Experiment 1

Aim: Creating a One and Two-Dimensional Array (Row / Column Vector) (Matrix of given size) then,

* Performing Arithmetic Operations - Addition, Subtraction, Multiplication and Exponentiation
* Performing Matrix operations - Inverse, Transpose, Rank with plots.

Software used: Octave/Octave online (<https://octave-online.net>)

Procedure:

|  |
| --- |
| %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) |

Output:

|  |
| --- |
| 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:

The vectors in this experiment - row vector, column vector and matrices are created using the shorthand method, all arithmetic operations are performed using both the predefined functions that octave provisions along with using operator notation as well. Matrix operations such as determinant, transpose etc. are also performed with both predefined functions and element wise operations.

Result:

The vectors and matrices were created and output for both the methods – predefined functions and element wise operations is same.

Conclusion:

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

* Relational Operations -
* Logical Operations -

Software used: Octave/Octave online (<https://octave-online.net>)

Procedure:

|  |
| --- |
| %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) |

Output:

|  |
| --- |
| 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:

In this experiment matrices are created, and matrix manipulation functions are applied on the same matrices. The functions include rotation using , sorting using , reshaping using , flipping the matrices using . Logical operations are applied simply using the logical operators.

Result:

Outputs for every function mentioned above applied on the defined matrices are projected in the output section.

Conclusion:

Several functions were successfully applied on the matrices.

Experiment 3A

Aim: Generating a set of Commands on a given Vector (example: ) to

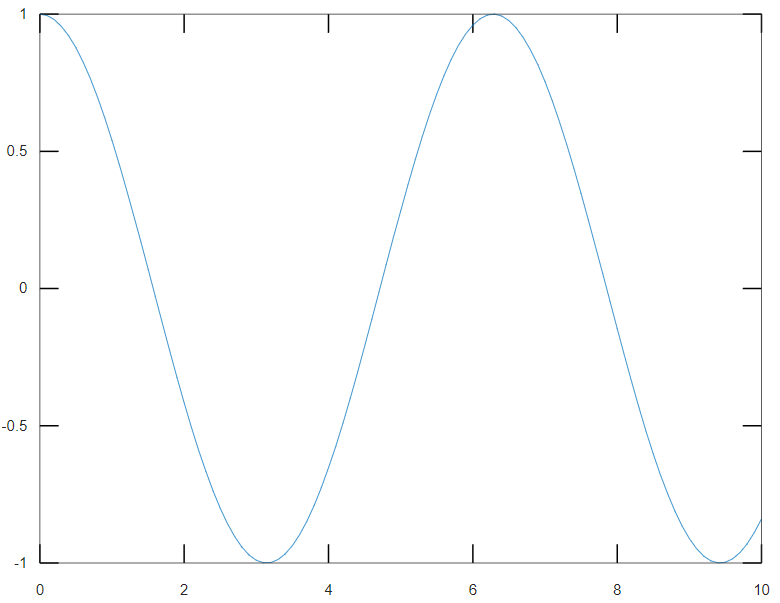
* Add up the values of the elements (Check with sum)
* Compute the Running Sum (Check with sum), where Running Sum for element j = the sum of the elements from 1 to j, inclusive
* Generating a Random Sequence using functions and plot them.

Software used: Octave/Octave online (<https://octave-online.net>)

Procedure:

|  |
| --- |
| 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]  #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  #Cumulative sum of elements of a VECTOR  sca = cumsum(A)  #Cumulative sum of elements of a MATRIX  scb = cumsum(B) #cumulative sum of elements columnwise  scbc = cumsum(B,1) #cumulative su  m of elements columnwise  scbr = cumsum(B,2) #cumulative sum of elements rowwise  #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) |

Output:



Discussion:

In this experiment vectors are created and their row-wise and column-wise, along with their cumulative sum is calculated with functions and respectively. Vectors and matrices are also created using rand function. Finally, sine, exponential and cosine plots of a vector are generated using , and functions respectively.

Result:

Sum and cumulative sum of matrices, and plots of vectors were plotted successfully.

