**B.TECH. (2020-24)**

**Artificial Intelligence**

**Lab File**

on

**Basic Simulation Lab**

**[ES204]**

**Logo

Description automatically generated**

Submitted To

**Mr. Lala Bhaskar**

Submitted By

**Hitesh**

**A023119820027**

**4CSE11 (AI)**

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

AMITY SCHOOL OF ENGINEERING AND TECHNOLOGY

AMITY UNIVERSITY UTTAR PRADESH

NOIDA (U.P)

**EXPERIMENT- 1**

**AIM**

(A) Creating a One-Dimensional Array (Row / Column Vector)

(B) Creating a Two-Dimensional Array (Matrix of given size)

(C) Performing Arithmetic Operations - Addition, Subtraction, Multiplication and Exponentiation

(D) Performing Matrix operations - Inverse, Transpose and Rank.

**SOFTWARE USED**

Octave Online - <https://octave-online.net/>

**THEORY**

MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation.

MATLAB is an interactive system whose basic data element is an array that does not require dimensioning.

MATLAB is an abbreviation for "**mat**rix **lab**oratory." While other programming languages mostly work with numbers one at a time, MATLAB is designed to operate primarily on whole matrices and arrays.

All MATLAB variables are multidimensional arrays, no matter what type of data. A matrix is a two-dimensional array often used for linear algebra.

(I) Creating a One-Dimensional Array (Row / Column Vector)

To create a 1D Array, you need to assign a list of numbers separated with comma (,) or space ( ), to any variable say (A) for a row vector or separated with semi-colon (;) for a column vector.

For instance, to create an array with four elements in a single row, separate the elements with either a comma (,) or a space.

|  |
| --- |
| A = [1 2 3 4] |

This type of array is a row vector.

To create a column vector with 3 elements, separate the (row) elements with semicolons.

|  |
| --- |
| B = [1; 2; 7] |

(II) Creating a Two-Dimensional Array (Matrix of given size)

To create a 2D array or matrix of given size (say 3x3) that has multiple rows, separate the rows with semicolons.

|  |
| --- |
| C = [1 3 5; 2 4 6; 7 8 10] |

(III) Performing Arithmetic Operations - Addition, Subtraction, Multiplication and Exponentiation

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

Consider that A and B are two 3x3 matrices,

1. **Addition**: Using operator (+) as such , adds arrays A and B by adding corresponding elements. If one input is a string array, then plus appends the corresponding elements as strings. The sizes of A and B must be the same or be compatible.

Alternatively, using MATLAB command plus() as such to execute A + B.

1. **Subtraction**: Using operator (-) as such , subtracts array B from array A by subtracting corresponding elements. The sizes of A and B must be the same or be compatible.

Alternatively, using MATLAB command minus() as such to execute A – B.

1. **Multiplication**: Using operator (\*) as such for the matrix product of A and B. If A is an m-by-p and B is a p-by-n matrix, then C is an m-by-n matrix defined by

A picture containing text, watch

Description automatically generated

Alternatively, using MATLAB command mtimes() as such to execute A\*B.

1. **Hadamard Product (Element-wise Multiplication):** Using operator (.\*) as such , multiplies arrays A and B by multiplying corresponding elements. The sizes of A and B must be the same or be compatible.

Alternatively, using MATLAB command times() as such to execute A.\*B.

1. **Division**: Using operator (./) as such , divides each element of A by the corresponding element of B. The sizes of A and B must be the same or be compatible.

Alternatively, using MATLAB commands rdivide() for right divide and ldivide() for left divide as such and to divide A by B.

1. **Element-by-element Exponentiation**: Using MATLAB command exp() as such , returns the exponential ex for each element in array X.
2. **Matrix Exponentiation**: Using MATLAB command expm() as such , computes the matrix exponential of X. Although it is not computed this way, if X has a full set of eigenvectors V with corresponding eigenvalues D, then [V,D] = eig(X) and expm(X) = V\*diag(exp(diag(D)))/V.

It returns the result of matrix exponential eAt, i.e,

A picture containing text, watch, clock

Description automatically generated

(IV) Performing Matrix operations - Inverse, Transpose and Rank.

1. **Inverse**: Using operator {^(-1)} as such , computes the inverse of square matrix X.

Alternatively, using MATLAB command inv() as such to get the inverse of square matrix X.

1. **Transpose**: Using operator (.') as such , returns the nonconjugate transpose of A, that is, interchanges the row and column index for each element.

Alternatively, using MATLAB command transpose() as such to compute the nonconjugate transpose of A.

1. **Rank**: Using MATLAB command rank() as such , returns the rank of matrix A.

**PROGRAM CODE**

|  |  |
| --- | --- |
| %CREATION OF MATRIX  A = [1 2 3] %Row Vector  B = [1;2;3] %Column Vector  C = [1 2 3;4,6,6;7,8,9]  D = [0,0,0;4,5,6;7,8,9]  E = [1,2,4,5,3;1,2,4,5,6;5,6,4,3,3;1,2,4,5,5]  %ADDITION  F = C + D  G = plus(C,D)  %SUBTRACTION  H = C - D  I = minus(C,D)  %MULTIPLICATION  H = A \* B  s = mtimes(A,B)  %ELEMENT WISE MULTIPLICATION (Hadamard Product)  I = C .\* D  J = times(C,D)  %DIVISION  K = [2 4 6 8; 3 5 7 9]  L = [10,10,10,10;10,10,10,10]  M = K./L  N = rdivide(K,L)  O = ldivide(L,K) | %EXPONENTIAL  p = [1 1 0; 0 0 2; 0 0 -1]  q = exp(p)  r = expm(p)  %Transpose of matrix  s = C.'  t = transpose(C)  %DETERMINANT OF matrix  u = det(C)  %Inverse of matrix  v = C^(-1)  w = inv(C)  x = v\*C  y = w\*C  %Rank of matrix  z = rank(C)  %ones  zz = 10 \* ones(2,3)  zzz = zeros(2,4) |

**RESULTS**

|  |  |  |
| --- | --- | --- |
| A =  1 2 3  B =  1  2  3  C =  1 2 3  4 6 6  7 8 9  D =  0 0 0  4 5 6  7 8 9  E =  1 2 4 5 3  1 2 4 5 6  5 6 4 3 3  1 2 4 5 5  F =  1 2 3  8 11 12  14 16 18  G =  1 2 3  8 11 12  14 16 18  H =  1 2 3  0 1 0  0 0 0  I =  1 2 3  0 1 0  0 0 0 | H = 14  s = 14  I =  0 0 0  16 30 36  49 64 81  J =  0 0 0  16 30 36  49 64 81  K =  2 4 6 8  3 5 7 9  L =  10 10 10 10  10 10 10 10  M =  0.2000 0.4000 0.6000 0.8000  0.3000 0.5000 0.7000 0.9000  N =  0.2000 0.4000 0.6000 0.8000  0.3000 0.5000 0.7000 0.9000  O =  0.2000 0.4000 0.6000 0.8000  0.3000 0.5000 0.7000 0.9000  p =  1 1 0  0 0 2  0 0 -1  q =  2.7183 2.7183 1.0000  1.0000 1.0000 7.3891  1.0000 1.0000 0.3679  r = | 2.7183 1.7183 1.0862  0 1.0000 1.2642  0 0 0.3679  s =  1 4 7  2 6 8  3 6 9  t =  1 4 7  2 6 8  3 6 9  u = -12.000  v =  -0.5000 -0.5000 0.5000  -0.5000 1.0000 -0.5000  0.8333 -0.5000 0.1667  w =  -0.5000 -0.5000 0.5000  -0.5000 1.0000 -0.5000  0.8333 -0.5000 0.1667  x =  1.0000 0 0.0000  0.0000 1.0000 -0.0000  0.0000 0.0000 1.0000  y =  1.0000 0 0.0000  0.0000 1.0000 -0.0000  0.0000 0.0000 1.0000  z = 3  zz =  10 10 10  10 10 10  zzz =  0 0 0 0  0 0 0 0 |

**DISCUSSION and CONCLUSION**

The several MATLAB commands have been explored and successfully used to create a One-Dimensional Array (Row / Column Vector), a Two-Dimensional Array (Matrix of given size) and perform the required Arithmetic Operations and Matrix Operations on the Octave Online platform.

**PRECAUTIONS**

* Don’t forget to save the code after every change you make.
* Use MATLAB properly.
* MATLAB requires a stable network connection.
* Save the file after compiling the code and take the required notes and screenshots, so that you don’t have to open octave and do everything again.

|  |  |  |  |
| --- | --- | --- | --- |
| **Criteria** | **Total Marks** | **Marks Obtained** | **Comments** |
| Concept (A) | 2 |  |  |
| Implementation (B) | 2 |  |  |
| Performance (C) | 2 |  |  |
| Total | 6 |  |  |

**EXPERIMENT- 2**

**AIM**

Performing Matrix Manipulations – Concatenating, Indexing, Sorting, Shifting, Reshaping, Resizing and Flipping about a Vertical Axis / Horizontal Axis; Creating Arrays X & Y of given size (1 x N) and Performing

1. Relational Operations – (>, <, ==, <=, >=, ~=)
2. Logical Operations – (~, &, |, XOR)

**SOFTWARE USED**

Octave Online - <https://octave-online.net/>

**THEORY**

MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation.

MATLAB is an interactive system whose basic data element is an array that does not require dimensioning.

MATLAB is an abbreviation for "**mat**rix **lab**oratory." While other programming languages mostly work with numbers one at a time, MATLAB is designed to operate primarily on whole matrices and arrays.

All MATLAB variables are multidimensional arrays, no matter what type of data. A matrix is a two-dimensional array often used for linear algebra.

(I) Matrix Manipulations

1. **Concatenation:** Using MATLAB command cat() as such to concatenate B to the end of A along dimension dim when A and B have compatible sizes (the lengths of the dimensions match except for the operating dimension dim).

concatenates A1, A2, … , An along dimension dim.

Using square bracket operator [] to concatenate. For example, [A,B] or [A B] concatenates arrays A and B horizontally, and [A; B] concatenates them vertically.

1. **Indexing:** The most common way is to explicitly specify the indices of the elements. As such, to access a single element of a matrix, specify the row number followed by the column number of the element.

For example, , e is the element in the 3,2 position (third row, second column) of A.

