HW6

April 29, 2024

1 Section 6.2

1.1 5

1.1.1 given

$$w^T w = 1$$

$$A = ww^T$$

1.1.2 to prove that $A^2 = A$ (i.e., A is idempotent)

$$A^2 = (ww^T)(ww^T)$$

- Matrix multiplication is associative, so we can rearrange the terms in the product.

$$A^2 = ww^T ww^T$$

- Since $w^T w$ is a scalar and we know that $w^T w = 1$, we can rewrite the above equation as

$$A^2 = w(1)w^T = ww^T$$

- We know that $A = ww^T$, so

$$A^2 = A$$

- A is idempotent.

1.2 15

1.2.1 given

$$AB = 0$$

- 0 is the zero matrix.
- A and B have all nonzero elements.
- A and B are of order 2.

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}, \quad B = \begin{bmatrix} e & f \\ g & h \end{bmatrix}$$

$$AB = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} ae+bg & af+bh \\ ce+dg & cf+dh \end{bmatrix}$$

Condition for AB to be the zero matrix:

$$ae+bg=0, af+bh=0, ce+dg=0, cf+dh=0.$$

Example

$$A = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix}$$

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

1.2.2 non singular vs singular

- A matrix is nonsingular if its determinant is nonzero. The determinants of the matrices A and B are given by

$$\det(A) = 1\cdot 1 - 1\cdot 1 = 0, \det(B) = 1\cdot(-1) - 1\cdot(-1) = 0.$$

- Therefore, both matrices are singular since $\det(A) = 0$ and $\det(B) = 0$.

Thoughts for all cases not just the one we produced.

- (I wasn't sure if we needed to just show one example or prove for all cases so...)
- Given two order 2 matrixes A, B such that all elements are nonzero, and $AB = 0$.
- A matrix can be nonsingular only if it is full rank, meaning its determinant is non-zero.
- Since $AB = 0$, this means at least one of the matrices, A or B, has to be singular
  - Why? Well, if both were nonsingular, their product wouldn't be the zero matrix.
  - Even though all elements in A and B are nonzero, it doesn't prevent them from being singular. A matrix can still be singular if its rows (or columns) are linearly dependent.
- In our case, both A and B must actually be singular.
  - If either A or B were nonsingular, we can multiply $AB = 0$ by the inverse of the nonsingular matrix, which should result in the other matrix equalling zero. That contradicts our initial condition that all elements are nonzero.
- Thus, both A and B are singular because that's the only way they don't conflict with the nonzero element condition.

1.3 17

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

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

$$AB = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} a+2b & 2a+b \\ c+2d & 2c+d \end{bmatrix}$$

$$BA = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} a+2c & b+2d \\ 2a+c & 2b+d \end{bmatrix}$$

1.3.1 Commutativity

- Corresponding elements of AB and BA must be equal.

$$a + 2b = a + 2c$$

$$2a + b = b + 2d$$

$$c + 2d = 2a + c$$

$$2c + d = 2b + d$$

1.3.2 Simplifying

$$b = ca = d$$

- For A and B to commute, the elements of A must satisfy $a = d$ and $b = c$.

$$A = \begin{bmatrix} a & b \\ b & a \end{bmatrix}$$

2 Section 6.3

2.1 3

- We recreated the Matlab code in python below.

2.1.1 n = 2

$$n = 2$$
$$Solution = \begin{bmatrix} 1.0 \\ 0 \end{bmatrix}$$

2.1.2 n = 5

$$n = 5$$
$$Solution = \begin{bmatrix} 1.0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

2.1.3 n = 10

$$n = 10$$

$$Solution = \begin{bmatrix} 1.0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

2.1.4 n = 20

$$n = 20$$

$$Solution = \begin{bmatrix} 1.0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

2.2 6

2.2.1 c

$$A = \begin{bmatrix} 1 & 2 & 4 \\ 1 & 3 & 9 \\ 1 & 4 & 16 \end{bmatrix}$$

Inverse of A

- From the textbook we will apply gaussian elimination to A and I to get A^{-1}.

