Problem 1 (Manipulating matrices)

There are many useful functions in Matlab for manipu- lating matrices and vectors, like the diag function from the previous problem. Some other examples include the tril, triu, reshape, and repmat functions. The tril and triu can be used to return any portion of the lower and upper part of a matrix A, respectively. The reshape command allows you to change the dimensions of a matrix from n-by-m to p-by-q, provided mn = pq. For example,

```
% part (a)
A1 = reshape(1:25, [5 5]);
triu(A1,-1)
```

```
ans = 5×5
     1     6    11    16    21
     2     7    12    17    22
     0     8    13    18    23
     0     0    14    19    24
     0     0     0    20    25
```

```
tril(A1,1)
```

```
ans = 5×5
     1     6     0     0     0
     2     7    12     0     0
     3     8    13    18     0
     4     9    14    19    24
     5    10    15    20    25
```

```
upper_triangualr_ones = triu(ones(5),2); lower_triangular = tril(A1, -2);
% combine A1 with upper triangualr and minus the lower triangular.
A3 = (upper_triangualr_ones.*A1)-lower_triangular
```

```
A3 = 5×5
     0     0    11    16    21
     0     0     0    17    22
    -3     0     0     0    23
    -4    -9     0     0     0
    -5   -10   -15     0     0
```

```


clc
% Part (b)
B1 = reshape(50:-2:2, [5 5])
```

```
B1 = 5×5
    50    40    30    20    10
    48    38    28    18     8
    46    36    26    16     6
    44    34    24    14     4
    42    32    22    12     2
```

```
upper_triangualr_B1 = triu(ones(5),1);
lower_triangular = tril(B1, -1); B2 =
(upper_triangualr_B1.*B1)+lower_triangular
```

```
B2 = 5×5
     0    40    30    20    10
    48     0    28    18     8
    46    36     0    16     6
```

```
    44    34    24     0     4
    42    32    22    12     0
```

```
v = [-2 -2 -2 -2 -2];
B = diag(v);
B3 = B2+B
```

```
B3 = 5×5
    -2    40    30    20    10
    48    -2    28    18     8
    46    36    -2    16     6
    44    34    24    -2     4
    42    32    22    12    -2
```

```

clc
% Part (c)
C1 = reshape(ones(5),[5 5]);
C1(2:5,2) = 4; C1(2,2:5) = 4;
C1(3:5,3) = 7; C1(3,3:5) = 7;
C1(4:5,4) = 10; C1(4,4:5) = 10;
C1(5:5,5) = 13
```

```
C1 = 5×5
     1     1     1     1     1
     1     4     4     4     4
     1     4     7     7     7
     1     4     7    10    10
     1     4     7    10    13
```

```
C = reshape(ones(5),[5 5]);
C(2,1:2) = 4; C(1,2) = 4;
C(3,1:3) = 7; C(1:2,3) = 7;
C(4,1:4) = 10; C(1:3,4) = 10;
C(5,1:5) = 13; C(1:4,5) = 13
```

```
C = 5×5
     1     4     7    10    13
     4     4     7    10    13
     7     7     7    10    13
    10    10    10    10    13
    13    13    13    13    13
```

```

upper_triangualr_C2 = triu(ones(5),1);
lower_triangular_C2 = -tril(C, -1);
C2 = (upper_triangualr_C2.*C)+lower_triangular_C2
```

```
C2 = 5×5
     0     4     7    10    13
    -4     0     7    10    13
    -7    -7     0    10    13
   -10   -10   -10     0    13
   -13   -13   -13   -13     0
```

```

upper_triangualr_C2 = triu(ones(5),1);
```

```
lower_triangular_C2 = tril(C, -1);
C_3 = (upper_triangualr_C2.*C)+lower_triangular_C2;
C3 = C.*(-eye(5))+C_3
```

```
C3 = 5×5
    -1     4     7    10    13
     4    -4     7    10    13
     7     7    -7    10    13
    10    10    10   -10    13
    13    13    13    13   -13
```

```
% Part (d)
d1 = [4 25 64; 9 36 81; 16 49 100];
d = repmat(d1,[2 3])
```

```
d = 6×9
     4    25    64     4    25    64     4    25    64
     9    36    81     9    36    81     9    36    81
    16    49   100    16    49   100    16    49   100
     4    25    64     4    25    64     4    25    64
     9    36    81     9    36    81     9    36    81
    16    49   100    16    49   100    16    49   100
```

Problem 2 (Toeplitz matrices)

