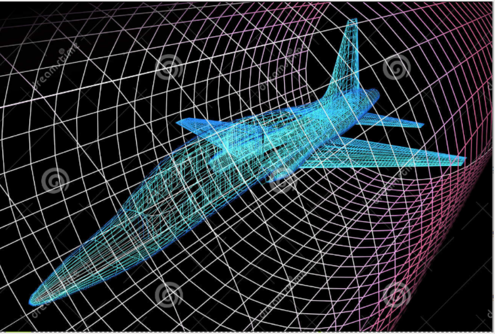
MATLAB

Fourth Generation Programming Language



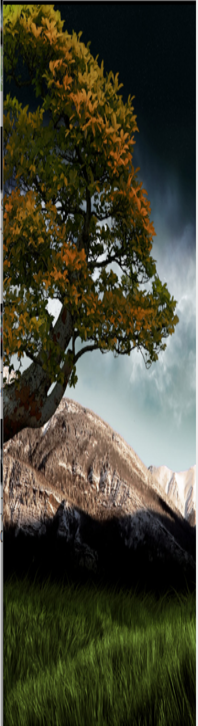
QAZI EJAZ UR REHMAN

Avionics Engineer

Graduate Research Assistant
Aeronautics & Astronautics Department
Institute of Space Technology
Islamabad

February 10, 2016

ist Institute of Space Technology



Outline (Lec 01)

Institute of Space Technology

1. Start
2. Matlab Basics
3. Scripts: Overview
4. Arrays
5. Practical Problems
6. Built in
7. indexing
8. Plotting

Course Layout

Institute of
Space Technology

- Lectures
  - Variables, Scripts and Operations
  - Visualization and Programming
  - Solving Equations, Fitting
  - Images, Animations, Advanced Methods
  - Optional: Symbolic Math, Simulink
- Archimedes and π approximation
- Computational Fluid Dynamics

Course Layout

- Lectures
  - Variables, Scripts and Operations
  - Visualization and Programming
  - Solving Equations, Fitting
  - Images, Animations, Advanced Methods
  - Optional: Symbolic Math, Simulink
- Archimedes and π approximation
- Computational Fluid Dynamics

Course Layout

- Lectures
  - Variables, Scripts and Operations
  - Visualization and Programming
  - Solving Equations, Fitting
  - Images, Animations, Advanced Methods
  - Optional: Symbolic Math, Simulink
- Archimedes and π approximation
- Computational Fluid Dynamics

Course Layout

- Lectures
  - Variables, Scripts and Operations
  - Visualization and Programming
  - Solving Equations, Fitting
  - Images, Animations, Advanced Methods
  - Optional: Symbolic Math, Simulink
- Archimedes and π approximation
- Computational Fluid Dynamics

Course Layout

- Lectures
  - Variables, Scripts and Operations
  - Visualization and Programming
  - Solving Equations, Fitting
  - Images, Animations, Advanced Methods
  - Optional: Symbolic Math, Simulink
- Archimedes and π approximation
- Computational Fluid Dynamics

Course Layout

- Lectures
  - Variables, Scripts and Operations
  - Visualization and Programming
  - Solving Equations, Fitting
  - Images, Animations, Advanced Methods
  - Optional: Symbolic Math, Simulink
- Archimedes and π approximation
- Computational Fluid Dynamics

Course Layout

- Lectures
  - Variables, Scripts and Operations
  - Visualization and Programming
  - Solving Equations, Fitting
  - Images, Animations, Advanced Methods
  - Optional: Symbolic Math, Simulink
- Archimedes and π approximation
- Computational Fluid Dynamics

Course Layout

- Lectures
  - Variables, Scripts and Operations
  - Visualization and Programming
  - Solving Equations, Fitting
  - Images, Animations, Advanced Methods
  - Optional: Symbolic Math, Simulink
- Archimedes and π approximation
- Computational Fluid Dynamics