- To find the inverse of matrix A, we solve the equation $AX = I$ by applying Gaussian elimination for each column of the identity matrix I_3. Each solution vector x_{*i} becomes a column of the inverse matrix A^{-1}.

$$[A|I] = \left[\begin{array}{ccc|ccc} 1 & 2 & 4 & 1 & 0 & 0 \\ 1 & 3 & 9 & 0 & 1 & 0 \\ 1 & 4 & 16 & 0 & 0 & 1 \end{array}\right]$$

Row operations

- $R2 = R2 - R1$
- $R3 = R3 - R1$

$$\left[\begin{array}{ccc|ccc} 1 & 2 & 4 & 1 & 0 & 0 \\ 0 & 1 & 5 & -1 & 1 & 0 \\ 0 & 2 & 12 & -1 & 0 & 1 \end{array}\right]$$

- $R3 = R3 - 2 \times R2$

$$\left[\begin{array}{ccc|ccc} 1 & 2 & 4 & 1 & 0 & 0 \\ 0 & 1 & 5 & -1 & 1 & 0 \\ 0 & 0 & 2 & 1 & -2 & 1 \end{array}\right]$$

- $R3 = R3/2$
- $R2 = R2 - 5 \times R3$
- $R1 = R1 - 4 \times R3$

$$\left[\begin{array}{ccc|ccc} 1 & 2 & 0 & 1 & 4 & 0 \\ 0 & 1 & 0 & -1 & 6 & 0 \\ 0 & 0 & 1 & 0.5 & -1 & 0.5 \end{array}\right]$$

- $R1 = R1 - 2 \times R2$

Solution Matrix

$$\left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 6 & -8 & 3 \\ 0 & 1 & 0 & -3.5 & 6 & -2.5 \\ 0 & 0 & 1 & 0.5 & -1 & 0.5 \end{array}\right]$$

Constructing A^{-1}

$$A^{-1} = \begin{bmatrix} 6 & -8 & 3 \\ -3.5 & 6 & -2.5 \\ 0.5 & -1 & 0.5 \end{bmatrix}$$

2.2.2 e

Given Matrix A:

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \\ 1 & 3 & 6 & 10 \\ 1 & 4 & 10 & 20 \end{bmatrix}$$

Augmented Matrix

$$[A|I] = \left[\begin{array}{cccc|cccc} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 2 & 3 & 4 & 0 & 1 & 0 & 0 \\ 1 & 3 & 6 & 10 & 0 & 0 & 1 & 0 \\ 1 & 4 & 10 & 20 & 0 & 0 & 0 & 1 \end{array}\right]$$

Row Operations

$R2 = R2 - R1 \rightarrow \begin{bmatrix} 0 & 1 & 2 & 3 & -1 & 1 & 0 & 0 \end{bmatrix}$ $R3 = R3 - R1 \rightarrow \begin{bmatrix} 0 & 2 & 5 & 9 & -1 & 0 & 1 & 0 \end{bmatrix}$ $R4 = R4 - R1 \rightarrow [0 \quad 3$

$R3 = R3 - 2 \times R2 \rightarrow \begin{bmatrix} 0 & 0 & 1 & 3 & 1 & -2 & 1 & 0 \end{bmatrix}$ $R4 = R4 - 3 \times R2 \rightarrow \begin{bmatrix} 0 & 0 & 3 & 10 & 2 & -3 & 0 & 1 \end{bmatrix}$

$$\left[\begin{array}{cccc|cccc} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 3 & 1 & -2 & 1 & 0 \\ 0 & 0 & 3 & 10 & 2 & -3 & 0 & 1 \end{array}\right]$$

$$R4 = R4 - 3 \times R3 \rightarrow \begin{bmatrix} 0 & 0 & 0 & 1 & -1 & 3 & -3 & 1 \end{bmatrix}$$

Solution Matrix

$$\left[\begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & 4 & -6 & 4 & -1 \\ 0 & 1 & 0 & 0 & -6 & 14 & -11 & 3 \\ 0 & 0 & 1 & 0 & 4 & -11 & 10 & -3 \\ 0 & 0 & 0 & 1 & -1 & 3 & -3 & 1 \end{array}\right]$$

Constructing A^{-1}

$$A^{-1} = \begin{bmatrix} 4 & -6 & 4 & -1 \\ -6 & 14 & -11 & 3 \\ 4 & -11 & 10 & -3 \\ -1 & 3 & -3 & 1 \end{bmatrix}$$

