Table of Contents


```
% a) Learning POD Summary
% POD members: Raymond, Meghan
% Lab struggle: I did not know how to eliminate question to one expression.
% I originally did not think to use what expression to link the two parts.
% discussion with POD help: After discussed with my podmates, I learned
% that I could use "," to connect the two random portion.
```

1.5.2

1.

What is M(2,3)? Ans: M(2,3) = 6

```
M = [1, 3, 5, 7, 9; 2, 4, 6, 8, 10]
M(2,3)
```

```
M =

     1     3     5     7     9
     2     4     6     8    10


ans =
```

```
     6
```

2.

What will happen if you type MP(2,3) Ans: there is an error message Index in position 2 exceeds array bounds. Index must not exceed 2.

```
MP = M'
% MP(2, 3) // need to comm


MP =

     1     2
     3     4
     5     6
     7     8
     9    10
```

3.

What will happen if type MP(4,2)? Ans: 8

```
MP(4,2)


ans =

     8
```

1.5.4

4.

Create a row vector fo the numbers 1-100 in reverse order

```
A = flip(1:100)


A =

  Columns 1 through 13

   100    99    98    97    96    95    94    93    92    91    90    89    88

  Columns 14 through 26

    87    86    85    84    83    82    81    80    79    78    77    76    75
```

```
  Columns 27 through 39

    74    73    72    71    70    69    68    67    66    65    64    63    62

  Columns 40 through 52

    61    60    59    58    57    56    55    54    53    52    51    50    49

  Columns 53 through 65

    48    47    46    45    44    43    42    41    40    39    38    37    36

  Columns 66 through 78

    35    34    33    32    31    30    29    28    27    26    25    24    23

  Columns 79 through 91

    22    21    20    19    18    17    16    15    14    13    12    11    10

  Columns 92 through 100

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

5.

10x10 matrix with 5 for all diagonal entries and -3 for off diagonal entries

```
B = eye(10,10) * 8 + (-3) * ones(10,10)
```

```
B =

     5    -3    -3    -3    -3    -3    -3    -3    -3    -3
    -3     5    -3    -3    -3    -3    -3    -3    -3    -3
    -3    -3     5    -3    -3    -3    -3    -3    -3    -3
    -3    -3    -3     5    -3    -3    -3    -3    -3    -3
    -3    -3    -3    -3     5    -3    -3    -3    -3    -3
    -3    -3    -3    -3    -3     5    -3    -3    -3    -3
    -3    -3    -3    -3    -3    -3     5    -3    -3    -3
    -3    -3    -3    -3    -3    -3    -3     5    -3    -3
    -3    -3    -3    -3    -3    -3    -3    -3     5    -3
    -3    -3    -3    -3    -3    -3    -3    -3    -3     5
```

6.

Create a 5x3 matrix comprising random numbers selected uniformly from the range 10 to 20