Course Layout

- Problem Sets / Office Hours
  - One per day, should take about 3 hours to do
  - Submit doc or pdf (include code, figures)
  - No set office hours but available by email
- Requirements for passing
  - Attend all lectures
  - Complete all problem sets (-, $\surd$, +)
- Prerequisites
  - Basic familiarity with programming
  - Basic linear algebra, differential equations, and probability

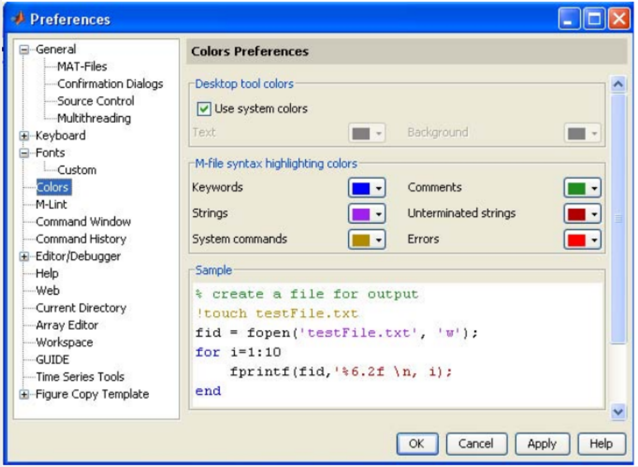
Making Folders

Institute of Space Technology

- Use folders to keep your programs organized
- Folder making
    - Type mkdir foldername in command window (e.g mkdir aero)
- Access that folder by
    - Typing cd foldername (e.g. cd aero)
- Inquiry of Content of Folder
    - type ls, dir
    - pwd command will tell you where are you
    - what will list categorized folder contents

Customization

- File → Preferences
  - Allows you personalize your MATLAB experience



Colors Preferences

Desktop tool colors

☑ Use system colors

Text ▨ Background ▨

M-file syntax highlighting colors

Keywords 🟦 Comments 🟩
Strings 🟪 Unterminated strings 🟥
System commands 🟨 Errors 🟥

Sample

```
% create a file for output
!touch testFile.txt
fid = fopen('testFile.txt', 'w');
for i=1:10
    fprintf(fid,'%6.2f \n, i);
end
```

MATLAB Basics

Institute of Space Technology

- MATLAB can be thought of as a super-powerful graphing calculator
  - Remember the TI-83 from calculus?
  - With many more buttons (built-in functions)
- In addition it is a programming language
  - MATLAB is an interpreted language, like Java
  - Commands executed line by line

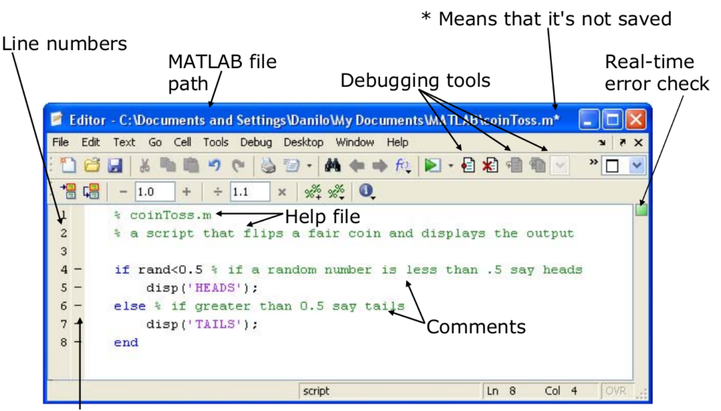
Help/Docs

- ≫ help
  - The most important function for learning MATLAB on your own
- To get info on how to use a function:
  ≫ help sin
  - Help lists related functions at the bottom and links to the doc
- To get a nicer version of help with examples and easy-to-read descriptions:
  ≫ doc sin
- To search for a function by specifying keywords:
  ≫ doc + search

Scripts: Overview

- Scripts are
  - collection of commands executed in sequence
  - written in the MATLAB editor
  - saved as MATLAB files (.m extension)
- To create an MATLAB file from command-line
  - edit helloWorld.m

