Exercise 1: Consider the 3x3 real matrix,

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

1. Compute the eigenvalues of A,

   Solution:
   First we consider the characteristic equation of A,

$$|A - I\hat{\lambda}| = 0$$
$$|\begin{bmatrix} 2-\lambda & 1 & 1 \\ 4 & -\lambda & -2 \\ 2 & 2 & 3-\lambda \end{bmatrix}| = 0$$

   Solving for λ when the determinant is zero using co-factor expansion we get the following,

$$\begin{aligned} (2-\lambda)[-\lambda(3-\lambda)+2(2)] - [4(3-\lambda)+2(2)] + [4(2)+2\lambda] &= 0 \\ -\lambda^3 + 5\lambda^2 - 10\lambda + 8 + 4\lambda - 16 + 2\lambda + 8 &= 0 \\ -\lambda^3 + 5\lambda^2 - 4\lambda &= 0 \\ \lambda(-\lambda^2 + 5\lambda - 4) &= 0 \\ \lambda(\lambda-1)(\lambda-4) &= 0 \end{aligned}$$

   Finally we get that the eigenvalues for the matrix A are $\lambda = 0, 1, 4$

2. Compute the rank of A,

   Solution:
   To compute the rank of A let's first get the matrix in row echelon form,

$$\begin{bmatrix} 2 & 1 & 1 \\ 4 & 0 & -2 \\ 2 & 2 & 3 \end{bmatrix} = \begin{bmatrix} 2 & 1 & 1 \\ 0 & -2 & -4 \\ 2 & 2 & 3 \end{bmatrix} = \begin{bmatrix} 2 & 1 & 1 \\ 0 & -2 & -4 \\ 0 & 1 & 2 \end{bmatrix} = \begin{bmatrix} 2 & 1 & 1 \\ 0 & -2 & -4 \\ 0 & 0 & 0 \end{bmatrix}$$

   Since there are only 2 non-zero pivot values in the matrix we that the rank of the matrix is 2.

3. Compute the determinant of A,

   Solution:
   We can compute the determinant of A by hand using co-factor expansion,

$$det(A) = 2(0+4) - 1(12+4) + 1(8+0) = 0$$

   We get that the determinant of the matrix A is 0. This is expected as the matrix is not full rank and therefore not invertible.

4. Compute the inverse of A(if possible),

   Solution:
   As we have demonstrated, A is not full rank and it has the property that det(A) = 0. Therefore A is not invertible.

5. Compute the inverse of $B = A(2:3, 1:2)$

   Solution:
   Consider the matrix B,
$$B = \begin{bmatrix} 4 & 0 \\ 2 & 2 \end{bmatrix}$$
   We can quickly see that the $det(B) = 8$, and thus B is invertible. We can compute the inverse fairly quickly with the 2x2 matrix inverse formula,
$$B^{-1} = \frac{1}{8}\begin{bmatrix} 2 & 0 \\ -2 & 4 \end{bmatrix} = \begin{bmatrix} \frac{1}{4} & 0 \\ -\frac{1}{4} & \frac{1}{2} \end{bmatrix}$$

6. Solve the linear system of $Ax = b$ where $b = [-1, 8, -6]^*$,

   Solution:
   Consider the following augmented matrix,
$$\left[\begin{array}{ccc|c} 2 & 1 & 1 & -1 \\ 4 & 0 & -2 & 8 \\ 2 & 2 & 3 & -6 \end{array}\right]$$
   Reducing to REF we get that,
$$\left[\begin{array}{ccc|c} 2 & 1 & 1 & -1 \\ 4 & 0 & -2 & 8 \\ 2 & 2 & 3 & -6 \end{array}\right] = \left[\begin{array}{ccc|c} 2 & 1 & 1 & -1 \\ 0 & -2 & -4 & 10 \\ 2 & 2 & 3 & -6 \end{array}\right] = \left[\begin{array}{ccc|c} 2 & 1 & 1 & -1 \\ 0 & -2 & -4 & 10 \\ 0 & 1 & 2 & -5 \end{array}\right] = \left[\begin{array}{ccc|c} 2 & 1 & 1 & -1 \\ 0 & -2 & -4 & 10 \\ 0 & 0 & 0 & 0 \end{array}\right] = \left[\begin{array}{ccc|c} 1 & \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \\ 0 & 1 & 2 & -5 \\ 0 & 0 & 0 & 0 \end{array}\right]$$
   Note that x_3 is a free variable, so in our solution we get the following,
$$Ax = x_3\begin{bmatrix} -\frac{3}{2} \\ 2 \\ 1 \end{bmatrix} + \begin{bmatrix} 2 \\ -5 \\ 0 \end{bmatrix}$$

