Homework 4

1/21/22

Sections 2.1, 2.2

2.1.5

$\frac{2n^3}{3}$ is approximately the number of operations for Gaussian Elimination to solve n equations in n unknowns. When n tripled, $\frac{2(3n)^3}{3} = \frac{2(27n^3)}{3} = \frac{54n^3}{3}$. Compared to $\frac{2n^3}{3}$, $\frac{\frac{54n^3}{3}}{\frac{2(n)^3}{3}} = 27$. Thus, it is 27 times as long for n tripled.

2.2.2 **and** 2.2.4

Find LU Factorization of the given matrices. Check by matrix multiplication.

a)

$$\begin{bmatrix} 3 & 1 & 2 \\ 6 & 3 & 4 \\ 3 & 1 & 5 \end{bmatrix} \xrightarrow[R_2-2R_1, R_3-R_1]{} \begin{bmatrix} 3 & 1 & 2 \\ 0 & 1 & 0 \\ 0 & 0 & 3 \end{bmatrix}$$

$$\mathbf{U} = \begin{bmatrix} 3 & 1 & 2 \\ 0 & 1 & 0 \\ 0 & 0 & 3 \end{bmatrix} \qquad \mathbf{L} = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

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

b)

$$\begin{bmatrix} 4 & 2 & 0 \\ 4 & 4 & 2 \\ 2 & 2 & 3 \end{bmatrix} \xrightarrow[R_2-R_1, R_3-\frac{1}{2}R_1]{} \begin{bmatrix} 4 & 2 & 0 \\ 0 & 2 & 2 \\ 0 & 1 & 3 \end{bmatrix} \xrightarrow[R_3-\frac{1}{2}R_1]{} \begin{bmatrix} 4 & 2 & 0 \\ 0 & 2 & 2 \\ 0 & 0 & 2 \end{bmatrix}$$

$$\mathbf{U} = \begin{bmatrix} 4 & 2 & 0 \\ 0 & 2 & 2 \\ 0 & 0 & 2 \end{bmatrix} \qquad \mathbf{L} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ \frac{1}{2} & \frac{1}{2} & 1 \end{bmatrix}$$

$$\mathbf{LU} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ \frac{1}{2} & \frac{1}{2} & 1 \end{bmatrix} \begin{bmatrix} 4 & 2 & 0 \\ 0 & 2 & 2 \\ 0 & 0 & 2 \end{bmatrix} = \begin{bmatrix} 4 & 2 & 0 \\ 4 & 4 & 2 \\ 2 & 2 & 3 \end{bmatrix}$$

c)

$$\begin{bmatrix}1 & -1 & 1 & 2 \\ 0 & 2 & 1 & 0 \\ 1 & 3 & 4 & 4 \\ 0 & 2 & 1 & -1\end{bmatrix} \xrightarrow[R_3-R_1]{} \begin{bmatrix}1 & -1 & 1 & 2 \\ 0 & 2 & 1 & 0 \\ 0 & 4 & 3 & 2 \\ 0 & 2 & 1 & -1\end{bmatrix} \xrightarrow[R_3-2R_2, R4-R_2]{} \begin{bmatrix}1 & -1 & 1 & 2 \\ 0 & 2 & 1 & 0 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & -1\end{bmatrix}$$

$$\mathbf{U} = \begin{bmatrix}1 & -1 & 1 & 2 \\ 0 & 2 & 1 & 0 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & -1\end{bmatrix} \qquad \mathbf{L} = \begin{bmatrix}1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 2 & 1 & 0 \\ 0 & 1 & 0 & 1\end{bmatrix}$$

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

4)

Solve the system by finding the LU decomposition and then carrying out the two-step back substitution.

a)

$$\begin{bmatrix}3&1&2\\6&3&4\\3&1&5\end{bmatrix}\xrightarrow[R_2-2R_1,R_3-R_1]{}\begin{bmatrix}3&1&2\\0&1&0\\0&0&3\end{bmatrix}$$

$$\mathbf{U}=\begin{bmatrix}3&1&2\\0&1&0\\0&0&3\end{bmatrix}\qquad \mathbf{L}=\begin{bmatrix}1&0&0\\2&1&0\\1&0&1\end{bmatrix}$$

