

%QS: 1. Basic Mathematical Operations

% 1.1

pi

ans = 3.1416

sqrt(2)

ans = 1.4142

exp(1)

ans = 2.7183

format long
pi

ans =
3.141592653589793

sqrt(2)

ans =
1.414213562373095

exp(1)

ans =
2.718281828459046

% 1.2

1e20

ans =
1.000000000000000e+20

3.54e-5

ans =
3.540000000000000e-05

% 1.3

exp(3)

ans =
20.085536923187668

tan(1)

ans =
1.557407724654902

asin(sin(5))

ans =

-1.283185307179586

`sin (asin(5))`

ans =
5.000000000000000 - 0.000000000000000i

`log(exp(-2 + 3*i))`

ans =
-2.000000000000000 + 3.000000000000000i

% 1.4

`z = 5.32 - 3.24*i`

z =
5.320000000000000 - 3.240000000000000i

`zc = conj(z)`

zc =
5.320000000000000 + 3.240000000000000i

`modzsq = z*zc`

modzsq =
38.800000000000004

`abs(z)^2`

ans =
38.800000000000011

`sqrtz = sqrt(modzsq^0.5)*exp(angle(z)*0.5i)`

sqrtz =
2.403015251819990 - 0.674153024527434i

`sqrt(z)`

ans =
2.403015251819990 - 0.674153024527434i

`2*real(z)`

ans =
10.640000000000001

`z+zc`

ans =
10.640000000000001

`2*imag(z)`

ans =
-6.480000000000000

`z-zc`

```
ans =
    0.000000000000000 - 6.480000000000000i
```

```
% 1.5
```

```
e = 2.718
```

```
e =
    2.718000000000000
```

```
absolute_error = (exp(1) - e)
```

```
absolute_error =
    2.818284590455633e-04
```

```
relative_error = absolute_error/exp(1)
```

```
relative_error =
    1.036788960198905e-04
```

```
%QS: 2. Vector Operations
```

```
% 2.1
```

```
u = [4,3,7]
```

```
u = 1×3
     4     3     7
```

```
% 2.2
```

```
v = [3;5;2]
```

```
v = 3×1
     3
     5
     2
```

```
t_v = [3,5,2]'
```

```
t_v = 3×1
     3
     5
     2
```

```
%2.3
```

```
u+v
```

```
ans = 3×3
     7     6    10
     9     8    12
     6     5     9
```

```
element_wise_mult = u.*v
```

```
element_wise_mult = 3×3
    12     9    21
```

```

20    15    35
8      6    14

```

```
inner_product = dot(u,v)
```

```
inner_product =
    41
```

```
u-1
```

```
ans = 1×3
      3      2      6
```

```
% 2.4
```

```
% 2.4.1
```

```
u1 = (1:5)
```

```
u1 = 1×5
      1      2      3      4      5
```

```
% 2.4.2
```

```
u2 = (1:.5:3)
```

```
u2 = 1×5
      1.000000000000000      1.500000000000000      2.000000000000000      2.500000000000000 ...
```

```
% 2.4.3
```

```
u3 = (5:-1:1)
```

```
u3 = 1×5
      5      4      3      2      1
```

```
% 2.5
```

```
row = linspace(1,10,50 )
```

```
row = 1×50
      1.000000000000000      1.183673469387755      1.367346938775510      1.551020408163265 ...
```

```
% 2.6
```

```
enteries = length((1:pi:exp(10)))
```

```
enteries =
    7011
```

```
% 2.7
```

```
rand_int = rand(100,1).'
```

```
rand_int = 1×100
      0.709364830858073      0.754686681982361      0.276025076998578      0.679702676853675 ...
```

```
[val, index] = min(rand_int)
```

```
val =
      0.004634224134067
index =
```

75

% 2.8

```
% element wise multiplication of row vector of (1x5) with
% numbers varying from 0 to pi, with another row vector of
% (1x5) such that only the elements of index 2, 5 are printed
% rest are 0.
```

```
linspace(0, pi, 5) .* [0 1 0 0 1]
```

```
ans = 1x5
      0      0.785398163397448      0      0 ...
```