7. Now check your solutions with MATLAB.
   Terminal:

```
>> A = [2 1 1; 4 0 -2; 2 2 3]
A =
     2     1     1
     4     0    -2
     2     2     3

>> format long
>> eig(A)
ans =
   4.000000000000000
  -0.000000000000002
   1.000000000000001

>> rank(A)
ans =
     2

>> det(A)
ans =
     0

>> inv(A)
Warning: Matrix is singular to working precision.
ans =
   Inf   Inf   Inf
   Inf   Inf   Inf
   Inf   Inf   Inf


>> b = [-1 8 -6]
b =
    -1     8    -6

>> A\b'
Warning: Matrix is singular to working precision.
ans =
   NaN
   NaN
   NaN
```

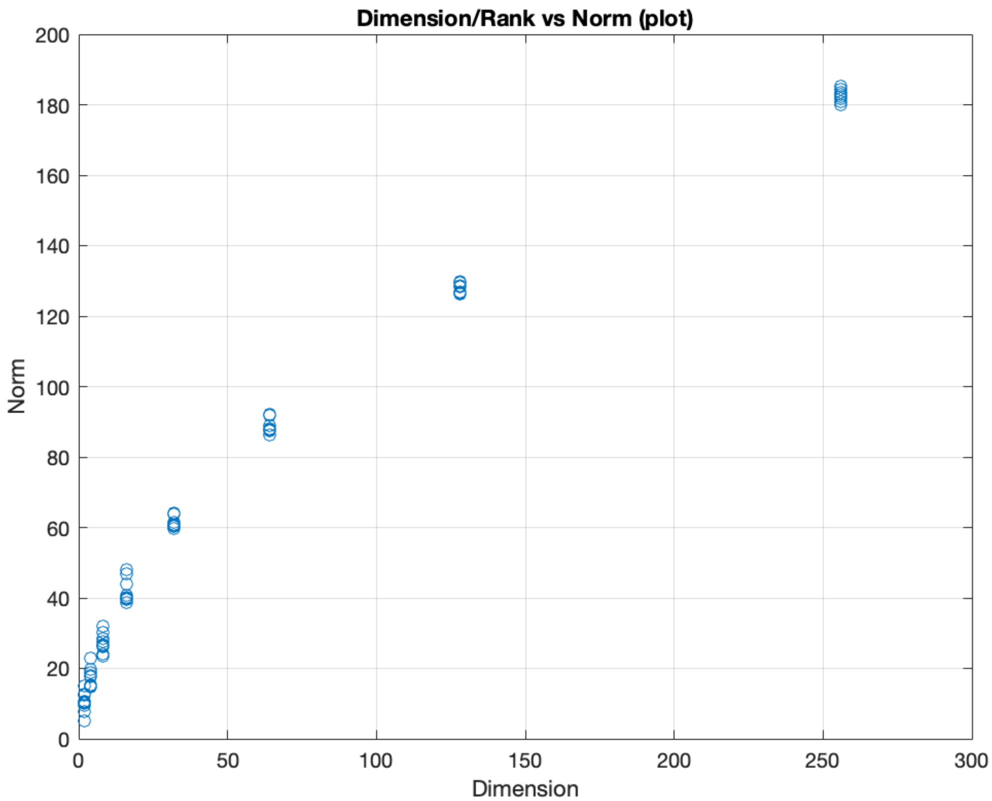
Exercise 2: Write a Matlab script which generates 10 random matrixes of size *mxm* for each of these powers of two: $m : 2, 4, 8, ..., 256$. Every matrix will have entries which are random real numbers uniformly distributed on $[-10, 10]$. For each of these matrices compute the rank, the 2-norm, and the absolute value of the determinant. Communicate these data using plots in reasonable ways; a significant part of your script will be devoted to generating plots.

