Introduction to MATLAB

Gianluca Bianchin
gbian001@ucr.edu

Graduate Quantitative Methods Center
University of California, Riverside

LFSC1425 - January 25, 2018

Why MATLAB?

- Perform mathematical computation and signal (data) processing
- Analyzing and visualizing data
- Model and simulate physical systems
- Testing your hypothesis / design

What is MATLAB

- MATLAB = Matrix Laboratory, developed by MathWorks
- Is a numerical computing environment
- Includes a proprietary programming language
- Includes optional toolboxes for specific applications
- Includes Simulink package, for simulation of dynamical and embedded systems and a number of toolboxes developed for specific applications (Computer Vision, SimBiology, Econometrics, ...)
- MathWorks introduced great features for integration with R, C++, LaTeX, ...

Outline

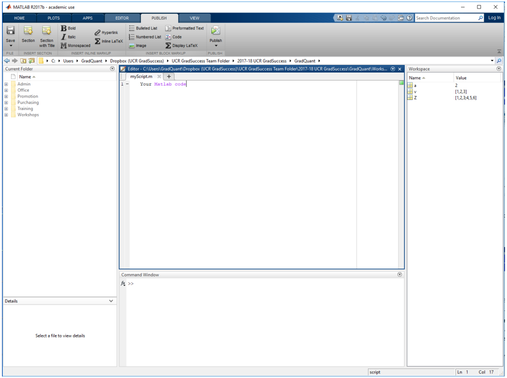
1. Using the Command Window

2. Storing and processing data: Arrays and Matrices

3. Processing data and arrays

4. Saving and Loading data

Desktop interface



MATLAB R2017b - academic use

C: ▸ Users ▸ GradQuant ▸ Dropbox (UCR GradSuccess) ▸ UCR GradSuccess Team Folder ▸ 2017-18 UCR GradSuccess ▸ GradQuant ▸

Admin
Office
Promotion
Purchasing
Training
Workshops

Select a file to view details

Editor - C:\Users\GradQuant\Dropbox (UCR GradSuccess)\UCR GradSuccess Team Folder\2017-18 UCR GradSuccess\GradQuant\Works...

```
Your Matlab code
```

```
>>
```

Name	Value
a	2
v	[1,2,3]
Z	[1,2,3;4,5,6]

script Ln 1 Col 17

Figure: Matlab interface.

- Current folder
- Editor
- Workspace
- Command window

Using the Command Window

Operations, operators, and variables

- Operations: `3*2; 5*2^3+4*(3);`
- Operators: `+, -, *, /, ^`
- Variables assignment:
  `a = 3; b = 2; c = a*b; month = 'August';`
- Variables can be visualized in the correspondent section
- Some notes:
  1. No need to define variable types!!
  2. Variable names must begin with a letter
  3. Case sensitive
  4. Avoid names that correspond to functions

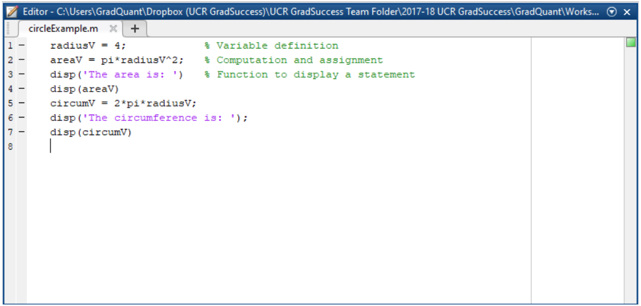
Pre-defined variables and functions

- Variables: `pi`=3.14159, `i = j =`$\sqrt{-1}$, `inf`, `NaN`
- Functions:
  `sqrt(x)`, `sin(x)`, `cos(x)`, `tan(x)`, `exp(x)`, `log(x)`
  `round(x)`, `floor(x)`, `ceil(x)`, `...`
- To obtain function description: `help 'functionName'` or click "help" from the toolbar

Script files

A script file is a collection of MATLAB commands that are executed in sequence

- Extension .m
- Click on the new script icon
- To run: Hit the green arrow in the toolbar



Editor - C:\Users\GradQuant\Dropbox (UCR GradSuccess)\UCR GradSuccess Team Folder\2017-18 UCR GradSuccess\GradQuant\Works...

```
radiusV = 4;            % Variable definition
areaV = pi*radiusV^2;   % Computation and assignment
disp('The area is: ')   % Function to display a statement
disp(areaV)
circumV = 2*pi*radiusV;
disp('The circumference is: ');
disp(circumV)
```

Figure: A script is a collection of commands.

