# ALADS - Assignment - 1

Chenglong Li

April 2024

# 1 Question 1

## 1.1 Question 1(a)

```
#get I1-II6
def solve_I(b1,b2,b3,b4,b5,b6):
    import numpy as np
    A = np.array ([[0,6,-2,0,0,0],[0,4,1,2,2,0],[0,0,0,0,2,4],
    \[1,-1,0,1,0,0],[0,0,1,0,-1,1],[0,0,0,1,-1,1]])
    b = np.array([b1,b2,b3,b4,b5,b6])
    x = np.linalg.solve(A,b)
    return x

I = solve_I(10,17,14,0,0,0)
import matplotlib.pyplot as plt
plt.figure(figsize=(8, 6))
plt.bar(range(1, 7), I, color='skyblue')
plt.xlabel('Current␣Index')
plt.ylabel('Current␣Value')
plt.title('Linear␣System')
plt.xticks(range(1, 7), [f'I_{i}' for i in range(1, 7)])
plt.grid(axis='y')
plt.show()
```

Output look at the Figure 1.

## 1.2 Question 1(b)

- What is the rank of matrix?
  - The rank of a matrix is the number of independent columns.
  - The dimension of the columns space, $dim(\mathbf{C}(A))$.

```
import numpy as np
A = np.array ([[0,6,-2,0,0,0],[0,4,1,2,2,0]
\,[0,0,0,0,2,4],[1,-1,0,1,0,0]
\,[0,0,1,0,-1,1],[0,0,0,1,-1,1]])
```

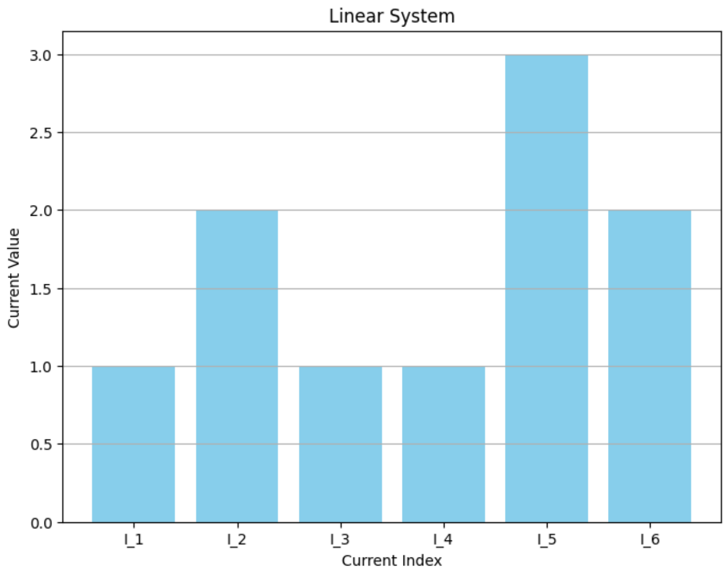


Figure 1: Question 1(a)-Output

```
rank_A = np.linalg.matrix_rank(A)
#print(A)
print(rank_A)
```

- What is the dimension of the four fundamental subspaces $\mathbf{C}(A), \mathbf{N}(A), \mathbf{C}(A^T), \mathbf{N}(A^T)$ in this case?
  - $\mathbf{C}(A) = 6$
  - $\mathbf{N}(A) = 0$
  - $\mathbf{C}(A^T) = 6$
  - $\mathbf{N}(A^T) = 0$
- Which subspace does the solution vector belong to?
  - Column Space, $\mathbf{C}(A)$.
- Is the solution to the equation system unique given a certain right-hand-side? Motivate you answer.
  - Yes. For any right-hand-side vector $\mathbf{b}$, as long as $\mathbf{b} \in \mathcal{R}^n$ , in this case $\mathbf{b} \in \mathcal{R}^6$,there is a always unique solution.
- Which subspace does the right-hand-side vector belong to?
  - Since the matrix $A$ is full rank, there will always be a solution for any $\mathbf{b}$.

- Is it possible to find a right-hand-side where no solution exists? If so, examplify and figure out which subspace that right-hand-side vector belong to?
  - Since matrix $A$ is full rank, there is no situation where there would be no solution. Any $\mathbf{b} \in \mathbf{C}(A)$.

# 2 Question 2

## 2.1 Question 2(a)

```
import numpy as np
import time

def matrix_fac_v1(A,B):
    m, p1 = A.shape
    p2, n = B.shape
    if p1 != p2 :
        print("Can not Mutiply!")
        return
    C = np.zeros((m,n))
    for i in range(m):
        for j in range(n):
            for k in range(p1):
                C[i,j] += A[i,k] * B[k,j]
    return C

def matrix_fac_v2(A,B):
    m, p1 = A.shape
    p2, n = B.shape
    if p1 != p2:
        print("Can not Mutiply!")
        return
    C = np.zeros((m,n))
    for k in range(p1):
        for j in range(n):
            for i in range(m):
                C[i,j] += A[i,k] * B[k,j]
    return C


A = np.random.rand(500,500)
B = np.random.rand(500,500)

start_time_v1 = time.process_time()
matrix_fac_v1(A,B)
end_time_v1 = time.process_time()
time_v1 = end_time_v1 - start_time_v1
```

```
print("Version 1:",time_v1)

start_time_v2 = time.process_time()
matrix_fac_v2(A,B)
end_time_v2 = time.process_time()
time_v2 = end_time_v2 - start_time_v2
print("Version 2:",time_v2)
```

Output look at Figure 2.

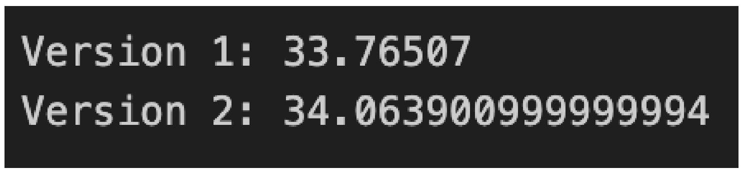


Figure 2: Question 2(a)-Output

## 2.2 Question 2(b)

**The leftmost algorithm is based on repeated dot-products, but what basic operation is the 2nd (rightmost) algorithm based on?**
The right algorithm is based on repeated additions to the elements of the result Matrix $A$. Each element $C(i, j)$ is accumulated by traversing over a row of $A$ and a column of $B$, multiplying the corresponding elements and adding them up. This approach is sometimes referred to as the outer product method, where the outer productions of rows of $A$ and columns of $B$ are computed one element at a time and added to the accumulating matrix $C$.