Conclusion:

Experiment 3B

Aim:

Software used: Octave/Octave online (<https://octave-online.net>)

Procedure:

|  |
| --- |
| A = 1:10  sum = 0  for i=1 : length(A)  sum = sum + A(i)  end  B = [1 2 3;4 5 6]  for i=1 : length(B([1:end],1))  s = 0  for j=1 : length(B)  s = s + B(i,j)  end  end  disp('')  for i=1 : size(B,1)  s = 0  for j=1 : size(B,2)  s = s + B(i,j)  end  end |

Output:

|  |
| --- |
| A =  1 2 3 4 5 6 7 8 9 10  sum = 0  sum = 1  sum = 3  sum = 6  sum = 10  sum = 15  sum = 21  sum = 28  sum = 36  sum = 45  sum = 55  B =  1 2 3  4 5 6  s = 0  s = 1  s = 3  s = 6  s = 0  s = 4  s = 9  s = 15  s = 0  s = 1  s = 3  s = 6  s = 0  s = 4  s = 9  s = 15 |

Discussion:

This experiment constitutes calculating the sum of a vector A using a for loop rather than the predefined function Octave provisions, for a deeper understanding of indexing. It also encompasses calculation of row-wise sum of a matrix B.

Result:

The sum of vector A using for loop is the same as of the output of function, same goes for row-wise sum of matrix B.

Conclusion:

Sum of vector and row-wise sum of matrix were calculated using for loop successfully.

Experiment 4A

Aim: Evaluating a given expression and rounding it to the nearest integer value using

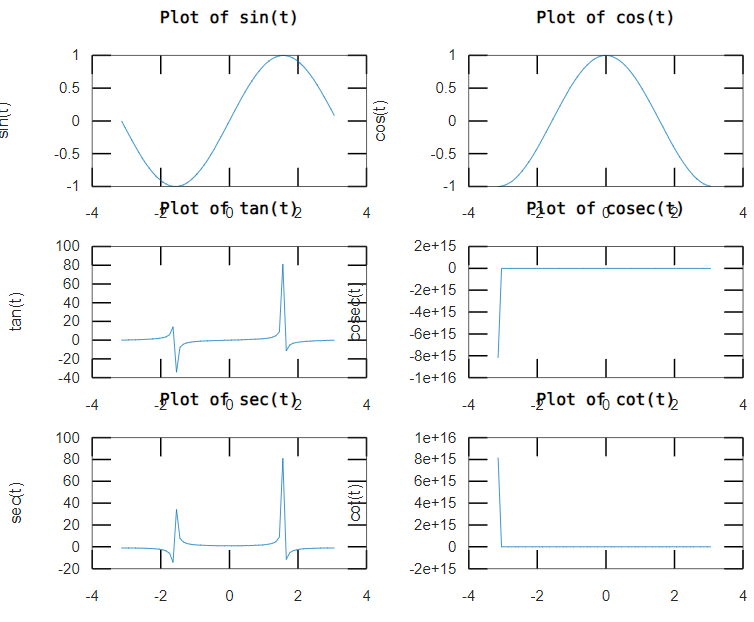
* and Fix functions; Also, generating and Plots of
* Trigonometric Functions - and for a given duration, .

Software used: Octave/Octave online (<https://octave-online.net>)

Procedure:

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

Output:



Discussion:

In this experiment all trigonometric functions – sine, cosine, tangent, cosecant, secant and cotangent are plotted using , , , , and functions provided by Octave between the range of to with a step size of 0.1.

Result:

Plots of every trigonometric functions in the range of to were generated successfully.

Conclusion:

, , , , and functions can be used to generate trigonometric plots provided a vector of values, easily.

