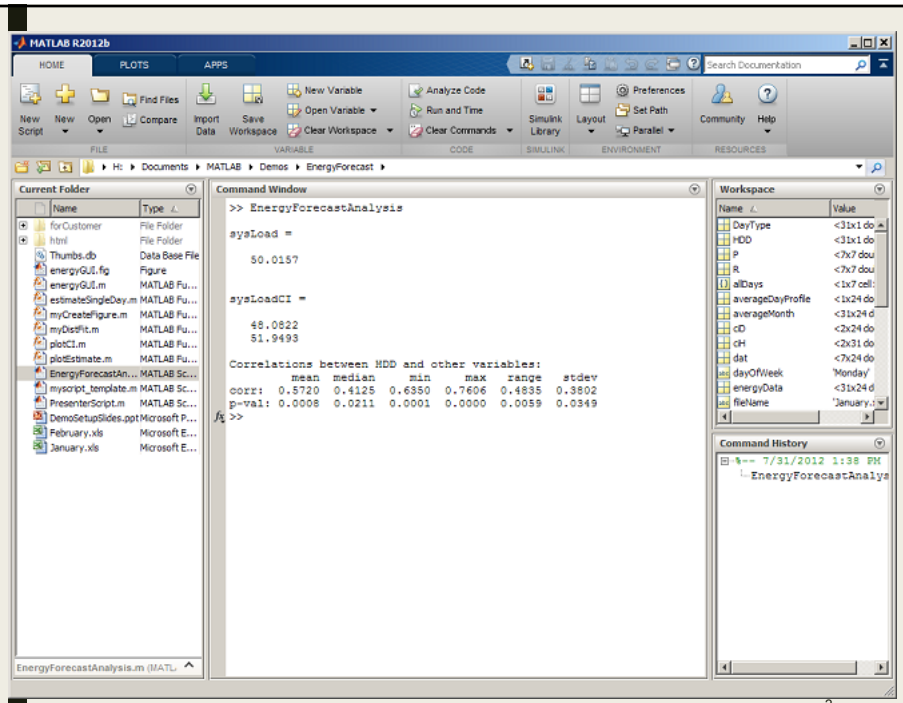
Introduction to Matlab

1



2

Introduction to Matlab

- Simple introduction to some basic Matlab syntax.
- Declaration of a variable
- [ ] Matrices or vectors
- Some special (useful) syntax.
- Control statements
- Functions
- Input and output

3

Declaration of a variable

- One of the first things you might want to do in a programming language is to declare a variable.
- In Matlab you can declare and assign a value as below:

```
a=1
b=2
```

4

Simple operations

- We can add subtract, multiply and divide.

a + b

a - b

The answer is returned in the system generated variable "ans"

- *Alternatively we can assign the answer to our own new variable*

c = a + b

d = cos(a)

5

Creating Vectors

MATLAB is an array based language where variables can be vectors, matrices or N dimensional arrays. You use square brackets to construct array. To create a row vector you can type:

t=[1 2 3 4 5]

You can use the colon operator to simplify the creation of equally spaced arrays.

t = 1:5

You can recall previously entered commands by dragging them from the command history here or by pressing the up-arrow key.

t=0:.01:1 *%t goes from 0 in steps of .01 to 1*

Adding a semicolon avoids command output being echoed to the command window.

t=0:.01:1;

6

- To see what variables you have created so far type,

  whos

...or view a list in the **workspace browser**

To see the value of a variable just type its name such as,

  b

- Vector Operations. You carry-out operations on vectors just like simple scalars. For example,

  y = sin(2*pi*t)

This makes use of the constant pi, pre-defined in MATLAB.

- Basic plotting. You can plot y against t with the plot command:

  plot(t,y)

7

Creating Matrices.

- You enter matrices using the semicolon in the following way,

a = [1 2 3; 4 5 6; 7 8 10]

- You can find the dimensions of an array with the |size| function.

size(a)

data=rand(5,5) %Uniformly distributed random numbers

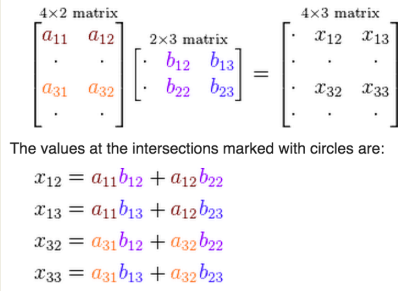
r = randi([0 10],2,5) %Random Integers

8

- Matrix Operations

You can perform matrix operations such as...

- **b = a'** *% b = the transpose of a.*
- **c = a*b** % c = a times b, which performs matrix multiplication,...



