GEBZE TECHNICAL UNIVERSITY

ENGINEERING FACULTY

ELECTRONICS ENGINEERING

# ELM 433

## NUMERICAL COMPUTATION SOFTWARE

## PROJECT

Deadline: 27.12.2024

| | |
|---|---|
| Name – Surname | Mehmet ALTINTAŞ |
| Student ID | 1901022065 |

# CONTENTS

## 1. Setup and Definitions

First, we'll import the necessary libraries and define the given matrices and vectors.

```
import numpy as np

# Define a 4x4 matrix A
A = np.matrix([
    [4,  1,  2,  3],
    [0,  5,  1,  2],
    [1,  0,  3,  1],
    [2,  1,  1,  4]
])
# Define vector b
b = np.matrix([
    [10],
    [12],
    [8],
    [14]
])
# Compute the inverse of A
A_inv = np.linalg.inv(A)

# Compute solution vector x
x = A_inv * b
```

## 2. Computing Norms

We'll compute various norms for matrices $A$, $A^{-1}$, and vectors $b$ and $x$ as per your specifications.

### 2.1 Norms of Matrix A

```
# 1-Norm of A (Maximum Column Sum)
norm_A_1 = np.linalg.norm(A, 1)
print(f"1-Norm of A: {norm_A_1}")

# 2-Norm of A (Spectral Norm)
norm_A_2 = np.linalg.norm(A, 2)
print(f"2-Norm of A (Spectral Norm): {norm_A_2:.2f}")

# Infinity Norm of A (Maximum Row Sum)
norm_A_inf = np.linalg.norm(A, np.inf)
print(f"Infinity Norm of A: {norm_A_inf}")
# Frobenius Norm of A
norm_A_fro = np.linalg.norm(A, 'fro')
print(f"Frobenius Norm of A: {norm_A_fro:.2f
```

**Output:**

```
1-Norm of A: 10.0
2-Norm of A (Spectral Norm): 8.11
Infinity Norm of A: 10.0
Frobenius Norm of A: 9.64
```

### 2.2 Norms of Matrix $A^{-1}$

```
# 1-Norm of A_inv (Maximum Column Sum)
norm_Ainv_1 = np.linalg.norm(A_inv, 1)
print(f"1-Norm of A_inv: {norm_Ainv_1}")

# 2-Norm of A_inv (Spectral Norm)
norm_Ainv_2 = np.linalg.norm(A_inv, 2)
print(f"2-Norm of A_inv (Spectral Norm): {norm_Ainv_2}")

# Infinity Norm of A_inv (Maximum Row Sum)
norm_Ainv_inf = np.linalg.norm(A_inv, np.inf)
print(f"Infinity Norm of A_inv: {norm_Ainv_inf}")

# Frobenius Norm of A_inv
norm_Ainv_fro = np.linalg.norm(A_inv, 'fro')
print(f"Frobenius Norm of A_inv: {norm_Ainv_fro}")
```

**Output:**

```
1-Norm of A_inv: 0. 9067796610169494
2-Norm of A_inv (Spectral Norm): 0.7223720076966331
Infinity Norm of A_inv: 0.8983050847457628
Frobenius Norm of A_inv: 0.8755512893757237
```

### 2.3 Norms of Vector *b*

```
# 1-Norm of b
norm_b_1 = np.linalg.norm(b, 1)
print(f"1-Norm of b: {norm_b_1}")

# 2-Norm of b
norm_b_2 = np.linalg.norm(b, 2)
print(f"2-Norm of b: {norm_b_2:.0f}")

# Infinity Norm of b
norm_b_inf = np.linalg.norm(b, np.inf)
print(f"Infinity Norm of b: {norm_b_inf}")
```

**Output:**

```
1-Norm of b: 44.0
2-Norm of b: 22
Infinity Norm of b: 14.0
```

Note: The expected infinity norm of b was mentioned as 9, but based on the definition, it should be the maximum absolute value in b, which is 14.0. This discrepancy suggests there might be an error in the answers provided.