Experiment 4B

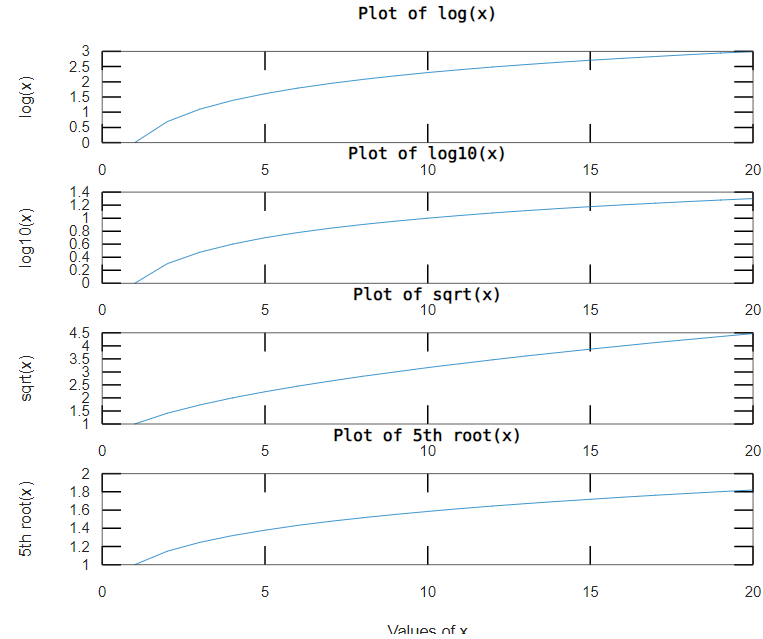
Aim:

Software used: Octave/Octave online (<https://octave-online.net>)

Procedure:

|  |
| --- |
| 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)') |

Output:



Discussion:

In this experiment, natural logarithm, logarithm to the base 10, square root and nth root of a vector are calculated using , , , and functions provided by Octave. Further plots of these functions are plotted using the function.

Result:

Plots of these functions were generated successfully.

Conclusion:

, , , and functions can be used to create logarithmic and root plots

Experiment 5

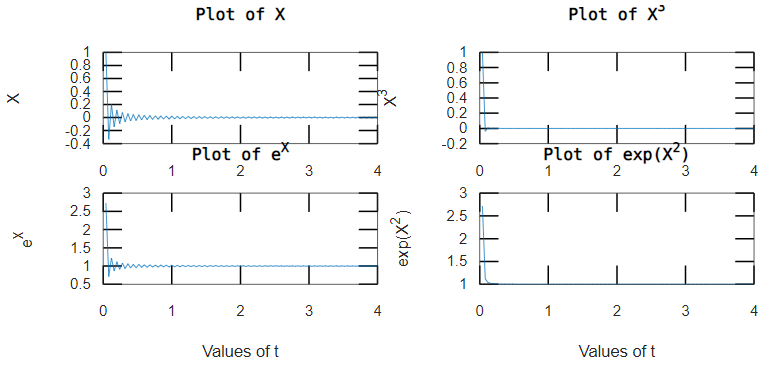
Aim: Creating a vector with elements, and adding up 100 elements of the vector, X; and plotting the functions, over the interval 0 < x < 4 (by choosing appropriate mesh values for x to obtain smooth curves), on a rectangular plot

Software used: Octave/Octave online (<https://octave-online.net>)

Procedure:

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

Output:



Discussion:

In this experiment, exponential curves and cubic curves of a vector are plotted using element wise method of exponentiation . The curves were possible with the use of function , which gives exponential values of the argument.

Result:

Exponential curves and cubic curves were plotted successfully.

Conclusion:

function can be used to get exponential values of vector and matrices.

