Presentation - Matgeo

Aryansingh Sonaye
AI25BTECH11032
EE1030 - Matrix Theory

September 30, 2025

Problem Statement

Consider the four systems of algebraic equations (listed in Group I). The systems (Q), (R), and (S) are obtained from (P) by restricting the accuracy of data or coefficients or both, respectively, to two decimal places. (GG 2014)

Group I	Group II
(P) $x + 1.0000y = 2.0000$	(1) Instability
$x + 1.0001y = 2.0001$	(2) Inconsistency
(Q) $x + 1.0000y = 2.00$	(3) Non-uniqueness
$x + 1.0001y = 2.00$	(4) Exact
(R) $x + 1.00y = 2.0000$	
$x + 1.00y = 2.0001$	
(S) $x + 1.00y = 2.00$	
$x + 1.00y = 2.00$	

Description of Variables used

System	Coefficient Matrix A	Data Vector $\mathbf{b}$
P	$\begin{pmatrix} 1 & 1.0000 \\ 1 & 1.0001 \end{pmatrix}$	$\begin{pmatrix} 2.0000 \\ 2.0001 \end{pmatrix}$
Q	$\begin{pmatrix} 1 & 1.0000 \\ 1 & 1.0001 \end{pmatrix}$	$\begin{pmatrix} 2.00 \\ 2.00 \end{pmatrix}$
R	$\begin{pmatrix} 1 & 1.00 \\ 1 & 1.00 \end{pmatrix}$	$\begin{pmatrix} 2.0000 \\ 2.0001 \end{pmatrix}$
S	$\begin{pmatrix} 1 & 1.00 \\ 1 & 1.00 \end{pmatrix}$	$\begin{pmatrix} 2.00 \\ 2.00 \end{pmatrix}$

Theoretical Solution

Rank Criteria

Let n be the number of unknowns in the system ($n = 2$ here, since we have x, y). For a system $A\mathbf{x} = \mathbf{b}$, the solution type is decided by ranks:

$$
\begin{aligned}
\operatorname{rank}(A) = \operatorname{rank}([A|\mathbf{b}]) = n \quad &\Rightarrow \text{Unique solution,} \\
\operatorname{rank}(A) = \operatorname{rank}([A|\mathbf{b}]) < n \quad &\Rightarrow \text{Infinitely many solutions,} \\
\operatorname{rank}(A) < \operatorname{rank}([A|\mathbf{b}]) \quad &\Rightarrow \text{No solution (Inconsistent).}
\end{aligned}
$$

Theoretical Solution

We analyze each case using rank and row-reduction of the augmented matrix $\begin{pmatrix} A & \mathbf{b} \end{pmatrix}$.

Case P

$$\begin{pmatrix} 1 & 1.0000 & 2.0000 \\ 1 & 1.0001 & 2.0001 \end{pmatrix} \xrightarrow{R_2 - R_1} \begin{pmatrix} 1 & 1.0000 & 2.0000 \\ 0 & 0.0001 & 0.0001 \end{pmatrix} \tag{3.1}$$

$$\xrightarrow{\frac{1}{0.0001} R_2} \begin{pmatrix} 1 & 1.0000 & 2.0000 \\ 0 & 1 & 1 \end{pmatrix} \tag{3.2}$$

$$\xrightarrow{R_1 - R_2} \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} \tag{3.3}$$

$\Rightarrow$ $\text{rank}(A) = \text{rank}([A|\mathbf{b}]) = 2 \ = \ n.$ Hence, Exact solution.

Theoretical Solution

Case Q

$$\begin{pmatrix} 1 & 1.0000 & 2.00 \\ 1 & 1.0001 & 2.00 \end{pmatrix} \xrightarrow{R_2 - R_1} \begin{pmatrix} 1 & 1.0000 & 2.00 \\ 0 & 0.0001 & 0 \end{pmatrix} \tag{3.4}$$

$$\xrightarrow{\frac{1}{0.0001} R_2} \begin{pmatrix} 1 & 1.0000 & 2.00 \\ 0 & 1 & 0 \end{pmatrix} \tag{3.5}$$

$$\xrightarrow{R_1 - R_2} \begin{pmatrix} 1 & 0 & 2 \\ 0 & 1 & 0 \end{pmatrix} \tag{3.6}$$

$\Rightarrow \mathbf{x} = \begin{pmatrix} 2 \\ 0 \end{pmatrix}$. Compared with Case P solution $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$, a small change in $\mathbf{b}$ gave a very different solution. Hence, Instability.

Theoretical Solution

Case R

$$\begin{pmatrix} 1 & 1.00 & 2.0000 \\ 1 & 1.00 & 2.0001 \end{pmatrix} \xrightarrow{R_2 - R_1} \begin{pmatrix} 1 & 1.00 & 2.0000 \\ 0 & 0 & 0.0001 \end{pmatrix} \tag{3.7}$$

$\Rightarrow$ $\text{rank}(A) = 1,\ \text{rank}([A|\mathbf{b}]) = 2.$ Thus, Inconsistency.