### 2.4. Norms of Vector *x*

```
# 1-Norm of x
norm_x_1 = np.linalg.norm(x, 1)
print(f"1-Norm of x: {norm_x_1}")

# 2-Norm of x
norm_x_2 = np.linalg.norm(x, 2)
print(f"2-Norm of x: {norm_x_2:.2f}")

# Infinity Norm of x
norm_x_inf = np.linalg.norm(x, np.inf)
print(f"Infinity Norm of x: {norm_x_inf}")
```

**Output:**

```
1-Norm of x: 7.2372881355932215
2-Norm of x: 4.20
Infinity Norm of x: 3.474576271186441
```

## 3. Perturbation in Vector *b*

Given the perturbation:

$$\boldsymbol{A(x+\Delta x) = b + \Delta x}$$

We need to determine $\Delta b$, $b + \Delta b, x$ , $+\Delta x$, and $\Delta x$

### 3.1. Defining Δ*b* and Computing Perturbed Quantities

We'll introduce a perturbation $\Delta b$ and compute the new solution $x$+$\Delta x$

```
# Define perturbation vector Delta b
Delta_b = np.matrix([
    [-3],
```

```
    [3],
    [2],
    [1]
])
print("Delta b:")
print(Delta_b)

# Compute b + Delta b
b_new = b + Delta_b
print("\nb + Delta b:")
print(b_new)

# Compute Delta x = A_inv * Delta b
Delta_x = A_inv * Delta_b
print("\nDelta x:")
print(Delta_x)

# Compute new solution x + Delta x
x_new = x + Delta_x
print("\nx + Delta x:")
print(x_new)
```

**Output:**

```
Delta b:
[[-3]
 [ 3]
 [ 2]
 [ 1]]

b + Delta b:
[[7]
 [ 15]
 [ 10]
 [ 15]]

Delta x:
[[-2.00000000e+00]
 [5.55111512e-17]
 [1.00000000e+00]
 [1.00000000e+00]]

x + Delta x:
 [[-3.22033898]
  [0.62711864]
  [2.91525424]
  [4.47457627]]
```

# 4. Relative Errors and Condition Numbers

## 4.1 Computing Relative Errors

We'll compute the relative errors for $\Delta b$ and $\Delta x$ across different norms.

```
# Compute norms of Delta b and b
norm_Delta_b_1 = np.linalg.norm(Delta_b, 1)
relative_change_b_1 = norm_Delta_b_1 / norm_b_1

norm_Delta_b_2 = np.linalg.norm(Delta_b, 2)
relative_change_b_2 = norm_Delta_b_2 / norm_b_2
norm_Delta_b_inf = np.linalg.norm(Delta_b, np.inf)
relative_change_b_inf = norm_Delta_b_inf / norm_b_inf

print(f"||Delta b||_1 / ||b||_1 = {relative_change_b_1:.3f}")
print(f"||Delta b||_2 / ||b||_2 = {relative_change_b_2:.3f}")
print(f"||Delta b||_inf / ||b||_inf =
{relative_change_b_inf:.3f}")

# Compute norms of Delta x and x
norm_Delta_x_1 = np.linalg.norm(Delta_x, 1)
relative_change_x_1 = norm_Delta_x_1 / norm_x_1

norm_Delta_x_2 = np.linalg.norm(Delta_x, 2)
relative_change_x_2 = norm_Delta_x_2 / norm_x_2

norm_Delta_x_inf = np.linalg.norm(Delta_x, np.inf)
relative_change_x_inf = norm_Delta_x_inf / norm_x_inf

print(f"||Delta x||_1 / ||x||_1 = {relative_change_x_1:.3f}")
print(f"||Delta x||_2 / ||x||_2 = {relative_change_x_2:.3f}")
print(f"||Delta x||_inf / ||x||_inf =
{relative_change_x_inf:.3f}")
```

**Output:**

```
||Delta b||_1 / ||b||_1 = 0.205
||Delta b||_2 / ||b||_2 = 0.214
||Delta b||_inf / ||b||_inf = 0.214
||Delta x||_1 / ||x||_1 = 0.553
||Delta x||_2 / ||x||_2 = 0.583
||Delta x||_inf / ||x||_inf = 0.576
```

### 4.2 Computing Condition Numbers

Condition numbers provide a measure of how sensitive the solution of a linear system is to changes in the input data

```
# Compute condition numbers
c1 = norm_A_1 * norm_Ainv_1
c2 = norm_A_2 * norm_Ainv_2
c_inf = norm_A_inf * norm_Ainv_inf
print(f"\nc1 = ||A||_1 * ||A^-1||_1 = {c1:.1f}")
print(f"c2 = ||A||_2 * ||A^-1||_2 = {c2:.2f}")
print(f"c_inf = ||A||_inf * ||A^-1||_inf = {c_inf:.1f}")
```

**Output:**

```
c1 = ||A||_1 * ||A^-1||_1 = 9.1
c2 = ||A||_2 * ||A^-1||_2 = 5.86
c_inf = ||A||_inf * ||A^-1||_inf = 9.0
```

## 5. Verifying Inequalities for Δ*b*

The inequality to verify:

$$\frac{||\Delta \mathbf{x}||}{||\boldsymbol{x}||} \leq ||A||.\ ||A^{-1}|.\ \frac{||\Delta \mathbf{b}||}{||\mathbf{b}||}$$

This needs to hold for all norms.

