Grade: 40/50

**HW 4- Factorization and Decomposition**
**University of Chicago - Linear Algebra and Python**

Professor Chao-Jen Chen

Jennifer Gunawan
Duke University
October 2023

Great job.

Your answers are thorough, the examples are well thought out, and the code snippets are excellent.

10/10

## Problem 1

False, the set of vectors $b$ that are not in the column space $C(A)$ of a matrix $A$ does not form a subspace. Below is a counter-example. Suppose we have a 2X2 matrix $A$:

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

The column space of matrix A is given by:

$$C(A) = span\left\{ \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right\}$$

Suppose we have two vectors that are not in C(A):

$$b1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$b2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Great job coming up with the simplest example and further elaborating in a footnote.

While b1 and b2 are not in C(A), they do not form a subspace. Specifically, if we consider the sum of b1 and b2, it is not in C(A).

$$b1 + b2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Additionally, it is also not in the set of vectors that are not in C(A). Therefore, the set of vectors that are not in C(A) does not form a subspace since it is not closed under vector addition.[1]

## Problem 2

False, if the column space of matrix A (C(A)) contains only the zero vector, it doesn't necessarily imply that matrix A must be a zero matrix with all entries being zero. Below is a counterexample. Suppose we have the matrix A:

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

[1] Some conditions that must hold for it to be a subspace: (1) the zero vector must be in the set, i.e. [0,0] is in the set of vectors not in C(A); (2) the set must be closed under vector addition, i.e. taking the two vectors not in C(A) doesn't necessarily mean that their sum is in the set (e.g. [2,2] and [3,3] are both not in C(A), but their sum is [5,5] which is also not in C(A). This means that this set is not closed under vector addition); (3) the set must be closed under scalar multiplication, i.e., if you take a vector not in C(A) and multiply it by a scalar, the result may not necessarily be in the set (e.g. 2*[2,2] = [4,4] which is not in C(A). This means that this set is not closed under scalar multiplication.

This is actually a claim showing it's true. Namely that C(A) is actually just the X-axis of the plane. So it contains more vectors than 0, eg. [2 0], [3 0], etc.

Suppose we have C(A) = {0}, then that implies that any linear combination of c1a1, c2a2, etc. must *always* be 0, for any c1, c2..., the only way this is satisfied is if all the vectors in A are 0.

The column space of matrix A is:

$$C(A) = span\left\{\begin{bmatrix}1\\0\end{bmatrix}\right\}$$

C(A) contains only the zero vector $\begin{bmatrix}0\\0\end{bmatrix}$. However, matrix A is not a zero matrix since it has non-zero entry in the first row and first column. Therefore, the claim is false.

## Problem 3 10/10

To perform LU decomposition on matrix A, we need to decompose A into the product of a lower triangular matrix L, and an upper triangular matrix U such that their products equal A (A = LU).
To derive L and U from A, we can use the Gaussian Elimination method- we'll do it row by row to eliminate entries below the main diagonal. Specifically,

1. Initialize matrices L and U as identity matrices of the same size as A, where A = $\begin{bmatrix}1 & 1 & 1\\1 & 2 & 3\\1 & 3 & 6\end{bmatrix}$

2. Perform Gaussian Elimination to transform U into an upper triangular matrix (i.e. make the elements below the main diagonal into zeroes). Specifically,

   (a) Create elimination matrix (E1) to eliminate elements below the (1,1) entry, such that

   $$E1 = \begin{bmatrix}1 & 0 & 0\\-1 & 1 & 0\\-1 & 0 & 1\end{bmatrix}$$

   And calculate U1 by multiplying E1*A:

   $$U1 = E1 \times A$$
   $$U1 = \begin{bmatrix}1 & 1 & 1\\0 & 1 & 2\\0 & 2 & 5\end{bmatrix}$$

   (b) Eliminate elements below (2,2) in U1 by creating elimination matrix (E2), such that

   $$E2 = \begin{bmatrix}1 & 0 & 0\\0 & 1 & 0\\0 & -1 & 1\end{bmatrix}$$

   Then calculate U2 by multiplying E2 and U1:

   $$U2 = E2 \times U1$$
   $$U2 = \begin{bmatrix}1 & 1 & 1\\0 & 1 & 2\\0 & 0 & 1\end{bmatrix}$$

   (c) Eliminate elements below (3,3) in U2 by creating elimination matrix (E3), such that:

   $$E3 = \begin{bmatrix}1 & 0 & 0\\0 & 1 & 0\\0 & 1 & 1\end{bmatrix}$$

Then calculate U3 by multiplying E3 and U2:

$$U3 = E3 \times U2$$

$$U3 = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix}$$

This is the upper triangular matrix U obtained by applying elimination matrices to A. We can mathematically verify this by multiplying E3, E2, and E1 together to geth the combined elimination matrix E and verify that multiplying E to A equals U3.

(d) To get L, we simply have to transpose U ($L = U^T$), such that

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

$$U^T = L = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

(e) We now have L and U. L is the lower triangular matrix with ones on the diagonal; U is the upper triangular matrix.

