Week 4 MATLAB Activities

Contents

Time required: 120 minutes

Reading

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

Sections 2.3, 2.4

MATLAB Assignment Script

1. Create a MATLAB script named **Wk04Lastname.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: Transpose of a Matrix

Transpose is swapping the rows for the columns. The ' (single quote) is the transpose operator.

```
%% transpose matrix
% Create a matrix using linspace takes two steps
clc
% Create a Vector Using linspace:
vector = linspace(1, 9, 9);

% Reshape the Vector into a 3x3 Matrix:
aMatrix = reshape(vector, [3, 3]);

% You can use the transpose operator '
aTranspose = aMatrix';
% or the tranpose function
aTranspose = transpose(aMatrix);

disp("Original matrix");
disp(aMatrix);
disp('Transpose of A:');
disp(aTranspose);
```

Example run:

```
Original matrix
     1     4     7
     2     5     8
     3     6     9

Transpose of A:
     1     2     3
     4     5     6
     7     8     9
```

Tutorial 2: MATLAB mean (Average) Function

MATLAB has many built-in functions that work on a vector or matrix.

The **mean** function in MATLAB is used to calculate the average value of a matrix.

Syntax:

```
avg = mean(matrix)
avg = mean(matrix, dim)
```

- matrix: The input matrix.
- dim: The dimension along which to calculate the mean. If not specified, the mean (average) is calculated along the first non-singleton dimension.

Mean of a vector

```
%% mean function with vector
myArray = [1, 2, 3, 4, 5];
myArray = mean(myArray);
disp(myArray);  % Ouput: 3
```

Calculate the Mean of each Column

```
%% mean function with array
myArrayA = [1, 2, 3; 4, 5, 6; 7, 8, 9];
myArray = mean(myArrayA);  % Column-wise mean
disp(arrayMeanM);  % Output: [4 5 6]
```

Calculate the Mean of each Row

```
%% mean along a specific dimension
myArray = [1, 2, 3; 4, 5, 6; 7, 8, 9];
arrayMean = mean(myArray, 2);  % Row-wise mean
disp(arrayMean);  % Output: [2; 5; 8]
```

Tutorial 3: MATLAB min and max Functions

The min and max functions in MATLAB are used to find the minimum and maximum values in a vector or matrix.

min and max of a vector

Syntax

```
minValue = min(vec)
maxValue = max(vec)
```

```
%% min and max of a vector
myArray = [1, 2, 3, 4, 5];
minValue = min(myArray);
maxValue = max(myArray);
fprintf("Min: %d\n", minValue);  % Output: Min: 1
fprintf("Max: %d\n", maxValue);  % Output: Max: 5
```

Example run:

```
Min: 1
Max: 5
```

min and max of a matrix

Syntax

```
minValue = min(matrix)          % Columns
maxValue = max(matrix, [], 2)   % rows
```

```
%% min and max of a matrix
clc
myArray = [1, 2, 3; 4, 5, 6; 7, 8, 9];
minYValues = min(myArray);  % Column-wise minimum
maxYValues = max(myArray);  % Column-wise maximum
minXValues = min(myArray, [], 2);  % Row-wise minimum
maxXValues = max(myArray, [], 2);  % Row-wise maximum
disp("Column Min Values")
disp(minYValues);
disp("Column Max Values")
disp(maxYValues);
disp("Row Min Values")
disp(minXValues);
disp("Row Min Values")
disp(maxXValues);
```

Example run:

```
Column Min Values
     1     2     3

Column Max Values
     7     8     9

Row Min Values
     1
     4
     7

Row Min Values
     3
     6
     9
```

Tutorial 4: Random Numbers in MATLAB

- **rand()** - without arguments generates a single random number between 0 and 1.
- **rand(m, n)** - generates an m-by-n matrix of random numbers, where each element is independently sampled from a uniform distribution between 0 and 1.
- **randn(size)** - generates random numbers from a standard normal distribution, also known as a Gaussian distribution or a bell curve. **Size** specifies the size of the matrix or array you want to generate. The distribution of random numbers follows a standard normal distribution, which is a symmetric bell-shaped curve.
- **randperm(n)** - generates a random permutation of a sequence of integers from 1 – n.
- **randperm(n, m)** - Creates a random permutation of m integers from 1 to n.
- **randi(n)** - Generates a random scalar integer between 1 and n.
- **randi(n, m)** - Generates a m sized matrix with random integers values between 1 and n.
- **randi(n, m, l)** - Generates a m x l matrix with random integer values between 1 and n.

Add the following program to your MATLAB file.

```
% Generates a column vector with 5 random values between 0 and 1
rand(5, 1)

% Generates a 5x5 matrix of random numbers
% with a mean of 0 and a standard deviation of 1
randn(5)

% Creates a random permutation of the integers from 1 to 9
randperm(9)

% Creates a random permutation of 3 integers from 1 to 9
randperm(9, 3)

% Generates 3 unique random integers from 1 to 5, then adds 2 to each
2 + randperm(5, 3)

% Generates a random scalar integer between 1 and 5
randi(5)

% Generates a 3x3 column vector with random integer values between 1 and 5
randi(5, 3)

% Generates a 3x2 matrix with random integer values between 1 and 5
randi(5, 3, 2)

% Generates a 3x2 matrix with random integer values
% between 1 and 5 and assigns it to variable x
x = randi(5, 3, 2)

% Generates a 3x2 matrix with random integer values between 1 and 4,
% then adds 1 to each element
x = 1 + randi(4, 3, 2)
```

Tutorial 5: The : (Colon) Operator with Matrices

The colon operator is a powerful tool in MATLAB for creating vectors, selecting array elements, and extracting rows or columns from matrices.

```
%% : Colon operator with a matrix
clc
% Create a  3 x 3 matrix
myMatrix = [1, 2, 3; 4, 5, 6; 7, 8, 9];
```

Extract a Single Row

To extract a single row from a matrix, use the colon operator to specify all columns of that row. The result is a vector.