You can also reference multiple elements at a time by specifying their indices in a vector. For example, to access the first and third elements of the second row of A, the command would be .

To access elements in a range of rows or columns, use the colon. For example, to access the elements in the first through third row and the second through fourth column of A, the command is .

An alternative way to compute r is to use the keyword “end” to specify the second column through the last column. This approach lets you specify the last column without knowing exactly how many columns are in A.

If you want to access all the rows or columns, use the colon operator by itself. For example, to return the entire third column of A, the command would be .

1. **Sorting:** Using MATLAB command sort() as such sorts the elements of A in ascending order.

* If A is a vector, then sorts the vector elements.
* If A is a matrix, then treats the columns of A as vectors and sorts each column.
* If A is a multidimensional array, then operates along the first array dimension whose size does not equal 1, treating the elements as vectors.

returns the sorted elements of A along dimension dim. For example, if A is a matrix, then sort(A,2) sorts the elements of each row.

returns sorted elements of A in the order specified by direction using any of the previous syntaxes. 'ascend' indicates ascending order (the default) and 'descend' indicates descending order.

1. **Shifting:** Using MATLAB command circshift() as such , you can circularly shift the elements in array A by K positions. If K is an integer, then circshift() shifts along the first dimension of A whose size does not equal 1. If K is a vector of integers, then each element of K indicates the shift amount in the corresponding dimension of A.

circularly shifts the values in array A by K positions along dimension dim. Inputs K and dim must be scalars.

1. **Reshaping and Resizing**: Using MATLAB command reshape() as such reshapes A using the size vector, sz, to define size(B). For example, reshape(A,[2,3]) reshapes A into a 2-by-3 matrix. sz must contain at least 2 elements, and prod(sz) must be the same as numel(A).

reshapes A into a sz1-by-...-by-szN array where sz1,...,szN indicates the size of each dimension. You can specify a single dimension size of [] to have the dimension size automatically calculated, such that the number of elements in B matches the number of elements in A.

For example, if A is a 10-by-10 matrix, then reshapes the 100 elements of A into a 2-by-2-by-25 array.

1. **Flipping**: Using MATLAB command flip() as such , returns array B the same size as A, but with the order of the elements reversed. The dimension that is reordered in B depends on the shape of A: -

* If A is vector, then flip(A) reverses the order of the elements along the length of the vector.
* If A is a matrix, then flip(A) reverses the elements in each column.
* If A is an N-D array, then flip(A) operates on the first dimension of A in which the size value is not 1.

reverses the order of the elements in A along dimension dim.

For example, if A is a matrix, then reverses the elements in each column, and reverses the elements in each row.

1. **Rotation**: Using MATLAB command rot90() as such rotates array A counterclockwise by 90 degrees. For multidimensional arrays, rot90 rotates in the plane formed by the first and second dimensions.

rotates array A counterclockwise by k\*90 degrees, where k is an integer.

Using Octave command rotdim() as such , it returns a copy of x with the elements rotated counterclockwise in 90-degree increments.

The second argument n is optional and specifies how many 90-degree rotations are to be applied (the default value is 1). Negative values of n rotate the matrix in a clockwise direction.

The third argument is also optional and defines the plane of the rotation. If present, plane is a two-element vector containing two different valid dimensions of the matrix. When plane is not given the first two non-singleton dimensions are used.

(II) Performing Relational Operations on 2 arrays X & Y of given size (1xN)

Relational operators can also work on both scalar and non-scalar data. Relational operators for arrays perform element-by-element comparisons between two arrays and return a logical array of the same size, with elements set to logical 1 (true) where the relation is true and elements set to logical 0 (false) where it is not.

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No.** | **Operator** | **MATLAB Command/Function** | **Description** |
| **1** | < | lt(a,b) | Tests whether a is less than b |
| **2** | <= | le(a,b) | Tests whether a is less than or equal to b |
| **3** | > | gt(a,b) | Tests whether a is greater than b |
| **4** | >= | ge(a,b) | Tests whether a is greater than or equal to b |
| **5** | == | eq(a,b) | Tests whether a is equal to b |
| **6** | ~= | ne(a,b) | Tests whether a is not equal to b |
| **7** |  | isequal(a,b) | Determine array equality, treating NaN values as unequal |
| **8** |  | isequaln(a,b) | Determine array equality, treating NaN values as equal |

(III) Performing Logical Operations on 2 arrays X & Y of given size (1xN)

MATLAB represents Boolean data using the logical data type. This data type represents true and false states using the numbers 1 and 0, respectively. Certain MATLAB functions and operators return logical values to indicate fulfilment of a condition.

1. **Logical NOT**: Using tilde operator (~) as such , returns a logical array of the same size as A. The array contains logical 1 (true) values where A is zero and logical 0 (false) values where A is nonzero.

Alternatively, use MATLAB command to execute ~A.

1. **Logical** **AND**: Using ampersand operator (&) as such to perform a logical AND of arrays A and B and to return an array containing elements set to either logical 1 (true) or logical 0 (false). An element of the output array is set to logical 1 (true) if both A and B contain a nonzero element at that same array location. Otherwise, the array element is set to 0.

Using MATLAB command is an alternate way to execute A & B.

1. **Logical** **OR**: Using operator (|) as such to perform a logical OR of arrays A and B and to return an array containing elements set to either logical 1 (true) or logical 0 (false). An element of the output array is set to logical 1 (true) if either A or B contain a nonzero element at that same array location. Otherwise, the array element is set to 0.

Using MATLAB command is an alternate way to execute A | B.

1. **Logical** **Exclusive-OR**: Using MATLAB command to perform a logical exclusive-OR of arrays A and B and to return an array containing elements set to either logical 1 (true) or logical 0 (false). An element of the output array is set to logical 1 (true) if A or B, but not both, contains a nonzero element at that same array location. Otherwise, the array element is set to 0.

**PROGRAM CODE**

|  |  |
| --- | --- |
| %CONCATENATE  A = [1 2 3 4;5 6 7 8]  B = [3 5 2 1;7 4 9 8]  C = [A B]  D = [A;B]  CC = cat (2,A,B)  DD = cat (1,A,B)  %INDEXING  E = A([1:2],[2:4])  EE = A(6)  %SORTING BY COLUMNWISE  F = sort(B)  G = sort(B,'descend')  %SORTING BY ROW WISE  H = sort(B,2)  I = sort(B,2,'descend')  %SHIFTING  J = circshift([1 2 3 4 5 6 7],2)  JJ = circshift([1;2;3;4;5;6;7],2)  %RESHAPING  K = reshape ([1, 2, 3, 4], 2, 2)  L = reshape ([1, 2, 3, 4], 2, [])  M = reshape ([1, 2;3, 4], 1,[])  %FLIPPING  N = flip(A)  O = flip(A,2)  %ROTATION  rot90 ([1, 2; 3, 4], -1)  rot90 ([1, 2; 3, 4], 1)  rot90 ([1, 2; 3, 4], 7) | rotdim ([1, 2; 3, 4], -1)  rotdim ([1, 2; 3, 4], -1, [1, 2])  rotdim ([1, 2; 3, 4], 1, [1, 2])  %CREATING X AND Y MATRICES  x = [2 3 5 6 0 1 9 8]  y = [1 4 7 2 0 6 2 0]  less = x < y  more = x > y  lessequal = x <= y  moreequal = x>= y  equal = x == y  notequal = x ~= y  less2 = lt(x,y)  more2 = gt(x,y)  lessequal2 = le(x,y)  moreequal2 = ge(x,y)  equal2 = eq(x,y)  notequal2 = ne(x,y)  p = [1 2 3 4 NaN]  q = [1 2 3 4 NaN]  equality = isequal(p,q)  noteq = isequaln(p,q)  %LOGICAL OPERATORS  or1 = x | y  or2 = or(x,y)  and1 = x & y  and2 = and(x,y)  notA = not(y)  nta = ~y  exor = xor(x,y) |

**RESULTS**

|  |  |  |
| --- | --- | --- |
| A =  1 2 3 4  5 6 7 8  B =  3 5 2 1  7 4 9 8  C =  1 2 3 4 3 5 2 1  5 6 7 8 7 4 9 8  D =  1 2 3 4  5 6 7 8  3 5 2 1  7 4 9 8  CC =  1 2 3 4 3 5 2 1  5 6 7 8 7 4 9 8  DD =  1 2 3 4  5 6 7 8  3 5 2 1  7 4 9 8  E =  2 3 4  6 7 8  EE = 7  F =  3 4 2 1  7 5 9 8  G =  7 5 9 8  3 4 2 1  H =  1 2 3 5  4 7 8 9  I =  5 3 2 1  9 8 7 4  J =  6 7 1 2 3 4 5  JJ =  6  7  1  2  3  4  5 | K =  1 3  2 4  L =  1 3  2 4  M =  1 3 2 4  N =  5 6 7 8  1 2 3 4  O =  4 3 2 1  8 7 6 5  ans =  3 1  4 2  ans =  2 4  1 3  ans =  3 1  4 2  ans =  3 1  4 2  ans =  3 1  4 2  ans =  2 4  1 3  x =  2 3 5 6 0 1 9 8  y =  1 4 7 2 0 6 2 0  less =  0 1 1 0 0 1 0 0  more =  1 0 0 1 0 0 1 1  lessequal =  0 1 1 0 1 1 0 0 | moreequal =  1 0 0 1 1 0 1 1  equal =  0 0 0 0 1 0 0 0  notequal =  1 1 1 1 0 1 1 1  less2 =  0 1 1 0 0 1 0 0  more2 =  1 0 0 1 0 0 1 1  lessequal2 =  0 1 1 0 1 1 0 0  moreequal2 =  1 0 0 1 1 0 1 1  equal2 =  0 0 0 0 1 0 0 0  notequal2 =  1 1 1 1 0 1 1 1  p =  1 2 3 4 NaN  q =  1 2 3 4 NaN  equality = 0  noteq = 1  or1 =  1 1 1 1 0 1 1 1  or2 =  1 1 1 1 0 1 1 1  and1 =  1 1 1 1 0 1 1 0  and2 =  1 1 1 1 0 1 1 0  notA =  0 0 0 0 1 0 0 1  nta =  0 0 0 0 1 0 0 1  exor =  0 0 0 0 0 0 0 1 |

**DISCUSSION and CONCLUSION**

The several MATLAB commands have been explored and successfully used to perform Matrix Manipulations – Concatenating, Indexing, Sorting, Shifting, Reshaping, Resizing and Flipping about a Vertical Axis / Horizontal Axis, and to create Arrays X & Y of given size (1 x N) and performing relational and logical operations on them.