- **c = a.*b** % c = a dot times b, which performs element-wise multiplication, where the corresponding elements of each matrix are multiplied.

9

- Indexing. You can select elements or sections of an array by indexing. For the variable a, here is the value at...

  a(2,3) % ...the second row and third column

- ...here is the section from,

  data(1:3,2:end) % rows 1 to 3 and columns 2 to the end.

- You can set values in this way too. For example, with data, you could

  data(1:2, :) = 0 % set rows 1:2 and all the columns to zero.

- The colon operator used on its own in indexing, specifies "all elements", in this case, all columns.
- Note that array indices in MATLAB start at 1.

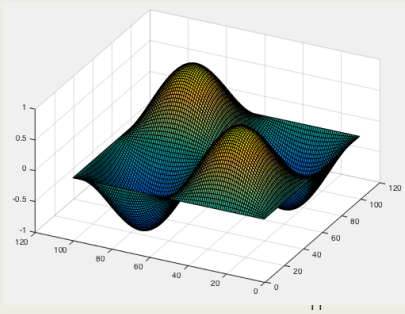
10

Plotting Matrices

- And you can plot matrices as well. If you wanted to display the matrix w=y'*y;
- generated by multiplying the transpose of the sine wave vector y, with itself. you could enter...

surf(w);

which creates a surface plot.



Controls (if & for)

```
A=3
for J=1:8
    A=A+1
end

B=7
if  B==7
    C=10
elseif B==5
    C=11
else
    C=100
end
```

```
for var=start value: end value

    Statements ……

end  %of for


for var=start value: step(optional) : end
value

    Statements ……

end  %of for
```

12

Controls (while)

```
C=1
while C~=4
    C=C+1
end
```

- *Use Matlab help (command line) to find switch, break and continue.*
- *Example "help switch"*

13

m-files

- Matlab scripts can be written and saved as an "m-file".
- They have a ".m" extension.
- There are simply a series of program steps saved for later execution.
- To run the script type the name of the m-file.
- Make the above "for" loop into a script m-file.

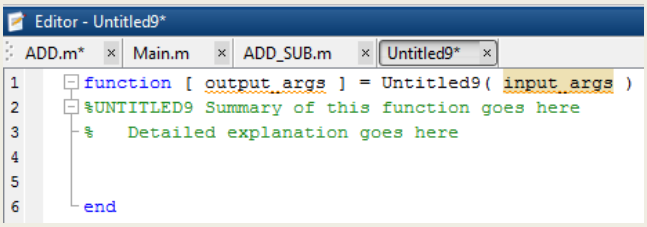
14

Functions

- M-file with the function keyword.
- First line:

function [myout1, myout2,..] = myfunction(a, b,…)

- Functions are **called using the name of m-file**, **not using the function name**, **but keep them the same**.
- Remember to save modifications.



15

Functions & variables

- Unlike scripts, function variables have local scope.
- Clear the work space variables (“clear”) (and observe workspace window)
- So if we repeat the “for” loop example as a function the Matlab workspace will not know anything about any variables A & B.
- But we can retrieve the answer by assignment
  - *D=myfor()* *(if we saved that function as “myfor.m”)*

16

Functions with global variables

- Global keyword allows global access to the variables.
- Declare variable global:
  - ***global x (after top line in function)***
- Check the scope of x in the workspace and in other functions.
- Need to declare x as global everywhere we need to access it as global.

17

Function & Script examples

Editor - C:\Users\Naseer\Matlab files\ADD.m*

ADD.m* | Main.m | ADD_SUB.m

```
1   function [C] = ADD( A,B )
2   %UNTITLED Summary of this function goes here
3   %   Detailed explanation goes here
4   global N
5   C= A+B;
6
7   N=N+1;
8   end
```

Function "ADD"

Editor - C:\Users\Naseer\Matlab files\ADD_SUB.m

ADD.m | Main.m | ADD_SUB.m

```
1   function [ A, S ] = ADD_SUB( AA, BB )
2   %UNTITLED7 Summary of this function goes here
3   %   Detailed explanation goes here
4   global N;
5   A= AA+BB;
6   S= AA-BB;
7   N=N+1;
8   end
```

Function "ADD_SUB"

Script "Main"

Editor - C:\Users\Naseer\Matlab files\Main.m

ADD.m | Main.m | ADD_SUB.m

```
1   clear all
2   clc
3   A=15; B=30;
4   global N
5   N=10;
6   [C,D]= ADD_SUB(A,B);
7   N=N+1;
8   CC= ADD (A,B);
9   N=N+1;
10
```

Workspace

Name	Value	Min
A	15	15
B	30	30
C	45	45
CC	45	45
D	-15	-15
N	14	14

Command Window

New to MATLAB? Watch this Video, see Examples, or read Getting Started.

```
>> Main
>>
```

18

Input/Output

- a = input(‘give me some input: ’)
- a = input(‘give me some input’, ‘s’) % string input
- a = input(‘give me some input \n’) % what ‘\n’ do?
- disp(‘Hello world’)
  - *Hello world*
- str= ‘Hello there…’
- disp(str)
  - *Hello there…*

19

Exercise (do it yourself)

- Write a function to input a name and output it after asking for a response.
- The function should only display the prompts and typed input.
- You will have to use some control structure (use ‘y’ and ‘n’).

