5.13.68

EE25BTECH11043 - Nishid Khandagre

October 3 , 2025

Question

For what value of k do the following system of equations possess a non trivial solution over the set of rationals Q?

$$
\begin{aligned}
x + ky + 3z &= 0 \\
3x + ky - 2z &= 0 \\
2x + 3y - 4z &= 0
\end{aligned}
$$

For that value of k, find all the solutions of the system.

Theoretical Solution

The given system of equations can be written in augmented matrix form as:

$$\left(\begin{array}{ccc|c} 1 & k & 3 & 0 \\ 3 & k & -2 & 0 \\ 2 & 3 & -4 & 0 \end{array}\right) \tag{1}$$

Apply Gaussian elimination to find the row echelon form. First, perform row operations: $R_2 \to R_2 - 3R_1$ and $R_3 \to R_3 - 2R_1$.

$$\left(\begin{array}{ccc|c} 1 & k & 3 & 0 \\ 0 & k-3k & -2-9 & 0 \\ 0 & 3-2k & -4-6 & 0 \end{array}\right) \tag{2}$$

This simplifies to:

$$\left(\begin{array}{ccc|c} 1 & k & 3 & 0 \\ 0 & -2k & -11 & 0 \\ 0 & 3-2k & -10 & 0 \end{array}\right) \tag{3}$$

Theoretical Solution

For a non-trivial solution, the rank of the coefficient matrix must be less than 3. If $-2k = 0$, then $k = 0$. In this case, the matrix becomes:

$$\left(\begin{array}{ccc|c} 1 & 0 & 3 & 0 \\ 0 & 0 & -11 & 0 \\ 0 & 3 & -10 & 0 \end{array}\right) \tag{4}$$

This matrix has rank 3, which would lead to only the trivial solution. So $k \neq 0$.
If $k \neq 0$, we can proceed R_2: $R_2 \to -R_2$:

$$\left(\begin{array}{ccc|c} 1 & k & 3 & 0 \\ 0 & 2k & 11 & 0 \\ 0 & 3-2k & -10 & 0 \end{array}\right) \tag{5}$$

Theoretical Solution

$R_3 \to 2kR_3 - (3-2k)R_2$:

$$\left(\begin{array}{ccc|c} 1 & k & 3 & 0 \\ 0 & 2k & 11 & 0 \\ 0 & 0 & 2k-33 & 0 \end{array}\right) \tag{6}$$

For a non-trivial solution, the rank of matrix must be less than 3,therefore

$$2k - 33 = 0 \tag{7}$$

$$2k = 33 \tag{8}$$

$$k = \frac{33}{2} \tag{9}$$

Theoretical Solution

Now, find the solutions for $k = \frac{33}{2}$. The augmented matrix is:

$$\left(\begin{array}{ccc|c} 1 & 33/2 & 3 & 0 \\ 3 & 33/2 & -2 & 0 \\ 2 & 3 & -4 & 0 \end{array}\right) \tag{10}$$

Perform $R_2 \to R_2 - 3R_1$:

$$\left(\begin{array}{ccc|c} 1 & 33/2 & 3 & 0 \\ 0 & -33 & -11 & 0 \\ 2 & 3 & -4 & 0 \end{array}\right) \tag{11}$$

Perform $R_3 \to R_3 - 2R_1$:

$$\left(\begin{array}{ccc|c} 1 & 33/2 & 3 & 0 \\ 0 & -33 & -11 & 0 \\ 0 & -30 & -10 & 0 \end{array}\right) \tag{12}$$

Theoretical Solution

$R_2 \to R_2/(-11)$ and $R_3 \to R_3/(-10)$:

$$\left(\begin{array}{ccc|c} 1 & 33/2 & 3 & 0 \\ 0 & 3 & 1 & 0 \\ 0 & 3 & 1 & 0 \end{array}\right) \tag{13}$$

Perform $R_3 \to R_3 - R_2$:

$$\left(\begin{array}{ccc|c} 1 & 33/2 & 3 & 0 \\ 0 & 3 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right) \tag{14}$$

The rank of the coefficient matrix is 2, which is less than 3, so there are non-trivial solutions.

From the second row: $3y + z = 0 \Rightarrow z = -3y$.

From the first row: $x + \frac{33}{2}y + 3z = 0$. Substitute $z = -3y$: $x = -\frac{15}{2}y$.