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

$$\mathbf{Lc}=\mathbf{b}\implies\begin{bmatrix}1&0&0\\2&1&0\\1&0&1\end{bmatrix}\begin{bmatrix}c_1\\c_2\\c_3\end{bmatrix}=\begin{bmatrix}0\\1\\3\end{bmatrix}$$

$$\begin{aligned}c_1&=0\\2c_1+c_2=1\implies c_2&=1\\c_1+c_3=3\implies c_3&=3\end{aligned}$$

$$\mathbf{c}=\begin{bmatrix}0\\1\\3\end{bmatrix}$$

$$\mathbf{Ux}=\mathbf{c}\implies\begin{bmatrix}3&1&2\\0&1&0\\0&0&3\end{bmatrix}\begin{bmatrix}x_1\\x_2\\x_3\end{bmatrix}=\begin{bmatrix}0\\1\\3\end{bmatrix}$$

$$\begin{aligned}3x_1+x_2+2x_3=0\implies 3x_1=-x_2-2x-3=-1-2\implies x_1&=-1\\x_2&=1\\3x_3=3\implies x_3&=1\end{aligned}$$

$$\mathbf{x} = \begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix}$$

To check:

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

b)

$$\begin{bmatrix} 4 & 2 & 0 \\ 4 & 4 & 2 \\ 2 & 2 & 3 \end{bmatrix} \xrightarrow[R_2 - R_1, R_3 - \frac{1}{2}R_1]{} \begin{bmatrix} 4 & 2 & 0 \\ 0 & 2 & 2 \\ 0 & 1 & 3 \end{bmatrix} \xrightarrow[R_3 - \frac{1}{2}R_1]{} \begin{bmatrix} 4 & 2 & 0 \\ 0 & 2 & 2 \\ 0 & 0 & 2 \end{bmatrix}$$

$$\mathbf{U} = \begin{bmatrix} 4 & 2 & 0 \\ 0 & 2 & 2 \\ 0 & 0 & 2 \end{bmatrix} \qquad \mathbf{L} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ \frac{1}{2} & \frac{1}{2} & 1 \end{bmatrix}$$

$$\mathbf{LU} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ \frac{1}{2} & \frac{1}{2} & 1 \end{bmatrix} \begin{bmatrix} 4 & 2 & 0 \\ 0 & 2 & 2 \\ 0 & 0 & 2 \end{bmatrix} = \begin{bmatrix} 4 & 2 & 0 \\ 4 & 4 & 2 \\ 2 & 2 & 3 \end{bmatrix}$$

$$\mathbf{Lc} = \mathbf{b} \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ \frac{1}{2} & \frac{1}{2} & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \\ 6 \end{bmatrix}$$

$$\begin{aligned} c_1 &= 2 \\ c_2 &= 4 - c_1 = 4 - 2 = 2 \\ c_3 &= -\frac{1}{2}c_1 - \frac{1}{2}c_2 + 6 = -1 - 1 + 6 = 4 \end{aligned}$$

$$c = \begin{bmatrix} 2 \\ 2 \\ 4 \end{bmatrix}$$

$$\mathbf{Ux} = \mathbf{c} \implies \begin{bmatrix} 4 & 2 & 0 \\ 0 & 2 & 2 \\ 0 & 0 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \\ 4 \end{bmatrix}$$

$$\begin{aligned} 4x_1 + 2x_2 = 2 \implies 2x_1 = 1 - x_2 \implies x_1 = 1 \\ 2x_2 + 2x_3 = 2x_2 = 1 - x_3 \implies x_2 = 1 - 2 = -1 \\ x_3 = 2 \end{aligned}$$

$$\mathbf{x} = \begin{bmatrix} 1 \\ -1 \\ 2 \end{bmatrix}$$

To check:

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

Computer Problems

2.1.1

The program I wrote for the three systems is as follows. The the results are below the code.

```
 def systemsofEquations():
    f1 = [[2,-2,-1], [4,1,-2], [-2,1,-1]]
    b1 = [-2, 1, -3]
    f2 = [[1, 2, -1], [0, 3, 1], [2, -1, 1]]
    b2 = [2, 4, 2]
    f3 = [[2, 1, -4], [1, -1, 1], [-1, 3, -2]]
    b3 = [-7, -2, 6]
    return f1, b1, f2, b2, f3, b3

# System 1
def gaussianElimination():
    syst1 = systemsofEquations()[0]
    b1 = systemsofEquations()[1]
    m = len(b1)
    x = [0.0]*m

    # Forward Elimination - Row major
    for row in range(m-1):  # for each row loop over row, m =
    ↪  number of rows
        if abs(syst1[row][row]) == 0.0:
            raise ValueError('zero pivot encountered')
        for col in range(row+1, m):  # loop over each column, n =
        ↪  number of cols
            ratio = syst1[col][row]/syst1[row][row]
            for c in range(row, m):
                syst1[col][c] = syst1[col][c] -
                ↪  ratio*syst1[row][c]
            b1[col] += -ratio * b1[row]
    print('syst1, b1', syst1, b1)

     #Back substitution - row major
    x[m - 1] = b1[m - 1] / syst1[m - 1][m - 1]
    for col in range(m-1, -1, -1):
        for row in range(col+1, m):
            b1[col] = b1[col] - (syst1[col][row] * x[row])
        x[col] = b1[col] / syst1[col][col]
    return syst1, b1, x
```

```
# System 2
def gaussianElimination2():
    syst2 = systemsofEquations()[2]
    b2 = systemsofEquations()[3]
    m = len(b2)
    x = [0.0]*m

    # Forward Elimination - Row major
    for row in range(m-1):  # for each row loop over row, m =
    ↪  number of rows
        if abs(syst2[row][row]) == 0.0:
            raise ValueError('zero pivot encountered')
        for col in range(row+1, m):  # loop over each column, n =
        ↪  number of cols
            ratio = syst2[col][row]/syst2[row][row]
            for c in range(row, m):
                syst2[col][c] = syst2[col][c] -
                ↪  ratio*syst2[row][c]
            b2[col] += -ratio * b2[row]
    print('syst2, b2', syst2, b2)

     #Back substitution - row major
    x[m - 1] = b2[m - 1] / syst2[m - 1][m - 1]
    for col in range(m-1, -1, -1):
        for row in range(col+1, m):
            b2[col] = b2[col] - (syst2[col][row] * x[row])
        x[col] = b2[col] / syst2[col][col]
    return syst2, b2, x

# System 3
def gaussianElimination3():
    syst3 = systemsofEquations()[4]
    b3 = systemsofEquations()[5]
    m = len(b3)
    x = [0.0]*m

    # Forward Elimination - Row major
    for row in range(m-1):  # for each row loop over row, m =
    ↪  number of rows
        if abs(syst3[row][row]) == 0.0:
            raise ValueError('zero pivot encountered')
        for col in range(row+1, m):  # loop over each column, n =
        ↪  number of cols
            ratio = syst3[col][row]/syst3[row][row]
            for c in range(row, m):
                syst3[col][c] = syst3[col][c] -
                ↪  ratio*syst3[row][c]
            b3[col] += -ratio * b3[row]
    print('syst3, b3', syst3, b3)

     #Back substitution - row major
    x[m - 1] = b3[m - 1] / syst3[m - 1][m - 1]
    for col in range(m-1, -1, -1):
        for row in range(col+1, m):
            b3[col] = b3[col] - (syst3[col][row] * x[row])
        x[col] = b3[col] / syst3[col][col]
    return syst3, b3, x


if __name__ == "__main__":
    syst1, b1, syst2, b2, syst3, b3 = systemsofEquations()
    print('system 1:', syst1, 'b1:', b1, "\n"
          'system 2:', syst2, 'b1:', b2, "\n"
          'system 3:', syst3, 'b1:', b3, "\n")
    ge1, b1, x1 = gaussianElimination()
    print(f'Result of GE is {ge1}, b1 is {b1}, and the solution x
    ↪  is {x1}.')
    print(' ')
    ge2, b2, x2 = gaussianElimination2()
    print(f'Result of GE is {ge2}, b1 is {b2}, and the solution x
    ↪  is {x2}.')
    print(' ')
    ge3, b3, x3 = gaussianElimination3()
    print(f'Result of GE is {ge3}, b1 is {b3}, and the solution x
    ↪  is {x3}.')
```

The solutions to the systems of exercise 2 are:

- a) The resulting system after GE is $[[2, -2, -1], [0.0, 5.0, 0.0], [0.0, 0.0, -2.0]]$, $\vec{b_1}$ is $[2.0, 5.0, -4.0]$, and the solution $\vec{x}$ is $[1.0, 1.0, 2.0]$.
- b) The resulting system after GE is $[[1, 2, -1], [0.0, 3.0, 1.0], [0.0, 0.0, 4.666666666666667]]$, $\vec{b_2}$ is $[1.0, 3.0, 4.666666666666667]$, and the solution $\vec{x}$ is $[1.0, 1.0, 1.0]$.
- c) The resulting system after GE is $[[2, 1, -4], [0.0, -1.5, 3.0], [0.0, 0.0, 3.0]]$, $\vec{b_3}$ is $[-2.0, -4.5, 6.0]$, and the solution $\vec{x}$ is $[-1.0, 3.0, 2.0]$.