Experiment 6

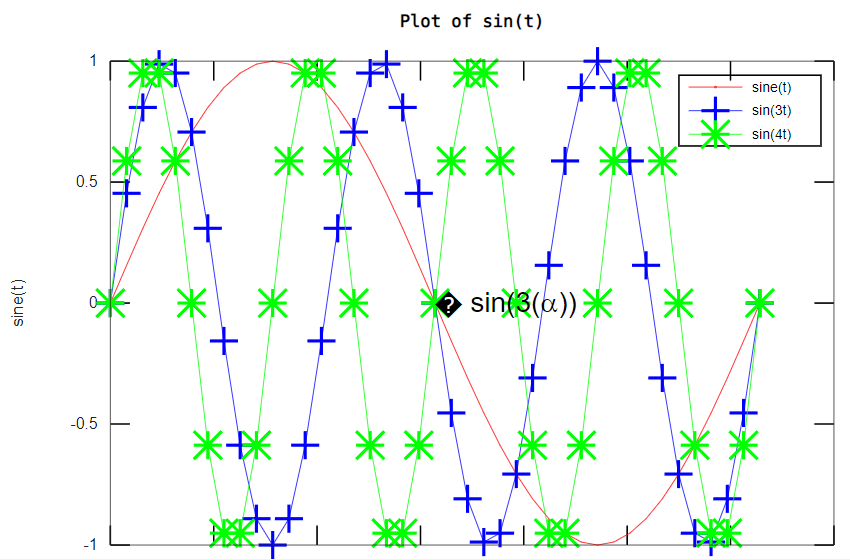
Aim: Generating a Sinusoidal Signal of a given frequency with titling, label, adding text, adding legends, printing text in Greek Letters. Time scale the generated signal for different values. E.g. 2X, 4X, 0.25X, 0.0625X

Software used: Octave/Octave online (<https://octave-online.net>)

Procedure:

|  |
| --- |
| 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 existig 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 |

Output:



Discussion:

To add the requirements specified in the question , and functions are used to add the respective. Also function is with different arguments is used to generate the sine curve.

Result:

and are added successfully.

Conclusion:

and can be used to add labels, legend and text.

Experiment 7

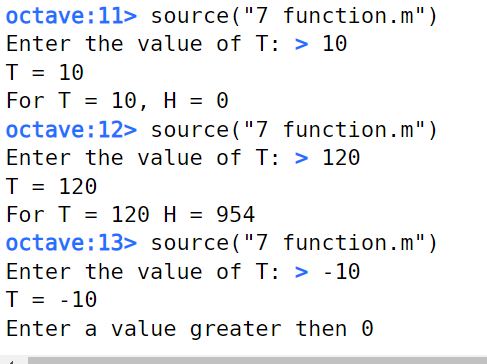
Aim: Writing brief Scripts starting each Script with a request for input (using input) to Evaluate the function using if-else statement, where

Software used: Octave/Octave online (<https://octave-online.net>)

Procedure:

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

Output:



Discussion:

To solve the problem statement if-else statements were used to process different kinds of inputs based on the range defined in the question.

Result:

The function was written successfully.

Conclusion:

If-else statements can be used to process range dependent data.

Experiment 8

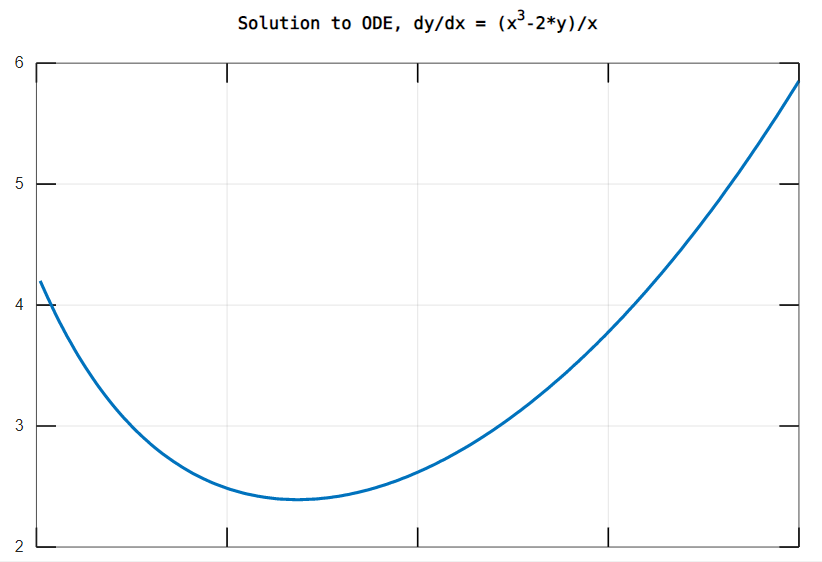
Aim: Solving First, Second and third Order Ordinary Differential Equation using Built-in Functions and plot. Consider the following ordinary differential equation

Software used: Octave/Octave online (<https://octave-online.net>)

Procedure:

|  |
| --- |
| 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') |

Output:



Discussion:

The 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.