[ ]:
```python
import numpy as np

def GEpivot(A, b):
    # Check if A is square and b is of appropriate dimensions
    m, n = A.shape
    if m != n:
        raise ValueError('Matrix A must be square.')
    if len(b) != n:
        raise ValueError('Matrix and vector dimensions do not match.')

    # Initialize the pivot vector
    piv = np.arange(n)
    A = A.astype(float)  # Ensure A is float for divisions
    b = b.astype(float)  # Ensure b is float for proper subtraction and division
```

```
    # Perform elimination
    for k in range(n - 1):
        # Find the index of the max element in the pivot column
        index = np.argmax(np.abs(A[k:n, k])) + k
        if index != k:
            # Swap rows in A
            A[[k, index], k:n] = A[[index, k], k:n]
            # Swap elements in b
            b[k], b[index] = b[index], b[k]
            # Swap pivot indices
            piv[k], piv[index] = piv[index], piv[k]

        # Form the multipliers and carry out the elimination step
        A[k+1:n, k] /= A[k, k]
        for i in range(k+1, n):
            A[i, k+1:n] -= A[i, k] * A[k, k+1:n]
            b[i] -= A[i, k] * b[k]

    # Solve the upper triangular system
    x = np.zeros(n)
    x[n-1] = b[n-1] / A[n-1, n-1]
    for i in range(n-2, -1, -1):
        x[i] = (b[i] - np.dot(A[i, i+1:n], x[i+1:n])) / A[i, i]

    # LU factorization matrix with pivot included
    lu = np.tril(A, -1) + np.eye(n) + np.triu(A)

    return x, lu, piv

# Example usage
A = np.array([[2, 1, 1], [1, 3, 2], [1, 0, 0]], dtype=float)
b = np.array([4, 5, 6], dtype=float)
x, lu, piv = GEpivot(A, b)
```

[ ]:
```
import numpy as np

def generate_matrix_vector(n, matrix_rule=None, vector_rule=None):
    """
    Generate a matrix A and vector b of size n based on given rules.

    Parameters:
        n (int): The size of the matrix and vector.
        matrix_rule (callable, optional): A function that defines the rule to␣
 ↪generate elements of A.
                                          Should accept two parameters i and j.
```

```
        vector_rule (callable, optional): A function that defines the rule to␣
↪generate elements of b.
                                          Should accept one parameter i.

    Returns:
        A (ndarray): Generated matrix of size n x n.
        b (ndarray): Generated vector of size n.
    """
    if matrix_rule is None:
        # Default rule for matrix if none provided
        matrix_rule = lambda i, j: min(i, j)
    if vector_rule is None:
        # Default rule for vector if none provided
        vector_rule = lambda i: 1

    # Create the matrix A according to the rule
    A = np.fromfunction(np.vectorize(lambda i, j: matrix_rule(i+1, j+1)), (n,␣
↪n), dtype=int)

    # Create the vector b according to the rule
    b = np.fromfunction(np.vectorize(lambda i: vector_rule(i+1)), (n,),␣
↪dtype=int)

    return A, b
```

[ ]:
```
import sympy as sp

def matrix_to_latex(A, b, x, lu, piv):
    """
    Converts matrices and vectors into LaTeX format.

    Parameters:
        A (numpy.ndarray): The original matrix A from the linear system Ax = b.
        b (numpy.ndarray): The right-hand side vector b of the linear system.
        x (numpy.ndarray): The solution vector x.
        lu (numpy.ndarray): The LU decomposition matrix.
        piv (numpy.ndarray): The pivot indices vector.

    Returns:
        dict: A dictionary containing the LaTeX strings for each matrix and␣
↪vector.
    """
    A_sym = sp.Matrix(A)
    b_sym = sp.Matrix(b)
    x_sym = sp.Matrix(x)
    lu_sym = sp.Matrix(lu)
    piv_sym = sp.Matrix(piv)
```

```python
    latex_dict = {
        'A': sp.latex(A_sym),
        'b': sp.latex(b_sym),
        'x': sp.latex(x_sym),
        'LU': sp.latex(lu_sym),
        'Pivot Vector': sp.latex(piv_sym)
    }

    return latex_dict
```

