Presentation - Matgeo

Aryansingh Sonaye
AI25BTECH11032
EE1030 - Matrix Theory

September 22, 2025

Problem Statement

Problem 4.13.101. Let p, q amd r be nonzero real numbers that are the $10^{th}, 100^{th}$ and 1000^{th} terms of a harmonic progression, respectively. Consider the following system of linear equations

$$x + y + z = 1 \tag{1.1}$$

$$10x + 100y + 1000z = 0 \tag{1.2}$$

$$qrx + pry + pqz = 0 \tag{1.3}$$

(I) If $\frac{q}{r} = 10$, then the system of linear equations has

(II) If $\frac{p}{r} \neq 100$, then the system of linear equations has

(III) If $\frac{p}{q} \neq 10$, then the system of linear equations has

(IV) If $\frac{p}{q} = 10$, then the system of linear equations has

Problem Statement

(A) $x = 0,\ y = \dfrac{10}{9},\ z = -\dfrac{1}{9}$ as a solution

(B) $x = \dfrac{10}{9},\ y = -\dfrac{1}{9},\ z = 0$ as a solution

(C) infinitely many solutions

(D) no solution

(E) at least one solution

Description of Variables used

Input Data

Given scalars:	$p, \ q, \ r$
HP relation (reciprocals in AP):	$\dfrac{1}{p} = a + 9d, \ \dfrac{1}{q} = a + 99d,$ $\dfrac{1}{r} = a + 999d$
Coefficient matrix (M) rows:	$\mathbf{R}_1 = (1, 1, 1), \ \mathbf{R}_2 = (10, 100, 1000),$ $\mathbf{R}_3 = (qr, pr, pq)$
RHS vector:	$\mathbf{b} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$

Table: Input data (scalars and vectors) derived from problem statement

Theoretical Solution

Given system of equations is:

$$\mathbf{x} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}.$$

$$\begin{pmatrix} 1 & 1 & 1 \\ 10 & 100 & 1000 \\ qr & pr & pq \end{pmatrix} \mathbf{x} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}.$$

Theoretical Solution

From the system of equations, the augmented matrix formed is:

$$[\,(M)\mid \mathbf{b}\,] = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 10 & 100 & 1000 & 0 \\ qr & pr & pq & 0 \end{pmatrix} \tag{3.1}$$

Eliminate first-column below row1: do $R_2 \leftarrow R_2 - 10R_1$ and $R_3 \leftarrow R_3 - (qr)R_1$:

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 10 & 100 & 1000 & 0 \\ qr & pr & pq & 0 \end{pmatrix} \xrightarrow{R_2-10R_1,\ R_3-qrR_1} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 90 & 990 & -10 \\ 0 & pr-qr & pq-qr & -qr \end{pmatrix} \tag{3.2}$$

Theoretical Solution

Now eliminate the (3,2) entry using row2. Set

$$s = \frac{pr - qr}{90}, \tag{3.3}$$

and do $R_3 \leftarrow R_3 - sR_2$. Compute the new third-row entries explicitly:

$$\begin{aligned}
(3,3)\colon \quad & (pq - qr) - s \cdot 990 && (3.4)\\
& = pq - qr - 990 \cdot \frac{pr - qr}{90} && (3.5)\\
& = pq - qr - 11(pr - qr) && (3.6)\\
& = pq - 11\,pr + 10\,qr \;:=\; D, && (3.7)
\end{aligned}$$

Theoretical Solution

$$
\begin{aligned}
(3,4)\colon \qquad & -qr - s(-10) && (3.8)\\
&= -qr + 10 \cdot \frac{pr - qr}{90} && (3.9)\\
&= -qr + \frac{pr - qr}{9} && (3.10)\\
&= \frac{pr - 10\,qr}{9} \;:=\; E. && (3.11)
\end{aligned}
$$

Thus the matrix in row-echelon form is

$$
[\,(M) \mid \mathbf{b}\,] = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 90 & 990 & -10 \\ 0 & 0 & D & E \end{pmatrix}, \tag{3.12}
$$

with

$$
D = pq - 11\,pr + 10\,qr, \qquad E = \frac{pr - 10\,qr}{9}. \tag{3.13}
$$

Theoretical Solution

Conclusions from the echelon form (standard linear algebra facts):

If $D \neq 0,$ the system has a unique solution.