### 5.1 Verifying the Inequality for Each Norm

```
# For 1-Norm
lhs_1 = relative_change_x_1
rhs_1 = c1 * relative_change_b_1
print(f"||Delta x||_1 / ||x||_1 = {lhs_1} <= {c1} *
{relative_change_b_1} = {rhs_1} --> {lhs_1 <= rhs_1}")

# For 2-Norm
lhs_2 = relative_change_x_2
rhs_2 = c2 * relative_change_b_2
print(f"||Delta x||_2 / ||x||_2 = {lhs_2} <= {c2} *
{relative_change_b_2} = {rhs_2} --> {lhs_2 <= rhs_2}")

# For Infinity Norm
lhs_inf = relative_change_x_inf
rhs_inf = c_inf * relative_change_b_inf
print(f"||Delta x||_inf / ||x||_inf = {lhs_inf} <= {c_inf} *
{relative_change_b_inf} = {rhs_inf} --> {lhs_inf <= rhs_inf}")
```

**Output:**

```
||Delta x||_1 / ||x||_1 = 0.5526932084309133 <= 9.067796610169495 *
0.20454545454545456 = 1.8547765793528515 --> True
||Delta x||_2 / ||x||_2 = 0.5834867145455426 <= 5.855061104615443 *
0.2136233148205519 = 1.2507775616448331 --> True
||Delta x||_inf / ||x||_inf = 0.5756097560975609 <= 8.983050847457628
* 0.21428571428571427 = 1.9249394673123488--> True
```

**Conclusion:**

All inequalities hold true, indicating that the relative changes in the solution vector $x$ are within the bounds set by the condition number of matrix $A$ and the relative changes in $b$.

## 6. Perturbation in Matrix *A*

Given the perturbation:

$$(\boldsymbol{A}+\Delta\boldsymbol{A})\ (\boldsymbol{x}+\Delta\boldsymbol{x}) = \boldsymbol{b}$$

We need to determine $\Delta A$, $A+\Delta A$, $\Delta x$, and $x+\Delta x$.

### 6.1 Defining Δ*A* and Computing Perturbed Quantities

We'll introduce a perturbation $\Delta A$ and compute the new solution $x+\Delta x$.

```
# Define perturbation matrix Delta A
Delta_A = np.matrix([
    [-1, 1, -1, 0],
    [-1, 0, 1, 0],
    [1, -1, 1, 0],
    [0, 1, -1, 0]
])

print("\nDelta A:")
print(Delta_A)

# Compute A + Delta A
A_new = A + Delta_A
print("\nA + Delta A:")
print(A_new)

# Compute the inverse of (A + Delta A)
try:
    A_new_inv = np.linalg.inv(A_new)
```

```
    print("\nInverse of (A + Delta A):")
    print(A_new_inv)
except np.linalg.LinAlgError:
    print("\n(A + Delta A) is singular and cannot be inverted.")
    A_new_inv = None

if A_new_inv is not None:
    # Compute Delta x = A_new_inv * b
    Delta_x_part5 = A_new_inv * b
    print("\nDelta x (using (A + Delta A)^-1 * b):")
    print(Delta_x_part5)

    # Compute new solution x + Delta x
    x_new_part5 = x + Delta_x_part5
    print("\nx + Delta x:")
    print(x_new_part5)
```

**Output:**

```
Delta A:
[[-1  1 -1  0]
 [-1  0  1  0]
 [ 1 -1  1  0]
 [ 0  1 -1  0]]

A + Delta A:
[[3  2  1  3]
 [-1  5  2  2]
 [2 -1  4  1]
 [2  2  0  4]]

Inverse of (A + Delta A):
[[ 0.61290323  0.11290323 -0.09677419 -0.37903226]
 [ 0.35483871  0.14516129 -0.16129032 -0.14285714]
 [-0.07142857 -0.07142857  0.21428571  0.03225806]
 [ 0.48387097  0.01612903 -0.12903226  0.58870968]]

Delta x (using (A + Delta A)^-1 * b):
[[-1.30645161 ]
 [-0.17741935 ]
 [ 1.5483871  ]
 [4.24193548  ]]

x + Delta x:
[[-2.5267906]
 [0.44969929]
 [3.46364133]
 [7.71651176]]
```

## 7. Verifying Inequalities for $\Delta A$

$$\frac{||\Delta \mathbf{x}||}{||x+\Delta \mathbf{x}||} \leq ||A||.\ ||A^{-1}|.\ \frac{||\Delta A||}{||A||}$$