[ ]:
```python
# Example usage within your loop
values_of_n = [2, 5, 10, 20]

for n in values_of_n:
        A, b = generate_matrix_vector(n)
        x, lu, piv = GEpivot(A, b)
        latex_results = matrix_to_latex(A, b, x, lu, piv)
        print(f"n = {n}")
        print("Solution = ", latex_results['x'])

        print("Matrix A = ", latex_results['A'])
        print("Vector b = ", latex_results['b'])
        print("LU Matrix = ", latex_results['LU'])
        print("Pivot Vector = ", latex_results['Pivot Vector'])

        print("\n")
```

```
n = 2
Solution =  \left[\begin{matrix}1.0\\0\end{matrix}\right]
Matrix A =  \left[\begin{matrix}1 & 1\\1 & 2\end{matrix}\right]
Vector b =  \left[\begin{matrix}1\\1\end{matrix}\right]
LU Matrix =  \left[\begin{matrix}2.0 & 1.0\\1.0 & 2.0\end{matrix}\right]
Pivot Vector =  \left[\begin{matrix}0\\1\end{matrix}\right]



n = 5
Solution =  \left[\begin{matrix}1.0\\0\\0\\0\\0\end{matrix}\right]
Matrix A =  \left[\begin{matrix}1 & 1 & 1 & 1 & 1\\1 & 2 & 2 & 2 & 2\\1 & 2 & 3
& 3 & 3\\1 & 2 & 3 & 4 & 4\\1 & 2 & 3 & 4 & 5\end{matrix}\right]
Vector b =  \left[\begin{matrix}1\\1\\1\\1\\1\end{matrix}\right]
LU Matrix =  \left[\begin{matrix}2.0 & 1.0 & 1.0 & 1.0 & 1.0\\1.0 & 2.0 & 1.0 &
1.0 & 1.0\\1.0 & 1.0 & 2.0 & 1.0 & 1.0\\1.0 & 1.0 & 1.0 & 2.0 & 1.0\\1.0 & 1.0 &
1.0 & 1.0 & 2.0\end{matrix}\right]
Pivot Vector =  \left[\begin{matrix}0\\1\\2\\3\\4\end{matrix}\right]



n = 10
```

```
Solution =
\left[\begin{matrix}1.0\\0\\0\\0\\0\\0\\0\\0\\0\\0\end{matrix}\right]
Matrix A =  \left[\begin{matrix}1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1\\1 & 2 & 2
& 2 & 2 & 2 & 2 & 2 & 2 & 2\\1 & 2 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3\\1 & 2 & 3 &
4 & 4 & 4 & 4 & 4 & 4 & 4\\1 & 2 & 3 & 4 & 5 & 5 & 5 & 5 & 5 & 5\\1 & 2 & 3 & 4
& 5 & 6 & 6 & 6 & 6 & 6\\1 & 2 & 3 & 4 & 5 & 6 & 7 & 7 & 7 & 7\\1 & 2 & 3 & 4 &
5 & 6 & 7 & 8 & 8 & 8\\1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 9\\1 & 2 & 3 & 4 & 5
& 6 & 7 & 8 & 9 & 10\end{matrix}\right]
Vector b =  \left[\begin{matrix}1\\1\\1\\1\\1\\1\\1\\1\\1\\1\end{matrix}\right]
LU Matrix =  \left[\begin{matrix}2.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 &
1.0 & 1.0\\1.0 & 2.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0\\1.0 & 1.0
& 2.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0\\1.0 & 1.0 & 1.0 & 2.0 & 1.0 &
1.0 & 1.0 & 1.0 & 1.0 & 1.0\\1.0 & 1.0 & 1.0 & 1.0 & 2.0 & 1.0 & 1.0 & 1.0 & 1.0
& 1.0\\1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 2.0 & 1.0 & 1.0 & 1.0 & 1.0\\1.0 & 1.0 &
1.0 & 1.0 & 1.0 & 1.0 & 2.0 & 1.0 & 1.0 & 1.0\\1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0
& 1.0 & 2.0 & 1.0 & 1.0\\1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 2.0 &
1.0\\1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 &
2.0\end{matrix}\right]
Pivot Vector =
\left[\begin{matrix}0\\1\\2\\3\\4\\5\\6\\7\\8\\9\end{matrix}\right]


n = 20
Solution =  \left[\begin{matrix}1.0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0
\\0\\0\\0\\0\end{matrix}\right]
Matrix A =  \left[\begin{array}{cccccccccccccccccccc}1 & 1 & 1 & 1 & 1 & 1 & 1 &
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1\\1 & 2 & 2 & 2 & 2 & 2 & 2 & 2
& 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2\\1 & 2 & 3 & 3 & 3 & 3 & 3 & 3 &
3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3\\1 & 2 & 3 & 4 & 4 & 4 & 4 & 4 & 4
& 4 & 4 & 4 & 4 & 4 & 4 & 4 & 4 & 4 & 4 & 4\\1 & 2 & 3 & 4 & 5 & 5 & 5 & 5 & 5 &
5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5\\1 & 2 & 3 & 4 & 5 & 6 & 6 & 6 & 6 & 6
& 6 & 6 & 6 & 6 & 6 & 6 & 6 & 6 & 6 & 6\\1 & 2 & 3 & 4 & 5 & 6 & 7 & 7 & 7 & 7 &
7 & 7 & 7 & 7 & 7 & 7 & 7 & 7 & 7 & 7\\1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 8 & 8 & 8
& 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8\\1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 9 & 9 &
