1. Scalar variables.

Make the following variables

a) $a = 10$

b) $b = 2.5 \times 10^{23}$

c) $c = 2 + 3i$, where **i** is the imaginary number

d) $d = e^{j2\pi/3}$, where **j** is the imaginary number and e is Euler's number (use **exp, pi**).

Code

```
clc
clear all
close all
a = 10
b = 2.5 * 10^23
c = 2 + 3i
d = exp((j*2*pi)/3)
```

Output (Command Window and/or Plots)

```
a =

    10


b =

   2.5000e+23


c =

   2.0000 + 3.0000i


d =

  -0.5000 + 0.8660i
```

2. Vector variables.

Make the following variables

a) $aVec = [3.14\ 15\ 9\ 26]$

b) $bVec = \begin{bmatrix} 2.71 \\ 8 \\ 28 \\ 182 \end{bmatrix}$

c) $cVec = [5, 4.8, \cdots -4.8, -5]$ (all the numbers from 5 to -5 in increments of -0.2)

d) $dVec = [10^0\ 10^{0.01}\ \cdots\ 10^{0.99}\ 1]$ (Logarithmically spaced numbers between 1 and 10, use **logspace**, make sure you get the length right!)

e) $eVec = Hello$ ($eVec$ is a string, which is a vector of characters)

Code

```
clc
clear all
close all
aVec = [3.14 15 9 26]
bVec = [2.71;8;28;182]
cVec = [5:-0.2:-5]
dVec = logspace (1,10)
eVec = 'Hello'
```

Output (Command Window and/or Plots)

```
aVec =

    3.1400   15.0000    9.0000   26.0000


bVec =

    2.7100
    8.0000
   28.0000
  182.0000


cVec =

  Columns 1 through 12

    5.0000    4.8000    4.6000    4.4000    4.2000    4.0000    3.8000
3.6000    3.4000    3.2000    3.0000    2.8000

  Columns 13 through 24

    2.6000    2.4000    2.2000    2.0000    1.8000    1.6000    1.4000
1.2000    1.0000    0.8000    0.6000    0.4000

  Columns 25 through 36
```

```
    0.2000         0   -0.2000   -0.4000   -0.6000   -0.8000   -1.0000   -
1.2000   -1.4000   -1.6000   -1.8000   -2.0000

  Columns 37 through 48

   -2.2000   -2.4000   -2.6000   -2.8000   -3.0000   -3.2000   -3.4000   -
3.6000   -3.8000   -4.0000   -4.2000   -4.4000

  Columns 49 through 51

   -4.6000   -4.8000   -5.0000


dVec =

   1.0e+10 *

  Columns 1 through 11

    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
0.0000    0.0000    0.0000    0.0000

  Columns 12 through 22

    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
0.0000    0.0000    0.0000    0.0000

  Columns 23 through 33

    0.0000    0.0000    0.0000    0.0000    0.0001    0.0001    0.0001
0.0002    0.0003    0.0005    0.0008

  Columns 34 through 44

    0.0012    0.0018    0.0027    0.0041    0.0063    0.0095    0.0146
0.0222    0.0339    0.0518    0.0791

  Columns 45 through 50

    0.1207    0.1842    0.2812    0.4292    0.6551    1.0000


eVec =

    'Hello'
```

3. Matrix variables.

Make the following variables

f) $aMat = \begin{bmatrix} 2 & \dots & 2 \\ \dots & \ddots & \dots \\ 2 & \dots & 2 \end{bmatrix}$ a 9x9 matrix full of 2's (use **ones** or **zeros**)

a) $bMat = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$

[1 2 3 4 5 4 3 2 1] on the main diagonal (use **zeros, diag**).

b) $cMat = \begin{bmatrix} 1 & 11 & \cdots & 91 \\ 2 & 12 & \ddots & 92 \\ \vdots & \vdots & \ddots & \vdots \\ 10 & 20 & \cdots & 100 \end{bmatrix}$ a 10x10 matrix where the vector 1:100 runs down the columns (use **reshape**).

c) $dMat = \begin{bmatrix} NaN & NaN & NaN & NaN \\ NaN & NaN & NaN & NaN \\ NaN & NaN & NaN & NaN \\ NaN & NaN & NaN & NaN \end{bmatrix}$ a 3x4 NaN matrix (use **nan**)

d) $eMat = \begin{bmatrix} 13 & -1 & 5 \\ -22 & 10 & 87 \end{bmatrix}$

e) Make $fMat$ be a 5x3 matrix of random integers with values on the range -3 to 3 (use **rand** and **floor** or **ceil**)