20

Cell arrays

- Use with care.
- Each element can contain different types.
- a = { 56, ‘fifty-seven’, 8}
- Compare with:
  - *a = [56, ‘fifty-seven’, 8]*

hint: char()

21

Cell array example

C1 = {'one', 'two', 'three'; 4, 5, 6; 7,8,'nine'}

C2= C1(2:3,2:3)

C3=C1;
C3(2,1:2) = {'four','five'}

```
C1 =

    'one'    'two'    'three'
    [  4]    [  5]    [    6]
    [  7]    [  8]    'nine'


C2 =

    [5]    [   6]
    [8]    'nine'


C3 =

    'one'     'two'     'three'
    'four'    'five'    [    6]
    [   7]    [   8]    'nine'
```

22

Class

- Class(a) returned variables class or type.
- Generally variables are of type double, but can be cast.

```
a = 5
class(a) is double precision
b = uint8(a)
class(b) is 8 bit unsigned integer
```

- Some operations +- etc not applicable to certain types.

23

Variable numbers of inputs to functions

- Suppose we don't know how many inputs will be given to a function.
- Example:
  - *Optional arguments.*
  - *A list of undetermined length.*
- Use varargin after any compulsory arguments.

24

Variable numbers of inputs to functions

- a = myfunction(a, b, varargin)
- After a and b have been input (compulsory in this case) any extra inputs are put into the cell array varargin and can be used in your function.
- See “help varargin”

25

Variable numbers of inputs to functions

```
function varlist2(X,Y,varargin)
   fprintf('Total number of inputs = %d\n',nargin);

   nVarargs = length(varargin);
   fprintf('Inputs in varargin(%d):\n',nVarargs)
   for k = 1:nVarargs
      fprintf('   %d\n', varargin{k})
   end
```

Call varlist2 with more than two inputs.

```
varlist2(10,20,30,40,50)
```

```
Total number of inputs = 5
Inputs in varargin(3):
   30
   40
   50
```

26

Variable numbers of outputs from functions.

- Suppose we don't know how many outputs will be taken from a function.
- Example:
  - *General functions*
  - *Extracting the first answers from a list.*
- Use varargout after any necessary output variables.

27

Variable numbers of outputs from functions

- function [s,varargout] = myfunction(x)
- You can fill the cell array varargout with as many different variables as you like.
- When you call the function you then don't have to take all of the variables.
- [a, b, c] = myfunction(x) takes s and the first two variables in varargout.

28

Exercises (do it yourself)

- Write a Matlab function to take three numbers in and return 2.
- Test it with the wrong numbers of inputs and outputs.
- Write a Matlab function to take a least 2 inputs and return 2. (use varargin/out)
- Test it with different numbers of inputs and outputs.

29

Matlab (2015a) for images

- imfinfo(filename)
  returns a structure whose fields contain information about an image in a graphics file, filename.
- A = imread(filename)
- A = imread(filename,fmt)
  additionally specifies the format of the file.
- imwrite(A, filename, fmt);
- image(A) or imagesc(A)

30

Matlab (2015a) for sound

- audioinfo(filename)
  returns information about the contents of the audio file specified by filename.
- [y,Fs] = audioread(filename)
  reads data from the file named filename, and returns sampled data, y, and a sample rate for that data, Fs. [y,Fs] = audioread(filename, samples)
  reads the selected range of audio samples in the file, where samples is a vector of the form [start,finish].
- audiowrite(filename, y, Fs)
  writes a matrix of audio data, y, with sample rate Fs to a file called filename.
- sound(y,Fs)
  sends audio signal y to the speaker at sample rate Fs.
- **We'll use *wavread*, *wavwrite* for wav format media.**

31

Matlab (2015a) for video

- vr = VideoReader(filename)
  creates object v to read video data from the file named filename.
- f = readFrame(vr)
- vw = VideoWriter(filename, profile)
  creates a VideoWriter object and applies a set of properties tailored to a specific file format (such as 'MPEG-4' or'Uncompressed AVI').
- open(vw); writeVideo(vw,f); close(vw);
  Write the image in f to the video file.

32