```
C = 10*rand(5,3) + 10
```

```
C =

   10.7818   17.7491   12.5987
   14.4268   18.1730   18.0007
   11.0665   18.6869   14.3141
   19.6190   10.8444   19.1065
   10.0463   13.9978   11.8185
```

7.

Create a vector of length 20, whose first 15 numbers are randomly chosen between 2 and 3, and whose last 5 numbers are randomly chosen between -1 and -2

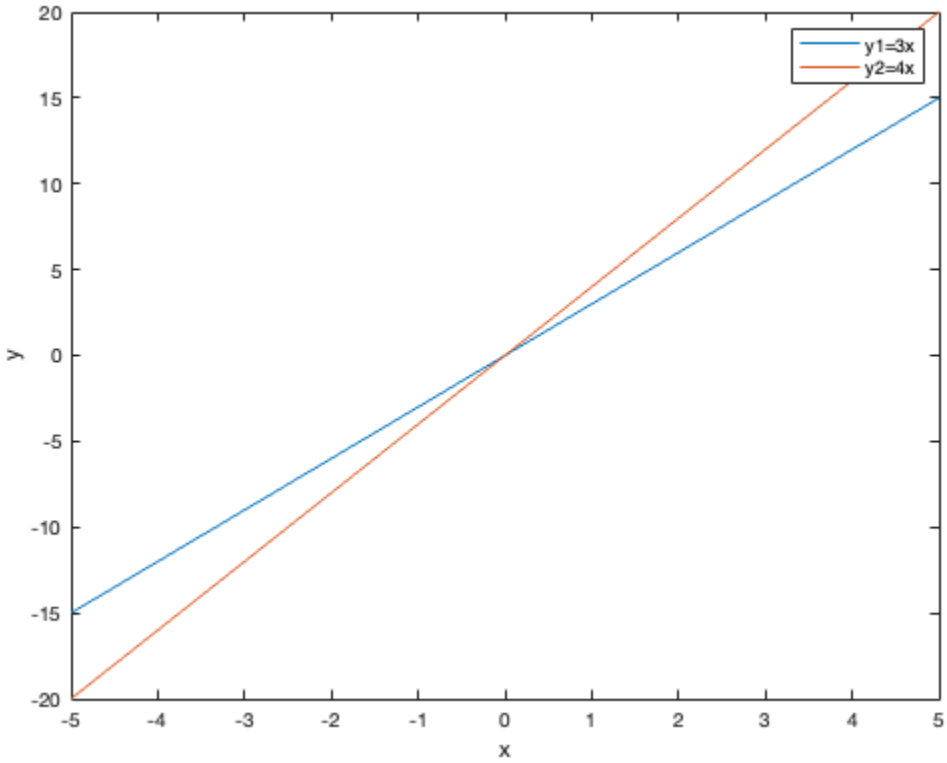
```
vector = [2 + (3-2).*rand(1,15), -2 + (-1+2).*rand(1,5)]
```

```
vector =

  Columns 1 through 7

    2.2638    2.1455    2.1361    2.8693    2.5797    2.5499    2.1450

  Columns 8 through 14

    2.8530    2.6221    2.3510    2.5132    2.4018    2.0760    2.2399

  Columns 15 through 20

    2.1233   -1.8161   -1.7600   -1.5827   -1.9503   -1.0973
```

8.

plot y = 3x for a range of x from -5 to +5 with steps of 0.1. On the same graph, in a different color, plot y = 4x. Label the axes and indicate with a legend which color corresponds to which line

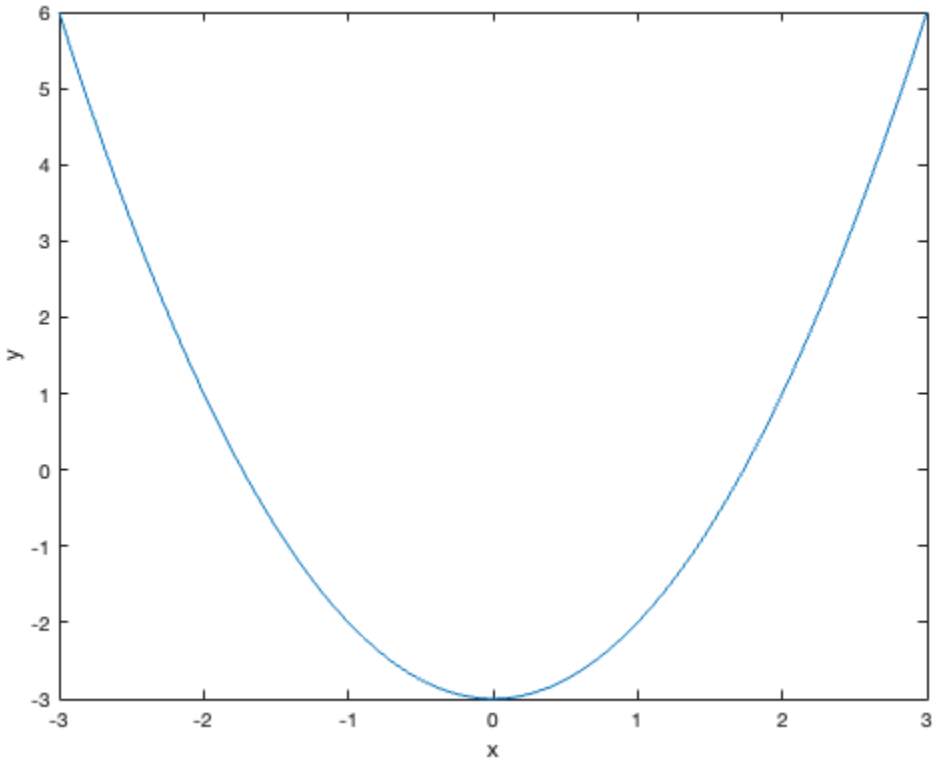
```
x = -5:0.1:5;
y1 = 3 * x;
plot(x,y1)