Your First Script file

Write a script that computes the area and circumference of a circle and displays them as an output

```
radiusV = 4;                % Variable definition
areaV = pi*radiusV^2;       % Computation and assignment
disp('The area is: ')       % Function to display a statement
disp(areaV)
circumV = 2*pi*radiusV;
disp('The circumference is: ');
disp(circumV)
```

- Semicolon ; is used to drop output

Your First Script file

Write a script that computes the area and circumference of a circle and displays them as an output

```
radiusV = 4;            % Variable definition
areaV = pi*radiusV^2;   % Computation and assignment
disp('The area is: ')   % Function to display a statement
disp(areaV)
circumV = 2*pi*radiusV;
disp('The circumference is: ');
disp(circumV)
```

- Semicolon ; is used to drop output

Some useful notes

- Matlab is an interpreted language (no need to compile)
- When you pick a name for a script you must follow the same rules as for naming variables.
- The script file must be in your current directory
- Begin your script with `clear`, `close all`, `clc`
- Debug button, Publish button

Storing and processing data: Arrays and Matrices

Arrays (and Matrices)

Matrices and Arrays are extremely important because they are used to store data we would like to process, visualize, ...

- While other programming languages mostly work with one number at a time, MATLAB is designed to operate primarily on whole matrices and arrays
- A matrix is a two-dimensional array often used for linear algebra

$$v = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} \qquad w = \begin{bmatrix} 2 & -5 & .9 \end{bmatrix} \qquad M = \begin{bmatrix} 3 & 1 & 0 \\ 2 & 3 & 1 \\ 4 & 2 & 3 \end{bmatrix}$$

Arrays definition and addressing

- To create an array with four elements in a single row, separate the elements with either a comma (,) or a space

  ```
  v = [1 2 3 4]
  ```

- To create a multidimensional array, separate the rows with semicolons (;)

  ```
  M = [3 1 0; 2 3 1; 4 2 3]
  ```

- To address entries of an array:

  ```
  v(1), v(3:end), M(1:2,:)
  ```

- Colon (:) indicates an interval of indexes
- Entries can be removed from an array

  ```
  v(3)= []
  ```

Arrays definition

- Another way to create a matrix is to use a function, such as `ones(), zeros(), or rand()`
- `v = rand(1,4)` creates a 1×4 array with random numbers within $[0,1]$
- To create a vector with values from 1 to 10 and spaced by 0.01 `t = 1:.01:10;`

Operations on Arrays

MATLAB allows you to process all of the values in a matrix using a single arithmetic operator or function.

- `v + 10, v'`
- Function computed over vectors return a vector with the output `sqrt(v), sin(v), exp(v), ...`
- Matrix multiplication MN: `M*N`
- Matrix power M^2: `Mˆ2`
- Element-wise multiplication $M \circ N$: `M.*N`
- Concatenation: `V = [v, v], V = [v;v]`

Functions on Arrays

- Some useful functions for arrays:
  `max(), min()`
  `mean(), median(), cov(), var()`
  `sum(), diff(), cumsum()`
- Sorting: `sort(v)`
- Find: `find(v==3), find(v>1)`
- Size: `length(v), size(M)`

Hands on: Room temperature

Assume we the average temperature in a room during a day (8am-8pm) is $75°$F and that the thermostat introduces Gaussian noise ($\mu = 0$, $\sigma = 1$). We are requested to describe this data, display the max and min temperature, and plot the temperature over time.

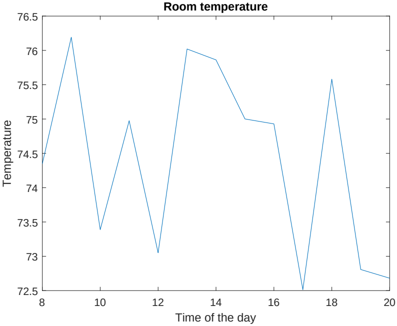


Figure: Room temperature over time.