Code

```
clc
clear all
close all
aMat = 2*ones(9,9)
bMat = diag([1 2 3 4 5 4 3 2 1])
cMat = reshape(1:100,[10,10])
dMat = nan(3,4)
eMat = [13 -1 5;-22 10 87]
fMat = randi([-3 3],5,3)
```

Output (Command Window and/or Plots)

```
aMat =

     2     2     2     2     2     2     2     2     2
     2     2     2     2     2     2     2     2     2
     2     2     2     2     2     2     2     2     2
     2     2     2     2     2     2     2     2     2
     2     2     2     2     2     2     2     2     2
     2     2     2     2     2     2     2     2     2
     2     2     2     2     2     2     2     2     2
     2     2     2     2     2     2     2     2     2
     2     2     2     2     2     2     2     2     2
```

```
bMat =

     1     0     0     0     0     0     0     0     0
     0     2     0     0     0     0     0     0     0
     0     0     3     0     0     0     0     0     0
     0     0     0     4     0     0     0     0     0
     0     0     0     0     5     0     0     0     0
     0     0     0     0     0     4     0     0     0
     0     0     0     0     0     0     3     0     0
     0     0     0     0     0     0     0     2     0
     0     0     0     0     0     0     0     0     1

cMat =

     1    11    21    31    41    51    61    71    81    91
     2    12    22    32    42    52    62    72    82    92
     3    13    23    33    43    53    63    73    83    93
     4    14    24    34    44    54    64    74    84    94
     5    15    25    35    45    55    65    75    85    95
     6    16    26    36    46    56    66    76    86    96
     7    17    27    37    47    57    67    77    87    97
     8    18    28    38    48    58    68    78    88    98
     9    19    29    39    49    59    69    79    89    99
    10    20    30    40    50    60    70    80    90   100

dMat =

   NaN   NaN   NaN   NaN
   NaN   NaN   NaN   NaN
   NaN   NaN   NaN   NaN

eMat =

    13    -1     5
   -22    10    87


fMat =

     2     3     2
    -2     0    -2
     0    -3     2
     1    -2    -2
     3    -2     3
```

4. Scalar equations.

Using the variables created in 1, calculate x, y, and z.

a) $x = \frac{1}{1+e^{(-(a-16)/5)}}$

b) $y = \left(\sqrt{a} + \sqrt[21]{b}\right)^{\pi}$

c) $z = \frac{\log \Re[(c+d)(c-d)] \sin\left(\frac{a\pi}{3}\right)}{c\bar{c}}$ where $\Re$ indicates the real part of the complex number in brackets, c is the complex conjugate of $\bar{c}$, and log is the natural log (use **`real, conj, log`**).

Code

```
clc
clear all
close all
a = 10;
b = 2.5 * 10^23;
c = 2 + 3i;
d = exp((j*2*pi)/3);
x = 1/(1+exp(-(a-16)/5))
y = (sqrt(a)+(b)^(1/21))^pi
z = (log(real((c+d)*(c-d)) * sin(a*pi/3))) / (c * conj(c))
```

Output (Command Window and/or Plots)

```
x =

    0.2315


y =

   6.2696e+03


z =

    0.1046
```

5. Vector equations.

Using the variables created in 2, solve the equations below, elementwise. For example, in part a, the first element of $xVec$ should just be the function evaluated at the value of the first element of $cVec$

$$xVec_1 = \frac{1}{\sqrt{2\pi 2.5^2}} e^{-cVec_1^2/(2\cdot 2.5^2)}$$

and similarly for all the other elements so that $xVec$ and $cVec$ have the same size. Use the elementwise operators **.*, ./, .^.**

a) $xVec = \frac{1}{\sqrt{2\pi 2.5^2}} e^{-\frac{cVec^2}{2\cdot 2.5^2}}$

b) $yVec = \sqrt{(aVec^T)^2 + bVec^2}$, where $aVec^T$ is the transpose of $aVec$.

c) $zVec = \log_{10}\left(\frac{1}{dVec}\right)$

Code

```
clc
clear all
close all
aVec = [3.14 15 9 26];
bVec = [2.71;8;28;182];
cVec = [5:-0.2:-5];
dVec = logspace (1,10);

xVec = ((1)./(sqrt(2*pi*(2.5)^2)).*exp(-(cVec.^2)./(2*2.5^2)))
yVec = sqrt((aVec').^2 + bVec.^2)
zVec = log10(1./dVec)
```