**PRECAUTIONS**

* Don’t forget to save the code after every change you make.
* Use MATLAB properly.
* MATLAB requires a stable network connection.
* Save the file after compiling the code and take the required notes and screenshots, so that you don’t have to open octave and do everything again.

|  |  |  |  |
| --- | --- | --- | --- |
| **Criteria** | **Total Marks** | **Marks Obtained** | **Comments** |
| Concept (A) | 2 |  |  |
| Implementation (B) | 2 |  |  |
| Performance (C) | 2 |  |  |
| Total | 6 |  |  |

**EXPERIMENT- 3**

**AIM**

Generating a set of Commands on a given Vector (Example: X = [1 8 3 9 0 1]) to

(A). Add up the values of the elements (Check with sum)

(B). Compute the Running Sum (Check with sum), where Running Sum for element j = the sum of the elements from 1 to j, inclusive.

(C) Generating a Random Sequence using functions and plot them.

**SOFTWARE USED**

Octave Online - <https://octave-online.net/>

**PROGRAM CODE**

**%** **Generating a set of Commands on a given Vector (Example: X = [1 8 3 9 0 1])**

|  |
| --- |
| A = 1:10 #vector from 1 to 10  AA = 1:0.5:10 #vector from 1 to 10 with step size of 0.5  AAA = 1:0.1:10 #vector from 1 to 10 with step size of 0.1  B = [1 2 3;4 5 6;7 8 9] |

**% (A). Add up the values of the elements (Check with sum)**

|  |
| --- |
| #Sum of elements of a VECTOR  saa = sum(AA)  saaa = sum(AAA)  sa = sum(A)  #Sum of elements of MATRIX  sb = sum(B) #sum of elements columnwise  sbc = sum(B,1) #sum of elements columnwise  sbr = sum(B,2 #sum of elements rowwise |

**% (B). Compute the Running Sum (Check with sum), where Running Sum for element j = the sum of the elements from 1 to j, inclusive.**

|  |
| --- |
| #Cumulative sum of elements of a VECTOR  sca = cumsum(A)  ssA = 0  for i=1 : length(A)  ssA = ssA + A(i)  end  #Cumulative sum of elements of a MATRIX  scb = cumsum(B) #cumulative sum of elements columnwise  scbc = cumsum(B,1) #cumulative sum of elements columnwise  scbr = cumsum(B,2) #cumulative sum of elements rowwise  #Cumulative Sum of elements rowwise using loop construct  for i=1 : size(B,1)  sb = 0  for j=1 : size(B,2)  sb = sb + B(i,j)  end  end |

**%(C) Generating a Random Sequence using rand() / randn() functions and plot them.**

|  |
| --- |
| #Generating random real no.s between 0 and 1  C = rand  D = rand  #Generating matrix with its elements as random real no.s between 0 and 1  E = rand(5,3)  F = rand(5)  #Ploting with the help of plot()  X = 0:0.1:10  Yexp = exp(X)  Ysin = sin(X)  Ycos = cos(X)  pa = plot(10:-1:1)  pexp = plot(X,Yexp)  psin = plot(X,Ysin)  pcos = plot(X,Ycos) |

**RESULTS**

Generating a set of Commands on a given Vector (Example: X = [1 8 3 9 0 1])

|  |  |
| --- | --- |
| A =  1 2 3 4 5 6 7 8 9 10  AA =  Columns 1 through 7:  1.0000 1.5000 2.0000 2.5000 3.0000 3.5000 4.0000  Columns 8 through 14:  4.5000 5.0000 5.5000 6.0000 6.5000 7.0000 7.5000  Columns 15 through 19:  8.0000 8.5000 9.0000 9.5000 10.0000  AAA =  Columns 1 through 7:  1.0000 1.1000 1.2000 1.3000 1.4000 1.5000 1.6000  Columns 8 through 14:  1.7000 1.8000 1.9000 2.0000 2.1000 2.2000 2.3000  Columns 15 through 21:  2.4000 2.5000 2.6000 2.7000 2.8000 2.9000 3.0000  Columns 22 through 28:  3.1000 3.2000 3.3000 3.4000 3.5000 3.6000 3.7000  Columns 29 through 35:  3.8000 3.9000 4.0000 4.1000 4.2000 4.3000 4.4000 | Columns 36 through 42:  4.5000 4.6000 4.7000 4.8000 4.9000 5.0000 5.1000  Columns 43 through 49:  5.2000 5.3000 5.4000 5.5000 5.6000 5.7000 5.8000  Columns 50 through 56:  5.9000 6.0000 6.1000 6.2000 6.3000 6.4000 6.5000  Columns 57 through 63:  6.6000 6.7000 6.8000 6.9000 7.0000 7.1000 7.2000  Columns 64 through 70:  7.3000 7.4000 7.5000 7.6000 7.7000 7.8000 7.9000  Columns 71 through 77:  8.0000 8.1000 8.2000 8.3000 8.4000 8.5000 8.6000  Columns 78 through 84:  8.7000 8.8000 8.9000 9.0000 9.1000 9.2000 9.3000  Columns 85 through 91:  9.4000 9.5000 9.6000 9.7000 9.8000 9.9000 10.0000  B =  1 2 3  4 5 6  7 8 9 |

(A). Add up the values of the elements (Check with sum)

|  |
| --- |
| saa = 104.50  saaa = 500.50  sa = 55  sb =  12 15 18  sbc =  12 15 18  sbr =  6  15  24 |

(B). Compute the Running Sum (Check with sum), where Running Sum for element j = the sum of the elements from 1 to j, inclusive.

|  |  |
| --- | --- |
| sca =  1 3 6 10 15 21 28 36 45 55  ssA = 0  ssA = 1  ssA = 3  ssA = 6  ssA = 10  ssA = 15  ssA = 21  ssA = 28  ssA = 36  ssA = 45  ssA = 55  scb =  1 2 3  5 7 9  12 15 18 | scbc =  1 2 3  5 7 9  12 15 18  scbr =  1 3 6  4 9 15  7 15 24  sb = 0  sb = 1  sb = 3  sb = 6  sb = 0  sb = 4  sb = 9  sb = 15  sb = 0  sb = 7  sb = 15  sb = 24 |

(C) Generating a Random Sequence using functions and plot them.