If $D = 0$ but $E \neq 0,$ the system is inconsistent (no solution).

If $D = 0$ and $E = 0,$ the system has infinitely many solutions (rank 2).

Using the HP condition,
Because p, q, r are the $10^{\text{th}}, 100^{\text{th}}, 1000^{\text{th}}$ terms of an HP,

$$\frac{1}{p} = a + 9d, \qquad \frac{1}{q} = a + 99d, \qquad \frac{1}{r} = a + 999d \tag{3.14}$$

for some real a, d. Evaluate $D/(pqr)$ to simplify algebra:

$$\frac{D}{pqr} = \frac{1}{r} - \frac{11}{q} + \frac{10}{p} \tag{3.15}$$

$$= (a + 999d) - 11(a + 99d) + 10(a + 9d) \tag{3.16}$$

$$= a + 999d - 11a - 1089d + 10a + 90d = 0. \tag{3.17}$$

Theoretical Solution

Hence

$$\boxed{D \equiv 0 \text{ for every valid HP triple } (p, q, r).} \tag{3.18}$$

Therefore the coefficient matrix is singular and a unique solution is impossible.
Next compute $E/(pqr)$:

$$\frac{E}{pqr} = \frac{1}{9}\Big(\frac{1}{q} - \frac{10}{p}\Big) \tag{3.19}$$

$$= \frac{1}{9}\big((a + 99d) - 10(a + 9d)\big) \tag{3.20}$$

$$= \frac{1}{9}(-9a + 9d) = d - a. \tag{3.21}$$

Thus

$$\boxed{E = pqr\,(d - a).} \tag{3.22}$$

Theoretical Solution

So the system is consistent (infinitely many solutions) exactly when $E = 0$, i.e. when $d = a$. Equivalently,

$$d = a \quad \Longrightarrow \quad \frac{1}{p} = 10a, \ \frac{1}{q} = 100a, \ \frac{1}{r} = 1000a \tag{3.23}$$

$$\Longrightarrow \quad p : q : r = 100 : 10 : 1. \tag{3.24}$$

Parametric solution when consistent. If $d = a$ (equivalently $p : q : r = 100 : 10 : 1$) then the third equation is redundant and we can solve the first two:

$$x + y + z = 1, \tag{3.25}$$

$$10x + 100y + 1000z = 0. \tag{3.26}$$

Set $z = t$. Then $y = 1 - t - x$. Substitute into the second:

$$10x + 100(1 - t - x) = -1000t \tag{3.27}$$

$$-90x + 100 - 100t = -1000t \tag{3.28}$$

$$-90x = -900t - 100 \tag{3.29}$$

Theoretical Solution

$$x = 10t + \frac{10}{9}. \tag{3.30}$$

Thus the solution family is

$$\mathbf{x} = \begin{pmatrix} 10t + \frac{10}{9} \\ -11t - \frac{1}{9} \\ t \end{pmatrix}, \qquad t \in \mathbb{R}. \tag{3.31}$$

Two convenient particular choices:

$$t = -\frac{1}{9} \implies \mathbf{x} = \begin{pmatrix} 0 \\ \frac{10}{9} \\ -\frac{1}{9} \end{pmatrix} \quad \text{(matches option A)}, \tag{3.32}$$

$$t = 0 \implies \mathbf{x} = \begin{pmatrix} \frac{10}{9} \\ -\frac{1}{9} \\ 0 \end{pmatrix} \quad \text{(matches option B)}. \tag{3.33}$$

So when consistent both A and B are valid particular solutions, and there are infinitely many of them (C).