Output (Command Window and/or Plots)

```
xVec =

  Columns 1 through 11

    0.0216    0.0253    0.0294    0.0339    0.0389    0.0444    0.0503
0.0566    0.0633    0.0703    0.0777

  Columns 12 through 22

    0.0852    0.0929    0.1007    0.1083    0.1159    0.1231    0.1300
0.1364    0.1422    0.1473    0.1516

  Columns 23 through 33

    0.1550    0.1575    0.1591    0.1596    0.1591    0.1575    0.1550
0.1516    0.1473    0.1422    0.1364

  Columns 34 through 44

    0.1300    0.1231    0.1159    0.1083    0.1007    0.0929    0.0852
0.0777    0.0703    0.0633    0.0566

  Columns 45 through 51
```

```
    0.0503    0.0444    0.0389    0.0339    0.0294    0.0253    0.0216


yVec =

    4.1477
   17.0000
   29.4109
  183.8478


zVec =

  Columns 1 through 11

   -1.0000   -1.1837   -1.3673   -1.5510   -1.7347   -1.9184   -2.1020   -
2.2857   -2.4694   -2.6531   -2.8367

  Columns 12 through 22

   -3.0204   -3.2041   -3.3878   -3.5714   -3.7551   -3.9388   -4.1224   -
4.3061   -4.4898   -4.6735   -4.8571

  Columns 23 through 33

   -5.0408   -5.2245   -5.4082   -5.5918   -5.7755   -5.9592   -6.1429   -
6.3265   -6.5102   -6.6939   -6.8776

  Columns 34 through 44

   -7.0612   -7.2449   -7.4286   -7.6122   -7.7959   -7.9796   -8.1633   -
8.3469   -8.5306   -8.7143   -8.8980

  Columns 45 through 50

   -9.0816   -9.2653   -9.4490   -9.6327   -9.8163  -10.0000
```

6. Matrix equations.

Using the variables created in 2 and 3, solve the equations below. Use matrix operators.

a) $xMat = (aVec \cdot bVec) \cdot aMat^2$

b) $yMat = (bVec \cdot aVec)$,

c) $zMat = |cMat| \cdot (aVec \cdot bVec)^T$, where $|cMat|$, is the determinant of $cMat$. (use **det**).

Code

```
clc
clear all
close all
aVec = [3.14 15 9 26];
bVec = [2.71;8;28;182];
aMat = 2*ones(9,9);
cMat = reshape(1:100,[10,10]);
xMat = (aVec*bVec)*aMat^2
yMat = (bVec*aVec)
zMat = det(cMat)*(aVec*bVec)'
```

Output (Command Window and/or Plots)

```
 xMat =
    1.0e+05 *

    1.8405    1.8405    1.8405    1.8405    1.8405    1.8405    1.8405
1.8405    1.8405
    1.8405    1.8405    1.8405    1.8405    1.8405    1.8405    1.8405
1.8405    1.8405
    1.8405    1.8405    1.8405    1.8405    1.8405    1.8405    1.8405
1.8405    1.8405
    1.8405    1.8405    1.8405    1.8405    1.8405    1.8405    1.8405
1.8405    1.8405
    1.8405    1.8405    1.8405    1.8405    1.8405    1.8405    1.8405
1.8405    1.8405
    1.8405    1.8405    1.8405    1.8405    1.8405    1.8405    1.8405
1.8405    1.8405
    1.8405    1.8405    1.8405    1.8405    1.8405    1.8405    1.8405
1.8405    1.8405
    1.8405    1.8405    1.8405    1.8405    1.8405    1.8405    1.8405
1.8405    1.8405
    1.8405    1.8405    1.8405    1.8405    1.8405    1.8405    1.8405
1.8405    1.8405

yMat =
    1.0e+03 *

    0.0085    0.0406    0.0244    0.0705
    0.0251    0.1200    0.0720    0.2080
    0.0879    0.4200    0.2520    0.7280
    0.5715    2.7300    1.6380    4.7320

zMat =

     0
```

7. Common functions and indexing

a) Make $cSum$ the column-wise sum of $cMat$. The answer should be a row vector (use **sum**).
b) Make $eMean$ the mean across the rows of $eMat$. The answer should be a column (use **mean**).
a) Replace the top row of $eMat$ with [1 1 1].
b) Make $cSub$ the submatrix of $cMat$ that only contains rows 2 through 9 and columns 2 through 9.
c) Make the vector $lin = [1\ 2\ \cdots\ 20]$ (the integers from 1 to 20), and then make every other value in it negative to get $lin = [1\ -2\ 3\ -4\ \cdots\ -20]$.
a) Make a 1x5 vector using rand. Find the elements that have values <0.5 and set those values to 0 (use find).

