Week 2 MATLAB Activities

Contents

 No AI use

Time required: 90 minutes

How to Create Screenshots: Please use the Snipping Tool. Paste a screenshot of just the program you are working on. If you are snipping a virtual machine, make sure your focus is outside the virtual machine before you snip.

1. Press and hold down the **Windows key** & **Shift**, then type **S.** This brings up the on-screen snipping tool.
2. Click and Drag your mouse around whatever you want to snip.
3. Release the mouse button. This places the snip into the Windows Clipboard.

Go into a blank Word document or wherever you want to paste the snip. Hold down **CTRL**, then type **V** to paste the snip.

Reading

Matlab A Practical Introduction to Programming and Problem Solving (Stormy Attaway)

Sections 1.5, 1.6, 1.7, 1.8

MATLAB Assignment Script

1. Create a MATLAB script named **Wk02Lastname.m**
2. Save all programs in this script.
3. Include your name and date at the top of the script file as comments.
4. Put a Section Break between each program.

Tutorial 1: fprintf

The MATLAB fprintf function allows for much more precise formatting of numbers.

```
% fprintf example with string and float
firstName = "Bob";
age = 45;
fprintf("%s is %.0f years old.\n", firstName, age)
```

Example run:

```
Bob is 45 years old.
```

There is more involved in the function **fprintf** in order to produce the output we want. Let's look at the individual pieces of this example.

- **%** signs are NOT for comments this time (note: they aren't green), they are now used as place holders for the data.
- **%s** indicates that a string is expected as input.
- **%f** indicates that a double is expected.
- **%d** indicates that an integer is expected.
- **%.2f** indicates that we expect a floating point numeric input and display 2 decimal points.
- **\n** is an "\escape character" that creates a New Line.
- **firstName** and **age** are listed after this, separated by commas.
- When the command is run, Matlab places the first data value, **firstName** (i.e. Bob), in the **%s** position and the second data value, **age** (i.e. 25), in the **%.0f** position. Got it?

Hello World with the **fprintf** function. **fprintf** needs a **\n** new line character to move to the next line.

```
% Display a simple message
fprintf("Hello, MATLAB!\n");
fprintf("Time to code.\n");
```

Example run:

```
Hello, MATLAB!
Time to code.
```

An example of formatting integer numbers.

```
% Display integer values
x = 5;
y = 10;
fprintf("The values are x = %d and y = %d.\n", x, y);
```

Example run:

```
The values are x = 5 and y = 10.
```

1. The code defines two numeric variables, x and y, with values 5 and 10, respectively.
2. **fprintf** displays a message containing these numeric values.
3. **%** is the placeholder for the variable.
4. **d** formats integers.
5. The format specifier **%d** is used to indicate that the corresponding variables (x and y) should be treated as integers in the formatted output.

An example using floating point (decimal) numbers.

```
% Default floating-point
fprintf("Default: %f\n", pi);

% Scientific notation
fprintf("Scientific: %e\n", pi);

% Fixed-point with 2 decimal places
fprintf("Fixed-point: %.2f\n", pi);
```

Example run:

```
Default: 3.141593
Scientific: 3.141593e+00
Fixed-point: 3.14
```

- **%f** specifier is used to represent a floating-point number.
- **%e** specifier is used to display the value of pi in scientific notation.
- **.2f** is fixed-point notation and limits it to two decimal places.

You can also have multiple values. The order of values corresponds to the % symbol.

```
% Multiple values can be arranged in any order
started = 3;
gallons = 4;
miles = 5.02;
fprintf("Values: gallons=%d, started=%d, miles=%.2f\n", gallons, started, miles);
```

Example run:

```
Values: gallons=4, started=3, miles=5.02
```

Formatted Output

```
% fprintf Width and precision example
num1 = 123.456789;
num2 = 32123.456789;

% Minimum width of 10, 2 decimal places
fprintf("num1: %f\n", num1);
fprintf("num2: %f\n", num2);
fprintf("Width 10: %10.2f\n", num1);
fprintf("Width 10: %10.2f\n", num2);
```

- **fprintf('Width 10: %10.2f\n', num);** This line uses the **fprintf()** function to display the value of num in a formatted way.
- The format specifier **%10.2f** is used:
    - **%10** specifies a minimum width of 10 characters for the entire output. If the number is less than 10 characters wide, spaces will be added to the left to meet the width requirement.