|  |  |  |
| --- | --- | --- |
| C = 0.3927  D = 0.4648  E =  0.078051 0.172380 0.668933  0.217670 0.011819 0.771541  0.886589 0.315641 0.808936  0.046360 0.271458 0.493604  0.645525 0.759192 0.678757  F =  0.6599 0.8430 0.7434 0.1666 0.4927  0.1059 0.6563 0.7810 0.8541 0.2505  0.8910 0.7735 0.5439 0.3215 0.4518  0.1689 0.4789 0.7256 0.1398 0.1916  0.6553 0.2968 0.1714 0.7497 0.8807  X =  Columns 1 through 7:  0 0.1000 0.2000 0.3000 0.4000 0.5000 0.6000  Columns 8 through 14:  0.7000 0.8000 0.9000 1.0000 1.1000 1.2000 1.3000  Columns 15 through 21:  1.4000 1.5000 1.6000 1.7000 1.8000 1.9000 2.0000  Columns 22 through 28:  2.1000 2.2000 2.3000 2.4000 2.5000 2.6000 2.7000  Columns 29 through 35:  2.8000 2.9000 3.0000 3.1000 3.2000 3.3000 3.4000  Columns 36 through 42:  3.5000 3.6000 3.7000 3.8000 3.9000 4.0000 4.1000  Columns 43 through 49:  4.2000 4.3000 4.4000 4.5000 4.6000 4.7000 4.8000  Columns 50 through 56:  4.9000 5.0000 5.1000 5.2000 5.3000 5.4000 5.5000  Columns 57 through 63:  5.6000 5.7000 5.8000 5.9000 6.0000 6.1000 6.2000  Columns 64 through 70:  6.3000 6.4000 6.5000 6.6000 6.7000 6.8000 6.9000  Columns 71 through 77:  7.0000 7.1000 7.2000 7.3000 7.4000 7.5000 7.6000  Columns 78 through 84:  7.7000 7.8000 7.9000 8.0000 8.1000 8.2000 8.3000  Columns 85 through 91:  8.4000 8.5000 8.6000 8.7000 8.8000 8.9000 9.0000  Columns 92 through 98:  9.1000 9.2000 9.3000 9.4000 9.5000 9.6000 9.7000  Columns 99 through 101:  9.8000 9.9000 10.0000  Yexp =  Columns 1 through 6:  1.0000e+00 1.1052e+00 1.2214e+00 1.3499e+00 1.4918e+00 1.6487e+00  Columns 7 through 12:  1.8221e+00 2.0138e+00 2.2255e+00 2.4596e+00 2.7183e+00 3.0042e+00 | Columns 13 through 18:  3.3201e+00 3.6693e+00 4.0552e+00 4.4817e+00 4.9530e+00 5.4739e+00  Columns 19 through 24:  6.0496e+00 6.6859e+00 7.3891e+00 8.1662e+00 9.0250e+00 9.9742e+00  Columns 25 through 30:  1.1023e+01 1.2182e+01 1.3464e+01 1.4880e+01 1.6445e+01 1.8174e+01  Columns 31 through 36:  2.0086e+01 2.2198e+01 2.4533e+01 2.7113e+01 2.9964e+01 3.3115e+01  Columns 37 through 42:  3.6598e+01 4.0447e+01 4.4701e+01 4.9402e+01 5.4598e+01 6.0340e+01  Columns 43 through 48:  6.6686e+01 7.3700e+01 8.1451e+01 9.0017e+01 9.9484e+01 1.0995e+02  Columns 49 through 54:  1.2151e+02 1.3429e+02 1.4841e+02 1.6402e+02 1.8127e+02 2.0034e+02  Columns 55 through 60:  2.2141e+02 2.4469e+02 2.7043e+02 2.9887e+02 3.3030e+02 3.6504e+02  Columns 61 through 66:  4.0343e+02 4.4586e+02 4.9275e+02 5.4457e+02 6.0185e+02 6.6514e+02  Columns 67 through 72:  7.3510e+02 8.1241e+02 8.9785e+02 9.9227e+02 1.0966e+03 1.2120e+03  Columns 73 through 78:  1.3394e+03 1.4803e+03 1.6360e+03 1.8080e+03 1.9982e+03 2.2083e+03  Columns 79 through 84:  2.4406e+03 2.6973e+03 2.9810e+03 3.2945e+03 3.6410e+03 4.0239e+03  Columns 85 through 90:  4.4471e+03 4.9148e+03 5.4317e+03 6.0029e+03 6.6342e+03 7.3320e+03  Columns 91 through 96:  8.1031e+03 8.9553e+03 9.8971e+03 1.0938e+04 1.2088e+04 1.3360e+04  Columns 97 through 101:  1.4765e+04 1.6318e+04 1.8034e+04 1.9930e+04 2.2026e+04  Ysin =  Columns 1 through 8:  0 0.0998 0.1987 0.2955 0.3894 0.4794 0.5646 0.6442  Columns 9 through 16:  0.7174 0.7833 0.8415 0.8912 0.9320 0.9636 0.9854 0.9975  Columns 17 through 24:  0.9996 0.9917 0.9738 0.9463 0.9093 0.8632 0.8085 0.7457  Columns 25 through 32:  0.6755 0.5985 0.5155 0.4274 0.3350 0.2392 0.1411 0.0416  Columns 33 through 40:  -0.0584 -0.1577 -0.2555 -0.3508 -0.4425 -0.5298 -0.6119 -0.6878  Columns 41 through 48:  -0.7568 -0.8183 -0.8716 -0.9162 -0.9516 -0.9775 -0.9937 -0.9999  Columns 49 through 56:  -0.9962 -0.9825 -0.9589 -0.9258 -0.8835 -0.8323 -0.7728 -0.7055  Columns 57 through 64: | -0.6313 -0.5507 -0.4646 -0.3739 -0.2794 -0.1822 -0.0831 0.0168  Columns 65 through 72:  0.1165 0.2151 0.3115 0.4048 0.4941 0.5784 0.6570 0.7290  Columns 73 through 80:  0.7937 0.8504 0.8987 0.9380 0.9679 0.9882 0.9985 0.9989  Columns 81 through 88:  0.9894 0.9699 0.9407 0.9022 0.8546 0.7985 0.7344 0.6630  Columns 89 through 96:  0.5849 0.5010 0.4121 0.3191 0.2229 0.1245 0.0248 -0.0752  Columns 97 through 101:  -0.1743 -0.2718 -0.3665 -0.4575 -0.5440  Ycos =  Columns 1 through 7:  1.000000 0.995004 0.980067 0.955336 0.921061 0.877583 0.825336  Columns 8 through 14:  0.764842 0.696707 0.621610 0.540302 0.453596 0.362358 0.267499  Columns 15 through 21:  0.169967 0.070737 -0.029200 -0.128844 -0.227202 -0.323290 -0.416147  Columns 22 through 28:  -0.504846 -0.588501 -0.666276 -0.737394 -0.801144 -0.856889 -0.904072  Columns 29 through 35:  -0.942222 -0.970958 -0.989992 -0.999135 -0.998295 -0.987480 -0.966798  Columns 36 through 42:  -0.936457 -0.896758 -0.848100 -0.790968 -0.725932 -0.653644 -0.574824  Columns 43 through 49:  -0.490261 -0.400799 -0.307333 -0.210796 -0.112153 -0.012389 0.087499  Columns 50 through 56:  0.186512 0.283662 0.377978 0.468517 0.554374 0.634693 0.708670  Columns 57 through 63:  0.775566 0.834713 0.885520 0.927478 0.960170 0.983268 0.996542  Columns 64 through 70:  0.999859 0.993185 0.976588 0.950233 0.914383 0.869397 0.815725  Columns 71 through 77:  0.753902 0.684547 0.608351 0.526078 0.438547 0.346635 0.251260  Columns 78 through 84:  0.153374 0.053955 -0.046002 -0.145500 -0.243544 -0.339155 -0.431377  Columns 85 through 91:  -0.519289 -0.602012 -0.678720 -0.748647 -0.811093 -0.865435 -0.911130  Columns 92 through 98:  -0.947722 -0.974844 -0.992225 -0.999693 -0.997172 -0.984688 -0.962365  Columns 99 through 101:  -0.930426 -0.889191 -0.839072  pa = -6.7602  pexp = -3.4376  psin = -6.7385  pcos = -3.4401 |
|  | | |

**DISCUSSION and CONCLUSION**

The several MATLAB commands have been explored and successfully used to create vectors, and their row-wise and column-wise sum and cumulative sum are calculated with functions sum and cumsum respectively. Also, the running sum is computed with looping construct. Vectors and matrices are also created using rand function. Finally, sine, exponential and cosine plots of a vector X are generated using sin, exp and cosine functions respectively.

**PRECAUTIONS**

* Don’t forget to save the code after every change you make.
* Use MATLAB properly.
* MATLAB requires a stable network connection.
* Save the file after compiling the code and take the required notes and screenshots, so that you don’t have to open octave and do everything again.

|  |  |  |  |
| --- | --- | --- | --- |
| **Criteria** | **Total Marks** | **Marks Obtained** | **Comments** |
| Concept (A) | 2 |  |  |
| Implementation (B) | 2 |  |  |
| Performance (C) | 2 |  |  |
| Total | 6 |  |  |

**EXPERIMENT- 4**

**AIM**

Evaluating a given expression and rounding it to the nearest integer value using Round, Floor, Ceil

and Fix functions.

Also, generating and Plots of

1. Trigonometric Functions - sin(t), cos(t), tan(t), sec(t), cosec(t) and cot(t) for a given duration, ‘t’.
2. Logarithmic and other Functions – log(A), log10(A), Square root of A, Real nth root of A.

**SOFTWARE USED**

Octave Online - <https://octave-online.net/>

**PROGRAM CODE**

**%** **Evaluating a given expression and rounding it to the nearest integer value using Round, Floor, Ceil**

**and Fix functions.**

|  |
| --- |
| round(99.9987)  ceil(99.9)  ceil(-99.9)  floor(99.9)  floor(-99.9)  fix(50.55)  fix(-50.55) |

**% (A) Trigonometric Functions - sin(t), cos(t), tan(t), sec(t), cosec(t) and cot(t) for a given duration, ‘t’.**

|  |  |
| --- | --- |
| #Trigonomeetric Functions  t = -pi:0.1:pi  subplot(3,2,1)  plot(t,sin(t))  xlabel('t')  ylabel('sin(t)')  title('Plot of sin(t)')  subplot(3,2,2)  plot(t,cos(t))  xlabel('t')  ylabel('cos(t)')  title('Plot of cos(t)')  subplot(3,2,3)  plot(t,tangent)  xlabel('t')  ylabel('tan(t)')  title('Plot of tan(t)')  t1 = -pi+0.01:0.01:-0.01 | t2 = 0.01:0.01:pi-0.01  subplot(3,2,4)  plot(t,csc(t))  xlabel('t')  ylabel('cosec(t)')  title('Plot of cosec(t)')  t1 = -pi/2+0.01:0.01:pi/2-0.01;  t2 = pi/2+0.01:0.01:(3\*pi/2)-0.01;  subplot(3,2,5)  plot(t,sec(t))  xlabel('t')  ylabel('sec(t)')  title('Plot of sec(t)')  t1 = -pi+0.01:0.01:-0.01;  t2 = 0.01:0.01:pi-0.01;  subplot(3,2,6)  plot(t,cot(t))  xlabel('t')  ylabel('cot(t)')  title('Plot of cot(t)') |

**% (B) Logarithmic and other Functions – log(A), log10(A), Square root of A, Real nth root of A.**

|  |  |
| --- | --- |
| x = 1:20  subplot(4,1,1)  plot(x,log(x))  xlabel('Values of x')  ylabel('log(x)')  title('Plot of log(x)')  subplot(4,1,2)  plot(x,log10(x))  xlabel('Values of x')  ylabel('log10(x)')  title('Plot of log10(x)') | subplot(4,1,3)  plot(x,sqrt(x))  xlabel('Values of x')  ylabel('sqrt(x)')  title('Plot of sqrt(x)')  subplot(4,1,4)  plot(x,nthroot(x,5))  xlabel('Values of x')  ylabel('5th root(x)')  title('Plot of 5th root(x)') |

**RESULTS**

Evaluating a given expression and rounding it to the nearest integer value using Round, Floor, Ceil

and Fix functions.

|  |
| --- |
| ans = 100  ans = 100  ans = -99  ans = 99  ans = -100  ans = 50  ans = -50 |

(A) Trigonometric Functions - sin(t), cos(t), tan(t), sec(t), cosec(t) and cot(t) for a given duration, ‘t’.