2.1.2 In-Depth Problem

Hilbert matrices, H, are $n \times n$ and have the property that each (i, j) entry is $\frac{1}{(i+j-1)}$. To solve the equation $Hx = b$, where b is the vector of all ones, and find its determinant for $n = 2, n = 5, n = 10$, we must first create the Hilbert Matrix using a program, utilize Gaussian Elimination, then compute the determinant using a built in function in Python.

To create the Hilbert Matrix using a program, we first allocate the space of the matrix using a list comprehension, then loop over the rows and columns and divide each entry according to the property $\frac{1}{(i+j-1}$. Here we can call the numpy Python Linear Algebra Determinant package to calculate the determinant.

```
def hilbertMatrix(n):
# list comprehension--space allocation for hmatrix b, and x
hmatrix = [[1.0]*n for x in range(n)]
for row in range(n):   # select row
    for column in range(n):   # loop across row so loop
     ↪  across columns [index]
        hmatrix[row][column] = 1/((row+1) + (column+1) - 1 ).
         ↪  #add 1, since python starts from 0
detHM2 = numpy.linalg.det(hmatrix)
return hmatrix, detHM2
```

The output is the Hilbert Matrix for $n = 2, n = 5, n = 10$.
The result for $n = 2$ is

$$H = \begin{bmatrix} 1 & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{3} \end{bmatrix}$$

and $det(H) = 0.08\overline{333}$.

As we increase to $n = 5$,

$$H = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} \\ \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} & \frac{1}{8} \\ \frac{1}{5} & \frac{1}{6} & \frac{1}{7} & \frac{1}{8} & \frac{1}{9} \end{bmatrix}$$

and $det(H) = 3.7493e - 12$.

Finally for $n = 10$,

$$H = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} & \frac{1}{8} & \frac{1}{9} & \frac{1}{10} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} & \frac{1}{8} & \frac{1}{9} & \frac{1}{10} & \frac{1}{11} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} & \frac{1}{8} & \frac{1}{9} & \frac{1}{10} & \frac{1}{11} & \frac{1}{12} \\ \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} & \frac{1}{8} & \frac{1}{9} & \frac{1}{10} & \frac{1}{11} & \frac{1}{12} & \frac{1}{13} \\ \frac{1}{5} & \frac{1}{6} & \frac{1}{7} & \frac{1}{8} & \frac{1}{9} & \frac{1}{10} & \frac{1}{11} & \frac{1}{12} & \frac{1}{13} & \frac{1}{14} \\ \frac{1}{6} & \frac{1}{7} & \frac{1}{8} & \frac{1}{9} & \frac{1}{10} & \frac{1}{11} & \frac{1}{12} & \frac{1}{13} & \frac{1}{14} & \frac{1}{15} \\ \frac{1}{7} & \frac{1}{8} & \frac{1}{9} & \frac{1}{10} & \frac{1}{11} & \frac{1}{12} & \frac{1}{13} & \frac{1}{14} & \frac{1}{15} & \frac{1}{16} \\ \frac{1}{8} & \frac{1}{9} & \frac{1}{10} & \frac{1}{11} & \frac{1}{12} & \frac{1}{13} & \frac{1}{14} & \frac{1}{15} & \frac{1}{16} & \frac{1}{17} \\ \frac{1}{9} & \frac{1}{10} & \frac{1}{11} & \frac{1}{12} & \frac{1}{13} & \frac{1}{14} & \frac{1}{15} & \frac{1}{16} & \frac{1}{17} & \frac{1}{18} \\ \frac{1}{10} & \frac{1}{11} & \frac{1}{12} & \frac{1}{13} & \frac{1}{14} & \frac{1}{15} & \frac{1}{16} & \frac{1}{17} & \frac{1}{18} & \frac{1}{19} \end{bmatrix}$$

and $det(H) = 2.1642e - 53$.

Notice that the value of the determinant for larger and larger n, decreases drastically. By $n = 10$, it is already past the number of decimal places the computer can be considered accurate because of the small value of the determinant. Also notice, that there is more rounding, which will lead to round-off errors.

