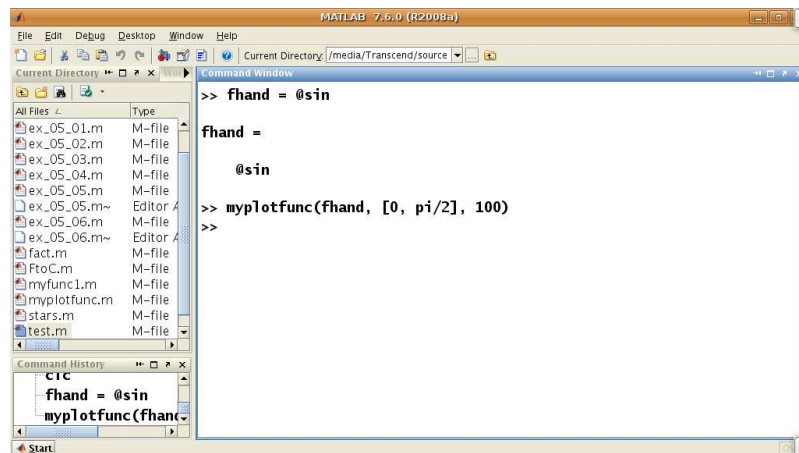
EXAMPLE:



FUNCTION HANDLES

Introduction to Scientific and Engineering Computing, BIL108E
Karaman

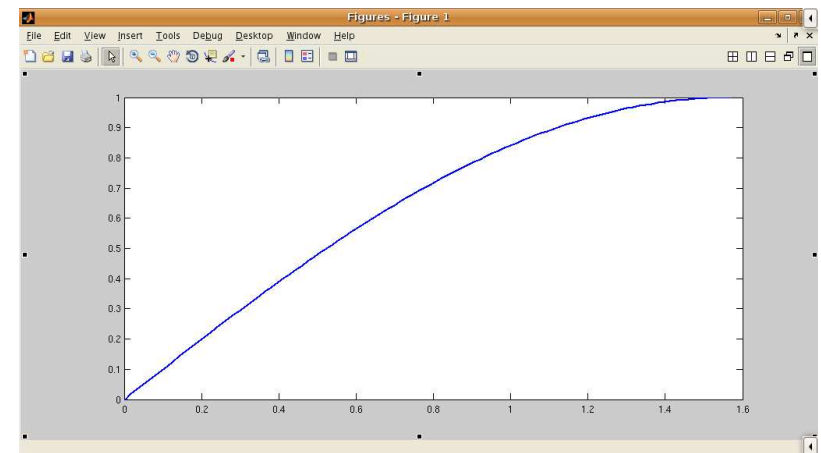
EXAMPLE:



FUNCTION HANDLES

Introduction to Scientific and Engineering Computing, BIL108E
Karaman

EXAMPLE:





FUNCTION HANDLES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

EXAMPLE:

```

MATLAB 7.6.0 (R2008a)
File Edit Debug Desktop Window Help
Current Directory: /media/Transcend/source
Command Window
>> fhand = @sin
fhand =
    @sin
>> myplotfunc(fhand, [0, pi/2], 100)
>> fhand = @cos
fhand =
    @cos
>> myplotfunc(fhand, [0, pi/2], 100)
>>
Command History
myplotfunc(fhand, [0, pi/2], 100)
fhand = @cos
myplotfunc(fhand, [0, pi/2], 100)