(a) The two matrices A and B below are examples of what are called Toeplitz matrices, which are matrices where each diagonal is a constant. These matrices occur quite often in applications. Read the online help for the Matlab function toeplitz and use this function to produce A and B below. Note: use vector concatenation and the function zeros instead of typing all those zeros. Your code should take one line to produce A and one to produce B.

```
% (a)
A = reshape(zeros(10),[10 10]);
I =A-2*eye(10);
A1 = I-diag(ones(1,9), 1)-diag(ones(1,9), -1)
```

```
A1 = 10×10
    -2    -1     0     0     0     0     0     0     0     0
    -1    -2    -1     0     0     0     0     0     0     0
     0    -1    -2    -1     0     0     0     0     0     0
     0     0    -1    -2    -1     0     0     0     0     0
     0     0     0    -1    -2    -1     0     0     0     0
     0     0     0     0    -1    -2    -1     0     0     0
     0     0     0     0     0    -1    -2    -1     0     0
     0     0     0     0     0     0    -1    -2    -1     0
     0     0     0     0     0     0     0    -1    -2    -1
     0     0     0     0     0     0     0     0    -1    -2
```

```
A = reshape(zeros(10),[10 10]);
I =A-2*eye(10);
B1 = I-diag(-ones(1,9), -1)-diag(-ones(1,9), 1)
```

```
B1 = 10×10
    -2     1     0     0     0     0     0     0     0     0
```

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

```
% (b)
c = [0 1 3 5 7 9 11];
r = [0 2 4 6 8 10 12];
A2 = toeplitz(c,r)
```

```
A2 = 7×7
     0     2     4     6     8    10    12
     1     0     2     4     6     8    10
     3     1     0     2     4     6     8
     5     3     1     0     2     4     6
     7     5     3     1     0     2     4
     9     7     5     3     1     0     2
    11     9     7     5     3     1     0
```

```
c = [-1 4 -7 10 -13 16 -19];
r = [-1 3 -5 7 -9 11 -13];
B2 = toeplitz(c,r)
```

```
B2 = 7×7
    -1     3    -5     7    -9    11   -13
     4    -1     3    -5     7    -9    11
    -7     4    -1     3    -5     7    -9
    10    -7     4    -1     3    -5     7
   -13    10    -7     4    -1     3    -5
    16   -13    10    -7     4    -1     3
   -19    16   -13    10    -7     4    -1
```

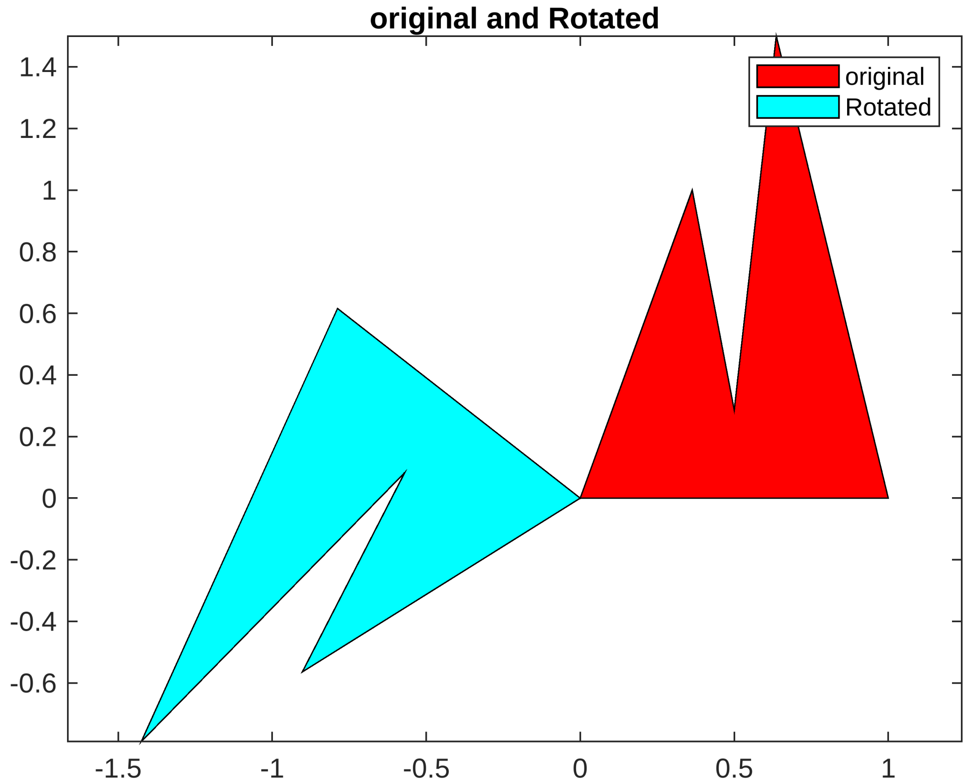
Problem 3 (Rotation matrices)

```
V = ([0 0; 1 0; 7/11 3/2; 1/2 2/7; 4/11 1])
```

```
V = 5×2
         0         0
    1.0000         0
    0.6364    1.5000
    0.5000    0.2857
    0.3636    1.0000
```

```
figure;
fill(V(:,1), V(:,2), 'r')
theta = deg2rad(142);
R = [cos(theta) -sin(theta); sin(theta) cos(theta)];
axis equal;
hold on;
```

```
rotation_matrix = (V*R')';
fill(rotation_matrix(1,:), rotation_matrix(2,:), 'c')
title('original and Rotated');
legend('original', 'Rotated');
```