Scripts: the Editor


* Means that it's not saved
Line numbers
MATLAB file path
Debugging tools
Real-time error check
Editor - C:\Documents and Settings\Danilo\My Documents\MATLAB\coinToss.m*
File Edit Text Go Cell Tools Debug Desktop Window Help
1.0
1.1
Help file
Comments
script
Ln 8
Col 4
OVR

```
% coinToss.m
% a script that flips a fair coin and displays the output

if rand<0.5 % if a random number is less than .5 say heads
    disp('HEADS');
else % if greater than 0.5 say tails
    disp('TAILS');
end
```

Scripts: Some Notes

- **COMMENT!**
  - Anything following a % is seen as a comment
  - The first contiguous comment becomes the script's help file
  - Comment thoroughly to avoid wasting time later
- Note that scripts are somewhat static, since there is no input and no explicit output
- All variables created and modified in a script exist in the workspace even after it has stopped running

Scripts: Exercise

Make a helloWorld script

- When run, the script should display the following text:

  Hello World!
  I am going to learn MATLAB!

- Hint: use disp to display strings. Strings are written between single quotes, like 'This is a string'

Scripts: Exercise

Make a helloWorld script

- When run, the script should display the following text:

  Hello World!
  I am going to learn MATLAB!

- Hint: use disp to display strings. Strings are written between single quotes, like 'This is a string'
- Open the editor and save a script as helloWorld.m. This is an easy script, containing two lines of code:

```
% helloWorld.m
% my first hello world program in MATLAB

disp('Hello World!');
disp('I am going to learn MATLAB!');
```

Variable Types

- MATLAB is a weakly typed language
  - No need to initialize variables!
- MATLAB supports various types, the most often used are

  ≫ 3.84
  - 64-bit double (default)
- ≫ 'a'
  - 16-bit char
- Most variables you'll deal with will be vectors or matrices of doubles or chars
- Other types are also supported: complex, symbolic, 16-bit and 8 bit integers, etc. You will be exposed to all these types through the homework

Naming variables

- To create a variable, simply assign a value to a name:
  ```
  ≫ var1=3.14
  ≫ myString='hello world'
  ```
- Variable names
  - first character must be a LETTER
  - after that, any combination of letters, numbers and _
  - CASE SENSITIVE! (var1 is different from Var1)
- Built-in variables. Dont use these names!
  - i and j can be used to indicate complex numbers
  - pi has the value 3.1415926...
  - ans stores the last unassigned value (like on a calculator)
  - Inf and -Inf are positive and negative infinity
  - NaN represents 'Not a Number'

Naming variables

- A variable can be given a value explicitly

  ```
  ≫ a = 10
  ```

  - shows up in workspace!
- Or as a function of explicit values and existing variables

  ```
  ≫ c = 1.3*45-2*a
  ```

- To suppress output, end the line with a semicolon

  ```
  ≫ cooldude = 13/3;
  ```

Arrays

- Like other programming languages, arrays are an important part of MATLAB
- Two types of arrays

    1. matrix of numbers (either double or complex)

    cell array of objects (more advanced data structure)

Column and Row vector

Institute of Space Technology

- Row vector: comma or space separated values between brackets

```
≫ row = [1 2 5.4 -6.6]
≫ row = [1, 2, 5.4, -6.6];
```

- Column vector: semicolon separated values between brackets

```
column = [4;2;7;4]
```

size & length

- You can tell the difference between a row and a column vector by:
  - Looking in the workspace
  - Displaying the variable in the command window
  - Using the size function
- To get a vector's length, use the length function