Syntax:

```
rowVec = matrix(row_index, :);
```

```
% Select all elements in the second row, return a vector
second_row_vector = aMatrix(2, :);
disp('Second row vector:');
disp(second_row_vector);
```

Example run:

```
Second row vector:
     4     5     6
```

Extract a Single Column

To extract a single column from a matrix, use the colon operator to specify all rows of that column. The result is a vector.

Syntax:

```
colVec = matrix(:, column_index);
```

```
% Select all elements in the third column, return a vector
third_column_vector = aMatrix(:, 3);
disp('Third column vector:');
disp(third_column_vector);
```

Example run:

```
Third column vector:
     3
     6
     9
```

Tutorial 6: Vector and Matrix Math

Try each of these out in your script to see how they work.

Describe how each operation works.

Define two vectors: v1 and v2.

Use the **fprintf** function to show the results

1. **Create Vectors**
    a. Use square brackets to define two vectors
    b. **v1 = [1, 2, 3];**
       v2 = [4, 3, 5];
2. **Vector Math Operations**
    a. Addition: **result = v1 + v2**
    b. Subtraction: **result = v1 - v2**
    c. Element-wise multiplication: **result = v1 .* v2**
    d. Dot product: **result = dot(v1, v2)**
    e. Element-wise division: **result = v1 ./ v2**
    f. Square each element: **result = v1 .^ 2**
3. **Create Matrices**
    a. Use square brackets to define two matrices
    b. m1 = [1, 2, 3; 4, 5, 6; 7, 8, 9];
       m2 = [3, 2, 1; 6, 5, 4; 9, 8, 7];
4. Matrix Math Operations
    a. Addition: **result = m1 + m2**; (m1 must have the same dimensions as m2).
    b. Subtraction: **result = m1 – m2**; (m1 must have the same dimensions as m2).
    c. Element-wise multiplication: **result = m1 .* m2;** (m1 must have the same dimensions as m2).
    d. Matrix multiplication: **result = m1 * m2**; (number of columns in m1 must equal the number of rows in m2).
    e. Element-wise division: **results = m1 / 9**

Practice 2.3

This assignment practices vector and matrix math with following vector and matrix.

```
myVector = [4, 11, 32, -5, 0,  9];
myMatrix = [1, 2, 3; 4, 5, 6; 7, 8, 9];
```

1. Create a vector variable and subtract 3 from every element in it.
2. Create a vector variable and multiply each element by 22.
3. Create a matrix variable and divide every element by 3.
4. Create a matrix variable and square every element.

Example run:

```
Matrix Element-wise subtraction
     1     8    29    -8    -3     6

Matrix Element-wise multiplication
    22   176   638  -176   -66   132

Matrix Element-wise Division
    0.3333    0.6667    1.0000
    1.3333    1.6667    2.0000
    2.3333    2.6667    3.0000

Matrix Element-wise exponent
     1     4     9
    16    25    36
    49    64    81
```

Exercise 2.1

1. Create the vector **vec** in three different ways
    a. using just square brackets
    b. using the colon operator
    c. using linspace:

vec = 5, 7, 9, 11

Exercise 2.6

1. Create a variable **myend**, which stores a random integer in the inclusive range from 5 to 9.

2. Using the colon operator, create a vector that iterates from 1 to myend in steps of 3.

NOTE: As this uses a random integer, your result may be different.

Example run:

```
vec =    1     4     7
```

Assignment 1: The Power Equation

The power equation, $\boldsymbol{P = VI}$, represents the relationship between electrical power (P), voltage (V), and current (I).

1. **Power (P)**:
    a. Power is the amount of energy transferred or converted per unit of time. In electrical terms, it is measured in watts (W).
    b. In the context of electrical circuits, power signifies the rate at which electrical energy is used or produced.
2. **Voltage (V)**:
    a. Voltage is the electric potential difference between two points in a circuit. It is the force that drives electric current.
    b. Voltage is measured in volts (V). It indicates the electric pressure or "push" that causes electrons to flow in a circuit.
3. **Current (I)**:
    a. Current is the flow of electric charge in a circuit. It represents the movement of electrons.
    b. Current is measured in amperes (A). It reflects the quantity of charge passing through a point in the circuit per unit of time.
4. The Power Equation

$$\boldsymbol{P = VI}:$$

    a. The power equation states that power (P) is equal to the product of voltage (V) and current (I).
    b. Mathematically, it can be written as $\boldsymbol{P = V\ x\ I}$.

If you have a higher voltage or a higher current in a circuit, the power will increase.

Power is the product of the force (voltage) driving the electrons and the quantity of electrons moving (current).

Example:

If you have a voltage of 12 volts (V) and a current of 2 amperes (A), the power would be

$$\boldsymbol{P = 12V\ x\ 2A = 24W}$$

Create a MATLAB program that uses the power equation to calculate the wattage from user input of voltage and amps. Remember to convert the equation variable names to an appropriate programming variable name.

Example run:

```
    -------- Wattage Calculator --------
Enter voltage: 24
Enter amperes: 10
The calculated wattage is: 240.00 watts
    -------- Wattage Calculator --------
Enter voltage: 12.5
Enter amperes: 6.2
The calculated wattage is: 77.50 watts
```

Assignment Submission

1. Submit properly named and commented script files to the assignment in Blackboard.
2. Attach a text file showing the successful execution of each script.
3. Attach all to the assignment in Blackboard.