Theoretical Solution

Let $y = 2t$ Then $x = -\frac{15}{2}(2t) = -15t$. And $z = -3(2t) = -6t$

The solutions are of the form: $\begin{pmatrix} x \\ y \\ z \end{pmatrix} = t \begin{pmatrix} -15 \\ 2 \\ -6 \end{pmatrix}$ for any $t \in \mathbb{Q}$.

C Code

```c
#include <stdio.h>

// Function to calculate the determinant of the 3x3 matrix
// | 1 k 3 |
// | 3 k -2 |
// | 2 3 -4 |
double calculateDeterminant(double k_val) {
    double det = 0.0;
    det = (1.0 * (k_val * -4.0 - (-2.0 * 3.0))) -
          (k_val * (3.0 * -4.0 - (-2.0 * 2.0))) +
          (3.0 * (3.0 * 3.0 - k_val * 2.0));
    return det;
}
```

C Code

```c
// Function to solve the system for a given k when det = 0
// It will express x, y in terms of z
// This function assumes a non-trivial solution exists (det = 0)
// For k = 33/2
// So, the solutions are of the form (5/2 * z, -1/3 * z, z) for
    any rational z.
void solveSystem(double k_val, double* x_coeff_z, double*
    y_coeff_z) {
    if (k_val == 33.0 / 2.0) { // Check if k is the correct value
        *x_coeff_z = 5.0 / 2.0;
        *y_coeff_z = -1.0 / 3.0;
    } else {
        // Handle cases where k is not the value that makes det
            =0,
        *x_coeff_z = 0.0;
        *y_coeff_z = 0.0;
    }
}
```

Python Code through shared output

```python
import ctypes
import numpy as np

# Load the shared library
lib_code = ctypes.CDLL(./code13.so)

# --- Part 1: Find k for non-trivial solution ---

# Define argument types and return type for calculateDeterminant
lib_code.calculateDeterminant.argtypes = [ctypes.c_double]
lib_code.calculateDeterminant.restype = ctypes.c_double

# Find k by iterating and checking the determinant
test_k_values = [16.0, 16.4, 16.5, 16.6, 17.0]
k_for_nontrivial = None
```

Python Code through shared output

```
print(Testing determinant for various k values:)
for k_val in test_k_values:
    det = lib_code.calculateDeterminant(k_val)
    print(f For k = {k_val:.2f}, Determinant = {det:.2f})
    if abs(det) < 1e-9: # A small epsilon for floating point
         comparison
        k_for_nontrivial = k_val
        break # Found k

# If not found directly, we know k = 33/2 = 16.5 analytically.
if k_for_nontrivial is None:
    k_for_nontrivial = 33.0 / 2.0
    print(f\nAnalytically determined k for non-trivial solution:
         {k_for_nontrivial:.2f})

print(f\nValue of k for which the system possesses a non-trivial
    solution: k = {k_for_nontrivial:.2f})
```

Python Code through shared output

```python
# --- Part 2: Find all solutions for that value of k ---
# Define argument types and return type for solveSystem
lib_code.solveSystem.argtypes = [
    ctypes.c_double,
    ctypes.POINTER(ctypes.c_double), # x_coeff_z
    ctypes.POINTER(ctypes.c_double) # y_coeff_z
]
lib_code.solveSystem.restype = None

# Create ctypes doubles to hold the coefficients
x_coeff_z_result = ctypes.c_double()
y_coeff_z_result = ctypes.c_double()
# Call the C function to get the coefficients
lib_code.solveSystem(
    k_for_nontrivial,
    ctypes.byref(x_coeff_z_result),
    ctypes.byref(y_coeff_z_result)
)
```

Python Code through shared output

```
x_coeff = x_coeff_z_result.value
y_coeff = y_coeff_z_result.value

print(f\nFor k = {k_for_nontrivial:.2f}, the solutions are of the form:)
print(fx = {x_coeff:.3f} * z)
print(fy = {y_coeff:.3f} * z)
print(fz = z (where z can be any rational number))
print(\nThis means the solution set is a subspace spanned by the vector:)
print(fSolution vector: ({x_coeff:.3f}, {y_coeff:.3f}, 1))
# Example non-trivial solution (let z = 6, to get integer values for x and y based on the derived coeffs)
if k_for_nontrivial == 33.0 / 2.0:
    print(\nExample non-trivial solution (let z = 6 to get integers):)
    example_z = 6
    example_x = x_coeff * example_z
    example_y = y_coeff * example_z
```