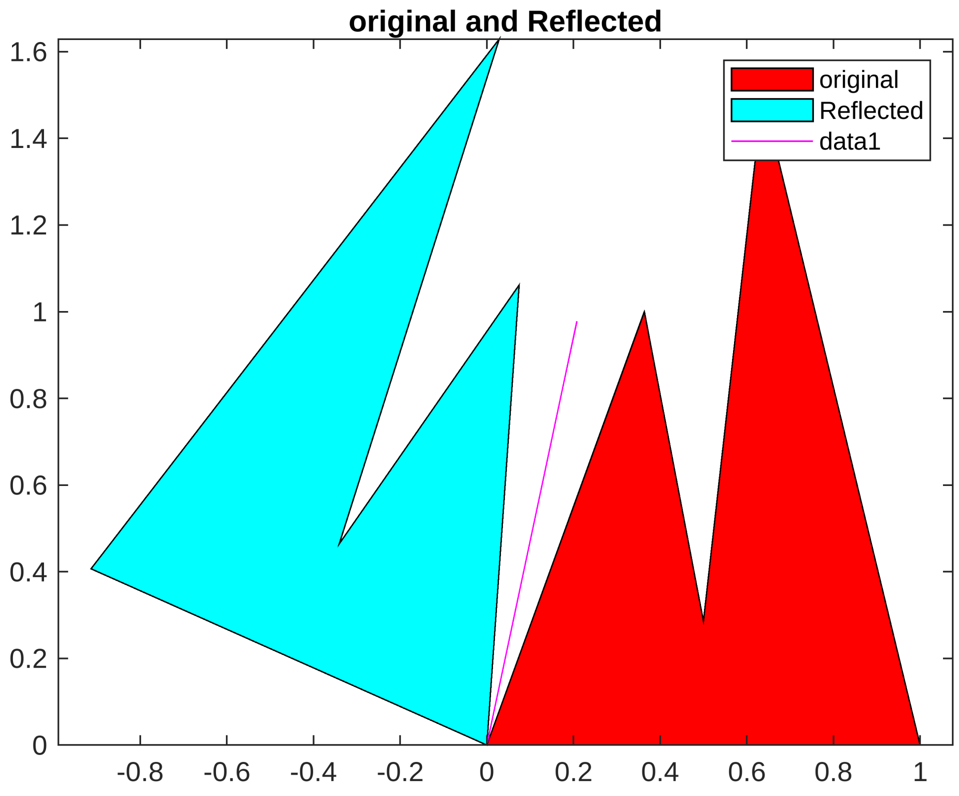


Problem 4 (Reflection matrices)

```
V = ([0 0; 1 0; 7/11 3/2; 1/2 2/7; 4/11 1]);

figure;
fill(V(:,1), V(:,2), 'r')
theta = deg2rad(78);
H = [cos(2*theta) sin(2*theta); sin(2*theta) -cos(2*theta)];
axis equal;
hold on;
rotation_matrix = (V*H')';
fill(rotation_matrix(1,:), rotation_matrix(2,:), 'c')
title('original and Reflected');
legend('original', 'Reflected');
hold on;
% line
x = [0,cos(theta)];
y = [0,sin(theta)];
plot(x, y, 'm');
```

```
hold off;
```



Problem 5 (Timing a linear solve)

In this problem, you will investigate the cost of solving linear systems and compare your results to the theoretical computational cost we have discussed in class.

```
clear;
% (a)
Mb = 1600; % MB
% Calculate M
M = sqrt((8*1024^2)/8);
fprintf('Maximum matrix dimension M: %d\n', M);
```

```
Maximum matrix dimension M: 1024
```

```
% nmax
Nmax = M/4
```

```
Nmax = 256
```

```
A = rand(Nmax, Nmax); % rand or randi random
B = rand(Nmax, Nmax);

tic % start
```

```
product = A*B;
time = toc; % stop
fprintf('Matrix multiplication took (size %dx%d) %.4f seconds.\n', Nmax,
Nmax, time);
```

```
Matrix multiplication took (size 256x256) 0.0021 seconds.
```

```
%flops
f = (2/3)*(Nmax)^3;
flops = f/time % flop rate flops/sec (Actual)
```

```
flops = 5.3286e+09
```

```
% actual value
actual_value = flops*4
```

```
actual_value = 2.1315e+10
```