|  |  |  |
| --- | --- | --- |
| t =  Columns 1 through 8:  -3.1416 -3.0416 -2.9416 -2.8416 -2.7416 -2.6416 -2.5416 -2.4416  Columns 9 through 16:  -2.3416 -2.2416 -2.1416 -2.0416 -1.9416 -1.8416 -1.7416 -1.6416  Columns 17 through 24:  -1.5416 -1.4416 -1.3416 -1.2416 -1.1416 -1.0416 -0.9416 -0.8416  Columns 25 through 32:  -0.7416 -0.6416 -0.5416 -0.4416 -0.3416 -0.2416 -0.1416 -0.0416  Columns 33 through 40:  0.0584 0.1584 0.2584 0.3584 0.4584 0.5584 0.6584 0.7584  Columns 41 through 48:  0.8584 0.9584 1.0584 1.1584 1.2584 1.3584 1.4584 1.5584  Columns 49 through 56:  1.6584 1.7584 1.8584 1.9584 2.0584 2.1584 2.2584 2.3584  Columns 57 through 63:  2.4584 2.5584 2.6584 2.7584 2.8584 2.9584 3.0584  t1 =  Columns 1 through 6:  -3.131593 -3.121593 -3.111593 -3.101593 -3.091593 -3.081593  Columns 7 through 12:  -3.071593 -3.061593 -3.051593 -3.041593 -3.031593 -3.021593  Columns 13 through 18:  -3.011593 -3.001593 -2.991593 -2.981593 -2.971593 -2.961593  Columns 19 through 24:  -2.951593 -2.941593 -2.931593 -2.921593 -2.911593 -2.901593  Columns 25 through 30:  -2.891593 -2.881593 -2.871593 -2.861593 -2.851593 -2.841593  Columns 31 through 36:  -2.831593 -2.821593 -2.811593 -2.801593 -2.791593 -2.781593  Columns 37 through 42:  -2.771593 -2.761593 -2.751593 -2.741593 -2.731593 -2.721593  Columns 43 through 48:  -2.711593 -2.701593 -2.691593 -2.681593 -2.671593 -2.661593  Columns 49 through 54:  -2.651593 -2.641593 -2.631593 -2.621593 -2.611593 -2.601593  Columns 55 through 60:  -2.591593 -2.581593 -2.571593 -2.561593 -2.551593 -2.541593  Columns 61 through 66:  -2.531593 -2.521593 -2.511593 -2.501593 -2.491593 -2.481593  Columns 67 through 72:  -2.471593 -2.461593 -2.451593 -2.441593 -2.431593 -2.421593  Columns 73 through 78:  -2.411593 -2.401593 -2.391593 -2.381593 -2.371593 -2.361593  Columns 79 through 84:  -2.351593 -2.341593 -2.331593 -2.321593 -2.311593 -2.301593  Columns 85 through 90:  -2.291593 -2.281593 -2.271593 -2.261593 -2.251593 -2.241593  Columns 91 through 96:  -2.231593 -2.221593 -2.211593 -2.201593 -2.191593 -2.181593  Columns 97 through 102:  -2.171593 -2.161593 -2.151593 -2.141593 -2.131593 -2.121593  Columns 103 through 108:  -2.111593 -2.101593 -2.091593 -2.081593 -2.071593 -2.061593  Columns 109 through 114:  -2.051593 -2.041593 -2.031593 -2.021593 -2.011593 -2.001593  Columns 115 through 120:  -1.991593 -1.981593 -1.971593 -1.961593 -1.951593 -1.941593  Columns 121 through 126:  -1.931593 -1.921593 -1.911593 -1.901593 -1.891593 -1.881593  Columns 127 through 132:  -1.871593 -1.861593 -1.851593 -1.841593 -1.831593 -1.821593  Columns 133 through 138:  -1.811593 -1.801593 -1.791593 -1.781593 -1.771593 -1.761593  Columns 139 through 144:  -1.751593 -1.741593 -1.731593 -1.721593 -1.711593 -1.701593  Columns 145 through 150:  -1.691593 -1.681593 -1.671593 -1.661593 -1.651593 -1.641593  Columns 151 through 156:  -1.631593 -1.621593 -1.611593 -1.601593 -1.591593 -1.581593  Columns 157 through 162:  -1.571593 -1.561593 -1.551593 -1.541593 -1.531593 -1.521593  Columns 163 through 168:  -1.511593 -1.501593 -1.491593 -1.481593 -1.471593 -1.461593  Columns 169 through 174:  -1.451593 -1.441593 -1.431593 -1.421593 -1.411593 -1.401593  Columns 175 through 180: | -1.391593 -1.381593 -1.371593 -1.361593 -1.351593 -1.341593  Columns 181 through 186:  -1.331593 -1.321593 -1.311593 -1.301593 -1.291593 -1.281593  Columns 187 through 192:  -1.271593 -1.261593 -1.251593 -1.241593 -1.231593 -1.221593  Columns 193 through 198:  -1.211593 -1.201593 -1.191593 -1.181593 -1.171593 -1.161593  Columns 199 through 204:  -1.151593 -1.141593 -1.131593 -1.121593 -1.111593 -1.101593  Columns 205 through 210:  -1.091593 -1.081593 -1.071593 -1.061593 -1.051593 -1.041593  Columns 211 through 216:  -1.031593 -1.021593 -1.011593 -1.001593 -0.991593 -0.981593  Columns 217 through 222:  -0.971593 -0.961593 -0.951593 -0.941593 -0.931593 -0.921593  Columns 223 through 228:  -0.911593 -0.901593 -0.891593 -0.881593 -0.871593 -0.861593  Columns 229 through 234:  -0.851593 -0.841593 -0.831593 -0.821593 -0.811593 -0.801593  Columns 235 through 240:  -0.791593 -0.781593 -0.771593 -0.761593 -0.751593 -0.741593  Columns 241 through 246:  -0.731593 -0.721593 -0.711593 -0.701593 -0.691593 -0.681593  Columns 247 through 252:  -0.671593 -0.661593 -0.651593 -0.641593 -0.631593 -0.621593  Columns 253 through 258:  -0.611593 -0.601593 -0.591593 -0.581593 -0.571593 -0.561593  Columns 259 through 264:  -0.551593 -0.541593 -0.531593 -0.521593 -0.511593 -0.501593  Columns 265 through 270:  -0.491593 -0.481593 -0.471593 -0.461593 -0.451593 -0.441593  Columns 271 through 276:  -0.431593 -0.421593 -0.411593 -0.401593 -0.391593 -0.381593  Columns 277 through 282:  -0.371593 -0.361593 -0.351593 -0.341593 -0.331593 -0.321593  Columns 283 through 288:  -0.311593 -0.301593 -0.291593 -0.281593 -0.271593 -0.261593  Columns 289 through 294:  -0.251593 -0.241593 -0.231593 -0.221593 -0.211593 -0.201593  Columns 295 through 300:  -0.191593 -0.181593 -0.171593 -0.161593 -0.151593 -0.141593  Columns 301 through 306:  -0.131593 -0.121593 -0.111593 -0.101593 -0.091593 -0.081593  Columns 307 through 312:  -0.071593 -0.061593 -0.051593 -0.041593 -0.031593 -0.021593  Column 313:  -0.011593  t2 =  Columns 1 through 6:  0.010000 0.020000 0.030000 0.040000 0.050000 0.060000  Columns 7 through 12:  0.070000 0.080000 0.090000 0.100000 0.110000 0.120000  Columns 13 through 18:  0.130000 0.140000 0.150000 0.160000 0.170000 0.180000  Columns 19 through 24:  0.190000 0.200000 0.210000 0.220000 0.230000 0.240000  Columns 25 through 30:  0.250000 0.260000 0.270000 0.280000 0.290000 0.300000  Columns 31 through 36:  0.310000 0.320000 0.330000 0.340000 0.350000 0.360000  Columns 37 through 42:  0.370000 0.380000 0.390000 0.400000 0.410000 0.420000  Columns 43 through 48:  0.430000 0.440000 0.450000 0.460000 0.470000 0.480000  Columns 49 through 54:  0.490000 0.500000 0.510000 0.520000 0.530000 0.540000  Columns 55 through 60:  0.550000 0.560000 0.570000 0.580000 0.590000 0.600000  Columns 61 through 66:  0.610000 0.620000 0.630000 0.640000 0.650000 0.660000  Columns 67 through 72:  0.670000 0.680000 0.690000 0.700000 0.710000 0.720000  Columns 73 through 78:  0.730000 0.740000 0.750000 0.760000 0.770000 0.780000  Columns 79 through 84:  0.790000 0.800000 0.810000 0.820000 0.830000 0.840000  Columns 85 through 90: | 0.850000 0.860000 0.870000 0.880000 0.890000 0.900000  Columns 91 through 96:  0.910000 0.920000 0.930000 0.940000 0.950000 0.960000  Columns 97 through 102:  0.970000 0.980000 0.990000 1.000000 1.010000 1.020000  Columns 103 through 108:  1.030000 1.040000 1.050000 1.060000 1.070000 1.080000  Columns 109 through 114:  1.090000 1.100000 1.110000 1.120000 1.130000 1.140000  Columns 115 through 120:  1.150000 1.160000 1.170000 1.180000 1.190000 1.200000  Columns 121 through 126:  1.210000 1.220000 1.230000 1.240000 1.250000 1.260000  Columns 127 through 132:  1.270000 1.280000 1.290000 1.300000 1.310000 1.320000  Columns 133 through 138:  1.330000 1.340000 1.350000 1.360000 1.370000 1.380000  Columns 139 through 144:  1.390000 1.400000 1.410000 1.420000 1.430000 1.440000  Columns 145 through 150:  1.450000 1.460000 1.470000 1.480000 1.490000 1.500000  Columns 151 through 156:  1.510000 1.520000 1.530000 1.540000 1.550000 1.560000  Columns 157 through 162:  1.570000 1.580000 1.590000 1.600000 1.610000 1.620000  Columns 163 through 168:  1.630000 1.640000 1.650000 1.660000 1.670000 1.680000  Columns 169 through 174:  1.690000 1.700000 1.710000 1.720000 1.730000 1.740000  Columns 175 through 180:  1.750000 1.760000 1.770000 1.780000 1.790000 1.800000  Columns 181 through 186:  1.810000 1.820000 1.830000 1.840000 1.850000 1.860000  Columns 187 through 192:  1.870000 1.880000 1.890000 1.900000 1.910000 1.920000  Columns 193 through 198:  1.930000 1.940000 1.950000 1.960000 1.970000 1.980000  Columns 199 through 204:  1.990000 2.000000 2.010000 2.020000 2.030000 2.040000  Columns 205 through 210:  2.050000 2.060000 2.070000 2.080000 2.090000 2.100000  Columns 211 through 216:  2.110000 2.120000 2.130000 2.140000 2.150000 2.160000  Columns 217 through 222:  2.170000 2.180000 2.190000 2.200000 2.210000 2.220000  Columns 223 through 228:  2.230000 2.240000 2.250000 2.260000 2.270000 2.280000  Columns 229 through 234:  2.290000 2.300000 2.310000 2.320000 2.330000 2.340000  Columns 235 through 240:  2.350000 2.360000 2.370000 2.380000 2.390000 2.400000  Columns 241 through 246:  2.410000 2.420000 2.430000 2.440000 2.450000 2.460000  Columns 247 through 252:  2.470000 2.480000 2.490000 2.500000 2.510000 2.520000  Columns 253 through 258:  2.530000 2.540000 2.550000 2.560000 2.570000 2.580000  Columns 259 through 264:  2.590000 2.600000 2.610000 2.620000 2.630000 2.640000  Columns 265 through 270:  2.650000 2.660000 2.670000 2.680000 2.690000 2.700000  Columns 271 through 276:  2.710000 2.720000 2.730000 2.740000 2.750000 2.760000  Columns 277 through 282:  2.770000 2.780000 2.790000 2.800000 2.810000 2.820000  Columns 283 through 288:  2.830000 2.840000 2.850000 2.860000 2.870000 2.880000  Columns 289 through 294:  2.890000 2.900000 2.910000 2.920000 2.930000 2.940000  Columns 295 through 300:  2.950000 2.960000 2.970000 2.980000 2.990000 3.000000  Columns 301 through 306:  3.010000 3.020000 3.030000 3.040000 3.050000 3.060000  Columns 307 through 312:  3.070000 3.080000 3.090000 3.100000 3.110000 3.120000  Column 313:  3.130000 |
|  | | |

(B) Logarithmic and other Functions – log(A), log10(A), Square root of A, Real nth root of A.

|  |
| --- |
| x =  Columns 1 through 16:  1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16  Columns 17 through 20:  17 18 19 20 |

**DISCUSSION and CONCLUSION**

The several MATLAB commands have been explored and successfully used in this experiment where natural logarithm, logarithm to the base 10, square root and nth root of a vector x are calculated using log, log10, sqrt, and nth root functions provided by Octave. Further plots of these functions are plotted using the plot function.