Now check cases (I)–(IV)

(I) If $\dfrac{q}{r} = 10.$

From reciprocals,

$$\frac{1/q}{1/r} = \frac{r}{q} = \frac{1}{10} \quad \Longrightarrow \quad \frac{a+99d}{a+999d} = \frac{1}{10}. \tag{3.34}$$

Multiply out:

$$10(a+99d) = a+999d \quad \Longrightarrow \quad 10a+990d = a+999d \quad \Longrightarrow \quad 9a = 9d, \tag{3.35}$$

so $a = d$. Therefore $E = 0$ and we are in the consistent case. Conclusion:

$$\text{(I)} \quad \boxed{\text{infinitely many solutions (option C). Also A and B are solutions.}} \tag{3.36}$$

(II) If $\dfrac{p}{r} \neq 100$.

Now $p/r \neq 100$ means $p \neq 100r$. Under the HP parametrisation, $p = 100r$ is equivalent to $a = d$ (see derivation above). Hence $p \neq 100r$ is equivalent to $a \neq d$. Then $E = pqr(d - a) \neq 0$. Since we already have $D \equiv 0$, $D = 0$ and $E \neq 0$ implies inconsistency. Conclusion:

$$\text{(II)} \quad \boxed{\text{no solution (option D).}} \tag{3.37}$$

(III) If $\dfrac{p}{q} \neq 10$.

Similarly $p/q = 10$ is equivalent to $a = d$ (check by $(a + 9d)/(a + 99d) = 1/10$ as in (I)). Therefore $p/q \neq 10$ implies $a \neq d$ and hence $E \neq 0$. With $D \equiv 0$ this gives inconsistency. Conclusion:

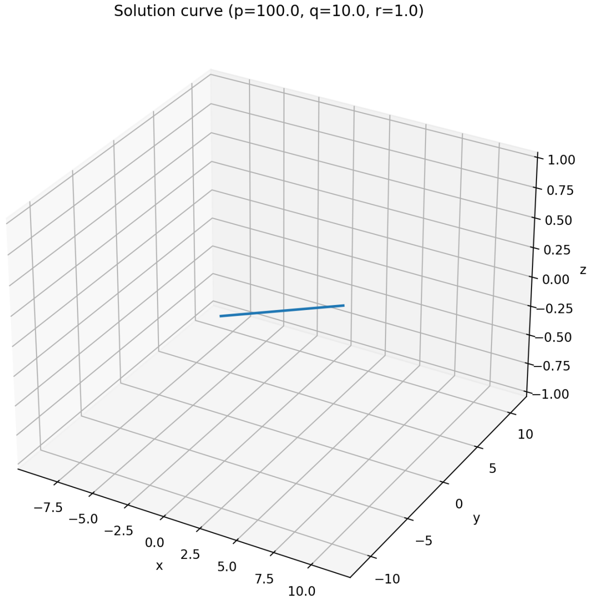
$$\text{(III)} \quad \boxed{\text{no solution (option D).}} \tag{3.38}$$

(IV) If $\dfrac{p}{q} = 10.$

As noted, $p/q = 10$ implies $a = d$. Thus $E = 0$ and the system is consistent with infinitely many solutions. Conclusion:

$$\text{(IV)} \quad \boxed{\text{infinitely many solutions (option C). Also A and B are solutions.}} \tag{3.39}$$

Plot


Solution curve (p=100.0, q=10.0, r=1.0)
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
z
10
5
0
−5
−10
y
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
x

Code - C

```
#include <stdio.h>
#include <math.h>
// Analyze the system: returns code (0=unique,1=no solution,2=infinitely
    many)
int analyze_system(double p, double q, double r, double *D_out,
    double *E_out){
double D = p*q - 11.0*p*r + 10.0*q*r;
double E = (p*r - 10.0*q*r) / 9.0;
if(D_out) *D_out = D;
if(E_out) *E_out = E;
const double tol = 1e-12;
if(fabs(D) > tol) return 0; // unique
if(fabs(E) > tol) return 1; // inconsistent
return 2; // infinitely many
}
```