hold on
y2 = 4 * x;
plot(x,y2)
legend("y1=3x", "y2=4x")
xlabel("x")
ylabel("y")

hold off
```



9.

Write a code that will plot y = x^2 - 3 for a range of x from -3 to +3 with steps of 0.1. Save the code as an .m file and run that file from the command window

```
x = -3:0.1:3;
y = x .^ 2 - 3;
plot(x, y)
xlabel("x")
ylabel("y")
```

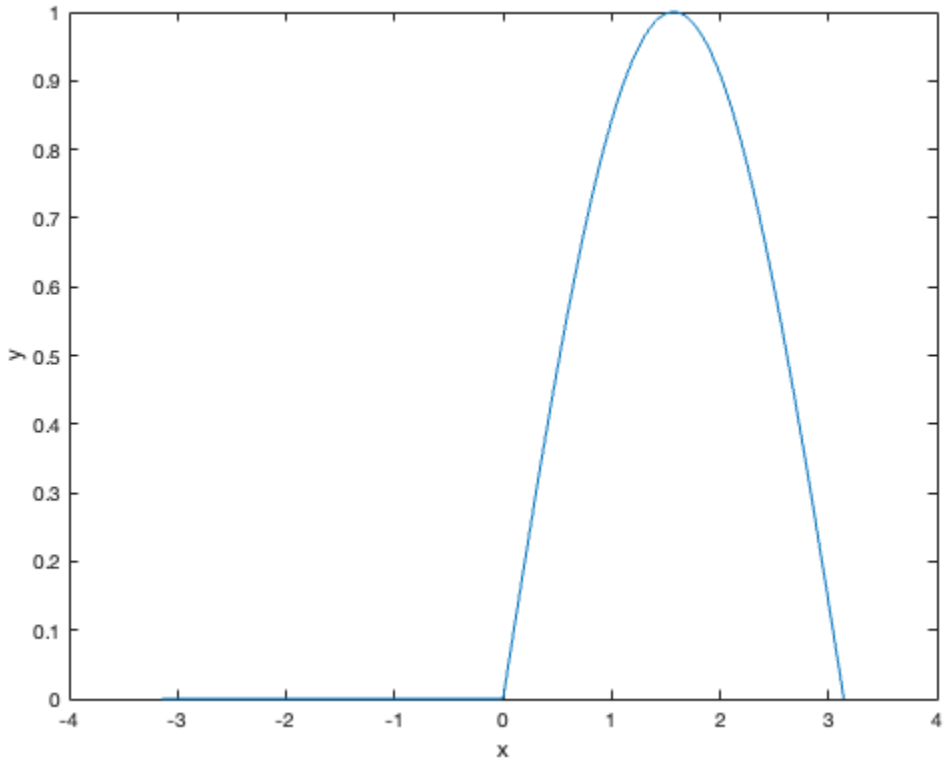


1.5.8

10.

Plot a rectified sine wave (negative values are fixed at zero) over the range of values -pi <= x <= pi

```
x = linspace(-pi, pi, 1000);
y = sin(x);
y_rectified = max(y, 0); % % The max() function compares each element of
%                           y with 0 and keeps the greater value
plot(x, y_rectified)
xlabel("x")
ylabel("y")
```



11.

Write a code to indicate whether the cube root of 6 is greater than the square root of 3.