```

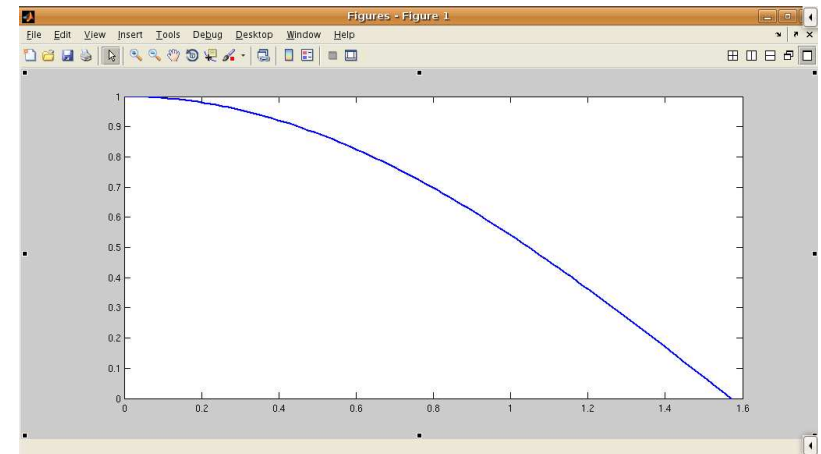


FUNCTION HANDLES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

EXAMPLE:



FUNCTION HANDLES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

EXAMPLE:

```

MATLAB 7.6.0 (R2008a)
File Edit Debug Desktop Window Help
Current Directory: /media/Transcend/source
Command Window
    @sin
>> myplotfunc(fhand, [0, pi/2], 100)
>> fhand = @cos
fhand =
    @cos
>> myplotfunc(fhand, [0, pi/2], 100)
>> fhand = @sin
fhand =
    @sin
>> hold('on')
>> myplotfunc(fhand, [0, pi/2], 100)
>>
Command History
fhand = @sin
hold('on')
myplotfunc(fhand, [0, pi/2], 100)

```

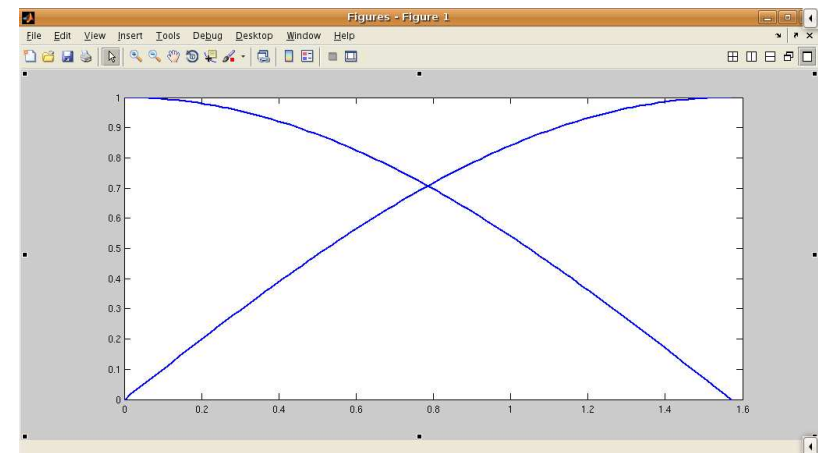


FUNCTION HANDLES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

EXAMPLE:





FUNCTION M-FILES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

EXAMPLE:

The function $\sin(x)$ can be written as a Taylor series by

$$\sin(x) = \sum_{k=1}^{\infty} \frac{(-1)^k x^{2k+1}}{(2k+1)!}$$

Write a user-defined function file that calculates $\sin(x)$ by using Taylor's series. For the function name and arguments use $y = T\sin(x, n)$. The input arguments are the angle x in degrees, and n the number of terms in the series. Use the function to calculate $\sin(150^\circ)$ using 3 and 7 terms.



STRUCTURES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

STRUCTURES

Arrays can store variables that may be all numeric or character. With structure different data types can be stored as one variable within a structure.

EXAMPLE:

Create a structure called student with one field for a student's name, a second for his/her student ID number, and a third for all her marks to date.

```
student.name = 'Can Ozgur';
student.id = 'N010080090';
student.marks=[80, 60, 40];
% student
% student.marks(2)
```



STRUCTURES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

STRUCTURES

Use subscripts to add more elements

EXAMPLE:

```
student(2).name = 'Ergun Yilmaz';
student(2).id = 'N010080091';
student(2).marks=[70, 30, 90];
% student
% student(2).marks(2)
```