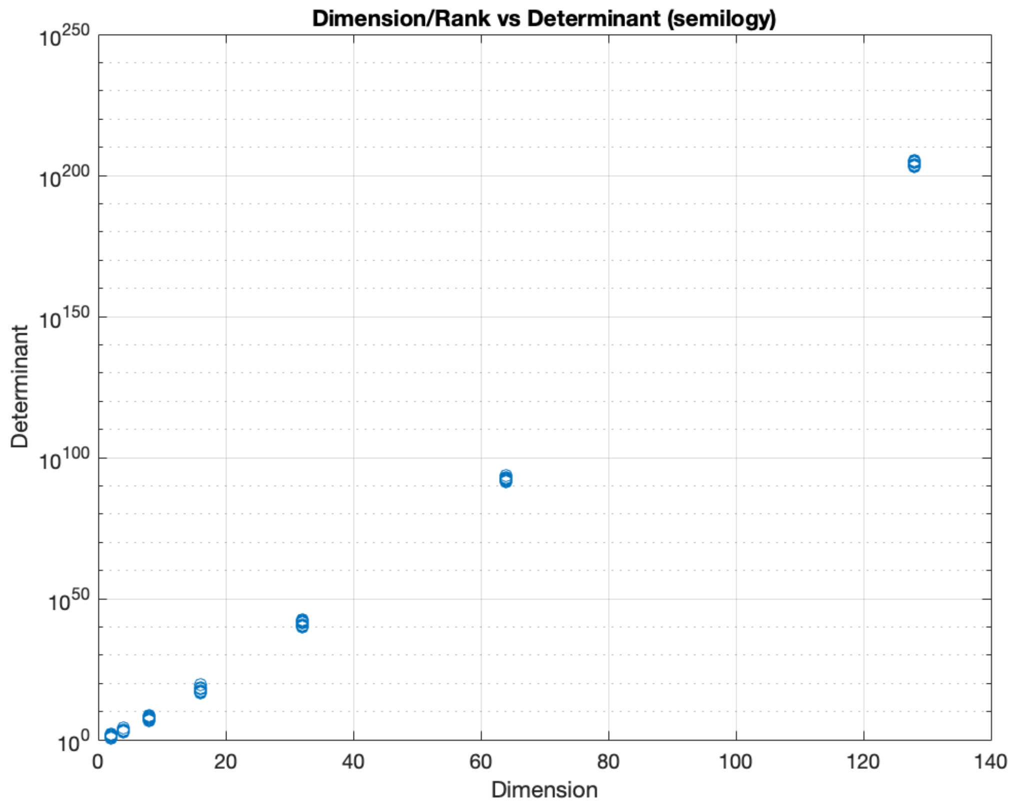
Solution:
Code:

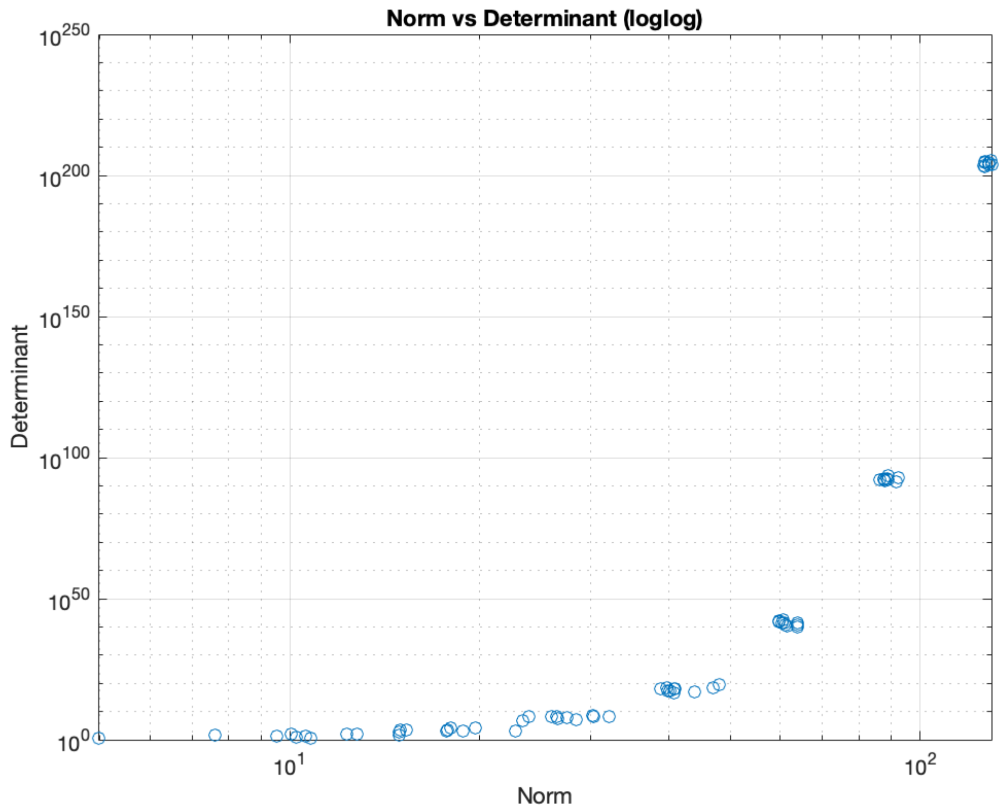
```
StoreMatrix = {}; %Initilize Storage Cell

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Generating Data
for i = 1:8

    %Initilizing individual data storage cells
    MatrixVec = {};
    NormVec = {};
    RankVec = {};
    DetVec = {};

    %Generating Matrices and Computing data
    for k = 1:10
        j = 20.*rand(2^i)-10; %Generate 2^ix2^i matrix

        % Data Computation and Storage to individual cells
        MatrixVec{1,k} = j;
        RankVec{1,k} = rank(j);
        NormVec{1,k} = norm(j);
        DetVec{1,k} = abs(det(j));
    end

        %Saving generated matrices
        StoreMatrix{i,1} = MatrixVec;
        %Saving the ranks
        StoreMatrix{i, 2} =RankVec;
        %Saving the 2-norms
        StoreMatrix{i, 3} = NormVec;
        %Saving the determinants
        StoreMatrix{i, 4} = DetVec;

end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Plotting Data
```

```
%Converting storage cells to plotable vectors
DegreeData = cell2mat([StoreMatrix{:,2}]);
NormData = cell2mat([StoreMatrix{:,3}]);
DetData = cell2mat([StoreMatrix{:,4}]);


tiledlayout(1,3)
ax1 = nexttile;
semilogy(ax1,DegreeData, NormData, 'o')
title(ax1,'Dimension/Rank vs Norm (plot)')
xlabel(ax1,'Dimension')
ylabel(ax1,'Norm')
grid on

ax2 = nexttile;
semilogy(ax2,DegreeData, DetData, 'o')
title(ax2,'Dimension/Rank vs Determinant (semilogy)')
xlabel(ax2,'Dimension')
ylabel(ax2,'Determinant')
grid on

ax3 = nexttile;
loglog(ax3,NormData, DetData, 'o')
title(ax3,'Norm vs Determinant (loglog)')
xlabel(ax3,'Norm')
ylabel(ax3,'Determinant')
grid on
```

Plots:


Dimension/Rank vs Norm (plot)
Norm
Dimension
0
20
40
60
80
100
120
140
160
180
200
0
50
100
150
200
250
300


Dimension/Rank vs Determinant (semilogy)
Determinant
Dimension
10^{250}
10^{200}
10^{150}
10^{100}
10^{50}
10^{0}
0
20
40
60
80
100
120
140



Exercise 3: 1. Consider the algorithm which computes the product of a rectangular matrix $A \in \mathbb{C}^{mxn}$ ans a column vector $v \in \mathbb{C}^{n+1}$. Count the number of floating point operations exactly, i.e as an expression in terms of m and n.

Solution:
Recall the definition of matrix vector multiplication, denoted as 1.2 in the reading

$$b = Av = \sum_{i=1}^{n} v_i a_i$$

Note that A is an mxn matrix and a_i denotes the i^{th} column of the matrix A. Since each a_i has m terms, each $v_i a_i$ term contains m multiplications. Over all n columns there are mn multiplications. Similarly, since there are only $n-1$ additions described in the sum, and each a_i column has m terms there are a total of $m(n-1)$ additions. In summation with mn multiplications and $m(n-1)$ additions we get the following,

$$Total_{FLOPS} = mn + m(n-1) = m(2n-1)$$

2. Implement the algorithm in a program *matvec.m* to multiply a matrix A and vector v, include error checking. Test your implementation against Matlab.
Code:

```
function [Count,x] = MatVec(A,v)
% This function takes a matrix A and a vector v
% and returns the product Av and the # of FLOPs.

if size(A,2) ~= size(v)
    error('Dimension Mismatch')
end


Count = 0; %Init Count
x = zeros(size(A,1),1);

    for m = 1:size(A,1) %Traverses Rows of A
        sum = 0;
        Count = Count-1; % Adjustment for initial addition

            for n = 1:size(v,1)% Traverses vector v
                sum = sum + A(m,n)*v(n);
                Count = Count + 2; % 1 multiplication and 1 addition
            end

        x(m) = sum;
    end
end
```