```
cubeRoot = nthroot(6, 3);
squareRoot = sqrt(3);
if cubeRoot > squareRoot
    disp('The cube root of 6 is greater than the square root of 3.');
else
    disp('The cube root of 6 is not greater than the square root of 3.');
end
```

```
The cube root of 6 is greater than the square root of 3.
```

1.5.9

12.

Sum all the cubed positive integers up to 15^3

```
sum = 0;
for i = 1:1:15
    sum = sum + i^3;
end
disp(sum)
```

14400

13.

find the positive integer n such that n + n^2 + n^3 + n^4 = 88740

```
n = 1;
while (n + n^2 + n^3 + n^4 ~= 88740)
    n = n+1;
end
disp(n)
```

17

1.5.10

14

14a. Write a function that takes as input a single vector and returns the sum of the squares of its elements as its single output.

```
function y = square_element_sum(x)
    y = sum(x .^ 2);
end

% 14b. Use that function to sum the square of the numbers from 27 to 37
% inclusive.
sum_vector = square_element_sum(27:37);
disp(sum_vector)
```

11374

15.

Write a function that takes as input a single vector of numbers and returns the mean, the mode, and the median as three separate variables for its output.

```
function [meanVal, modeVal, medianVal] = stats(vector)
    % Calculate mean
    sumVal = 0;
    for i = 1:length(vector)
        sumVal = sumVal + vector(i);
    end
    meanVal = sumVal / length(vector);

    % Calculate mode
    % Using hist to count occurrences and finding the value with the
    % maximum count
    [counts, values] = hist(vector, unique(vector));
```

```
    [~, maxIdx] = max(counts);
    modeVal = values(maxIdx);

    % Calculate median
    sortedVector = sort(vector);
    n = length(sortedVector);
    if mod(n, 2) == 0
        medianVal = (sortedVector(n/2) + sortedVector(n/2 + 1)) / 2;
    else
        medianVal = sortedVector((n + 1) / 2);
    end
end

% Test Case 1: Odd number of elements
vector1 = [10, 2, 8, 6, 3];
[meanVal1, modeVal1, medianVal1] = stats(vector1);
disp(['Test Case 1 - Mean: ', num2str(meanVal1), ', Mode: ', ...
    num2str(modeVal1), ', Median: ', num2str(medianVal1)]);

% Test Case 2: Even number of elements
vector2 = [1, 2, 3, 4, 5, 6];
[meanVal2, modeVal2, medianVal2] = stats(vector2);
disp(['Test Case 2 - Mean: ', num2str(meanVal2), ', Mode: ', ...
    num2str(modeVal2), ', Median: ', num2str(medianVal2)]);

% Test Case 3: Repeated elements
vector3 = [4, 2, 4, 2, 4, 3];
[meanVal3, modeVal3, medianVal3] = stats(vector3);
disp(['Test Case 3 - Mean: ', num2str(meanVal3), ', Mode: ', ...
    num2str(modeVal3), ', Median: ', num2str(medianVal3)]);

% Test Case 4: A single element
vector4 = [7];
[meanVal4, modeVal4, medianVal4] = stats(vector4);
disp(['Test Case 4 - Mean: ', num2str(meanVal4), ', Mode: ', ...
    num2str(modeVal4), ', Median: ', num2str(medianVal4)]);