Hands on: Room temperature (code)

Assume the average temperature in a room during a day (8am-8pm) is 75°F and that the thermostat introduces Gaussian noise ($\mu = 0$, $\sigma = 1$). We are requested to model this system, generate temperature data, display the max and min temperature, and plot the temperature over time.

```
time = 8:1:20;                % Use 24 hours format

noHours = length(time);       % Length of the interval considered

rTemp = 75*ones(1,noHours) + randn(1,noHours); % 75F + noise

disp('The max temperature is: ');
disp(max(rTemp))

disp('The min temperature is: ');
disp(min(rTemp))

figure; plot(time,rTemp)      % Plot
```

Hands on: Room temperature (code)

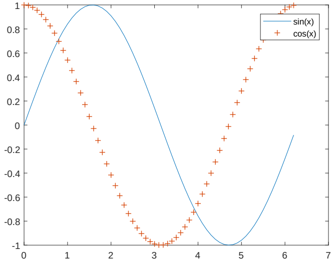
Assume the average temperature in a room during a day (8am-8pm) is $75°$F and that the thermostat introduces Gaussian noise ($\mu = 0$, $\sigma = 1$). We are requested to model this system, generate temperature data, display the max and min temperature, and plot the temperature over time.

```
time = 8:1:20;              % Use 24 hours format

noHours = length(time);     % Length of the interval considered

rTemp = 75*ones(1,noHours) + randn(1,noHours); % 75F + noise

disp('The max temperature is: ');
disp(max(rTemp))

disp('The min temperature is: ');
disp(min(rTemp))

figure; plot(time,rTemp)    % Plot
```

Plotting

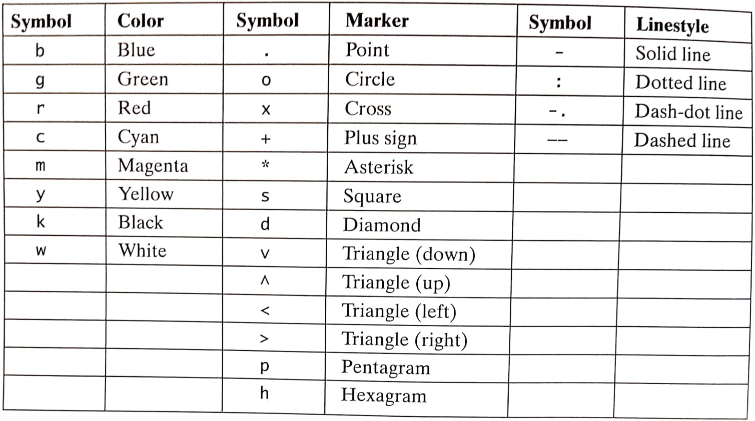
- Two-dimensional line and points are created with the command `plot(xdata,ydata)`

```
x = 0 : 0.1 : 2*pi;      % xdata
y1 = sin(x);             % first set of ydata
y2 = cos(x);             % second set of ydata
plot(x, y1, '-')
hold on                  % to plot on the same figure
plot(x, y2, '+')
```



Linestyle, markers, and colors

We can specify our own colors, markers, and linestyles by giving plot a third argument



Symbol	Color	Symbol	Marker	Symbol	Linestyle
b	Blue	.	Point	-	Solid line
g	Green	o	Circle	:	Dotted line
r	Red	x	Cross	-.	Dash-dot line
c	Cyan	+	Plus sign	--	Dashed line
m	Magenta	*	Asterisk		
y	Yellow	s	Square		
k	Black	d	Diamond		
w	White	v	Triangle (down)		
		^	Triangle (up)		
		<	Triangle (left)		
		>	Triangle (right)		
		p	Pentagram		
		h	Hexagram		

```
plot(x,y,'b:p'), plot(x,y,'c-'), plot(x,y,'m+')
```

Useful commands when plotting

- `hold on`: allows multiple plots on the same figure
- `grid on`: displays a grid in the background
- `axis([xmin xmax ymin ymax])`: sets axis limits
- `xlabel('')`: renames x axis
- `ylabel('')`: renames y axis
- `title('')`: sets a title for the figure
- `legend('')`: sets a legend for the figure

Plotting: Title, axis, and legend

- For example:

```
xlabel('x (radians)');               %  label the x-axis
ylabel('sine and cosine function');  %  label the y-axis
title('sin(x), cos(x)');
legend('sin(x)', 'cos(x)')
```

- Advanced formatting... in next MATLAB workshop!
  1. Advanced legends and adjust axis
  2. Adjust axis font (LaTeX) and size
  3. 3D plotting
  4. Axis scaling, logarithmic axis

Plotting: pie chart

We can create a pie chart:

```
a = [.5 1 1.6 1.2 .8 2.1];
pie(a,a==max(a))
title('Example pie chart')
```

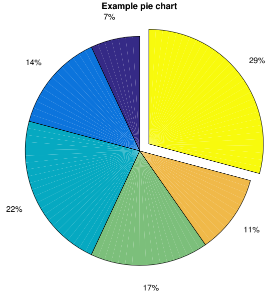


Figure: Pie chart created with code above.