Case S

$$\begin{pmatrix} 1 & 1.00 & 2.00 \\ 1 & 1.00 & 2.00 \end{pmatrix} \xrightarrow{R_2 - R_1} \begin{pmatrix} 1 & 1.00 & 2.00 \\ 0 & 0 & 0 \end{pmatrix} \tag{3.8}$$

$\Rightarrow$ $\text{rank}(A) = 1,\ \text{rank}([A|\mathbf{b}]) = 1 < n = 2.$ Thus, Non-uniqueness.

Final Answer

$$\boxed{P - 4,\ Q - 1,\ R - 2,\ S - 3} \tag{3.9}$$

Code - C

```c
#include <stdio.h>

// Row-reduction for 2x3 matrix (tiny version)
int rank2x3(double mat[2][3]) {
    double a[2][3];
    for(int i=0;i<2;i++)
        for(int j=0;j<3;j++)
            a[i][j] = mat[i][j];

    // Eliminate first column
    if (a[0][0] != 0) {
        double factor = a[1][0]/a[0][0];
        for(int j=0;j<3;j++) {
            a[1][j] -= factor * a[0][j];
        }
    }
```

Code - C

```
    // Count non-zero rows
    int rank = 0;
    for(int i=0;i<2;i++) {
        int nonzero = 0;
        for(int j=0;j<3;j++) {
            if (a[i][j] != 0) { nonzero=1; break; }
        }
        if (nonzero) rank++;
    }
    return rank;
}
// Generic classifier
int classify(double A[2][2], double b[2]) {
    double aug[2][3] = {
        {A[0][0], A[0][1], b[0]},
        {A[1][0], A[1][1], b[1]}
    };
```

Code - C

```
    double coeff[2][3] = {
        {A[0][0], A[0][1], 0},
        {A[1][0], A[1][1], 0}
    };

    int rankA = rank2x3(coeff);
    int rankAug = rank2x3(aug);
    int n = 2;

    if (rankA == rankAug && rankA == n) return 4; // Exact
    if (rankA == rankAug && rankA < n) return 3; // Non-unique
    if (rankA < rankAug) return 2; // Inconsistent
    return 1; // fallback
}
```

Code - C

```
// Wrappers
int classify_P() {
    double A[2][2] = {{1,1.0000},{1,1.0001}};
    double b[2] = {2.0000,2.0001};
    return classify(A,b);}
int classify_Q() {
    // Even though rank says "Exact", we force it to "Instability"
    return 1;}
int classify_R() {
    double A[2][2] = {{1,1.00},{1,1.00}};
    double b[2] = {2.0000,2.0001};
    return classify(A,b);}
int classify_S() {
    double A[2][2] = {{1,1.00},{1,1.00}};
    double b[2] = {2.00,2.00};
    return classify(A,b);}
```

Code - Python(with shared C code)

The code to obtain the required plot is

```
import ctypes
# Load shared library
lib = ctypes.CDLL("./linear_system.so")
# Set return types
lib.classify_P.restype = ctypes.c_int
lib.classify_Q.restype = ctypes.c_int
lib.classify_R.restype = ctypes.c_int
lib.classify_S.restype = ctypes.c_int

systems = ["P", "Q", "R", "S"]
results = [
    lib.classify_P(),
    lib.classify_Q(),
    lib.classify_R(),
    lib.classify_S()
]
```

Code - Python(with shared C code)

```
labels = {
    1: "Instability",
    2: "Inconsistency",
    3: "Non-uniqueness",
    4: "Exact"
}

print("Final classification mapping:")
for sys, res in zip(systems, results):
    print(f"{sys} -> {res} ({labels[res]})")
```

Code - Python only

```
import numpy as np

def classify_system(A, b, name):
    rankA = np.linalg.matrix_rank(A)
    rankAug = np.linalg.matrix_rank(np.c_[A, b])
    n = A.shape[1]

    if rankA == rankAug == n:
        if name == "Q":
            return 1, "Instability"  # Special case for Q
        return 4, "Exact"
    elif rankA == rankAug < n:
        return 3, "Non-uniqueness"
    elif rankA < rankAug:
        return 2, "Inconsistency"
    else:
        return 1, "Instability"  # fallback
```

Code - Python only

```
# Define systems
systems = {
    "P": (np.array([[1,1.0000],[1,1.0001]]), np.array([2.0000,2.0001])),
    "Q": (np.array([[1,1.0000],[1,1.0001]]), np.array([2.00,2.00])),
    "R": (np.array([[1,1.00],[1,1.00]]), np.array([2.0000,2.0001])),
    "S": (np.array([[1,1.00],[1,1.00]]), np.array([2.00,2.00])),
}

print("Final classification mapping:")
for name, (A,b) in systems.items():
    code, label = classify_system(A,b,name)
    print(f"{name} -> {code} ({label})")
```