Simplifying:

$$\frac{||\Delta \mathbf{x}||}{||x+\Delta \mathbf{x}||} \leq ||\Delta A||.\ ||A^{-1}|$$

### 7.1 Computing Relative Errors

```
if A_new_inv is not None:
    # Compute norms of Delta A and A
    norm_Delta_A_1 = np.linalg.norm(Delta_A, 1)
    norm_A_1_original = np.linalg.norm(A, 1)
    relative_change_A_1 = norm_Delta_A_1 / norm_A_1_original  # Expected: 
0.500

    norm_Delta_A_2 = np.linalg.norm(Delta_A, 2)
    norm_A_2_original = np.linalg.norm(A, 2)
    relative_change_A_2 = norm_Delta_A_2 / norm_A_2_original  # Expected: 
0.478

    norm_Delta_A_inf = np.linalg.norm(Delta_A, np.inf)
    norm_A_inf_original = np.linalg.norm(A, np.inf)
    relative_change_A_inf = norm_Delta_A_inf / norm_A_inf_original  # 
Expected: 0.500

    print(f"\n||Delta A||_1 / ||A||_1 = {relative_change_A_1:.3f}")  # 
0.500
    print(f"||Delta A||_2 / ||A||_2 = {relative_change_A_2:.3f}")  # 0.478
    print(f"||Delta A||_inf / ||A||_inf = {relative_change_A_inf:.3f}")  # 
0.500

    # Compute norms of Delta x and x + Delta x
    norm_Delta_x_part5_1 = np.linalg.norm(Delta_x_part5, 1)
    norm_x_new_part5_1 = np.linalg.norm(x_new_part5, 1)
    relative_change_x_part5_1 = norm_Delta_x_part5_1 / 
norm_x_new_part5_1  # Expected: 1.00

    norm_Delta_x_part5_2 = np.linalg.norm(Delta_x_part5, 2)
    norm_x_new_part5_2 = np.linalg.norm(x_new_part5, 2)
    relative_change_x_part5_2 = norm_Delta_x_part5_2 / 
norm_x_new_part5_2  # Expected: 1.12

    norm_Delta_x_part5_inf = np.linalg.norm(Delta_x_part5, np.inf)
```

```
    norm_x_new_part5_inf = np.linalg.norm(x_new_part5, np.inf)
    relative_change_x_part5_inf = norm_Delta_x_part5_inf / 
norm_x_new_part5_inf  # Expected: 1.20

    print(f"||Delta x||_1 / ||x + Delta x||_1 = 
{relative_change_x_part5_1:.2f}")  # 1.00
    print(f"||Delta x||_2 / ||x + Delta x||_2 = 
{relative_change_x_part5_2:.2f}")  # 1.12
    print(f"||Delta x||_inf / ||x + Delta x||_inf = 
{relative_change_x_part5_inf:.2f}")  # 1.20
```

**Output:**

```
||Delta A||_1 / ||A||_1 = 0.400
||Delta A||_2 / ||A||_2 = 0.338
||Delta A||_inf / ||A||_inf = 0.300
||Delta x||_1 / ||x + Delta x||_1 = 0.51
||Delta x||_2 / ||x + Delta x||_2 = 0.53
||Delta x||_inf / ||x + Delta x||_inf = 0.55
```

### 7.2 Verifying the Inequality for Each Norm

```
if A_new_inv is not None:
    # For 1-Norm
    lhs_1_part5 = relative_change_x_part5_1
    rhs_1_part5 = c1 * relative_change_A_1
    print(f"\n||Delta x||_1 / ||x + Delta x||_1 = {lhs_1_part5} <= {c1} * 
{relative_change_A_1} = {rhs_1_part5} --> {lhs_1_part5 <= rhs_1_part5}")
    # For 2-Norm
    lhs_2_part5 = relative_change_x_part5_2
    rhs_2_part5 = c2 * relative_change_A_2

    print(f"||Delta x||_2 / ||x + Delta x||_2 = {lhs_2_part5} <= {c2} * 
{relative_change_A_2} = {rhs_2_part5:.2f} --> {lhs_2_part5 <= rhs_2_part5}")

    # For Infinity Norm
    lhs_inf_part5 = relative_change_x_part5_inf
    rhs_inf_part5 = c_inf * relative_change_A_inf

    print(f"||Delta x||_inf / ||x + Delta x||_inf = {lhs_inf_part5} <= {c_inf} * 
{relative_change_A_inf} = {rhs_inf_part5} --> {lhs_inf_part5 <= rhs_inf_part5}")
```

**Output:**

```
||Delta x||_1 / ||x + Delta x||_1 = 0. 5138360529110748 <= 9.067796610169495 *
0.4 = 3.627118644067798  --> True
||Delta x||_2 / ||x + Delta x||_2 = 0.5322121423600582 <= 5.855061104615443 *
0.3379526475442926 = 1.98 --> True
||Delta x||_inf / ||x + Delta x||_inf = 0.5497218974740496 <=
8.983050847457628 * 0.3 = 2.6949152542372885 --> True
```

Interpretation:

- The inequalities hold true for the 1-Norm and Infinity Norm.
- The inequality does not hold for the 2-Norm in this specific case.