STRUCTURES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

EXAMPLE:

```
MATLAB 7.6.0 (R2008a)
File Edit Debug Desktop Window Help
Current Directory: /media/Transcend/source
Directory: Workspace
Name Value
>> type ex_05_05
student.name = 'Can Ozgur';
student.id = 'N010080090';
student.marks=[80, 60, 40];
% student
% student.marks(2)
>> ex_05_05
Command History
clear
clc
type ex_05_05
```



STRUCTURES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

EXAMPLE:

```

MATLAB 7.6.0 (R2008a)
File Edit Debug Desktop Window Help
Current Directory: /media/Transcend/source
Workspace:
Name Value
student <1x1 struct>
Command Window
>> type ex_05_05
student.name = 'Can Ozgur';
student.id = 'N010080090';
student.marks=[80, 60, 40];
% student
% student.marks(2)

>> ex_05_05
>> student

student =

    name: 'Can Ozgur'
    id: 'N010080090'
    marks: [80 60 40]

Command History
type ex_05_05
ex_05_05
student
  
```



STRUCTURES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

EXAMPLE:

```

MATLAB 7.6.0 (R2008a)
File Edit Debug Desktop Window Help
Current Directory: /media/Transcend/source
Workspace:
Name Value
ans 80
student <1x1 struct>
Command Window
% student
% student.marks(2)
ans =
    80
>> ex_05_05
>> student

student =

    name: 'Can Ozgur'
    id: 'N010080090'
    marks: [80 60 40]

>> student.marks(1)

ans =

    80

Command History
ex_05_05
student
  
```



STRUCTURES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

EXAMPLE:

```

MATLAB 7.6.0 (R2008a)
File Edit Debug Desktop Window Help
Current Directory: /media/Transcend/source
Workspace:
Name Value
ans 80
student <1x1 struct>
Command Window
>> type ex_05_06.m
student(2).name = 'Ergun Yilmaz';
student(2).id = 'N010080091';
student(2).marks=[70, 30, 90];
% student
% student(2).marks(2)

>> |

Command History
student.marks(1)
clc
type ex_05_06.m
  
```



STRUCTURES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

EXAMPLE:

```

MATLAB 7.6.0 (R2008a)
File Edit Debug Desktop Window Help
Current Directory: /media/Transcend/source
Workspace:
Name Value
ans 80
student <1x2 struct array>
Command Window
>> type ex_05_06.m
student(2).name = 'Ergun Yilmaz';
student(2).id = 'N010080091';
student(2).marks=[70, 30, 90];
% student
% student(2).marks(2)

>> ex_05_06
>> student

student =

1x2 struct array with fields:
    name
    id
    marks

Command History
type ex_05_06.m
ex_05_06
student
  
```



STRUCTURES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

EXAMPLE:

```

MATLAB 7.6.0 (R2008a)
File Edit Debug Desktop Window Help
Current Directory: /media/Transcend/source
Workspace:
Name Value
ans <1x1 struct
student <1x2 struct
Command Window
>> ex_05_06
>> student
student =
1x2 struct array with fields:
    name
    id
    marks
>> student(2)
ans =
    name: 'Ergun Yilmaz'
    id: 'N010080091'
    marks: [70 30 90]
Command History
ex_05_06
student
student(2)
Start

```



STRUCTURES

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

EXAMPLE:

```

MATLAB 7.6.0 (R2008a)
File Edit Debug Desktop Window Help
Current Directory: /media/Transcend/source
Workspace:
Name Value
ans 90
student <1x2 struct
Command Window
name
id
marks
>> student(2)
ans =
    name: 'Ergun Yilmaz'
    id: 'N010080091'
    marks: [70 30 90]
>> student(2).marks(3)
ans =
    90
Command History
student
student(2)
student(2).marks(3)
Start

```



M-FILE DEBUGGING

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

M-FILE DEBUGGING

The Editor / Debugger enables you to get inside a function, while it is running.



References

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

References for Week 5

- 1 Brian Hahn, Daniel T.Valentine, Essential Matlab for Engineers and Scientists, Elsevier, 2010.