% Test Case 5: Negative numbers
vector5 = [-3, -1, -4, -2, -5];
[meanVal5, modeVal5, medianVal5] = stats(vector5);
disp(['Test Case 5 - Mean: ', num2str(meanVal5), ', Mode: ', ...
    num2str(modeVal5), ', Median: ', num2str(medianVal5)]);
```

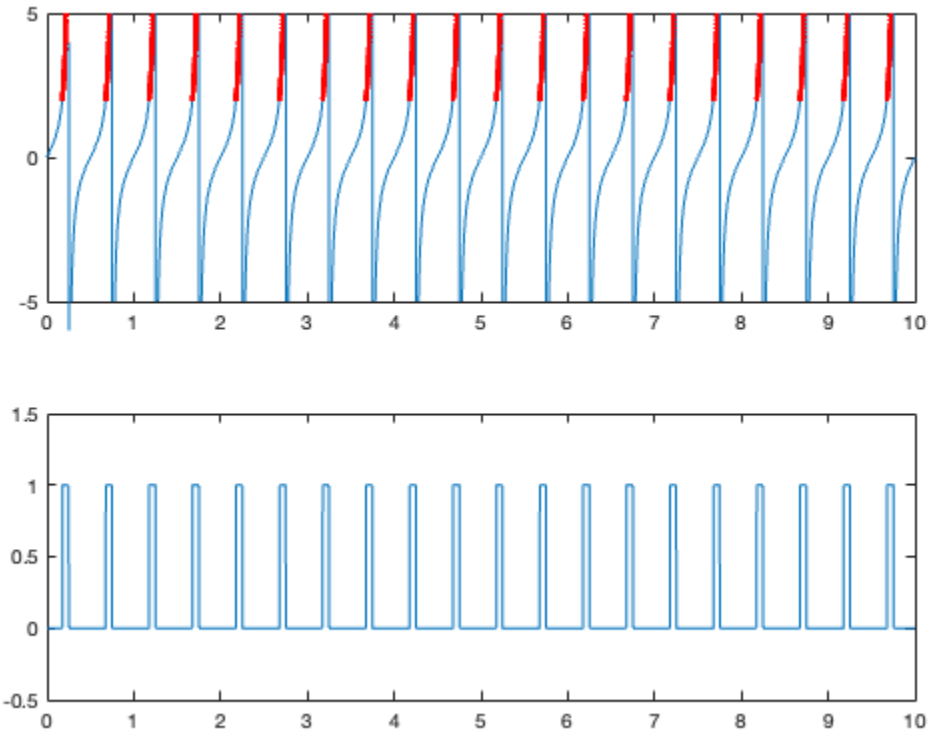
```
Test Case 1 - Mean: 5.8, Mode: 2, Median: 6
Test Case 2 - Mean: 3.5, Mode: 1, Median: 3.5
Test Case 3 - Mean: 3.1667, Mode: 4, Median: 3.5
Test Case 4 - Mean: 7, Mode: 7, Median: 7
Test Case 5 - Mean: -3, Mode: -5, Median: -3
```

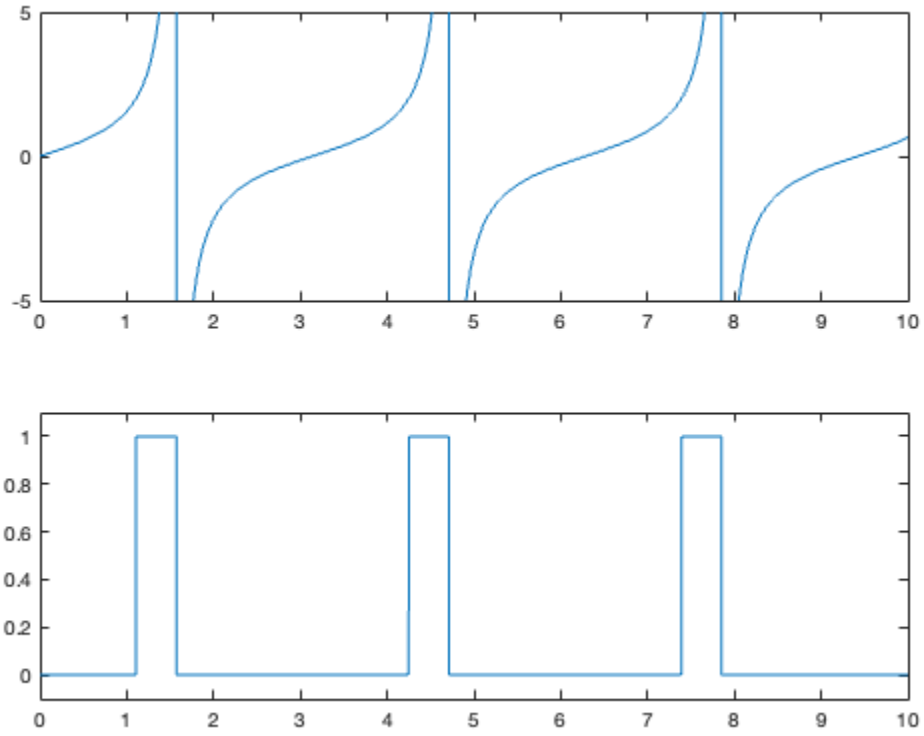
1.5.11

Suppose you want to write a code that indicates whenever the tangent function has a value greater than a threshold, here set as 2.