**PRECAUTIONS**

* Don’t forget to save the code after every change you make.
* Use MATLAB properly.
* MATLAB requires a stable network connection.
* Save the file after compiling the code and take the required notes and screenshots, so that you don’t have to open octave and do everything again.

|  |  |  |  |
| --- | --- | --- | --- |
| **Criteria** | **Total Marks** | **Marks Obtained** | **Comments** |
| Concept (A) | 2 |  |  |
| Implementation (B) | 2 |  |  |
| Performance (C) | 2 |  |  |
| Total | 6 |  |  |

**EXPERIMENT- 5**

**AIM**

Creating a vector X with elements, and adding up 100 elements of the vector, X; And, plotting the functions, , , , over the interval 0 < x < 4 (by choosing appropriate mesh values for x to obtain smooth curves), on a Rectangular Plot.

**SOFTWARE USED**

Octave Online - <https://octave-online.net/>

**PROGRAM CODE**

**%** **Creating a vector X with elements, and adding up 100 elements of the vector X**

|  |
| --- |
| for n = 1:100  X(n) = ((-1)^(n+1))/(2\*n-1)  end  sumX = sum(X) |

**% Plotting the functions, , , , over the interval 0 < x < 4, on a Rectangular Plot**

|  |  |
| --- | --- |
| t = 0.04:0.04:4  subplot(4,2,1)  plot(t,X)  xlabel('Values of t')  ylabel('X')  title('Plot of X')  subplot(4,2,2)  plot(t,X.^3)  xlabel('Values of t')  ylabel('X^3')  title('Plot of X^3') | subplot(4,2,3)  plot(t,exp(X))  xlabel('Values of t')  ylabel('e^X')  title('Plot of e^X')  subplot(4,2,4)  plot(t,exp(X.^2))  xlabel('Values of t')  ylabel('exp(X^2)')  title('Plot of exp(X^2)') |

**RESULTS**

Creating a vector X with elements, and adding up 100 elements of the vector X

|  |  |  |
| --- | --- | --- |
| X =  Columns 1 through 6:  1.0000e+00 -3.3333e-01 2.0000e-01 -1.4286e-01 1.1111e-01 -9.0909e-02  Columns 7 through 12:  7.6923e-02 -6.6667e-02 5.8824e-02 -5.2632e-02 4.7619e-02 -4.3478e-02  Columns 13 through 18:  4.0000e-02 -3.7037e-02 3.4483e-02 -3.2258e-02 3.0303e-02 -2.8571e-02  Columns 19 through 24:  2.7027e-02 -2.5641e-02 2.4390e-02 -2.3256e-02 2.2222e-02 -2.1277e-02  Columns 25 through 30:  2.0408e-02 -1.9608e-02 1.8868e-02 -1.8182e-02 1.7544e-02 -1.6949e-02  Columns 31 through 36: | 1.6393e-02 -1.5873e-02 1.5385e-02 -1.4925e-02 1.4493e-02 -1.4085e-02  Columns 37 through 42:  1.3699e-02 -1.3333e-02 1.2987e-02 -1.2658e-02 1.2346e-02 -1.2048e-02  Columns 43 through 48:  1.1765e-02 -1.1494e-02 1.1236e-02 -1.0989e-02 1.0753e-02 -1.0526e-02  Columns 49 through 54:  1.0309e-02 -1.0101e-02 9.9010e-03 -9.7087e-03 9.5238e-03 -9.3458e-03  Columns 55 through 60:  9.1743e-03 -9.0090e-03 8.8496e-03 -8.6957e-03 8.5470e-03 -8.4034e-03  Columns 61 through 66:  8.2645e-03 -8.1301e-03 8.0000e-03 -7.8740e-03 7.7519e-03 -7.6336e-03  Columns 67 through 72: | 7.5188e-03 -7.4074e-03 7.2993e-03 -7.1942e-03 7.0922e-03 -6.9930e-03  Columns 73 through 78:  6.8966e-03 -6.8027e-03 6.7114e-03 -6.6225e-03 6.5359e-03 -6.4516e-03  Columns 79 through 84:  6.3694e-03 -6.2893e-03 6.2112e-03 -6.1350e-03 6.0606e-03 -5.9880e-03  Columns 85 through 90:  5.9172e-03 -5.8480e-03 5.7803e-03 -5.7143e-03 5.6497e-03 -5.5866e-03  Columns 91 through 96:  5.5249e-03 -5.4645e-03 5.4054e-03 -5.3476e-03 5.2910e-03 -5.2356e-03  Columns 97 through 100:  5.1813e-03 -5.1282e-03 5.0761e-03 -5.0251e-03  sumX = 0.7829 |

Plotting the functions, , , , over the interval 0 < x < 4, on a Rectangular Plot

|  |  |
| --- | --- |
| t =  Columns 1 through 6:  0.040000 0.080000 0.120000 0.160000 0.200000 0.240000  Columns 7 through 12:  0.280000 0.320000 0.360000 0.400000 0.440000 0.480000  Columns 13 through 18:  0.520000 0.560000 0.600000 0.640000 0.680000 0.720000  Columns 19 through 24:  0.760000 0.800000 0.840000 0.880000 0.920000 0.960000  Columns 25 through 30:  1.000000 1.040000 1.080000 1.120000 1.160000 1.200000  Columns 31 through 36:  1.240000 1.280000 1.320000 1.360000 1.400000 1.440000  Columns 37 through 42:  1.480000 1.520000 1.560000 1.600000 1.640000 1.680000  Columns 43 through 48:  1.720000 1.760000 1.800000 1.840000 1.880000 1.920000  Columns 49 through 54: | 1.960000 2.000000 2.040000 2.080000 2.120000 2.160000  Columns 55 through 60:  2.200000 2.240000 2.280000 2.320000 2.360000 2.400000  Columns 61 through 66:  2.440000 2.480000 2.520000 2.560000 2.600000 2.640000  Columns 67 through 72:  2.680000 2.720000 2.760000 2.800000 2.840000 2.880000  Columns 73 through 78:  2.920000 2.960000 3.000000 3.040000 3.080000 3.120000  Columns 79 through 84:  3.160000 3.200000 3.240000 3.280000 3.320000 3.360000  Columns 85 through 90:  3.400000 3.440000 3.480000 3.520000 3.560000 3.600000  Columns 91 through 96:  3.640000 3.680000 3.720000 3.760000 3.800000 3.840000  Columns 97 through 100:  3.880000 3.920000 3.960000 4.000000 |
|  | |

**DISCUSSION and CONCLUSION**

The several MATLAB commands have been explored and successfully used to create a vector with its individual elements are found by computing its function and as such, exponential curves and cubic curves of a vector X are plotted using element wise method of exponentiation X.^3, X.^2. The curves were possible with the use of function exp, which gives exponential values of the argument.

**PRECAUTIONS**

* Don’t forget to save the code after every change you make.
* Use MATLAB properly.
* MATLAB requires a stable network connection.
* Save the file after compiling the code and take the required notes and screenshots, so that you don’t have to open octave and do everything again.

|  |  |  |  |
| --- | --- | --- | --- |
| **Criteria** | **Total Marks** | **Marks Obtained** | **Comments** |
| Concept (A) | 2 |  |  |
| Implementation (B) | 2 |  |  |
| Performance (C) | 2 |  |  |
| Total | 6 |  |  |

**EXPERIMENT- 6**

**AIM**

Generating a sinusoidal signal of a given frequency (say 100Hz) and plotting with graphical enhancements: titling, labelling, adding text, adding legends, adding new plots to existing plots, printings text in Greek letters, plotting as multiple subplots.