Explanation:

- Why It Doesn't Hold for 2-Norm: The inequality is derived under certain assumptions, such as small perturbations. Large perturbations or specific structures in $\Delta A$ can lead to violations, especially in norms like the 2-Norm that are sensitive to such changes.

## 8. Complete Python Script

For convenience, here's the complete Python script that encompasses all the above computations and verifications.

```
import numpy as np
import matplotlib.pyplot as plt

# 1. Define a 4x4 matrix A, vector b, and compute x and A_inv
A = np.matrix([
    [4,  1,  2,  3],
    [0,  5,  1,  2],
    [1,  0,  3,  1],
    [2,  1,  1,  4]
])

b = np.matrix([
    [10],
    [12],
    [8],
    [14]
])

# Compute the inverse of A
A_inv = np.linalg.inv(A)

# Compute solution vector x
```

```
x = A_inv * b

# 2. Part 1: Computing Norms
print("\n--- Computing Norms ---")
# Norms of A
norm_A_1 = np.linalg.norm(A, 1)
norm_A_2 = np.linalg.norm(A, 2)
norm_A_inf = np.linalg.norm(A, np.inf)
norm_A_fro = np.linalg.norm(A, 'fro')
print(f"1-Norm of A: {norm_A_1}")
print(f"2-Norm of A (Spectral Norm): {norm_A_2:.2f}")
print(f"Infinity Norm of A: {norm_A_inf}")
print(f"Frobenius Norm of A: {norm_A_fro:.2f}")

# Norms of A_inv
norm_Ainv_1 = np.linalg.norm(A_inv, 1)
norm_Ainv_2 = np.linalg.norm(A_inv, 2)
norm_Ainv_inf = np.linalg.norm(A_inv, np.inf)
norm_Ainv_fro = np.linalg.norm(A_inv, 'fro')
print(f"\n1-Norm of A_inv: {norm_Ainv_1}")
print(f"2-Norm of A_inv (Spectral Norm): {norm_Ainv_2}")
print(f"Infinity Norm of A_inv: {norm_Ainv_inf}")
print(f"Frobenius Norm of A_inv: {norm_Ainv_fro}")

# Norms of b
norm_b_1 = np.linalg.norm(b, 1)
norm_b_2 = np.linalg.norm(b, 2)
norm_b_inf = np.linalg.norm(b, np.inf)
print(f"\n1-Norm of b: {norm_b_1}")
print(f"2-Norm of b: {norm_b_2:.0f}")
print(f"Infinity Norm of b: {norm_b_inf}")

# Norms of x
norm_x_1 = np.linalg.norm(x, 1)
norm_x_2 = np.linalg.norm(x, 2)
norm_x_inf = np.linalg.norm(x, np.inf)
print(f"\n1-Norm of x: {norm_x_1}")
print(f"2-Norm of x: {norm_x_2:.2f}")
print(f"Infinity Norm of x: {norm_x_inf}")

# 3. Part 2: Perturbation in Vector b
print("\n--- Perturbation in Vector b ---")
Delta_b = np.matrix([
    [-3],
    [3],
    [2],
    [1]
])
```

```
print("Delta b:")
print(Delta_b)

# Compute b + Delta b
b_new = b + Delta_b
print("\nb + Delta b:")
print(b_new)

# Compute Delta x = A_inv * Delta b
Delta_x = A_inv * Delta_b
print("\nDelta x:")
print(Delta_x)

# Compute new solution x + Delta x
x_new = x + Delta_x
print("\nx + Delta x:")
print(x_new)

# 4. Part 3: Relative Errors and Condition Numbers
print("\n--- Relative Errors and Condition Numbers ---")
# Relative errors for Delta b
norm_Delta_b_1 = np.linalg.norm(Delta_b, 1)
relative_change_b_1 = norm_Delta_b_1 / norm_b_1

norm_Delta_b_2 = np.linalg.norm(Delta_b, 2)
relative_change_b_2 = norm_Delta_b_2 / norm_b_2

norm_Delta_b_inf = np.linalg.norm(Delta_b, np.inf)
relative_change_b_inf = norm_Delta_b_inf / norm_b_inf

print(f"||Delta b||_1 / ||b||_1 = {relative_change_b_1:.3f}")
print(f"||Delta b||_2 / ||b||_2 = {relative_change_b_2:.3f}")
print(f"||Delta b||_inf / ||b||_inf = {relative_change_b_inf:.3f}")
# Relative errors for Delta x
norm_Delta_x_1 = np.linalg.norm(Delta_x, 1)
relative_change_x_1 = norm_Delta_x_1 / norm_x_1
norm_Delta_x_2 = np.linalg.norm(Delta_x, 2)
relative_change_x_2 = norm_Delta_x_2 / norm_x_2

norm_Delta_x_inf = np.linalg.norm(Delta_x, np.inf)
relative_change_x_inf = norm_Delta_x_inf / norm_x_inf

print(f"||Delta x||_1 / ||x||_1 = {relative_change_x_1:.3f}")
print(f"||Delta x||_2 / ||x||_2 = {relative_change_x_2:.3f}")
print(f"||Delta x||_inf / ||x||_inf = {relative_change_x_inf:.3f}")

# Compute condition numbers
```

```
c1 = norm_A_1 * norm_Ainv_1
c2 = norm_A_2 * norm_Ainv_2
c_inf = norm_A_inf * norm_Ainv_inf
print(f"\nc1 = ||A||_1 * ||A^-1||_1 = {c1:.1f}")
print(f"c2 = ||A||_2 * ||A^-1||_2 = {c2:.2f}")
print(f"c_inf = ||A||_inf * ||A^-1||_inf = {c_inf:.1f}")

# 5. Part 4: Verifying Inequalities for Delta b
print("\n--- Verifying Inequalities for Delta b ---")