Result:

The given ode was solved successfully.

Conclusion:

function can be used to solve ordinary differential equations.

Experiment 9

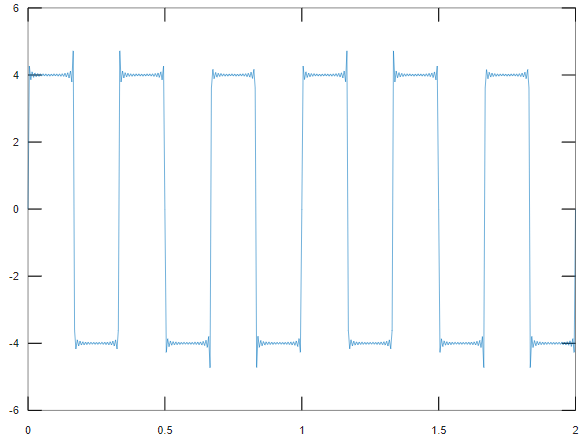
Aim: To generate a square wave using Fourier series as follows:

Software used: Octave/Octave online (<https://octave-online.net>)

Procedure:

|  |
| --- |
| t = 0:0.005:2  f = 3  A = 4  omega = 2\*pi\*f  sum = 0  for n = 1:2:100  y = sin(n\*omega\*t)/n  sum =sum + y  end  sum = sum\*(4\*A/pi)  plot(t, sum) |

Output:



Discussion:

In this experiment we are using a simple for loop, applying such operations such that odd Fourier series is obtained.

Result:

The series was plotted successfully

Conclusion:

For repetitive procedures, for loops can be used to ease up the process to very high extent.

Experiment 10

Aim: Basic 2d and 3d plots: parametric space curve, and vertices 3d contour lines, pie and bar chart.

Software used:

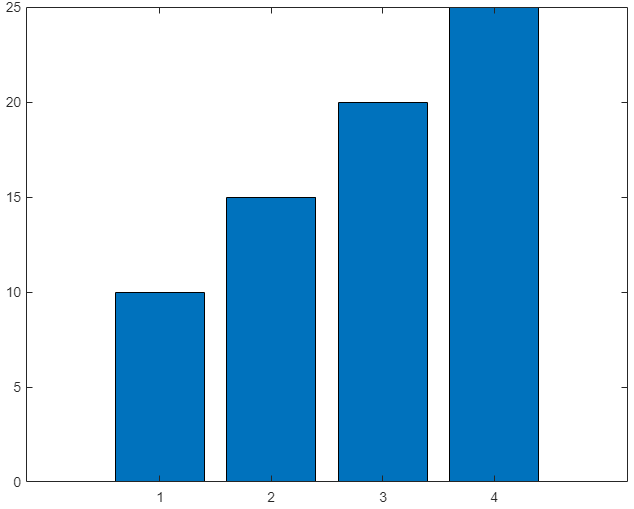
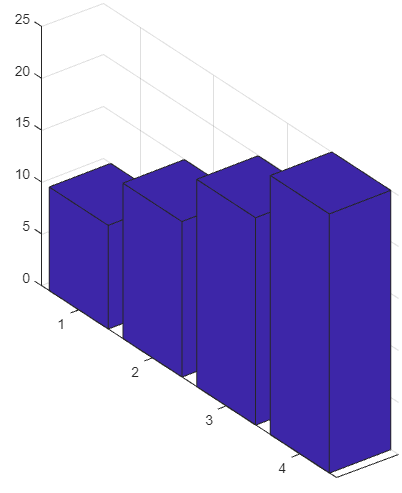
Octave/Octave online (<https://octave-online.net>)