Code - C

```
// Parametric solution (only valid if analyze_system returns 2)
void parametric_solution(double t, double *x, double *y, double *z){
if(x) *x = 10.0 * t + 10.0/9.0;
if(y) *y = -11.0 * t - 1.0/9.0;
if(z) *z = t;
}


// Row reduction for 3x4 augmented matrix
void row_reduce_3x4(const double A_in[3][4], double A_out[3][4]){
int i,j;
for(i=0;i<3;i++){
for(j=0;j<4;j++){
A_out[i][j] = A_in[i][j];
}
}
```

Code - C

```
// R2 <- R2 - 10 R1
for(j=0;j<4;j++){
A_out[1][j] -= 10.0 * A_out[0][j];
}
// R3 <- R3 - (qr) R1, qr = A_in[2][0]
double qr = A_in[2][0];
for(j=0;j<4;j++){
A_out[2][j] -= qr * A_out[0][j];
}
// s = (pr-qr)/90 ; pr = A_in[2][1]
double pr = A_in[2][1];
double s = (pr - qr) / 90.0;
for(j=0;j<4;j++){
A_out[2][j] -= s * A_out[1][j];
}
}
```

Code - Python(with shared C code)

The code to obtain the required plot is

```
# plot_hp_3d.py
import ctypes
from ctypes import c_double, POINTER, byref
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D # noqa: F401 (needed for 3D
    projection)

# load shared library (libhp.so should be in current working directory)
lib = ctypes.CDLL("./libhp.so")

# bind functions
lib.analyze_system.argtypes = (c_double, c_double, c_double,
                               POINTER(c_double), POINTER(
                                   c_double))
lib.analyze_system.restype = ctypes.c_int
```

Code - Python(with shared C code)

```
lib.parametric_solution.argtypes = (c_double, POINTER(c_double),
                                    POINTER(c_double), POINTER(
                                        c_double))
lib.parametric_solution.restype = None

# row_reduce signature: void row_reduce_3x4(const double A_in[3][4],
    double A_out[3][4]);
lib.row_reduce_3x4.argtypes = (POINTER(c_double), POINTER(c_double)
    )
lib.row_reduce_3x4.restype = None

# Python wrappers
def analyze(p, q, r):
    D = c_double(); E = c_double()
    code = lib.analyze_system(c_double(p), c_double(q), c_double(r),
                              byref(D), byref(E))
    return int(code), D.value, E.value
```

Code - Python(with shared C code)

```
def param_sol(t):
    x = c_double(); y = c_double(); z = c_double()
    lib.parametric_solution(c_double(t), byref(x), byref(y), byref(z))
    return x.value, y.value, z.value

def row_reduce(A):

    A = np.asarray(A, dtype=np.float64, order='C')
    if A.shape != (3,4):
        raise ValueError("A must be shape (3,4)")
    ArrayType = c_double * 12
    in_buf = ArrayType(*A.ravel().tolist())
    out_buf = ArrayType()
    lib.row_reduce_3x4(in_buf, out_buf)
    return np.frombuffer(out_buf, dtype=np.float64).reshape((3,4)).copy
        ()
```

Code - Python(with shared C code)

```
# ---- Main demo ----
if __name__ == "__main__":
    # change these to test other triples
    p, q, r = 100.0, 10.0, 1.0

    # Build augmented matrix [M | b]
    A = np.array([
        [1.0, 1.0, 1.0, 1.0],
        [10.0, 100.0, 1000.0, 0.0],
        [q*r, p*r, p*q, 0.0]
    ], dtype=np.float64)

    # Call row reduction and show result
    A_red = row_reduce(A)
    print("Reduced augmented matrix (after specified elimination steps):")
    np.set_printoptions(precision=6, suppress=True)
    print(A_red)
```

Code - Python(with shared C code)

```
# Analyze system
    code, D, E = analyze(p, q, r)
    print(f"analyze_system -> code={code}, D={D}, E={E}")
    # code: 0 = unique, 1 = no solution, 2 = infinitely many

    # If consistent (infinitely many), plot 3D solution curve
    if code == 2:
        ts = np.linspace(-1.0, 1.0, 401)
        xs = np.empty_like(ts); ys = np.empty_like(ts); zs = np.
            empty_like(ts)
        for i, t in enumerate(ts):
            xs[i], ys[i], zs[i] = param_sol(t)
```