Code

```
clc
clear all
close all
cMat = reshape(1:100,[10,10]);
eMat = [13 -1 5;-22 10 87];
cSum  = sum(cMat)
eMean = mean(eMat)'
eMat(1,:) = 1
cSub = cMat(2:9,2:9)
lin = [1:20];
lin(2:2:end) = -lin(2:2:end)
r1 = rand(1,5)
x = find(r1<0.5);
r1(x) = 0
```

Output (Command Window and/or Plots)

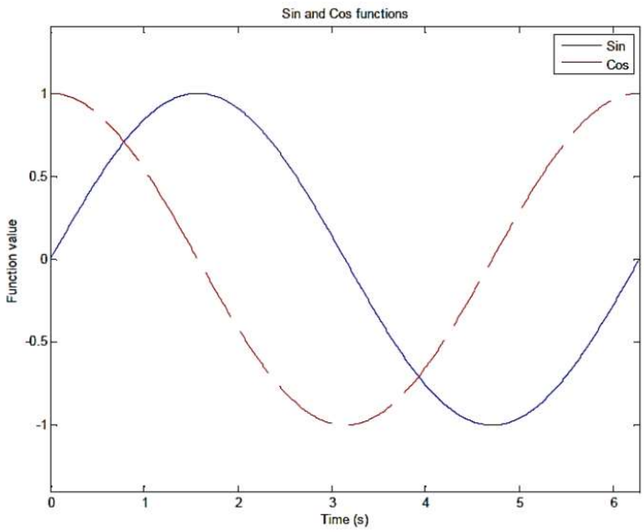
```
 cSum =

    55   155   255   355   455   555   655   755   855   955


eMean =

   -4.5000
    4.5000
   46.0000


eMat =

     1     1     1
   -22    10    87
```

```
cSub =

    12    22    32    42    52    62    72    82
    13    23    33    43    53    63    73    83
    14    24    34    44    54    64    74    84
    15    25    35    45    55    65    75    85
    16    26    36    46    56    66    76    86
    17    27    37    47    57    67    77    87
    18    28    38    48    58    68    78    88
    19    29    39    49    59    69    79    89


lin =

  Columns 1 through 19

     1    -2     3    -4     5    -6     7    -8     9   -10    11   -12
13   -14    15   -16    17   -18    19

  Column 20

   -20


r1 =

    0.3993    0.5269    0.4168    0.6569    0.6280


r1 =

         0    0.5269         0    0.6569    0.6280
```

8. Plotting multiple lines and colors.

In class we covered how to plot a single line in the default blue color on a plot. You may have noticed that subsequent plot commands simply replace the existing line. Here, we'll write a script to plot two lines on the same axes.

a) Open a script and name it **`twoLinePlot.m`**. Write the following commands in this script.
b) Make a new figure using figure
c) We'll plot a sine wave and a cosine wave over one period
   i) Make a time vector t from 0 to 2π with enough samples to get smooth lines
   ii) Plot *sin (t)*
   iii) Type hold on to turn on the '**`hold`**' property of the figure. This tells the figure not to discard lines that are already plotted when plotting new ones. Similarly, you can use **`hold off`** to turn off the hold property.
   iv) Plot $cos\ (t)$ using a red dashed line. To specify line color and style, simply add a third argument to your plot command (see the third paragraph of the **`plot`** help). This argument is a string specifying the line properties as described in the help file. For example, the string '**`k:`**' specifies a black dotted line.



d) Now, we'll add labels to the plot
   i) Label the x axis using **`xlabel`**.
   ii) Label the y axis using **`ylabel`**.
   iii) Give the figure a title using **`title`**.
e) Create a legend to describe the two lines you have plotted by using **`legend`** and passing to it the two strings '**`Sin`**' and '**`Cos`**'.
f) If you run the script now, you'll see that the x axis goes from 0 to 7 and y goes from -1 to +1. To make this look nicer, we'll manually specify the x and y limits. Use **`xlim`** to set the x axis to be from 0 to 2π and use **`ylim`** to set the y axis to be from -1.4 to 1.4.
g) Run the script to verify that everything runs right. You should see something like the figure shown above.