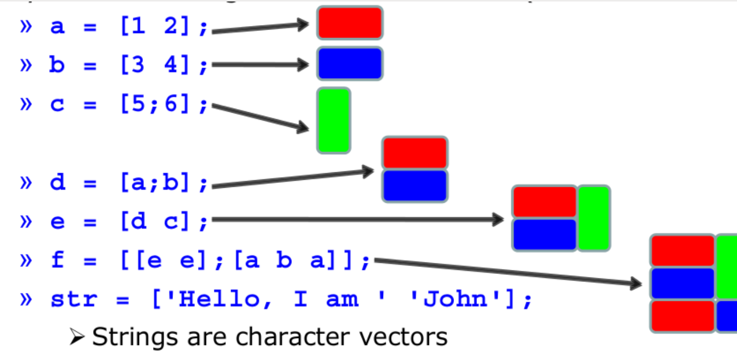
Matrices

- Make matrices like vectors

Element by element

- $\gg$a= [1 2;3 4]; $= \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$

- By concatenating vectors or matrices (dimension matters)



```
» a = [1 2];
» b = [3 4];
» c = [5;6];

» d = [a;b];
» e = [d c];
» f = [[e e];[a b a]];
» str = ['Hello, I am ' 'John'];
```

➢ Strings are character vectors

save/clear/load

- Use save to save variables to a file
  - ≫ save myFile a b
    - saves variables a and b to the file myfile.mat
    - myfile.mat file is saved in the current directory
    - Default working directory is
- Use clear to remove variables from environment
  - ≫ clear a b
    - look at workspace, the variables a and b are gone
- Use load to load variable bindings into the environment
  - ≫ load myFile
    - look at workspace, the variables a and b are back
- Can do the same for entire environment
  - ≫ save myenv; clear all; load myenv;

Exercise: Variables

Institute of Space Technology

Get and save the current date and time

- Create a variable start using the function clock
- What is the size of start? Is it a row or column?
- What does start contain? See help clock
- Convert the vector start to a string. Use the function datestr and name the new variable startString
- Save start and startString into a mat file named startTime

Exercise: Variables

Institute of Space Technology

Get and save the current date and time

- Create a variable start using the function clock
- What is the size of start? Is it a row or column?
- What does start contain? See help clock
- Convert the vector start to a string. Use the function datestr and name the new variable startString
- Save start and startString into a mat file named startTime

```
doc clock
start=clock;
size(start)
doc datestr
startString=datestr(start);
save startTime start startString
```

Exercise: Variables

Institute of Space Technology

Read in and display the current date and time

- In helloWorld.m, read in the variables you just saved using load
- Display the following text:

  I started learning MATLAB on *start date and time*
- Hint: use the disp command again, and remember that strings are just vectors of characters so you can join two strings by making a row vector with the two strings as sub- vectors.

```
≫ load startTime
≫ disp(['I started learning MATLAB on ' ... startString]);
```

Exercise: Variables

Institute of Space Technology

You will learn MATLAB at an exponential rate! Add the following to your helloWorld script:

- Your learning time constant is 1.5 days. Calculate the number of **seconds** in 1.5 days and name this variable tau
- This class lasts 5 days. Calculate the number of seconds in 5 days and name this variable endOfClass
- This equation describes your knowledge as a function of time t:

$$k = 1 - e^{\frac{-t}{\tau}}$$

- How well will you know MATLAB at endOfClass? Name this variable knowledgeAtEnd. (use exp)
- Using the value of knowledgeAtEnd, display the phrase:

  At the end of 6.094, I will know X% of MATLAB
- Hint: to convert a number to a string, use num2str

Exercise: Variables

```
secPerDay=60*60*24;
tau=1.5*secPerDay;
endOfClass=5*secPerDay
knowledgeAtEnd=1-exp(-endOfClass/tau);
disp(['At the end of 6.094, I will know ' ...
num2str(knowledgeAtEnd*100) '% of MATLAB'])
```

[1]

[1] Transpose, Addition and Subtraction, Element-Wise Functions

Exercise: Vector Operations

Institute of
Space Technology

Calculate how many seconds elapsed since the start of class

- In helloWorld.m, make variables called secPerMin, secPerHour, secPerDay, secPerMonth (assume 30.5 days per month), and secPerYear (12 months in year), which have the number of seconds in each time period.
- Assemble a row vector called secondConversion that has elements in this order: secPerYear, secPerMonth, secPerDay, secPerHour, secPerMinute, 1
- TMake a currentTime vector by using clock
- Compute elapsedTime by subtracting currentTime from start
- Compute t (the elapsed time in seconds) by taking the dot product of secondConversion and elapsedTime (transpose one of them to get the dimensions right)