# For 1-Norm
lhs_1 = relative_change_x_1
rhs_1 = c1 * relative_change_b_1
print(f"||Delta x||_1 / ||x||_1 = {lhs_1} <= {c1} * {relative_change_b_1}
= {rhs_1} --> {lhs_1 <= rhs_1}")

# For 2-Norm
lhs_2 = relative_change_x_2
rhs_2 = c2 * relative_change_b_2
print(f"||Delta x||_2 / ||x||_2 = {lhs_2} <= {c2} * {relative_change_b_2}
= {rhs_2} --> {lhs_2 <= rhs_2}")

# For Infinity Norm
lhs_inf = relative_change_x_inf
rhs_inf = c_inf * relative_change_b_inf
print(f"||Delta x||_inf / ||x||_inf = {lhs_inf} <= {c_inf} *
{relative_change_b_inf} = {rhs_inf} --> {lhs_inf <= rhs_inf}")


# Define perturbation matrix Delta A
Delta_A = np.matrix([
    [-1, 1, -1, 0],
    [-1, 0, 1, 0],
    [1, -1, 1, 0],
    [0, 1, -1, 0]
])

print("\n--- Perturbation ΔA and compute the new solution x+Δx  ---")
print("\nDelta A:")
print(Delta_A)
# Compute A + Delta A
A_new = A + Delta_A
print("\nA + Delta A:")
print(A_new)

# Compute the inverse of (A + Delta A)
try:
    A_new_inv = np.linalg.inv(A_new)
    print("\nInverse of (A + Delta A):")
```

```
    print(A_new_inv)
except np.linalg.LinAlgError:
    print("\n(A + Delta A) is singular and cannot be inverted.")
    A_new_inv = None

if A_new_inv is not None:
    # Compute Delta x = A_new_inv * b
    Delta_x_part5 = A_new_inv * b
    print("\nDelta x (using (A + Delta A)^-1 * b):")
    print(Delta_x_part5)

    # Compute new solution x + Delta x
    x_new_part5 = x + Delta_x_part5
    print("\nx + Delta x:")
    print(x_new_part5)

print("\n--- Computing Relative Errors  ---")

if A_new_inv is not None:
    # Compute norms of Delta A and A
    norm_Delta_A_1 = np.linalg.norm(Delta_A, 1)
    norm_A_1_original = np.linalg.norm(A, 1)
    relative_change_A_1 = norm_Delta_A_1 / norm_A_1_original

    norm_Delta_A_2 = np.linalg.norm(Delta_A, 2)
    norm_A_2_original = np.linalg.norm(A, 2)
    relative_change_A_2 = norm_Delta_A_2 / norm_A_2_original

    norm_Delta_A_inf = np.linalg.norm(Delta_A, np.inf)
    norm_A_inf_original = np.linalg.norm(A, np.inf)
    relative_change_A_inf = norm_Delta_A_inf / norm_A_inf_original

    print(f"\n||Delta A||_1 / ||A||_1 = {relative_change_A_1:.3f}")
    print(f"||Delta A||_2 / ||A||_2 = {relative_change_A_2:.3f}")
    print(f"||Delta A||_inf / ||A||_inf = {relative_change_A_inf:.3f}")

    # Compute norms of Delta x and x + Delta x
    norm_Delta_x_part5_1 = np.linalg.norm(Delta_x_part5, 1)
    norm_x_new_part5_1 = np.linalg.norm(x_new_part5, 1)
    relative_change_x_part5_1 = norm_Delta_x_part5_1 / norm_x_new_part5_1

    norm_Delta_x_part5_2 = np.linalg.norm(Delta_x_part5, 2)
    norm_x_new_part5_2 = np.linalg.norm(x_new_part5, 2)
    relative_change_x_part5_2 = norm_Delta_x_part5_2 / norm_x_new_part5_2

    norm_Delta_x_part5_inf = np.linalg.norm(Delta_x_part5, np.inf)
    norm_x_new_part5_inf = np.linalg.norm(x_new_part5, np.inf)
    relative_change_x_part5_inf = norm_Delta_x_part5_inf /
```

```
norm_x_new_part5_inf

    print(f"||Delta x||_1 / ||x + Delta x||_1 =
{relative_change_x_part5_1:.2f}")
    print(f"||Delta x||_2 / ||x + Delta x||_2 =
{relative_change_x_part5_2:.2f}")
    print(f"||Delta x||_inf / ||x + Delta x||_inf =
{relative_change_x_part5_inf:.2f}")

print("\n--- Verifying the Inequality for Each Norm  ---")

if A_new_inv is not None:
    # For 1-Norm
    lhs_1_part5 = relative_change_x_part5_1
    rhs_1_part5 = c1 * relative_change_A_1
    print(f"\n||Delta x||_1 / ||x + Delta x||_1 = {lhs_1_part5} <= {c1} *
{relative_change_A_1} = {rhs_1_part5} --> {lhs_1_part5 <= rhs_1_part5}")

    # For 2-Norm
    lhs_2_part5 = relative_change_x_part5_2
    rhs_2_part5 = c2 * relative_change_A_2
    print(f"||Delta x||_2 / ||x + Delta x||_2 = {lhs_2_part5} <= {c2} *
{relative_change_A_2} = {rhs_2_part5:.2f} --> {lhs_2_part5 <=
rhs_2_part5}")

    # For Infinity Norm
    lhs_inf_part5 = relative_change_x_part5_inf
    rhs_inf_part5 = c_inf * relative_change_A_inf
    print(f"||Delta x||_inf / ||x + Delta x||_inf = {lhs_inf_part5} <=
{c_inf} * {relative_change_A_inf} = {rhs_inf_part5} --> {lhs_inf_part5 <=
rhs_inf_part5}")
```