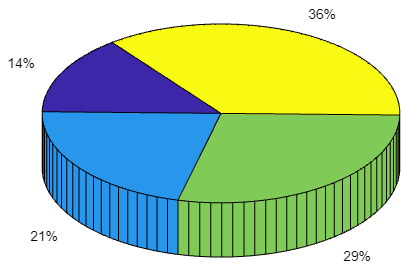
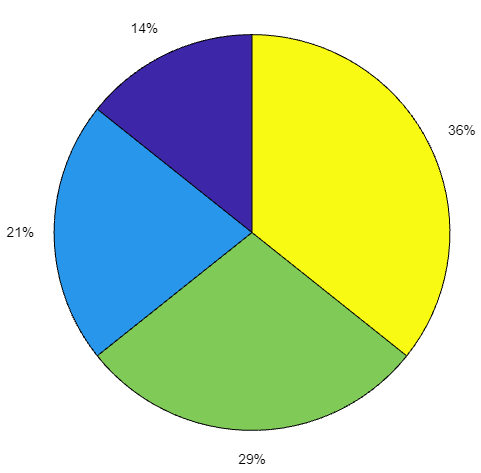
Matlab online ([https://matlab.mathworks.com](https://matlab.mathworks.comt))

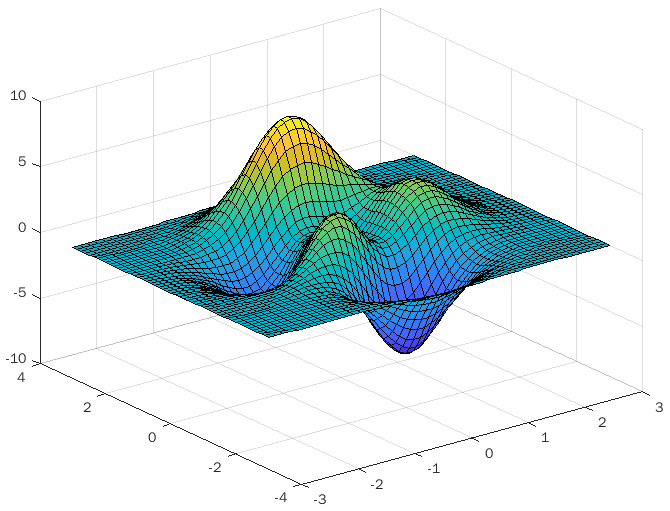
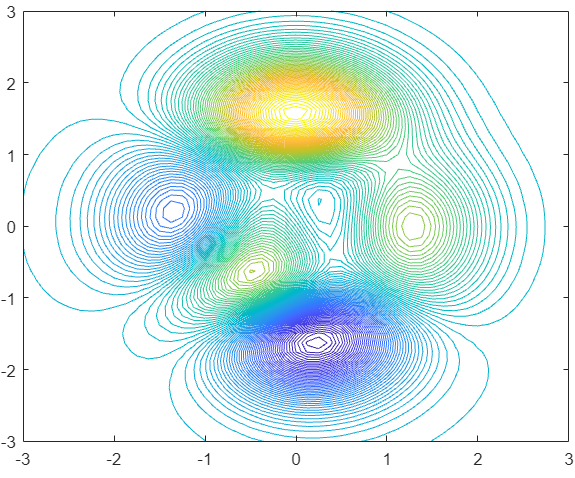
Procedure:

|  |
| --- |
| x = [10 15 20 25];  figure  bar(x) |
| x = [10 15 20 25];  figure  bar3(x) |
| x = [10 15 20 25];  figure  pie(x) |
| x = [10 15 20 25];  figure  pie3(x) |
| [X,Y,Z] = peaks;  figure  contour(X,Y,Z,100) |
| [X,Y,Z] = peaks;  figure  surf(X,Y,Z) |

Output:





Discussion:

Using some predefined functions in matlab, namely – and allows us to create bar graphs and pie charts in both 2-D and 3-D, and plots contour plots and surface plots.

Result:

2-D and 3-D plots were created successfully.

Conclusion:

The above mentioned functions allows us to plots 2-D and 3-D figures.