Plotting: histogram

We can create a histogram with:

```
x = -4:1:4;
y = [.1 .26 .31 .4 .5 .45 .38 .21 .1];
bar(x,y)
```

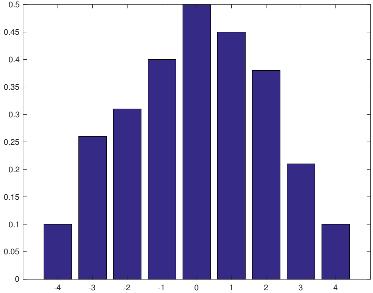


Figure: Histogram created with code above.

Processing data and arrays

Conditional Statements

- There are times when you want certain parts of your program to be executed only in limited circumstances

```
if (condition)
      (matlab commands)
end
```

- For instance:

```
a = 2;
b = 3;
if (a<b)
    j = -1;
end
```

Conditional Statements

- `condition` represents true and false through a logic expression.
- The logical operators are
  1. `==,~=`
  2. `<,>, >=, <=`
- To combine more than one conditional statement, use
  1. `&&, ||`

- Other useful operators are
  1. `all(condition)`: is the condition satisfied for all the elements?
  2. `any(condition)`: is the condition satisfied for any of the elements?
  3. `find(condition)`: find the elements that satisfy the condition
  4. `isempty(input)`: is the input a empty matrix ?
  5. `ischar(input)`: is the input a vector of characters?
  6. ...

Conditional Statements

- More complicated structures are also possible, including combinations like the following:

```
if  (condition statement)
    (matlab commands)
elseif  (condition statement)
    (matlab commands)
elseif (condition statement)
    (matlab commands)
.
.
.
else
    (matlab commands)
end
```

FOR loop

Loops are useful when we need to execute a certain operation to every entry of an array, or we have to execute a certain operation depending on the value of each entry of an array.

- Example: erase (set to zero) the entries greater than 3 in the following array: $v = [1\ 2\ 7\ 9\ 1\ 4\ 1]$
- Expected output: $v = [1\ 2\ 0\ 0\ 1\ 0\ 1]$

FOR loop: Syntax

- Loop a specific number of times, and keep track of each iteration with an incrementing index variable
- Syntax:

```
for index = values
    (statements)
end
```

FOR loop: Example

- Example: erase (set to zero) the entries greater than 3 in the following array: $v = [1\ 2\ 7\ 9\ 1\ 4\ 1]$
- Expected output: $v = [1\ 2\ 0\ 0\ 1\ 0\ 1]$
- Code:

```
v = [1 2 7 9 1 4 1];
arrayLen = length(v);
for i=1:arrayLen
    if v(i)>3
        v(i)=0;
    end
end
```

WHILE loop

- Loop as long as a condition remains true
- With respect to `for` loop we may not know the number if iterations at the beginning
- Example: generate random numbers (between 1 and 5) until number 1 appears
- Syntax:

```
while (expression)
    (statements)
end
```

Loops

Tip

If you inadvertently create an infinite loop (a loop that never ends on its own), stop execution of the loop by pressing Ctrl+C.

Saving and Loading data

Save and Load workspace

Variables in the workspace can be saved in a format naive to MATLAB, with extension .mat

- The `save` command can be used to store variables

  ```
  save filename
  save filename var1 var2 var 3
  ```

- To load a file use the `load` command `load filename`

  ```
  load filename var1 var2 var 3
  ```

Importing and exporting data

MATLAB includes functions to import and export data from most of the common file extensions

- `textread` reads formatted text from .txt file
- `xlsread` reads from Excel file
- `xlswrite` writes an Excel file

For example:

```
[NUM,TXT,RAW]=xlsread('myDataExcel.xlsx');
xlswrite('tmp.xls',NUM);
```

In “Advanced MATLAB.“.. On Feb 22, 2018

- Multidymensional Arrays
- Functions
- Files and I/O
- Try-Catch
- Cell Arrays
- Structures
- Data interpolation(310), integration and differentiation
- Symbolic manipulator
- Linear algebra & systems of equations
- Extrapolation and Regression
- Plot Formatting

GradQuat at UCR

GradQuant:

- Website: http://gradquant.ucr.edu
- Hours: Monday Thursday, 9 am - 3 pm
- Location: Life Sciences Building, Room #1425

If you seek help with MATLAB:

- Drop-in hours (Gianluca): Thursday 12pm-2pm
- Schedule a consultation (Gianluca)
- Email: GQstaff1@ucr.edu

MATLAB resources:
http://gradquant.ucr.edu/gq-calendar/workshop-resources/