```
% Writing and verifying that the two code options yield the same results
% both aim to find the points at which the tangent of a time vector,
% scaled by 2pi, exceeds a certain threshold. They then plot the tangent
% function and mark the points where it surpasses this threshold.

% The first script threshold_find.m uses a for-loop to iterate over each
% element of the time vector tvector, compute the tangent value, and
% check if it's above the threshold.

% The second script threshold_find2.m uses vectorized operations to
% compute the tangent values for all time points at once and create a
% logical vector findhigh indicating where the tangent exceeds the
% threshold. This approach is more efficient because it leverages
% MATLAB's ability to operate on entire arrays at once, instead of
% looping through individual elements.

% threshold_find.m
%clear
thresh = 2;
tmax = 10;
tvector = 0:0.001:tmax;
Nt = length(tvector);
tanval = zeros(size(tvector));          % to store tan of tvector
findhigh = zeros(size(tvector));        % stores when tan > thresh
for i=1:Nt                              % for all values of tvector
    tanval(i) = tan(2*pi*tvector(i));   % set tangent of t
    if (tanval(i) > thresh)                 % if tan is high
        findhigh(i) = 1;                % store value of t end
    end
end

% Now plot the results
figure(1)
subplot(2,1,1)
plot(tvector,tanval)                    % plot tan(2.pi.t) versus t
axis([0 tmax -5 5])
subplot(2,1,2)
plot(tvector,findhigh)                  %plot t where tan(2.pi.t)>2
axis([0 tmax -0.5 1.5])
% We can color the portions corresponding to findhigh=1 using
% MATLAB’s find command, which extracts the indices of the
% non-zero entries of a matrix:
highindices = find(findhigh);           % indices above threshold
subplot(2,1,1);
hold on
plot(tvector(highindices),tanval(highindices),'r.');


% threshold_find2.m
%clear
thresh = 2;
tmax = 10;
tvector = 0:0.001:tmax;
Nt = length(tvector);
```

```
tanval = tan(tvector);                  % operates on all values at once
findhigh = tanval>thresh;               % gives 1 or 0 for all entreis

% Now plot the results
figure(2)
%clf                                    % clears figure for a new plot
subplot(2,1,1)
plot(tvector,tanval)
axis([0 tmax -5 5])
subplot(2,1,2)
plot(tvector,findhigh)
axis([0 tmax -0.1 1.1])
```





Published with MATLAB® R2024a