- **.2** specifies that the number should be displayed with two digits after the decimal point.
- **f** indicates that the variable being formatted (num in this case) is a floating-point number.

Example run:

```
num1: 123.456789
num2: 32123.456789
Width 10:     123.46
Width 10:   32123.46
```

String formatting

```
name = "Alice";

% Normal string printing
fprintf("String: %s\n", name);

% String with specified width
fprintf("String (width 8): %8s\n", name);
```

Example run:

```
String: Alice
String (width 8):    Alice
```

Common placeholders for **fprintf()**

Placeholder	Useage
%f	Fixed point output (Most commonly used placeholder)
%s	Outputs a series of characters or a string
%i	Outputs an integer
%e or %E	Scientific notation with “e” displayed as a lowercase or uppercase, respectively
%g	Fixed point output (like %f) without trailing zeros

Please use **fprintf** for the rest of the assignments and the class.

Tutorial 2: Scalar, Vector, and Matrix

Vector and Matrix operations in MATLAB are much faster than in a traditional programming language. A traditional programming language would operate on each element one at a time. MATLAB operates on the entire vector or matrix all at once.

Add and run each example to your script.

Scalar

- Definition: A single numerical value.
- Declaration: **scalar = 5;**
- Example: **result = scalar * 10;** (Scalar multiplication)

```
% Scalar Declaration
scalarValue = 5;

% Scalar Operation
result = scalarValue * 10;

% Display Result
disp("Original value: " + scalarValue);
disp("Scalar multiplication: " + result);
```

Example run:

```
Original value: 5
Scalar multiplaction: 50
```

Vector

- Definition: An ordered array or set of numerical values.
- Declaration:
    - Using Colons: **myVector = 1:5;**
    - Using linspace: **myVector = linspace(0, 1, 4);**
    - Combining: **myVector = [2, 4, 6];**
- Example: **resultVector = myVector + 3;** (Element-wise addition)

```
% Vector Declaration with square brackets
myVector = [1, 2, 3, 4, 5];
disp("Vector Declaration:")
disp(myVector)

% Vector Operation (Element-wise Addition)
% Add 3 to each element of the vector
resultVector = myVector + 3;

% Display Result of calculation
disp("Vector Addition:")
disp(resultVector);
```

Example run:

```
Vector Declaration:
     1     2     3     4     5

Vector Addition:
     4     5     6     7     8
```

```
% Vector Declaration using the : (colon) operator
% Define the range of values from 1 to 10 with default increments of 1
% x = startValue:step:endValue;
% x = 0:10;
myVector = 1:10;
disp("Vector: ")
disp(myVector)

% Vector Operation (Element-wise multiplication)
resultVector = myVector * 3;

% Display Result
disp(resultVector);
```

Example run:

```
% Vector Declaration using the : (colon) operator
% Define the range of values from 1 to 10 with default increments of 1
% x = startValue:step:endValue;
% x = 0:10;
myVector = 1:10;
disp("Vector: ")
disp(myVector)

% Vector Operation (Element-wise multiplication)
resultVector = myVector * 3;

% Display Result
disp("Vector multiplication: ");
disp(resultVector);
```

Matrix

- Definition: A two-dimensional array of numerical values.
- Declaration:
    - Using square brackets: **myMatrix = [1, 2; 3, 4];**
    - Using zeros or ones functions: **myMatrix = zeros(2, 3);**
- Example: **resultMatrix = myMatrix * 2;** (Scalar multiplication for each element)