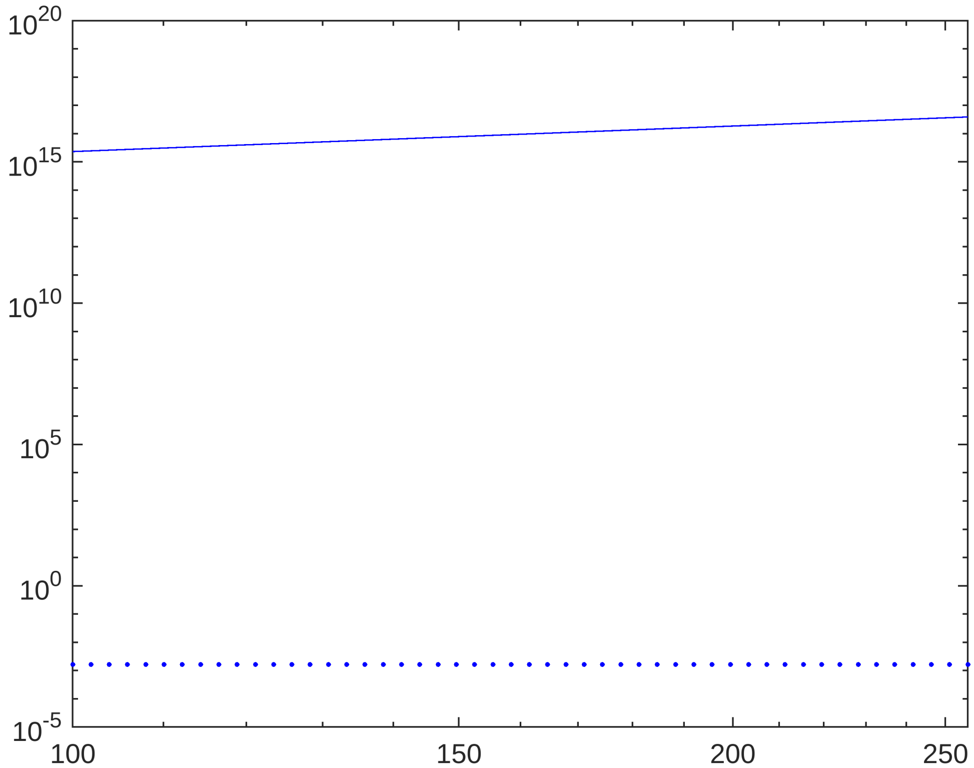
```
flops_cycle = 4;
clock_speed = 2.3; % clock speed of laptop (2.3 GHz)
% Theoretical value
Theoretical_flops_ = clock_speed*flops_cycle*10^9
```

```
Theoretical_flops_ = 9.2000e+09
```

```
% Part (c) Time a dense linear solve.
N = 100;
%Nvec = logspace(N, Nmax,50)
Nvec = logspace(log10(N), log10(Nmax),50);
for i = 1:1:length(N)
    A = rand(i,i);
    b = rand(i,1);
end
tic
x = A\b;
lutimes = toc
```

```
lutimes = 0.0016
```

```
% Part (d)
% Calculate the theoretical time based on the operation count
theoretical = (flops*Nvec.^3)/clock_speed;
figure;
loglog(Nvec, theoretical, 'b-', 'DisplayName', 'Theoretical');
hold on;
loglog(Nvec, lutimes, 'b.', 'DisplayName', 'Actual');
```




```
% Part (e)
% Using the value from the %Linpack Benchmark., what is the performance...
% of the world.s fastest supercomputer, measured in GFlops (= 109 flops)?

%9.95 EFlops => 9.95 * 10^18
Worlds_fastest = 9.95*10^9 %GFlops
```

```
Worlds_fastest = 9.9500e+09
```

```
%How much faster is the world's fastest computer than your computer or
laptop?
my_computer = flops/time %(s) versus 9.95*10^9
```

```
my_computer = 2.5387e+12
```

```
Worlds_fastest
```

```
Worlds_fastest = 9.9500e+09
```

```
% How recently would your computer have made it on the Top 500 list of the
world's
% fastest computers?

% not even close
```