To perform Gaussian Elimination using a software program we break down each step which includes Forward Elimination and Back Substitution. The program loops over each column in each row and performs the calculation of subtracting the ratio of the lower row entry by the first row entry in the given column multiplied by the row one entry of interest from the lower row entry just as it is done in Gaussian elimination. Then, going from bottom to top where we update b by subtracting the bottom rightmost entry times the last entry of x. Then, we update x from bottom to top by the fraction of b at that row by the corresponding entry in the system. Finally, we return x, the solution to $Hx = b$.

```
def gaussianElimination(syst1, b1):
m = len(b1)
x = [0.0]*m

# Forward elimination - column major
for col in range(m):  # for each row loop over row, m =
↪  number of rows
    if abs(syst1[col][col]) < numpy.finfo(float).eps:
    ↪  # less than epsilon
        raise ValueError('zero pivot encountered')
    for row in range(col + 1, m):  # loop over each column, n
    ↪  = number of cols
        ratio = syst1[row][col] / syst1[col][col]
```

```
            for c in range(col, m):
                syst1[row][c] = syst1[row][c] - ratio *
                ↪  syst1[col][c]
            b1[row] += -ratio * b1[col]
    print('syst1, b1', syst1, b1)

    # Back substitution - column major
    x[m - 1] = b1[m - 1] / syst1[m - 1][m - 1]
    for row in range(m - 1, -1, -1):
        for col in range(row + 1, m):
            b1[row] = b1[row] - (syst1[row][col] * x[col])
        x[row] = b1[row] / syst1[row][row]
    return x

    if __name__ == "__main__":
    n_list = [2, 5, 10]
    for n in n_list:
        hb = [1.0]*n
        HM, det = hilbertMatrix(n)
        print('HM', HM, hb, 'Determinant', det)
        hilbert = gaussianElimination(HM, hb)
        print('result of hilbert', hilbert)
```

After running the above program, we return the resulting solution following Gaussian Elimination Hilbert matrix, b, and the solution, x.
For $n = 2$,

$$H = \begin{bmatrix} 1 & \frac{1}{2} \\ 0 & 0.08\overline{33} \end{bmatrix}$$

$$b = \begin{bmatrix} 1 \\ \frac{1}{2} \end{bmatrix}$$

and the solution is

$$x = \begin{bmatrix} -2.\overline{000}1 \\ 6.\overline{000}2 \end{bmatrix}.$$

For $n = 5$,

$$H = \begin{bmatrix} 1.0 & 0.5 & 0.\overline{33} & 0.25 & 0.2 \\ 0.0 & 0.08\overline{33} & 0.08\overline{333} & 0.075 & 0.0\overline{666} \\ 0.0 & -1.3878e-17 & 0.00\overline{55} & 0.008\overline{33} & 0.0095 \\ 0.0 & 0.0 & 0.0 & 0.00036 & 0.0007 \\ 0.0 & 0.0 & 0.0 & 0.0 & 2.2676\mathrm{e}-5 \end{bmatrix}$$

$$b = \begin{bmatrix} 1.0 \\ 0.5 \\ 0.1\overline{666} \\ 0.04\overline{999} \\ 0.01429 \end{bmatrix}$$

and the solution is

$$x = \begin{bmatrix} 4.\overline{999} \\ -119.\overline{999} \\ 629.\overline{999} \\ -1119.\overline{000} \\ 629.\overline{999} \end{bmatrix}$$

Finally, for $n = 10$,

$$H = \begin{bmatrix}
1.0 & 0.5 & 0.\overline{33} & 0.25 & 0.2 & 0.1\overline{66} & 0.14286 & 0.125 & 0.\overline{111} & 0.1 \\
0.0 & 0.08\overline{33} & 0.08\overline{333} & 0.075 & 0.0\overline{666} & 0.0595 & 0.05357 & 0.0486 & 0.0\overline{444}, 0.04\overline{09} & \\
0.0 & -1.3878e-17 & 0.00\overline{55} & 0.008\overline{33} & 0.0095 & 0.0099 & 0.0099 & 0.0097 & 0.0094 & 0.0\overline{0909}6 \\
0.0 & 0.0 & 0.0 & 0.00036 & 0.0007 & 0.00099 & 0.00119 & 0.00132 & 0.00\overline{14}6 & 0.00147 \\
0.0 & 0.0 & 0.0 & 0.0 & 2.2676e-5 & 5.6689e-5 & 9.2764e-5 & 0.00013 & 0.0002 & 0.00018 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 1.4315e-6 & 4.2946e-6 & 8.0937e-6 & 1.2333e-5 & 1.6650e-5 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 9.009\,75e-8 & 3.1534e-7 & 6.7273e-7 & 1.1352e-6 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 5.659\,97e-9 & 2.263\,99e-8 & 5.3936e-8 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & -1.6941e-21 & 0.0 & 0.0 & 3.5513e-10 & 1.5981e-9 \\
0.0 & 0.0 & 0.0 & 2.1684e-19 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 2.2268e-11
\end{bmatrix}$$