Operations

- Scalar Operations:
    - Addition: **scalarResult = scalar1 + scalar2;**
    - Multiplication: **scalarResult = scalar1 * scalar2;**`
- Vector Operations:
    - Element-wise Addition: **resultVector = vector1 + vector2;**
    - Scalar Multiplication: **resultVector = scalar * vector;**
- Matrix Operations:
    - Element-wise Addition: **resultMatrix = matrix1 + matrix2;**
    - Scalar Multiplication: **resultMatrix = scalar * matrix;**
    - Matrix Multiplication: **resultMatrix = matrix1 * matrix2;**

```
% Matrix Declaration using Square Brackets
myMatrix = [1, 2; 3, 4];

% Matrix Operation (Element-wise multiplication)
resultMatrix = myMatrix * 2;

% Original matrix
disp("Original matrix: ");
disp(myMatrix);

% Display Result
disp("Result matrix: ")
disp(resultMatrix);
```

Example run:

```
Original matrix:
     1     2
     3     4

Matrix:
     2     4
     6     8
```

Indexing

- Scalar: No indexing required.
- Vector: Access elements by index, e.g., **element = myVector(3);**
- Matrix: Access elements by row and column indices, e.g., **element = myMatrix(2, 1);**

Assignment 1: Average Speed Calculator

Include comments in your code to explain each step.

$$\text{Average Speed } (S) = \text{Distance/Time}$$

1. Create variables **distance** and **time**
2. Get input from the user.
3. Calculate average speed. Assign to variable.
4. Use the **fprintf()** function to display average speed to 2 decimal places.

Example runs:

```
    ---- Average Speed Calculation ----
Enter the distance (in meters): 2.365
Enter the time (in seconds): 10
The average speed is: 0.24 meters per second
>> Wk02AverageSpeed
    ---- Average Speed Calculation ----
Enter the distance (in meters): 1.254
Enter the time (in seconds): 10.56
The average speed is: 0.12 meters per second
```

Assignment 2: Volume of a Sphere

Write a script that will calculate the volume of a sphere with user input.

$$V = \frac{4}{3}\pi r^3$$

Example run:

```
---   Calculate Volume of a Sphere   ---
Please enter the radius: 2.5
The volume of the sphere with radius 2.50 units is 65.45 cubic units.
---   Calculate Volume of a Sphere   ---
Please enter the radius: 3.254
The volume of the sphere with radius 3.25 units is 144.32 cubic units.
```

Assignment 3: Constant Acceleration and Distance

Include comments in your code to explain each step.

The distance formula is from kinematics, a branch of physics that deals with the motion of objects. The distance formula can be used to calculate the distance traveled by an object under constant acceleration over a certain period of time.

Assuming the initial velocity is 0 gives us this simpler distance formula. Here's a breakdown of the equation:

$$d = 0.5at^2$$

where d= distance, a=acceleration, t = time

- **distance:** This represents the total distance traveled by the object in a given time interval.
- **0.5:** This constant factor, also represented as 1221, is a result of the integration process when dealing with constant acceleration in physics.
- **acceleration:** This is the rate at which the object's velocity changes per unit of time. It is a crucial parameter influencing the object's motion.
- **time:** This term accounts for the squared time duration during which the object experiences the specified acceleration. Squaring the time is necessary when dealing with constant acceleration scenarios.

Unit Conversions

$$1\ ft/s^2 = 0.3048\ m/s^2 \text{ or } 1\ m/s^2 = 3.28084\ ft/s^2$$

1. Convert the above formula into MATLAB.
2. Get **acceleration** in ft per second squared and **time** in seconds from the user.
3. Convert from imperial to metric. Assign to variable.
4. Calculate distance. Assign to a variable.
5. Display the final distance using fprintf(). Display to 2 decimal places.

Example runs:

```
-------------- Calculate Distance --------------
Enter acceleration (ft/s^): 2.5
Enter the time (seconds): 12.3
Acceleration (m/s^) is 0.76
Time (seconds) is 12.30
Distance (m) is 57.64
```

```
-------------- Calculate Distance --------------
Enter acceleration (ft/s^): 120.1
Enter the time (seconds): 32.6
Acceleration (m/s^) is 36.61
Time (seconds) is 32.60
Distance (m) is 19451.95
```

Assignment Submission

1. Submit properly named and commented script file.

2. Attach a text file showing the successful execution of each script.
3. Attach all to the assignment in Blackboard.