% 2.9

```
A = 0:.1:10
```

```
A = 1x101
      0      0.100000000000000      0.200000000000000      0.300000000000000 ...
```

```
A = A.^2
```

```
A = 1x101
102 ×
      0      0.000100000000000      0.000400000000000      0.000900000000000 ...
```

```
result = sum(A)
```

```
result =
      3.383500000000000e+03
```

%QS: 3. Matrix Operations

% 3.1

```
ones(4,4)
```

```
ans = 4x4
      1      1      1      1
      1      1      1      1
      1      1      1      1
      1      1      1      1
```

```
eye(4)
```

```
ans = 4x4
      1      0      0      0
      0      1      0      0
      0      0      1      0
      0      0      0      1
```

% 3.2

```
M = diag([5.4, -3.9, 3.5, 8.2, 1])
```

```
M = 5x5
```

```

5.400000000000000      0      0      0 ...
      0 -3.900000000000000      0      0
      0      0 3.500000000000000      0
      0      0      0 8.199999999999999
      0      0      0      0

```

% 3.3

M1 = [1 2 3; 4 5 6; 3 8 9]

```

M1 = 3x3
      1      2      3
      4      5      6
      3      8      9

```

M2 = M1'

```

M2 = 3x3
      1      4      3
      2      5      8
      3      6      9

```

M1 + M2

```

ans = 3x3
      2      6      6
      6     10     14
      6     14     18

```

% 3.4

M3 = [3.2 -4.5; 7.6 8.1]

```

M3 = 2x2
      3.200000000000000 -4.500000000000000
      7.600000000000000  8.100000000000000

```

M3./max(M3,[],'all')

```

ans = 2x2
      0.395061728395062 -0.555555555555556
      0.938271604938272  1.000000000000000

```

% 3.5

M4 = [0.5 0.2; 0.4 0.7]

```

M4 = 2x2
      0.500000000000000  0.200000000000000
      0.400000000000000  0.700000000000000

```

M4*M4

```

ans = 2x2
      0.330000000000000  0.240000000000000
      0.480000000000000  0.570000000000000

```

M4^2

```

ans = 2x2

```

```
0.330000000000000 0.240000000000000
0.480000000000000 0.570000000000000
```

```
% 3.6
```

```
M5 = [1 1 1 1 1 1 1 1; -2 -1 0 1 2 3 4 5; 4 1 0 1 4 9 16 25]
```

```
M5 = 3×8
```

```
1 1 1 1 1 1 1 1
-2 -1 0 1 2 3 4 5
4 1 0 1 4 9 16 25
```

```
M5(3,7)
```

```
ans =
16
```

```
% 3.7
```

```
v1 = M5(3,:)
```

```
v1 = 1×8
```

```
4 1 0 1 4 9 16 25
```

```
v2 = M5(:,7)
```

```
v2 = 3×1
```

```
1
4
16
```

```
% 3.8
```

```
M
```

```
M = 5×5
```

```
5.400000000000000 0 0 0 ...
0 -3.900000000000000 0 0
0 0 3.500000000000000 0
0 0 0 8.199999999999999
0 0 0 0
```

```
% the 2nd row elements are multiplied by 3.2
```

```
M(2, :) = 3.2*M(2,:)
```

```
M = 5×5
```

```
5.400000000000000 0 0 0 ...
0 -12.480000000000000 0 0
0 0 3.500000000000000 0
0 0 0 8.199999999999999
0 0 0 0
```

```
% the 3rd row elements are added with 0.527 times the values
% of the elements of 1st row.
```

```
M(3, :) = M(3,:) + 0.527*M(1,:)
```

```
M = 5×5
```

```
5.400000000000000 0 0 0 ...
0 -12.480000000000000 0 0
```

```

2.8458000000000000      0      3.5000000000000000      0
0      0      0      8.199999999999999
0      0      0      0

```

```
% 3.9
```

```
M = [5 2 -1 1; 2 7 1 -2; 0 1 10 3; -1 3 2 9]
```

```

M = 4x4
     5     2    -1     1
     2     7     1    -2
     0     1    10     3
    -1     3     2     9

```

```
b = [13; 9; 1; 30]
```

```

b = 4x1
    13
     9
     1
    30

```

```
x = M \ b
```

```

x = 4x1
    1.000000000000000
    2.000000000000000
   -1.000000000000000
    3.000000000000000