$$b = \begin{bmatrix} 1.0 \\ 0.5 \\ 0.1\overline{66} \\ 0.04\overline{999} \\ 0.01428 \\ 0.00397 \\ 0.00108 \\ 0.00029 \\ 7.77001e-05 \\ 2.05681e-05 \end{bmatrix}$$

and the solution is

$$x = \begin{bmatrix} -9.9974 \\ 989.7719 \\ -23755.1338 \\ 240195.7143 \\ -1261048.5972 \\ 3783198.5012 \\ -6725765.4896 \\ 7000357.2379 \\ -3937735.4176 \\ 923673.4085 \end{bmatrix}$$

An important observation is that there are serious round-off errors as n increases. This is understandable considering how small the values are becoming that computer no longer has the necessary accuracy to compute them. Therefore, the error for the determinant and in the other computations seems to increase with increasing n.

Linear Algebra Review

True/False These questions relate to matrices **A**, **B**, and **C**. Provide justification for each of your answers.

$$\mathbf{A} = \begin{bmatrix} \frac{1}{3} & \frac{2}{3} & \frac{-2}{3} \\ \frac{-2}{3} & \frac{2}{3} & \frac{1}{3} \\ \frac{2}{3} & \frac{1}{3} & \frac{2}{3} \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} 1 & 2 & 3 \\ -2 & -4 & -6 \end{bmatrix} \qquad \mathbf{C} = \begin{bmatrix} \frac{1}{2} & \frac{-\sqrt{3}}{2} \\ \frac{\sqrt{3}}{2} & \frac{1}{2} \end{bmatrix}$$

1. The nullspace of B is spanned by $\left\{ \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix} \right\}$.

   Answer: False. By performing Gaussian Elimination and solving for U', we obtain

   $$\mathbf{U} = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 0 & 0 \end{bmatrix} \qquad \mathbf{U'} = \begin{bmatrix} -1 & 2 & 3 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}.$$

   The nullspace of B is spanned by

   $$\left\{ \begin{bmatrix} 2 \\ -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 3 \\ 0 \\ -1 \end{bmatrix} \right\}.$$

   The nullspace of B has rank 2 not rank 1.

2. False. A property of projection matrices is that they are $n \times n$ and $A^2 = A$. As a result of matrix multiplication we have

   $$A^2 = \begin{bmatrix} \frac{-7}{9} & \frac{4}{9} & \frac{-4}{9} \\ \frac{-4}{9} & \frac{1}{9} & \frac{8}{9} \\ \frac{4}{9} & \frac{8}{9} & \frac{1}{9} \end{bmatrix} \neq \begin{bmatrix} \frac{1}{3} & \frac{2}{3} & \frac{-2}{3} \\ \frac{-2}{3} & \frac{2}{3} & \frac{1}{3} \\ \frac{2}{3} & \frac{1}{3} & \frac{2}{3} \end{bmatrix}.$$

3. True. **C** applies a transformation on **x** by rotating **x** by $\frac{\pi}{3}$. This is because **C** is a rotation matrix, $\begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$. We know $\cos(\frac{\pi}{3}) = \frac{1}{2}$ and $\sin(\frac{\pi}{3}) = \frac{\sqrt{3}}{2}$ from the unit circle and goes counter-clockwise.