3. We can additionally form a sanity check by multiplying L and U to make sure that we get A as their product.

See Figure 1 for Python code. Running it will display matrices A, L, U, and L X U = A (which confirms that the LU decomposition is correct. The output is below:

- Matrix A: $\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \\ 1 & 3 & 6 \end{bmatrix}$
- Matrix L: $\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 2 & 1 \end{bmatrix}$
- Matrix U: $\begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix}$
- Product of L and U: $\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \\ 1 & 3 & 6 \end{bmatrix}$

The product of L and U is indeed equal to matrix A, confirming that the LU decomposition is correct.

## Problem 4 10/10

To determine whether vectors x1, x2, and x3 are linearly independent, we can create a matrix with these vectors as its columns and perform Gaussian Elimination (or row reduction) to check whether the reduced row echelon form (RREF) of the matrix contains a row of zeroes. If there are no zeroes in the RREF, it would mean that the vectors are linearly independent. Conversely,

if there is a row of zeroes, they are linearly dependent.
I used the Matrix.rref() method from sympy to figure it out. See Figure 2 for the Python code. The RREF of matrix A is: $\begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & -1 \\ 0 & 0 & 0 \end{bmatrix}$ Since the RREF has a row of all zeroes (the last row), we can say that the vectors x1, x2, and x3 are linearly dependent. Specifically, in the RREF, the last row corresponds to:

$$0x1 + 0x2 + 0x3 = 0$$

which implies that there exists a non-trivial solution to the homogenous equation involving these vectors which implies that they are linearly dependent. Therefore, vectors x1, x2, and x3 are linearly dependent.

## Problem 5 10/10

To express vector y as a linear combination of the vectors x1, x2, x3, we need to find scalars a, b, and c such that:

$$y = ax_1 + bx_2 + cx_3$$

We can set up an augmented matrix and use the Matrix.rref() method from the sympy library to solve the system of equations:

$$\begin{bmatrix} x1 & x2 & x3 & y \end{bmatrix}$$
$$\begin{bmatrix} 1 & 1 & 2 & 1 \\ 1 & 2 & -1 & -2 \\ 1 & 3 & 1 & 5 \end{bmatrix}$$

See Figure 3 for Python code. The reduced row echelon form (RREF) of matrix A is: $\begin{bmatrix} 1 & 0 & 1 & -6 \\ 0 & 1 & -3 & 3 \\ 0 & 0 & 0 & 2 \end{bmatrix}$
From the RREF, we can see that a, b, and c are -6, 3, and 2 respectively. Therefore, we can express vector y as a linear combination of vectors x1, x2, and x3 as follows:

$$y = -6x1 + 3x2 + 2x3$$
$$y = -6 \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} + 3 \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} + 2 \begin{bmatrix} 2 \\ -1 \\ 1 \end{bmatrix}$$
$$y = \begin{bmatrix} -6 \\ -6 \\ -6 \end{bmatrix} + \begin{bmatrix} 3 \\ 6 \\ 9 \end{bmatrix} + \begin{bmatrix} 4 \\ -2 \\ 2 \end{bmatrix}$$
$$y = \begin{bmatrix} (-6+3+4) \\ (-6+6-2) \\ (-6+9+2) \end{bmatrix}$$
$$y = \begin{bmatrix} 1 \\ 0 \\ 5 \end{bmatrix}$$

```python
import numpy as np

A = np.array([[1, 1, 1], [1, 2, 3], [1, 3, 6]], dtype=float)  # Convert to float to allow division
n = len(A)
L = np.eye(n, dtype=float)
U = np.copy(A).astype(float)  # Convert to float

for k in range(n-1):
    for i in range(k+1, n):
        factor = U[i, k] / U[k, k]
        L[i, k] = factor
        U[i, k:] -= factor * U[k, k:]

product = np.dot(L, U)
print("Matrix A:")
print(A)
print("Matrix L:")
print(L)
print("Matrix U:")
print(U)
print("Product of L and U:")
print(product)
```

```
Matrix A:
[[1. 1. 1.]
 [1. 2. 3.]
 [1. 3. 6.]]
Matrix L:
[[1. 0. 0.]
 [1. 1. 0.]
 [1. 2. 1.]]
Matrix U:
[[1. 1. 1.]
 [0. 1. 2.]
 [0. 0. 1.]]
Product of L and U:
[[1. 1. 1.]
 [1. 2. 3.]
 [1. 3. 6.]]
```

Figure 1: Problem 3 Python Code and Output

```python
import sympy as sp

# Define the vectors as columns of a matrix
A = sp.Matrix([[2, 1, 3], [-1, 1, -3], [3, -2, 8]])

# Calculate the RREF
rref_A = A.rref()[0]

# Display the RREF
print(rref_A)
```

[1] ✓ 1.0s

```
Matrix([[1, 0, 2], [0, 1, -1], [0, 0, 0]])
```

Figure 2: Problem 4 Python Code and Output

```python
from sympy import Matrix

A = Matrix([[1, 1, 2, 1], [1, 2, -1, -2], [1, 3, 1, 5]])
rref_A = A.rref()
solution = rref_A[0].col(-1)

print(f"a = {solution[0]}, b = {solution[1]}, c = {solution[2]}")
```

[1] ✓ 0.9s

```
a = -6, b = 3, c = 2
```

Figure 3: Problem 5 Python Code and Output