```

```
%QS: 4.    Conditionals, Loops and User Defined Functions
```

```
% 4.1
```

```

sum_for = 0;
sz = size(M5)

```

```

sz = 1x2
     3     8

```

```

for r = 1:sz(1,1)
    for c = 1:sz(1,2)
        sum_for=sum_for+M5(r,c);
    end
end

sum_for

```

```

sum_for =
    80

```

```
sum(sum(M5))
```

```

ans =
    80

```

```
% 4.2
```



```
k=0;

while 1/k > pi/1000
    k=k+1;
end

k
```

```
k =
    319
```

```
% 4.3
```

```
mysinc(0)
```

```
ans =
     1
```

```
mysinc(2)
```

```
ans =
    0.454648713412841
```

```
sin(2)/2
```

```
ans =
    0.454648713412841
```

```
% 4.4
```

```
mfun1(0:3)
```

```
ans = 1×4
      0      0.367879441171442      0.270670566473225      0.149361205103592
```

```
mfun2(0:3)
```

```
ans = 1×4
    1.000000000000000      0    -0.135335283236613    -0.099574136735728
```

```
% 4.5
```

```
h = 1e-5;
derivative_central_approx (@sin,1,h)
```

```
ans =
    0.540302305856999
```

```
(sin(1 + h) - sin(1 - h))/(2*h)
```

```
ans =
    0.540302305856999
```

```
derivative_central_approx (@mfun1, 1, h)
```

```
ans =
    1.387778780781446e-11
```

```
mfun2(1)
```

```
ans =  
    0
```

```
%QS: 5.    Plots
```

```
% 5.1
```

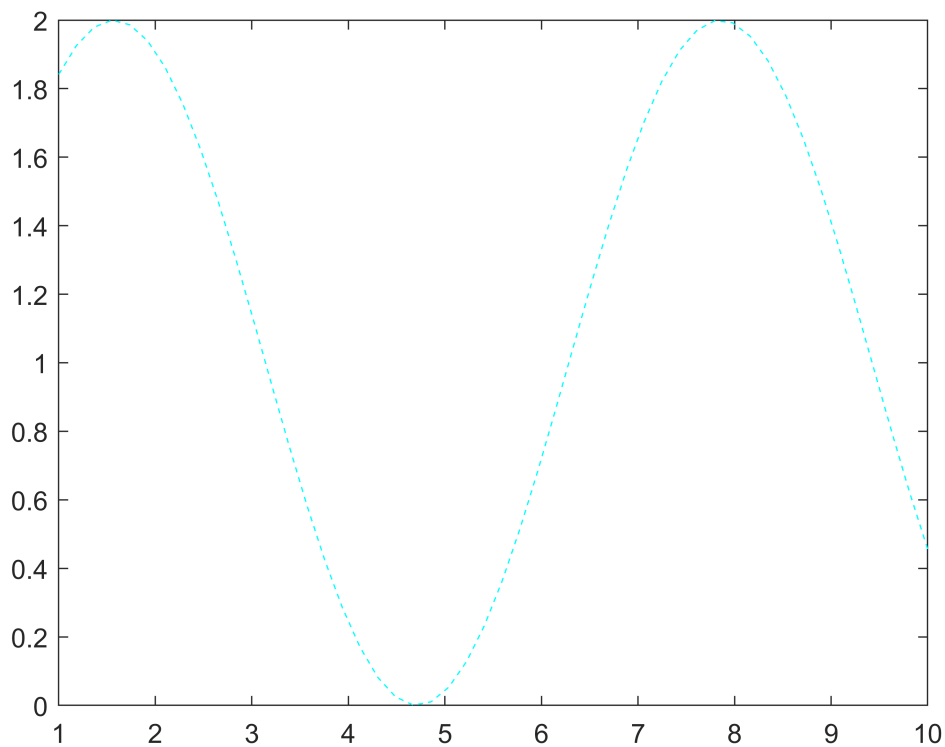
```
vec = linspace(1,10,50)
```

```
vec = 1×50  
    1.000000000000000    1.183673469387755    1.367346938775510    1.551020408163265 ...
```

```
y = sin(vec) + 1
```