4. False.

   $$\mathbf{R}(\mathbf{B}) = \begin{bmatrix} 1 & 2 & 3 \\ -2 & -4 & -6 \end{bmatrix} \xrightarrow[R_2+2R_1]{} \begin{bmatrix} 1 & 2 & 3 \\ 0 & 0 & 0 \end{bmatrix}$$

   $$\mathbf{R}(\mathbf{B}) = \left\{ \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \right\}$$

The projection matrix, $\mathbf{P} = \frac{aa^T}{||a^2||}$ where

$$\mathbf{a} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

Then,

$$\mathbf{P} = \frac{1}{14}\begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 6 \\ 3 & 6 & 9 \end{bmatrix}.$$

Onto means to be surjective. There are many vectors in R^3 that are not in $\mathbf{R}(\mathbf{B})$. Also, the number of columns is greater than or equal to the number of rows but does not equal the rank of $\mathbf{P}$, $n \geq m \neq$ rank $\mathbf{P}$. So it is not surjective.

5. False. $\mathbf{AB}$ doesn't obey matrix multiplication. $\mathbf{A}$ is of size 3×3 and $\mathbf{B}$ is of size 2×3 so they cannot be multiplied.

6. False. Matrix $\mathbf{B}$ is a rotation matrix, specifically it is a rotation matrix by $\frac{\pi}{3}$. So it cannot have an eigenvalue of 1 because it is not fixed (the identity) and it is not a rotation by 180 degrees which would be an eigenvalue of -1 anyways.

7. True. One property of orthogonal matrices is that for some matrix M, $MM^T = I$. $\mathbf{A}$ is orthogonal because

$$\begin{aligned}\mathbf{AA}^T &= \begin{bmatrix} \frac{1}{3} & \frac{2}{3} & \frac{-2}{3} \\ \frac{-2}{3} & \frac{2}{3} & \frac{1}{3} \\ \frac{2}{3} & \frac{1}{3} & \frac{2}{3} \end{bmatrix}\begin{bmatrix} \frac{1}{3} & \frac{2}{3} & \frac{-2}{3} \\ \frac{-2}{3} & \frac{2}{3} & \frac{1}{3} \\ \frac{2}{3} & \frac{1}{3} & \frac{2}{3} \end{bmatrix}^T \\ &= \begin{bmatrix} \frac{1}{3} & \frac{2}{3} & \frac{-2}{3} \\ \frac{-2}{3} & \frac{2}{3} & \frac{1}{3} \\ \frac{2}{3} & \frac{1}{3} & \frac{2}{3} \end{bmatrix}\begin{bmatrix} \frac{1}{3} & \frac{-2}{3} & \frac{2}{3} \\ \frac{2}{3} & \frac{2}{3} & \frac{1}{3} \\ \frac{-2}{3} & \frac{1}{3} & \frac{2}{3} \end{bmatrix} \\ &= I_3.\end{aligned}$$

$\mathbf{C}$ is orthogonal because

$$\begin{aligned}\mathbf{CC}^T &= \begin{bmatrix} \frac{1}{2} & \frac{-\sqrt{3}}{2} \\ \frac{\sqrt{3}}{2} & \frac{1}{2} \end{bmatrix}\begin{bmatrix} \frac{1}{2} & \frac{-\sqrt{3}}{2} \\ \frac{\sqrt{3}}{2} & \frac{1}{2} \end{bmatrix}^T \\ &= \begin{bmatrix} \frac{1}{2} & \frac{-\sqrt{3}}{2} \\ \frac{\sqrt{3}}{2} & \frac{1}{2} \end{bmatrix}\begin{bmatrix} \frac{1}{2} & \frac{\sqrt{3}}{2} \\ \frac{-\sqrt{3}}{2} & \frac{1}{2} \end{bmatrix} \\ &= I_2.\end{aligned}$$

8. False for $P_2 \to P_1$.

$$\begin{bmatrix} T(1) & T(x) & T(x^2) \end{bmatrix} = \begin{bmatrix} 1 & x \end{bmatrix} \begin{bmatrix} 3 & 2 & 1 \\ 1 & 2 & 3 \end{bmatrix}$$

Since $\begin{bmatrix} 3 & 2 & 1 \\ 1 & 2 & 3 \end{bmatrix} \neq \begin{bmatrix} 1 & 2 & 3 \\ -2 & -4 & -6 \end{bmatrix}$, the statement is false.

9. False. **CBA** doesn't obey matrix multiplication. **C** is 2×2, **B** is of size 2×3, and **A** is of size 3×3.

$$\begin{aligned} \mathbf{CBA} &= (2 \times 2) * (2 \times 3) * (3 \times 3) \\ &= (2 \times 3) * (3 \times 3) \\ &= 2 \times 3 \end{aligned}$$

By definition, the identity matrix must be $n \times n$ so it already fails since the dimension would result in a 2×3 matrix.