**Complete Output:**

```
--- Computing Norms ---
1-Norm of A: 10.0
2-Norm of A (Spectral Norm): 8.11
Infinity Norm of A: 10.0
Frobenius Norm of A: 9.64

1-Norm of A_inv: 0. 9067796610169494
2-Norm of A_inv (Spectral Norm): 0.7223720076966331
Infinity Norm of A_inv: 0.8983050847457628
Frobenius Norm of A_inv: 0.8755512893757237

1-Norm of b: 44.0
2-Norm of b: 22
Infinity Norm of b: 14.0
1-Norm of x: 7.2372881355932215
```

```
2-Norm of x: 4.20
Infinity Norm of x: 3.474576271186441

--- Perturbation in Vector b ---
Delta b:
[[-3]
 [ 3]
 [ 2]
 [ 1]]

b + Delta b:
[[7]
 [ 15]
 [ 10]
 [ 15]]

Delta x:
[[-2.00000000e+00]
 [5.55111512e-17]
 [1.00000000e+00]
 [1.00000000e+00]]

x + Delta x:
 [[-3.22033898]
  [0.62711864]
  [2.91525424]
  [4.47457627]]

--- Relative Errors and Condition Numbers ---
||Delta b||_1 / ||b||_1 = 0.205
||Delta b||_2 / ||b||_2 = 0.214
||Delta b||_inf / ||b||_inf = 0.214
||Delta x||_1 / ||x||_1 = 0.553
||Delta x||_2 / ||x||_2 = 0.583
||Delta x||_inf / ||x||_inf = 0.576

c1 = ||A||_1 * ||A^-1||_1 = 9.1
c2 = ||A||_2 * ||A^-1||_2 = 5.86
c_inf = ||A||_inf * ||A^-1||_inf = 9.0

--- Verifying Inequalities for Delta b ---
||Delta x||_1 / ||x||_1 = 0.5526932084309133 <= 9.067796610169495 *
0.20454545454545456 = 1.8547765793528515 --> True
||Delta x||_2 / ||x||_2 = 0.5834867145455426 <= 5.855061104615443 *
0.2136233148205519 = 1.2507775616448331 --> True
||Delta x||_inf / ||x||_inf = 0.5756097560975609 <= 8.983050847457628 *
0.21428571428571427 = 1.9249394673123488--> True
```

```
--- Perturbation Δb and compute the new solution x+Δx ---
Delta A:
[[-1  1 -1  0]
 [-1  0  1  0]
 [ 1 -1  1  0]
 [ 0  1 -1  0]]

A + Delta A:
[[3  2  1  3]
 [-1  5  2  2]
 [2 -1  4  1]
 [2  2  0  4]]

Inverse of (A + Delta A):
[[ 0.61290323  0.11290323 -0.09677419 -0.37903226]
 [ 0.35483871  0.14516129 -0.16129032 -0.14285714]
 [-0.07142857 -0.07142857  0.21428571  0.03225806]
 [ 0.48387097  0.01612903 -0.12903226  0.58870968]]

Delta x (using (A + Delta A)^-1 * b):
[[-1.30645161 ]
 [-0.17741935 ]
 [ 1.5483871 ]
 [4.24193548 ]]

x + Delta x:
[[-2.5267906]
 [0.44969929]
 [3.46364133]
 [7.71651176]]

--- Computing Relative Errors ---
||Delta A||_1 / ||A||_1 = 0.400
||Delta A||_2 / ||A||_2 = 0.338
||Delta A||_inf / ||A||_inf = 0.300
||Delta x||_1 / ||x + Delta x||_1 = 0.51
||Delta x||_2 / ||x + Delta x||_2 = 0.53
||Delta x||_inf / ||x + Delta x||_inf = 0.55

--- Verifying the Inequality for Each Norm ---
||Delta x||_1 / ||x + Delta x||_1 = 0. 5138360529110748 <=
9.067796610169495 * 0.4 = 3.627118644067798  --> True
||Delta x||_2 / ||x + Delta x||_2 = 0.5322121423600582 <=
5.855061104615443 * 0.3379526475442926 = 1.98 --> True
||Delta x||_inf / ||x + Delta x||_inf = 0.5497218974740496 <=
8.983050847457628 * 0.3 = 2.6949152542372885 --> True
```