**SOFTWARE USED**

Octave Online - <https://octave-online.net/>

**PROGRAM CODE**

**% Generating a sinusoidal signal of a given frequency (say 100Hz) and plotting with graphical enhancements: titling, labelling, adding text, adding legends, adding new plots to existing plots, printings text in Greek letters, plotting as multiple subplots.**

|  |  |
| --- | --- |
| t = 0:0.05\*pi:2\*pi  x = sin(t)  plot(t,x,'r-')  xlabel('t')  ylabel('sine(t)')  title('Plot of sin(t)')  legend('sine(t)')  text(pi,0,'\Leftarrow sin(\pi)','FontSize',18)  hold on  x1 = sin(3\*t)  plot(t,x1,'b+-')  xlabel('radians') | ylabel('amplitude')  title('Plot of sin(3{\alpha}) over existing wave sin({\alpha})')  legend('sine(t)','sin(3t)')  text(pi,0,'\Leftarrow sin(3({\alpha}))','FontSize',18)  x2 = sin(4\*t)  plot(t,x2,'g\*-')  xlabel('t')  ylabel('sine(t)')  title('Plot of sin(t)')  legend('sine(t)','sin(3t)','sin(4t)')  text(pi,0,'\Leftarrow sin(\pi)','FontSize',18)  hold off |

**RESULTS**

Generating a sinusoidal signal of a given frequency (say 100Hz) and plotting with graphical enhancements: titling, labelling, adding text, adding legends, adding new plots to existing plots, printings text in Greek letters, plotting as multiple subplots.

|  |  |
| --- | --- |
| t =  Columns 1 through 8:  0 0.1571 0.3142 0.4712 0.6283 0.7854 0.9425 1.0996  Columns 9 through 16:  1.2566 1.4137 1.5708 1.7279 1.8850 2.0420 2.1991 2.3562  Columns 17 through 24:  2.5133 2.6704 2.8274 2.9845 3.1416 3.2987 3.4558 3.6128  Columns 25 through 32:  3.7699 3.9270 4.0841 4.2412 4.3982 4.5553 4.7124 4.8695  Columns 33 through 40:  5.0265 5.1836 5.3407 5.4978 5.6549 5.8119 5.9690 6.1261  Column 41:  6.2832  x =  Columns 1 through 8:  0 0.1564 0.3090 0.4540 0.5878 0.7071 0.8090 0.8910  Columns 9 through 16:  0.9511 0.9877 1.0000 0.9877 0.9511 0.8910 0.8090 0.7071  Columns 17 through 24:  0.5878 0.4540 0.3090 0.1564 0.0000 -0.1564 -0.3090 -0.4540  Columns 25 through 32:  -0.5878 -0.7071 -0.8090 -0.8910 -0.9511 -0.9877 -1.0000 -0.9877  Columns 33 through 40:  -0.9511 -0.8910 -0.8090 -0.7071 -0.5878 -0.4540 -0.3090 -0.1564  Column 41:  -0.0000  ans = -20.950  x1 = | Columns 1 through 8:  0 0.4540 0.8090 0.9877 0.9511 0.7071 0.3090 -0.1564  Columns 9 through 16:  -0.5878 -0.8910 -1.0000 -0.8910 -0.5878 -0.1564 0.3090 0.7071  Columns 17 through 24:  0.9511 0.9877 0.8090 0.4540 0.0000 -0.4540 -0.8090 -0.9877  Columns 25 through 32:  -0.9511 -0.7071 -0.3090 0.1564 0.5878 0.8910 1.0000 0.8910  Columns 33 through 40:  0.5878 0.1564 -0.3090 -0.7071 -0.9511 -0.9877 -0.8090 -0.4540  Column 41:  -0.0000  ans = -20.950  x2 =  Columns 1 through 8:  0 0.5878 0.9511 0.9511 0.5878 0.0000 -0.5878 -0.9511  Columns 9 through 16:  -0.9511 -0.5878 -0.0000 0.5878 0.9511 0.9511 0.5878 0.0000  Columns 17 through 24:  -0.5878 -0.9511 -0.9511 -0.5878 -0.0000 0.5878 0.9511 0.9511  Columns 25 through 32:  0.5878 0.0000 -0.5878 -0.9511 -0.9511 -0.5878 -0.0000 0.5878  Columns 33 through 40:  0.9511 0.9511 0.5878 0.0000 -0.5878 -0.9511 -0.9511 -0.5878  Column 41:  -0.0000  ans = -20.950 |
|  | |

**DISCUSSION and CONCLUSION**

In this experiment, the specifications of a plot are added where to add the requirements specified in the question xlabel, ylabel, and text functions are used. Also sin() function is used with different arguments is used to generate the sine curve and some Greek letters are printed in the text of the plot.

**PRECAUTIONS**

* Don’t forget to save the code after every change you make.
* Use MATLAB properly.
* MATLAB requires a stable network connection.
* Save the file after compiling the code and take the required notes and screenshots, so that you don’t have to open octave and do everything again.

|  |  |  |  |
| --- | --- | --- | --- |
| **Criteria** | **Total Marks** | **Marks Obtained** | **Comments** |
| Concept (A) | 2 |  |  |
| Implementation (B) | 2 |  |  |
| Performance (C) | 2 |  |  |
| Total | 6 |  |  |

**EXPERIMENT- 7**

**AIM**

Writing brief script starting each script with a request for input (using input() function) to evaluate the function h(T) using if-else statement, where,

h(T) = (T-10) for 0<T<100

h(T) = (0.45T + 900) for T>100

**SOFTWARE USED**

Octave Online - <https://octave-online.net/>

**PROGRAM CODE**

**% Writing brief script starting each script with a request for input to evaluate the function h(T) using if-else statement**

|  |
| --- |
| H = 0  T = input('Enter the value of T: ')  if (T<0)  disp('Enter a value greater then 0')  else if (0<T) && (T<100)  fprintf('For T = %d,\n\t',T)  H = T-10  else if (T>100)  fprintf('For T = %d,\n\t',T)  H = (0.45\*T)+900  end  end  end |

**RESULTS**

Writing brief script starting each script with a request for input to evaluate the function h(T) using if-else statement

|  |  |
| --- | --- |
| **octave:27>** source("exp7.m")  H = 0  Enter the value of T: **>** -25  T = -25  Enter a value greater then 0 | **octave:29>** source("exp7.m")  H = 0  Enter the value of T: **>** 25  T = 25  For T = 25,  H = 15 |
| **octave:28>** source("exp7.m")  H = 0  Enter the value of T: **>** 150  T = 150  For T = 150,  H = 967.50 |  |

**DISCUSSION and CONCLUSION**

The MATLAB/ Octave functionality have been explored and successfully implemented to take input from user and to solve the problem statement if-else statements were used to process different kinds of inputs based on the range defined in the question.

**PRECAUTIONS**

* Don’t forget to save the code after every change you make.
* Use MATLAB properly.
* MATLAB requires a stable network connection.
* Save the file after compiling the code and take the required notes and screenshots, so that you don’t have to open octave and do everything again.

|  |  |  |  |
| --- | --- | --- | --- |
| **Criteria** | **Total Marks** | **Marks Obtained** | **Comments** |
| Concept (A) | 2 |  |  |
| Implementation (B) | 2 |  |  |
| Performance (C) | 2 |  |  |
| Total | 6 |  |  |

**EXPERIMENT- 8**

**AIM**

Solving first order differential equation using the built-in functions. Consider the following differential equation

, where , 1<x<3 and y = 4.2

**SOFTWARE USED**

Octave Online - <https://octave-online.net/>

**PROGRAM CODE**

**% Solving first order differential equation using the built-in functions**

|  |
| --- |
| ode1 = @(x,y)(x^3-2\*y)/x  [x,y] = ode45(ode1,[1.01:0.01:3],4.2)  plot(x,y,'linewidth',2)  xlabel('x')  ylabel('y')  grid on  title('Solution to ODE, dy/dx = (x^3-2\*y)/x') |

**RESULTS**

Solving first order differential equation using the built-in functions

|  |  |  |  |
| --- | --- | --- | --- |
| ode1 =  @(x, y) (x ^ 3 - 2 \* y) / x  x =  1.0100  1.0200  1.0300  1.0400  1.0500  1.0600  1.0700  1.0800  1.0900  1.1000  1.1100  1.1200  1.1300  1.1400  1.1500  1.1600  1.1700  1.1800  1.1900  1.2000  1.2100  1.2200  1.2300  1.2400  1.2500  1.2600  1.2700  1.2800  1.2900  1.3000  1.3100  1.3200  1.3300  1.3400  1.3500  1.3600  1.3700  1.3800  1.3900  1.4000  1.4100  1.4200  1.4300  1.4400  1.4500  1.4600  1.4700  1.4800  1.4900  1.5000  1.5100  1.5200  1.5300  1.5400  1.5500  1.5600  1.5700  1.5800  1.5900  1.6000  1.6100  1.6200  1.6300  1.6400  1.6500  1.6600  1.6700  1.6800  1.6900  1.7000  1.7100  1.7200  1.7300  1.7400  1.7500  1.7600  1.7700  1.7800  1.7900  1.8000  1.8100  1.8200  1.8300  1.8400  1.8500  1.8600  1.8700  1.8800  1.8900  1.9000  1.9100  1.9200  1.9300  1.9400  1.9500  1.9600  1.9700 | 1.9800  1.9900  2.0000  2.0100  2.0200  2.0300  2.0400  2.0500  2.0600  2.0700  2.0800  2.0900  2.1000  2.1100  2.1200  2.1300  2.1400  2.1500  2.1600  2.1700  2.1800  2.1900  2.2000  2.2100  2.2200  2.2300  2.2400  2.2500  2.2600  2.2700  2.2800  2.2900  2.3000  2.3100  2.3200  2.3300  2.3400  2.3500  2.3600  2.3700  2.3800  2.3900  2.4000  2.4100  2.4200  2.4300  2.4400  2.4500  2.4600  2.4700  2.4800  2.4900  2.5000  2.5100  2.5200  2.5300  2.5400  2.5500  2.5600  2.5700  2.5800  2.5900  2.6000  2.6100  2.6200  2.6300  2.6400  2.6500  2.6600  2.6700  2.6800  2.6900  2.7000  2.7100  2.7200  2.7300  2.7400  2.7500  2.7600  2.7700  2.7800  2.7900  2.8000  2.8100  2.8200  2.8300  2.8400  2.8500  2.8600  2.8700  2.8800  2.8900  2.9000  2.9100  2.9200  2.9300  2.9400  2.9500  2.9600  2.9700  2.9800  2.9900  3.0000 | y =  4.2000  4.1283  4.0589  3.9918  3.9270  3.8642  3.8036  3.7449  3.6882  3.6333  3.5803  3.5289  3.4793  3.4313  3.3849  3.3400  3.2966  3.2546  3.2141  3.1749  3.1371  3.1005  3.0652  3.0310  2.9981  2.9663  2.9357  2.9061  2.8776  2.8502  2.8237  2.7983  2.7738  2.7502  2.7276  2.7058  2.6850  2.6650  2.6458  2.6275  2.6099  2.5932  2.5772  2.5620  2.5475  2.5338  2.5207  2.5084  2.4967  2.4858  2.4755  2.4658  2.4568  2.4484  2.4406  2.4334  2.4269  2.4209  2.4155  2.4107  2.4064  2.4027  2.3996  2.3970  2.3949  2.3934  2.3924  2.3919  2.3919  2.3924  2.3934  2.3949  2.3968  2.3993  2.4022  2.4056  2.4095  2.4138  2.4186  2.4239  2.4296  2.4357  2.4423  2.4493  2.4567  2.4646  2.4729  2.4817  2.4908  2.5004  2.5104  2.5208  2.5316  2.5428  2.5544  2.5665  2.5789  2.5917  2.6049 | 2.6186  2.6326  2.6470  2.6618  2.6769  2.6925  2.7084  2.7248  2.7415  2.7586  2.7761  2.7939  2.8121  2.8307  2.8497  2.8691  2.8888  2.9089  2.9293  2.9502  2.9714  2.9930  3.0149  3.0372  3.0599  3.0829  3.1063  3.1301  3.1542  3.1787  3.2036  3.2288  3.2544  3.2803  3.3066  3.3333  3.3604  3.3878  3.4155  3.4436  3.4721  3.5010  3.5302  3.5598  3.5897  3.6200  3.6506  3.6817  3.7130  3.7448  3.7769  3.8093  3.8422  3.8754  3.9089  3.9428  3.9771  4.0118  4.0468  4.0822  4.1179  4.1540  4.1905  4.2273  4.2645  4.3021  4.3400  4.3783  4.4170  4.4561  4.4955  4.5353  4.5754  4.6159  4.6568  4.6981  4.7398  4.7818  4.8242  4.8669  4.9101  4.9536  4.9975  5.0417  5.0864  5.1314  5.1768  5.2226  5.2688  5.3153  5.3622  5.4096  5.4573  5.5053  5.5538  5.6026  5.6519  5.7015  5.7515  5.8019  5.8527 |
| Chart, line chart, histogram  Description automatically generated | | | |