Python Code through shared output

```
print(f Solution: ({example_x:.0f}, {example_y:.0f}, {example_z:.0f}))
print(\nLet's verify this solution with the original equations (with k=33/2):)
print(f 1({example_x}) + {k_for_nontrivial}({example_y}) + 3({example_z}) = {1*example_x + k_for_nontrivial*example_y + 3*example_z})
print(f 3({example_x}) + {k_for_nontrivial}({example_y}) - 2({example_z}) = {3*example_x + k_for_nontrivial*example_y - 2*example_z})
print(f 2({example_x}) + 3({example_y}) - 4({example_z}) = {2*example_x + 3*example_y - 4*example_z})
```

Python Code: Direct

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

def calculate_determinant(k_val):

    Calculates the determinant of the coefficient matrix for a
        given k.
    Matrix:
    | 1 k 3 |
    | 3 k -2 |
    | 2 3 -4 |

    det = (1 * (k_val * -4 - (-2 * 3))) - \
          (k_val * (3 * -4 - (-2 * 2))) + \
          (3 * (3 * 3 - k_val * 2))
    return det
```

Python Code: Direct

```python
def solve_system_coefficients(k_val):

    Solves the system for x and y in terms of z when det(A) = 0.
    This uses the analytical derivation. For a general solver,
    one would implement Gaussian elimination or a similar method on the matrix.
    Assumes k_val is such that a non-trivial solution exists.

    if k_val == 33.0 / 2.0: # Check if k is the correct value
        x_coeff_z = 5.0 / 2.0
        y_coeff_z = -1.0 / 3.0
        return x_coeff_z, y_coeff_z
    else:
        # If k is not the value for non-trivial solutions,
        # For homogeneous system, if det != 0, only trivial solution.
        return 0.0, 0.0
```

Python Code: Direct

```
# --- Part 1: Find k for non-trivial solution ---
# Analytically determined k
k_for_nontrivial = 33.0 / 2.0

# Verify the determinant for this k
det_at_k = calculate_determinant(k_for_nontrivial)
print(fFor k = {k_for_nontrivial:.2f}, the determinant is: {det_at_k:.6f} (should be close to zero))
if abs(det_at_k) < 1e-9:
    print(Determinant is effectively zero, confirming k for non-trivial solution.)
else:
    print(Warning: Determinant is not zero for this k. Check calculations.)
    exit()

print(f\nValue of k for which the system possesses a non-trivial solution: k = {k_for_nontrivial:.2f})
```

Python Code: Direct

```
# --- Part 2: Find all solutions for that value of k ---
x_coeff, y_coeff = solve_system_coefficients(k_for_nontrivial)

print(f\nFor k = {k_for_nontrivial:.2f}, the solutions are of the form:)
print(fx = {x_coeff:.6f} * z)
print(fy = {y_coeff:.6f} * z)
print(fz = z (where z can be any rational number))
print(\nThis means the solution set is a subspace spanned by the vector:)
print(fSolution basis vector: ({x_coeff:.6f}, {y_coeff:.6f}, 1))

# Example non-trivial solution (let z = 6, to get integer values for x and y based on the derived coeffs)
print(\nExample non-trivial solution (let z = 6):)
example_z = 6
example_x = x_coeff * example_z
example_y = y_coeff * example_z
```

Python Code: Direct

```
print(f If z = {example_z}, then x = {example_x:.0f}, y = {example_y:.0f})
print(f Solution: ({example_x:.0f}, {example_y:.0f}, {example_z:.0f}))
print(\nLet's verify this solution with the original equations (with k=33/2):)
k_val_for_verify = 33.0 / 2.0
eq1_result = (1 * example_x) + (k_val_for_verify * example_y) + (3 * example_z)
eq2_result = (3 * example_x) + (k_val_for_verify * example_y) - (2 * example_z)
eq3_result = (2 * example_x) + (3 * example_y) - (4 * example_z)

print(f x + ky + 3z = {eq1_result:.6f} (should be 0))
print(f 3x + ky - 2z = {eq2_result:.6f} (should be 0))
print(f 2x + 3y - 4z = {eq3_result:.6f} (should be 0))
```