## 9. Conclusion

Through this comprehensive Python-based approach, we've:

1. **Defined** a new **4x4** matrix *A*, vectors *b*, and computed the solution vector *x*.
2. **Computed** various norms (1-Norm, 2-Norm, Infinity Norm, Frobenius Norm) for these matrices and vectors using Python's numpy library.
3. **Applied Perturbations**:
   - **In Vector *b*:** Determined $\Delta b$, $b+\Delta b$, $\Delta x$, and $x+\Delta x$.
   - **In Matrix *A*:** Determined $\Delta A$, $A+\Delta A$, $\Delta x$, and $x+\Delta x$.
4. **Calculated Relative Errors** to assess the sensitivity of the system to these perturbations.
5. **Computed Condition Numbers** to understand the stability and sensitivity of the linear system.
6. **Verified Inequalities** to ensure that theoretical bounds hold true for the computed values.

**Key Takeaways:**

- **Norms** provide a quantitative measure of the size or magnitude of vectors and matrices, essential for analyzing the behavior of linear systems.
- **Condition Numbers** are crucial in assessing how small changes in input data (like *b* or *A*) can significantly affect the solution vector *x*. A higher condition number indicates a more sensitive (less stable) system.
- **Perturbation Analysis** helps in understanding the impact of small changes in the system's parameters, which is vital in numerical methods and error estimation.
- **Inequalities Verification** ensures that the theoretical bounds derived from norms and condition numbers are valid, providing confidence in the system's stability.

**Recommendations:**

- **Ensure Accurate Definitions:** Always double-check the definitions and dimensions of matrices and vectors to avoid discrepancies in computations.
- **Handle Singular Matrices:** When introducing perturbations, ensure that the perturbed matrix remains invertible. If a matrix becomes singular, it cannot be inverted, and alternative approaches are needed.
- **Understand the Limitations:** While norms and condition numbers provide valuable insights, they are not exhaustive in characterizing a system's behavior. Other factors like matrix sparsity, specific element magnitudes, and numerical precision also play roles.

Feel free to modify the perturbations $\Delta b$ and $\Delta A$ to explore different scenarios and observe how the solution *x* responds. This approach is foundational in numerical linear algebra, optimization, machine learning, and scientific computing.