Exercise: Vector Operations

```
secPerMin=60;
secPerHour=60*secPerMin;
secPerDay=24*secPerHour;
secPerMonth=30.5*secPerDay;
secPerYear=12*secPerMonth;
secondConversion=[secPerYear
   secPerMonth ...
secPerDay secPerHour secPerMin 1];
currentTime=clock;
elapsedTime=currentTime-start;
t=secondConversion*elapsedTime';
```

Exercise: Vector Operations

Display the current state of your knowledge

- Calculate currentKnowledgeusing the same relationship as before, and the t we just calculated:

$$k = 1 - e^{-t/\tau}$$

- Display the following text:

At this time, I know X% of MATLAB

Exercise: Vector Operations

Institute of Space Technology

Display the current state of your knowledge

- Calculate currentKnowledgeusing the same relationship as before, and the t we just calculated:

$$k = 1 - e^{-t/\tau}$$

- Display the following text:

At this time, I know X% of MATLAB

```
≫ tVec = linspace(0,endOfClass,10000);
≫ knowledgeVec=1-exp(-tVec/tau);
```

Automatic Initialization

- Initialize a vector of ones, zeros, or random numbers

  ≫ o=ones(1,10)

  - row vector with 10 elements, all 1

  ≫ z=zeros(23,1)

  - column vector with 23 elements, all 0

  ≫ r=rand(1,45)

  - column vector with 23 elements, all 0

  ≫ n=nan(1,69)

  - row vector of NaNs (useful for representing uninitialized variables)

The general function call is:

```
var=zeros(M,N);
```

Automatic Initialization

- To initialize a linear vector of values use linspace
  - ≫ a=linspace(0,10,5)
    - starts at 0, ends at 10 (inclusive), 5 values
- Can also use colon operator (:)
  - ≫ b=0:2:10
    - starts at 0, increments by 2, and ends at or before 10
    - increment can be decimal or negative
  - ≫ c=1:5
    - if increment isnt specified, default is 1
- To initialize logarithmically spaced values use logspace
    - similar to linspace, but see help

Exercise: Vector Functions

Calculate your learning trajectory

- In helloWorld.m, make a linear time vector tVec that has 10,000 samples between 0 and endOfClass
- Calculate the value of your knowledge (call it textcolorblueknowledgeVec) at each of these time points using the same equation as before:

$$k = 1 - e^{-t/\tau}$$

Exercise: Vector Functions

Institute of Space Technology

Calculate your learning trajectory

- In helloWorld.m, make a linear time vector tVec that has 10,000 samples between 0 and endOfClass
- Calculate the value of your knowledge (call it textcolorblueknowledgeVec) at each of these time points using the same equation as before:

$$k = 1 - e^{-t/\tau}$$

```
≫ tVec = linspace(0,endOfClass,10000);
≫ knowledgeVec=1-exp(-tVec/tau);
```

Vector Indexing

Institute of Space Technology

- MATLAB indexing starts with **1**, not **0**
  - We will not respond to any emails where this is the problem.
- a(n) returns the nth element

$$a = [13\ 5\ 9\ 10]$$

a(1) a(2) a(3) a(4)

- The index argument can be a vector. In this case, each element is looked up individually, and returned as a vector of the same size as the index vector.

$\gg$ x=[12 13 5 8];

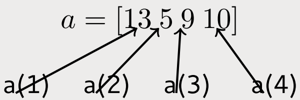
$\gg$a=x(2:3); $\longrightarrow$ a=[13 5];

$\gg$b=x(1:end-1); $\longrightarrow$b=[12 13 5];

Start
Matlab Basics
Help
Scripts: Overview
Exercise
Arrays
Practical Problems
Exercise
Built in
Exercise
indexing
Exercise
Plotting
Exercise

Vector Indexing

- MATLAB indexing starts with **1**, not **0**
  - We will not respond to any emails where this is the problem.
- a(n) returns the nth element



$$a = [13\ 5\ 9\ 10]$$

- The index argument can be a vector. In this case, each element is looked up individually, and returned as a vector of the same size as the index vector.