Code - Python(with shared C code)

```
    fig = plt.figure(figsize=(7,7))
    ax = fig.add_subplot(111, projection='3d')
    ax.plot(xs, ys, zs, lw=2)
    ax.set_xlabel('x'); ax.set_ylabel('y'); ax.set_zlabel('z')
    ax.set_title(f'Solution curve (p={p}, q={q}, r={r})')
    plt.tight_layout()
    outname = "hp_3d_only.png"
    fig.savefig(outname, dpi=200)
    print("Saved 3D plot:", outname)
else:
    print("System not consistent — nothing to plot.")
```

Code - Python only

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D # noqa: F401 (needed for 3D
    projection)

def row_reduce_3x4(A_in):
    A = np.asarray(A_in, dtype=float, order='C').copy()
    if A.shape != (3,4):
        raise ValueError("A_in must be shape (3,4)")
    # R2 <- R2 - 10 R1
    A[1,:] = A[1,:] - 10.0 * A[0,:]
    # R3 <- R3 - (qr) R1 (use qr from original A_in)
    qr = float(A_in[2,0])
    A[2,:] = A[2,:] - qr * A[0,:]
    # s = (pr - qr) / 90 (pr from original)
```

Code - Python only

```
    pr = float(A_in[2,1])
    s = (pr - qr) / 90.0
    A[2,:] = A[2,:] - s * A[1,:]
    return A

def analyze_system(p, q, r, tol=1e-12):
    D = p*q - 11.0*p*r + 10.0*q*r
    E = (p*r - 10.0*q*r) / 9.0
    if abs(D) > tol:
        return 0, D, E
    if abs(E) > tol:
        return 1, D, E
    return 2, D, E
```

Code - Python only

```
def parametric_solution(t):
    x = 10.0 * t + 10.0/9.0
    y = -11.0 * t - 1.0/9.0
    z = t
    return x, y, z

if __name__ == "__main__":
    # Choose (p,q,r). For consistent test use (100,10,1)
    p, q, r = 100.0, 10.0, 1.0

    # Build augmented matrix [M | b]
    A = np.array([
        [1.0, 1.0, 1.0, 1.0],
        [10.0, 100.0, 1000.0, 0.0],
        [q*r, p*r, p*q, 0.0]
    ], dtype=float)
```

Code - Python only

```
print("Input augmented matrix [M | b]:")
np.set_printoptions(precision=6, suppress=True)
print(A)

# Row-reduce with the exact steps used in math writeup
A_red = row_reduce_3x4(A)
print("\nReduced augmented matrix after specified elimination steps:")
print(A_red)

# Analyze with D,E
code, D, E = analyze_system(p, q, r)
status = {0: "unique solution (D != 0)", 1: "inconsistent (no solution)
    ", 2: "infinitely many solutions"}
print(f"\nanalyze_system -> code={code}, D={D:.6g}, E={E:.6g} 
    => {status[code]}")
```

Code - Python only

```
# If consistent, plot only the 3D solution curve
if code == 2:
    ts = np.linspace(-1.0, 1.0, 401)
    xs = np.empty_like(ts); ys = np.empty_like(ts); zs = np.
        empty_like(ts)
    for i,t in enumerate(ts):
        xs[i], ys[i], zs[i] = parametric_solution(t)

    fig = plt.figure(figsize=(7,7))
    ax = fig.add_subplot(111, projection='3d')
    ax.plot(xs, ys, zs, lw=2)
    ax.set_xlabel('x'); ax.set_ylabel('y'); ax.set_zlabel('z')
    ax.set_title(f'Solution curve (p={p}, q={q}, r={r})')
    plt.tight_layout()
    outname = "hp_3d_pure_python.png"
    fig.savefig(outname, dpi=200)
    print("\nSaved 3D plot:", outname)
```

Code - Python only

```
    else:
        print("\nSystem not consistent — nothing to plot.")
```