**DISCUSSION and CONCLUSION**

To achieve the solution for first order differential equations, the MATLAB/Octave’s built-in functions are used. The ode45() function is used to solve the ode given in the problem statement, it takes definition of the ode as the first argument, range as second parameter and initial condition as third parameter.

As such, the solution for the required differential equation is successfully obtained.

**PRECAUTIONS**

* Don’t forget to save the code after every change you make.
* Use MATLAB properly.
* MATLAB requires a stable network connection.
* Save the file after compiling the code and take the required notes and screenshots, so that you don’t have to open octave and do everything again.

|  |  |  |  |
| --- | --- | --- | --- |
| **Criteria** | **Total Marks** | **Marks Obtained** | **Comments** |
| Concept (A) | 2 |  |  |
| Implementation (B) | 2 |  |  |
| Performance (C) | 2 |  |  |
| Total | 6 |  |  |

**EXPERIMENT- 9**

**AIM**

Generating a square wave from sum of sine waves of certain amplitude and frequency

**SOFTWARE USED**

Octave Online - <https://octave-online.net/>

**PROGRAM CODE**

**%** **Generating a square wave from sum of sine waves of certain amplitude and frequency**

|  |  |
| --- | --- |
| disp(pi)  t = 0:0.05:2  length(t)  f = 1  A = 4  amp = 2\*pi\*f  sum = 0  sine = 0  for n = 1:2:100  y = sin(n\*amp\*t)/n  sum = sum + y    end | sum = sum \* (4\*A/pi)  for n = 1:100  z = sin(amp\*t)/n  sine = sine + z  end  hold on  plot(t,sum)  plot(t,sine,"g-+")  hold off |

**RESULTS**

Generating a square wave from sum of sine waves of certain amplitude and frequency

|  |  |
| --- | --- |
| 3.1416  t =  Columns 1 through 8:  0 0.0500 0.1000 0.1500 0.2000 0.2500 0.3000 0.3500  Columns 9 through 16:  0.4000 0.4500 0.5000 0.5500 0.6000 0.6500 0.7000 0.7500  Columns 17 through 24:  0.8000 0.8500 0.9000 0.9500 1.0000 1.0500 1.1000 1.1500  Columns 25 through 32:  1.2000 1.2500 1.3000 1.3500 1.4000 1.4500 1.5000 1.5500  Columns 33 through 40:  1.6000 1.6500 1.7000 1.7500 1.8000 1.8500 1.9000 1.9500  Column 41:  2.0000  ans = 41  f = 1  A = 4  amp = 6.2832  sum = 0  sine = 0  y =  Columns 1 through 7:  0 -0.003121 -0.005937 -0.008172 -0.009607 -0.010101 -0.009607  Columns 8 through 14:  -0.008172 -0.005937 -0.003121 -0.000000 0.003121 0.005937 0.008172  Columns 15 through 21:  0.009607 0.010101 0.009607 0.008172 0.005937 0.003121 0.000000  Columns 22 through 28:  -0.003121 -0.005937 -0.008172 -0.009607 -0.010101 -0.009607 -0.008172  Columns 29 through 35:  -0.005937 -0.003121 -0.000000 0.003121 0.005937 0.008172 0.009607  Columns 36 through 41:  0.010101 0.009607 0.008172 0.005937 0.003121 0.000000  sum =  Columns 1 through 8:  0 0.7692 0.7769 0.7792 0.7801 0.7804 0.7801 0.7792  Columns 9 through 16:  0.7769 0.7692 -0.0000 -0.7692 -0.7769 -0.7792 -0.7801 -0.7804  Columns 17 through 24:  -0.7801 -0.7792 -0.7769 -0.7692 0.0000 0.7692 0.7769 0.7792  Columns 25 through 32:  0.7801 0.7804 0.7801 0.7792 0.7769 0.7692 -0.0000 -0.7692  Columns 33 through 40:  -0.7769 -0.7792 -0.7801 -0.7804 -0.7801 -0.7792 -0.7769 -0.7692 | Column 41:  -0.0000  sum =  Columns 1 through 8:  0 3.9178 3.9567 3.9685 3.9732 3.9745 3.9732 3.9685  Columns 9 through 16:  3.9567 3.9178 -0.0000 -3.9178 -3.9567 -3.9685 -3.9732 -3.9745  Columns 17 through 24:  -3.9732 -3.9685 -3.9567 -3.9178 0.0000 3.9178 3.9567 3.9685  Columns 25 through 32:  3.9732 3.9745 3.9732 3.9685 3.9567 3.9178 -0.0000 -3.9178  Columns 33 through 40:  -3.9567 -3.9685 -3.9732 -3.9745 -3.9732 -3.9685 -3.9567 -3.9178  Column 41:  -0.0000  z =  Columns 1 through 7:  0 0.003090 0.005878 0.008090 0.009511 0.010000 0.009511  Columns 8 through 14:  0.008090 0.005878 0.003090 0.000000 -0.003090 -0.005878 -0.008090  Columns 15 through 21:  -0.009511 -0.010000 -0.009511 -0.008090 -0.005878 -0.003090 -0.000000  Columns 22 through 28:  0.003090 0.005878 0.008090 0.009511 0.010000 0.009511 0.008090  Columns 29 through 35:  0.005878 0.003090 0.000000 -0.003090 -0.005878 -0.008090 -0.009511  Columns 36 through 41:  -0.010000 -0.009511 -0.008090 -0.005878 -0.003090 -0.000000  sine =  Columns 1 through 8:  0 1.6030 3.0491 4.1967 4.9335 5.1874 4.9335 4.1967  Columns 9 through 16:  3.0491 1.6030 0.0000 -1.6030 -3.0491 -4.1967 -4.9335 -5.1874  Columns 17 through 24:  -4.9335 -4.1967 -3.0491 -1.6030 -0.0000 1.6030 3.0491 4.1967  Columns 25 through 32:  4.9335 5.1874 4.9335 4.1967 3.0491 1.6030 0.0000 -1.6030  Columns 33 through 40:  -3.0491 -4.1967 -4.9335 -5.1874 -4.9335 -4.1967 -3.0491 -1.6030  Column 41:  -0.0000 |
|  | |

**DISCUSSION and CONCLUSION**

In this experiment, the concept of loops and trigonometric functions are used for applying mathematical operations such that the terms for square wave using the sum of sine waves is obtained.

Using the computed terms and previously defined vector are used to plot the square wave.

**PRECAUTIONS**

* Don’t forget to save the code after every change you make.
* Use MATLAB properly.
* MATLAB requires a stable network connection.
* Save the file after compiling the code and take the required notes and screenshots, so that you don’t have to open octave and do everything again.

|  |  |  |  |
| --- | --- | --- | --- |
| **Criteria** | **Total Marks** | **Marks Obtained** | **Comments** |
| Concept (A) | 2 |  |  |
| Implementation (B) | 2 |  |  |
| Performance (C) | 2 |  |  |
| Total | 6 |  |  |

**EXPERIMENT- 10**

**AIM**

Basic 2D and 3D plots: parametric space curve, polygons with vertices, 3D contour lines and pie and bar charts.

**SOFTWARE USED**

Octave Online - <https://octave-online.net/>

MATLAB ONLINE - [https://matlab.mathworks.com](https://matlab.mathworks.comt)/

**PROGRAM CODE**

**%** **Basic 2D and 3D plots**

|  |
| --- |
| x = [10 15 20 25]  #Polygons with vertices  drawPolygon([0,0; 1,0; 1,1; 0,1])  drawPolygon([0,0; 2,0; 1,2])    #Bar Graphs  bar(x)  figure  bar3(x)  #Pie Charts  pie(x)  figure  pie3(x)  #3D Contour lines  [X,Y,Z] = peaks;  contour(X,Y,Z) |

**RESULTS**

**%** **Basic 2D and 3D plots**

|  |
| --- |
| x =  10 15 20 25 |

**DISCUSSION and CONCLUSION**

The various MATLAB commands have been explored and successfully used to plot the required 2D and 3D plots. Using some predefined functions in MATLAB, namely – drawPolygon, bar, bar3, pie, pie3 and contour, polygons with vertices, bar graphs & pie charts in both 2-D and 3-D, contour plots have been illustrated marking the completion of the experiment.

**PRECAUTIONS**

* Don’t forget to save the code after every change you make.
* Use MATLAB properly.
* MATLAB requires a stable network connection.
* Save the file after compiling the code and take the required notes and screenshots, so that you don’t have to open octave and do everything again.

|  |  |  |  |
| --- | --- | --- | --- |
| **Criteria** | **Total Marks** | **Marks Obtained** | **Comments** |
| Concept (A) | 2 |  |  |
| Implementation (B) | 2 |  |  |
| Performance (C) | 2 |  |  |
| Total | 6 |  |  |