≫ x=[12 13 5 8];

≫a=x(2:3); $\longrightarrow$ a=[13 5];

≫b=x(1:end-1); $\longrightarrow$ b=[12 13 5];

Advanced Indexing 1

Calculate your learning trajectory

- To select rows or columns of a matrix, use the :

$$c=\begin{bmatrix} 12 & 5 \\ -1 & 13 \end{bmatrix}$$

```
d=c(1,:);  ——————————→ d=[12 5];
e=c(:,2);  ——————————→ e=[5;13];
c(2,:)=[3 6]; %replaces second row of c
```

Advanced Indexing 2

Institute of Space Technology

- MATLAB contains functions to help you find desired values within a vector or matrix

  ≫ vec = [5 3 1 9 7]
- To get the minimum value and its index:

  ≫ [minVal,minInd] = min(vec);
  - max works the same way
- To find any the indices of specific values or ranges

  ≫ ind = find(vec == 9);

  ≫ ind = find(vec ¿ 2 & vec ¡ 6);
  - find expressions can be very complex, more on this later
- To convert between subscripts and indices, use ind2sub, and sub2ind. Look up help to see how to use them.

Start

Matlab Basics
Help

Scripts: Overview
Exercise

Arrays

Practical Problems
Exercise

Built in
Exercise

indexing
Exercise

Plotting
Exercise

Exercise: Indexing

When will you know 50% of MATLAB?

- First, find the index where knowledgeVec is closest to 0.5. Mathematically, what you want is the index where the value of —KNOWLEDGEVEC 0.5— is at a minimum (use abs and min).
- Next, use that index to look up the corresponding time in tVec and name this time halfTime.
- Finally, display the string: I will know half of MATLAB after X days Convert halfTime to days by using secPerDay

Start
Matlab Basics
Help
Scripts: Overview
Exercise
Arrays
Practical Problems
Exercise
Built in
Exercise
indexing
Exercise
Plotting
Exercise

Exercise: Indexing

When will you know 50% of MATLAB?

- First, find the index where knowledgeVec is closest to 0.5. Mathematically, what you want is the index where the value of —KNOWLEDGEVEC 0.5— is at a minimum (use abs and min).
- Next, use that index to look up the corresponding time in tVec and name this time halfTime.
- Finally, display the string: I will know half of MATLAB after X days Convert halfTime to days by using secPerDay

```
≫ [val,ind]=min(abs(knowledgeVec-0.5));
≫ halfTime=tVec(ind);
≫ disp(['I will know half of MATLAB after ' ...
≫ num2str(halfTime/secPerDay) ' days']);
```

Plotting

- Example
    - x=linspace(0,4*pi,10);
    - y=sin(x);
- Plot values against their index
    - plot(y);
- Usually we want to plot y versus x
    - plot(x,y);

What does plot do?

- plot generates dots at each (x,y) pair and then connects the dots with a line
- To make plot of a function look smoother, evaluate at more points

```
≫ x=linspace(0,4*pi,1000);
≫ plot(x,sin(x));
```

- x and y vectors must be same size or else youll get an error

```
≫ plot([1 2], [1 2 3])
```

Start

Matlab Basics
Help

Scripts: Overview
Exercise

Arrays

Practical Problems
Exercise

Built in
Exercise

indexing
Exercise

Plotting
Exercise

Exercise: Plotting

Plot the learning trajectory

- In helloWorld.m, open a new figure (use figure)
- Plot the knowledge trajectory using tVec and knowledgeVec. When plotting, convert tVec to days by using secPerDay
- Zoom in on the plot to verify that halfTime was calculated correctly

Exercise: Plotting [2]

Plot the learning trajectory

- In helloWorld.m, open a new figure (use figure)
- Plot the knowledge trajectory using tVec and knowledgeVec. When plotting, convert tVec to days by using secPerDay
- Zoom in on the plot to verify that halfTime was calculated correctly

```
≫figure
≫ plot(tVec/secPerDay, knowledgeVec);
```

[2] ▸ For further plotting options click here