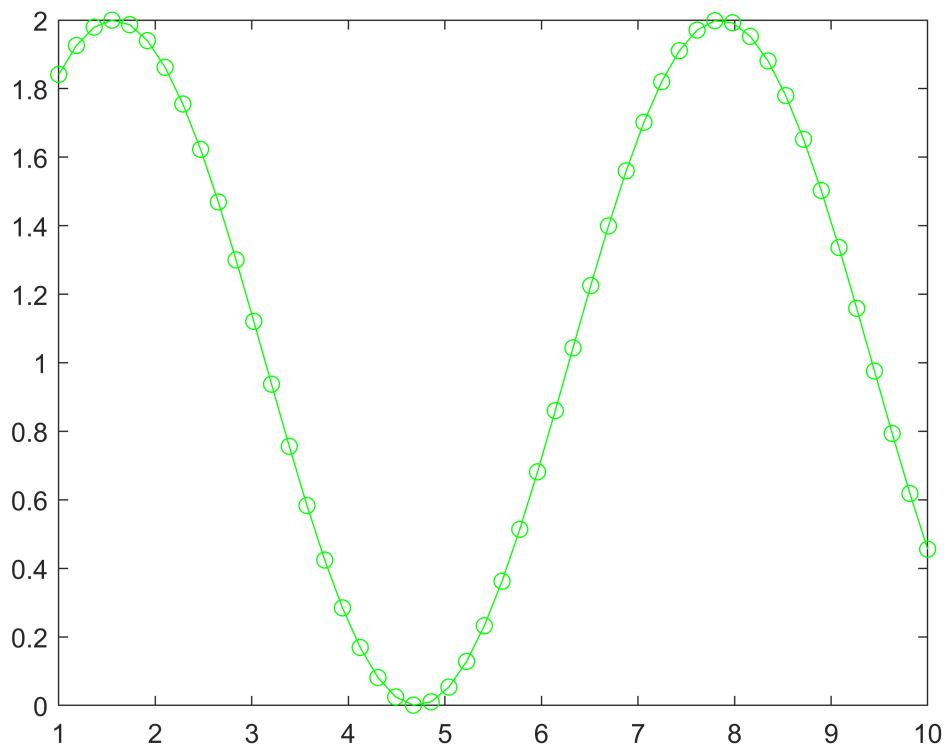
```
y = 1×50  
    1.841470984807897    1.925999086458024    1.979375461044357    1.999804462893930 ...
```

```
plot(vec,y,'c--')
```



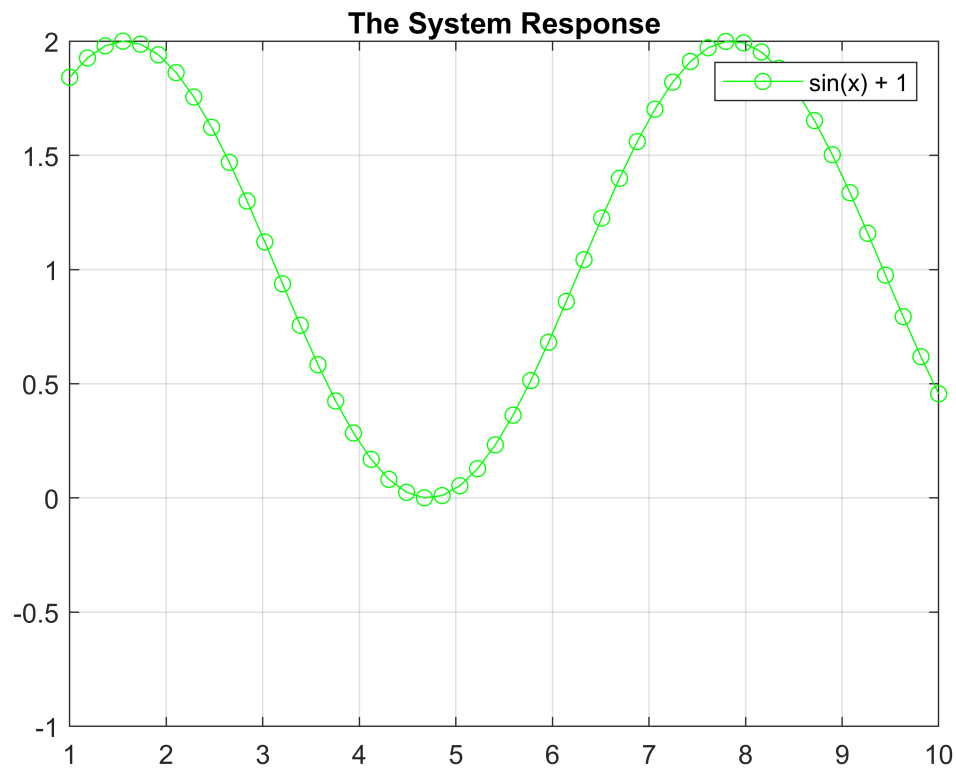
```
% 5.1.1
```

```
plot(vec,y,'go-')
```