Terminal:

```
>> A = rand(4,3)
A =
     0.3967     0.6648     0.3827
     0.9691     0.9111     0.2267
     0.5269     0.2030     0.1816
     0.0176     0.5845     0.3452

>> v = rand(3,1)
v =
     0.4955
     0.9074
     0.6154
```

```
>> [FLOPs, x] = MatVec(A,v)
FLOPs =
    20
x =
    1.0353
    1.4464
    0.5570
    0.7515

>> x - (A*v)
ans =
     0
     0
     0
     0
```

3. Write a function *matmat.m* for the product $C = AB$ of matrices $A \in \mathbb{R}^{mxn}$ and $B \in \mathbb{R}^{nxk}$. Count the number of operations. Check your code against the Matlab results for some example $m = 3$, $n = 4$, and $k = 3$.

Solution:
If we consider the columns of B as individual vectors we get the following idea,

$$AB = A[b_1|b_2|...|b_k] = [Ab_1|Ab_2|...|Ab_k].$$

Note that A is an mxn matrix and each column vector b_i is $1xn$. Recall from the previous problem that the total number of flops for a matrix-vector product is,

$$m(2n-1).$$

Since it takes k matrix-vector products to produce AB we know that,

$$Total_{FLOPS} = km(2n-1).$$

Code:

```
function [Count, x] = MatMat(A,v)
% This function takes a matrix A and another matrix v
% and returns the product Av and the # of FLOPs.

if size(A,2) ~= size(v,1)
    error('Dimension Mismatch')
end
```

```
Count = 0; %Init Count
x = zeros(size(A,1),size(v,2));

   for k = 1:size(v,2) %Traverses Columns of V

       for m = 1:size(A,1) %Traverses Rows of A
            sum = 0;
            Count = Count-1; % Adjustment for initial addition

         for n = 1:size(v,1)% Traverses vector v
             sum = sum + A(m,n)*v(n,k);
             Count = Count + 2; % 1 multiplication and 1 addition

         end
         x(m,k) = sum;
      end
   end
end
```

Terminal:

```
>> A = rand(4,3)
A =
    0.9412    0.5140    0.1830
    0.1128    0.6439    0.9694
    0.3806    0.8492    0.3358
    0.4997    0.9873    0.9615

>> v = rand(3,5)
v =
    0.8081    0.4580    0.8211    0.4860    0.6447
    0.9542    0.1672    0.9689    0.3047    0.8514
    0.7766    0.4877    0.4285    0.3143    0.5161

>> [FLOPs, x] = MatMat(A,v)
FLOPs =
   100
x =
    1.3931    0.6062    1.3492    0.6715    1.1388
    1.4583    0.6321    1.1318    0.5557    1.1211
    1.3787    0.4801    1.2792    0.5493    1.1417
    2.0925    0.8629    1.7788    0.8459    1.6589

>> x - (A*v)
ans =
```

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

Exercise 4: Let B be any $4x4$ matrix to which we apply the following operations in turn:

1. Interchange rows 1 and 3;

$$\begin{bmatrix}0&0&1&0\\0&1&0&0\\1&0&0&0\\0&0&0&1\end{bmatrix}B$$

2. Interchange Columns 2 and 4;

$$\begin{bmatrix}0&0&1&0\\0&1&0&0\\1&0&0&0\\0&0&0&1\end{bmatrix}B\begin{bmatrix}1&0&0&0\\0&0&0&1\\0&0&1&0\\0&1&0&0\end{bmatrix}$$

3. Double column 3;

$$\begin{bmatrix}0&0&1&0\\0&1&0&0\\1&0&0&0\\0&0&0&1\end{bmatrix}B\begin{bmatrix}1&0&0&0\\0&0&0&1\\0&0&1&0\\0&1&0&0\end{bmatrix}\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&2&0\\0&0&0&1\end{bmatrix}$$

4. Add row 3 to row 1;

$$\begin{bmatrix}1&0&1&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}\begin{bmatrix}0&0&1&0\\0&1&0&0\\1&0&0&0\\0&0&0&1\end{bmatrix}B\begin{bmatrix}1&0&0&0\\0&0&0&1\\0&0&1&0\\0&1&0&0\end{bmatrix}\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&2&0\\0&0&0&1\end{bmatrix}$$

5. Subtract row 2 from each of the other rows

$$\begin{bmatrix}1&-1&0&0\\0&1&0&0\\0&-1&1&0\\0&-1&0&1\end{bmatrix}\begin{bmatrix}1&0&1&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}\begin{bmatrix}0&0&1&0\\0&1&0&0\\1&0&0&0\\0&0&0&1\end{bmatrix}B\begin{bmatrix}1&0&0&0\\0&0&0&1\\0&0&1&0\\0&1&0&0\end{bmatrix}\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&2&0\\0&0&0&1\end{bmatrix}$$