Code

```
clc
clear all
close all
x = (0:0.000001:2*pi);
y = (sin(x));
plot (x,y)
hold;
y = (cos(x));
plot (x,y,'r--')
legend ('Sin','Cos')
xlabel ('Time(s)');
ylabel ('Function Value');
```

```
title ('Sin and Cos Function')
xlim ([0 2*pi])
ylim ([-1.4 1.4])
```

Output (Command Window and/or Plots)

9. Manipulating variables

Write a user-defined function with function call **`val = function_eval(f,a,b)`** where *f* is an inline function, and *a* and *b* are constants such that $a < b$. The function calculates the midpoint *m* of the interval *[a,b]* and returns the value of $f(a) + (1/2) f(m) + f(b)$. Execute the function for $f = e^{x/2}$, $a = -2$, $b = 4$.

Code

Function File:

```
function val = function_eval(f,a,b)
if (a>b)
    disp('Enter correct values for a and b (a < b)')
else
    fa = feval(f,a);
    fb = feval(f,b);
    m = (a+b)/2;
    fm = feval(f,m);
    val = fa + (0.5)*fm + fb;
end
```

Main Script File:

```
clc
clear all
close all
f = inline('exp(x/2)');
a = -2;
b = 4;
val = function_eval(f,a,b);
disp(f)
fprintf('a = %d,b = %d\n',a,b)
fprintf('Value = %d\n',val)
```

Output (Command Window and/or Plots)

```
 Inline function:
     f(x) = exp(x/2)

a = -2,b = 4
Value = 8.581296e+00
```

10. Flow Control

Write a script file that employs any combination of the flow control commands to generate

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 2 & 0 & -1 & 0 & 0 \\ 2 & 0 & 3 & 0 & -1 & 0 \\ 0 & 2 & 0 & 4 & 0 & -1 \\ 0 & 0 & 2 & 0 & 5 & 0 \\ 0 & 0 & 0 & 2 & 0 & 6 \end{bmatrix}$$

Code

```
clc
clear all
close all

% Define the size of the matrix
m = 6;
n = 6;

% Initialize a matrix A with zeros
A = zeros(m, n);

% Loop through rows and columns of the matrix
for i = 1:m
    for j = 1:n

        if i == j
            A(i, j) = i;      % Assigns value of i to diagonal elements

        elseif i == j - 2
            A(i, j) = -1;     % Assigns '-1' two positions above the diag

        elseif i == j + 2
            A(i, j) = 2;      % Assigns '2' two positions below the diag
        end
    end
end

disp(A)
```

Output (Command Window and/or Plots)

```
     1     0    -1     0     0     0
     0     2     0    -1     0     0
     2     0     3     0    -1     0
     0     2     0     4     0    -1
     0     0     2     0     5     0
     0     0     0     2     0     6
```

11. Usage of Function

Write a user-defined function with function call **[r, k] = root_finder(f,x0,kmax,tol)** where f is an inline function, x_0 is a specified value, k_{max} is the maximum number of iterations, and tol is a specified tolerance. The function sets $x_1 = x_0$, calculates $|f(x_1)|$, and if it is less than the tolerance, then x_1 approximates the root r. If not, it will increment x_1 by 0.01 to obtain x_2, repeat the procedure, and so on. The process terminates as soon as $|f(x_k)| < tol$ for some k. The outputs of the function are the approximate root and the number of iterations it took to find it. Execute the function for $f(x) = x^2 - 3.3x + 2.1$, $x_0 = 0.5, k_{max} = 50, tol = 10^{-2}$.

Code

Function File:

```
function [r, k] = root_finder(f, x0, kmax, tol)
 k = 1;
 x1 = x0;
 f1 = feval(f, x1);

 while abs(f1) > tol && k <= kmax
     x1 = x1 + 0.01;
     f1 = f(x1);
     k = k + 1;
 end

 r = x1;
```

Main Script File:

```
clc
clear all
close all
% Function execution and displaying results
f = inline('(x^2) - (3.3*x) + 2.1');
x0 = 0.5;
kmax = 50;
tol = 10^(-2);

[r, k] = root_finder(f, x0, kmax, tol);

disp(f)
fprintf('Root of the given function is %f\n', r);
fprintf('Number of iterations used is %d\n', k);
```

Output (Command Window and/or Plots)

```
     Inline function:
     f(x) = (x^2) - (3.3*x) + 2.1

Root of the given function is 0.860000
Number of iterations used is 37
```