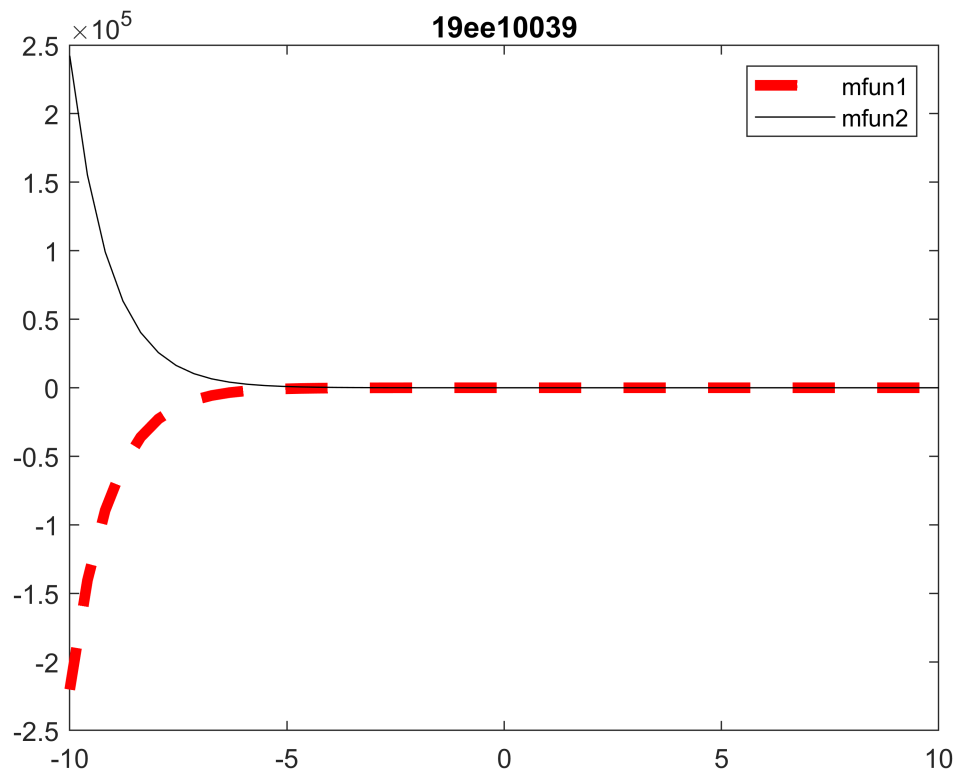
% 5.1.2

```
plot(vec,y,'go-')  
ylim([-1,2])  
grid on  
legend("sin(x) + 1")  
title("The System Response")
```



% 5.2

```
vec_2 = linspace(-10,10,50);
plot(vec_2,mfun1(vec_2),'r--','LineWidth',4.0)
hold on
plot(vec_2,mfun2(vec_2),'black')
legend('mfun1','mfun2')
title('19ee10039')
hold off
```



```
%defined functions
```

```
function [y] = derivative_central_approx (f,x,h)
y=(f(x+h)-f(x-h))/(2*h);
end

function [y] = mysinc(x)
if x==0
    y = 1;
else
    y = sin(x)/x;
end
end

function [y] = mfun2(x)
y=exp(-x)-x.*exp(-x);
end

function [y] = mfun1(x)
y = x.*exp(-x);
end
```