6. Replace column 3 with column 4;

$$\begin{bmatrix}1&-1&0&0\\0&1&0&0\\0&-1&1&0\\0&-1&0&1\end{bmatrix}\begin{bmatrix}1&0&1&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}\begin{bmatrix}0&0&1&0\\0&1&0&0\\1&0&0&0\\0&0&0&1\end{bmatrix}B\begin{bmatrix}1&0&0&0\\0&0&0&1\\0&0&1&0\\0&1&0&0\end{bmatrix}\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&2&0\\0&0&0&1\end{bmatrix}\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&0\\0&0&1&1\end{bmatrix}$$

7. Delete row 1;

$$\begin{bmatrix}0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}\begin{bmatrix}1&-1&0&0\\0&1&0&0\\0&-1&1&0\\0&-1&0&1\end{bmatrix}\begin{bmatrix}1&0&1&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}\begin{bmatrix}0&0&1&0\\0&1&0&0\\1&0&0&0\\0&0&0&1\end{bmatrix}B\begin{bmatrix}1&0&0&0\\0&0&0&1\\0&0&1&0\\0&1&0&0\end{bmatrix}\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&2&0\\0&0&0&1\end{bmatrix}\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&0\\0&0&1&1\end{bmatrix}$$

8. Simplify the product so it becomes a product of 3 matrices ABC where B is the same.

Solution:
Using the commutativity of matrix multiplication we can work from the inside out to simplify the product.

$$A=\begin{bmatrix}0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}\begin{bmatrix}1&-1&0&0\\0&1&0&0\\0&-1&1&0\\0&-1&0&1\end{bmatrix}\begin{bmatrix}1&0&1&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}\begin{bmatrix}0&0&1&0\\0&1&0&0\\1&0&0&0\\0&0&0&1\end{bmatrix}$$

$$A=\begin{bmatrix}0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}\begin{bmatrix}1&-1&0&0\\0&1&0&0\\0&-1&1&0\\0&-1&0&1\end{bmatrix}\begin{bmatrix}1&0&1&0\\0&1&0&0\\1&0&0&0\\0&0&0&1\end{bmatrix}$$

$$A=\begin{bmatrix}0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}\begin{bmatrix}1&-1&1&0\\0&1&0&0\\1&-1&0&0\\0&-1&0&1\end{bmatrix}$$

$$A=\begin{bmatrix}0&1&0&0\\1&-1&0&0\\0&-1&0&1\end{bmatrix}$$

Similarly we can compute C,

$$C=\begin{bmatrix}1&0&0&0\\0&0&0&1\\0&0&1&0\\0&1&0&0\end{bmatrix}\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&2&0\\0&0&0&1\end{bmatrix}\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&1\\0&0&0&0\end{bmatrix}$$

$$C=\begin{bmatrix}1&0&0&0\\0&0&0&1\\0&0&2&0\\0&1&0&0\end{bmatrix}\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&0\\0&0&1&1\end{bmatrix}$$

$$C=\begin{bmatrix}1&0&0&0\\0&0&1&1\\0&0&0&0\\0&1&0&0\end{bmatrix}$$

Finally we get that,

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix} B \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Exercise 1.3: Generalizing Example 1.3, we say that a square or rectangular matrix R with entries $r_{i,j}$ is upper-triangular if $r_{i,j} = 0$ for $i > j$. By considering what space is spanned by the first n columns of R and using (1.8), show that if R is a nonsingular mxm upper-triangular matrix, then R^{-1} is also upper triangular.

Solution:
Suppose that if R is a nonsingular mxm upper-triangular matrix. Note that by definition R is a square matrix in row echelon form with no pivot values and is therefore full-rank and fully invertible. Consider that by (1.8),

$$RR^{-1} = I,$$
$$R[r_1^{-1} r_2^{-1} \dots r_m^{-1}] = I.$$

So for some $k \in (1, m)$ we know that,

$$Rr_k^{-1} = e_k.$$

Note that since e_k has all zeros below the k^{th} row we know that the same must be true for r_k^{-1}, since R is lower triangular, a non zero value below the k^{th} row of r_k^{-1} would show up in e_k. This property extends to all columns in R^{-1} and therefore R^{-1} must be upper triangular.