9 & 9 & 9 & 9 & 9 & 9 & 9 & 9 & 9\\1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 10 &
10 & 10 & 10 & 10 & 10 & 10 & 10 & 10 & 10\\1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 &
10 & 11 & 11 & 11 & 11 & 11 & 11 & 11 & 11 & 11 & 11\\1 & 2 & 3 & 4 & 5 & 6 & 7
& 8 & 9 & 10 & 11 & 12 & 12 & 12 & 12 & 12 & 12 & 12 & 12 & 12\\1 & 2 & 3 & 4 &
5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 13 & 13 & 13 & 13 & 13 & 13 & 13\\1 & 2
& 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 14 & 14 & 14 & 14 & 14 &
14\\1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 15 & 15 &
15 & 15 & 15\\1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 &
16 & 16 & 16 & 16 & 16\\1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 &
14 & 15 & 16 & 17 & 17 & 17 & 17\\1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 &
12 & 13 & 14 & 15 & 16 & 17 & 18 & 18 & 18\\1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 &
10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 & 19 & 19\\1 & 2 & 3 & 4 & 5 & 6 & 7
& 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 & 19 & 20\end{array}\right]
Vector b =  \left[\begin{matrix}1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\
```

```
1\\1\\1\\1\end{matrix}\right]
LU Matrix =  \left[\begin{array}{cccccccccccccccccccc}2.0 & 1.0 & 1.0 & 1.0 &
1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 &
1.0 & 1.0 & 1.0\\1.0 & 2.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0
& 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0\\1.0 & 1.0 & 2.0 & 1.0 &
1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 &
1.0 & 1.0 & 1.0\\1.0 & 1.0 & 1.0 & 2.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0
& 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0\\1.0 & 1.0 & 1.0 & 1.0 &
2.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 &
1.0 & 1.0 & 1.0\\1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 2.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0
& 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0\\1.0 & 1.0 & 1.0 & 1.0 &
1.0 & 1.0 & 2.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 &
1.0 & 1.0 & 1.0\\1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 2.0 & 1.0 & 1.0 & 1.0
& 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0\\1.0 & 1.0 & 1.0 & 1.0 &
1.0 & 1.0 & 1.0 & 1.0 & 2.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 &
1.0 & 1.0 & 1.0\\1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 2.0 & 1.0
& 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0\\1.0 & 1.0 & 1.0 & 1.0 &
1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 2.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 &
1.0 & 1.0 & 1.0\\1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0
& 2.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0\\1.0 & 1.0 & 1.0 & 1.0 &
1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 2.0 & 1.0 & 1.0 & 1.0 & 1.0 &
1.0 & 1.0 & 1.0\\1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0
& 1.0 & 1.0 & 2.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0\\1.0 & 1.0 & 1.0 & 1.0 &
1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 2.0 & 1.0 & 1.0 &
1.0 & 1.0 & 1.0\\1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0
& 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 2.0 & 1.0 & 1.0 & 1.0\\1.0 & 1.0 & 1.0 & 1.0 &
1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 &
1.0 & 2.0 & 1.0\\1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0
& 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 2.0\end{array}\right]
Pivot Vector =  \left[\begin{matrix}0\\1\\2\\3\\4\\5\\6\\7\\8\\9\\10\\11\\12\\13
\\14\\15\\16\\17\\18\\19\end{matrix}\right]
```

[ ]:
```python
# Matrix 6c

A = np.array([[1, 2, 4], [1,3,9], [1, 4, 16]], dtype=int)

A_sym = sp.Matrix(A)

print(sp.latex(A_sym))
```

```
\left[\begin{matrix}1 & 2 & 4\\1 & 3 & 9\\1 & 4 & 16\end{matrix}\right]
```

[ ]:
```python

```