5.3.36

EE25BTECH11002 - Achat Parth Kalpesh

October 2,2025

Question

Solve the system of equations

$$\frac{bx}{a} - \frac{ay}{b} + a + b = 0 \tag{1}$$

$$bx - ay + 2ab = 0 \tag{2}$$

The above equation can be written as

$$\mathbf{n_1}^\top \mathbf{x} = c_1 \tag{3}$$

$$\mathbf{n_2}^\top \mathbf{x} = c_2 \tag{4}$$

$$\begin{pmatrix} \mathbf{n_1} \\ \mathbf{n_2} \end{pmatrix}^\top \mathbf{x} = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} \tag{5}$$

$$\mathbf{A} = \begin{pmatrix} \mathbf{n_1} \\ \mathbf{n_2} \end{pmatrix}^\top \quad \mathbf{b} = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} \tag{6}$$

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{7}$$

$$\begin{pmatrix} \frac{b}{a} & -\frac{a}{b} \\ b & -a \end{pmatrix} \mathbf{x} = \begin{pmatrix} -a - b \\ -2ab \end{pmatrix} \tag{8}$$

$$\mathbf{A}^\top \mathbf{A} \neq \mathbf{I} \tag{9}$$

Solution

Performing row operations:

$$\left(\begin{array}{cc|c} \frac{b}{a} & -\frac{a}{b} & -a-b \\ b & -a & -2ab \end{array}\right) \xleftrightarrow{R_1 \leftarrow R_1 - \frac{R_2}{b}} \left(\begin{array}{cc|c} \frac{b-a}{a} & 0 & a-b \\ b & -a & -2ab \end{array}\right) \tag{10}$$

$$\left(\begin{array}{cc|c} \frac{b-a}{a} & 0 & a-b \\ b & -a & -2ab \end{array}\right) \xleftrightarrow{R_2 \leftarrow -\frac{ab}{b-a} R_1 + R_2} \left(\begin{array}{cc|c} \frac{b-a}{a} & 0 & a-b \\ 0 & -a & -ab \end{array}\right) \tag{11}$$

$$\left(\begin{array}{cc|c} \frac{b-a}{a} & 0 & a-b \\ 0 & -a & -ab \end{array}\right) \xleftrightarrow{R_2 \leftarrow -\frac{R_2}{a}} \left(\begin{array}{cc|c} \frac{b-a}{a} & 0 & a-b \\ 0 & 1 & b \end{array}\right) \tag{12}$$

$$\left(\begin{array}{cc|c} \frac{b-a}{a} & 0 & a-b \\ 0 & 1 & -b \end{array}\right) \xleftrightarrow{R_1 \leftarrow \frac{a}{b-a} R_1} \left(\begin{array}{cc|c} 1 & 0 & -a \\ 0 & 1 & b \end{array}\right) \tag{13}$$

Solution

Thus,

$$\mathbf{x} = \begin{pmatrix} -a \\ b \end{pmatrix} \tag{14}$$

C Code

```c
#include <stdio.h>
void solve_system(double A[2][2], double b[2], double* x_sol,
    double* y_sol) { // Solve the 2x2 system using Cramer's rule;
     det(A)
    double determinant = A[0][0] * A[1][1] - A[0][1] * A[1][0];
    // Check if a unique solution exists.
    if (determinant != 0) {// det(Ax)
        double determinant_x = b[0] * A[1][1] - A[0][1] * b[1];
        // det(Ay)
        double determinant_y = A[0][0] * b[1] - b[0] * A[1][0];
        *x_sol = determinant_x / determinant;
        *y_sol = determinant_y / determinant;
    } else {
        // No unique solution, set results to 0 or an error
            indicator.
        *x_sol = 0;
        *y_sol = 0;
    }
}
```

Python Code

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

lib_path = './solver.so'
solver_lib = ctypes.CDLL(lib_path)
# Define the argument types and return type for the C function
# The function signature is: void solve_system(double A[2][2], double b[2], double* x, double* y)
solve_func = solver_lib.solve_system
solve_func.argtypes = [
    np.ctypeslib.ndpointer(dtype=np.float64, ndim=2, shape=(2,2)),
    np.ctypeslib.ndpointer(dtype=np.float64, ndim=1, shape=(2,)),
    ctypes.POINTER(ctypes.c_double),
    ctypes.POINTER(ctypes.c_double)
]
solve_func.restype = None
```

Python Code

```
# Define the coefficient matrix A and the constant vector b
# 2x - y = 10
# 3x + y = 5
A = np.array([[1, -1],
              [1, 1]], dtype=np.float64)
b = np.array([0, -2], dtype=np.float64)

# Create C-compatible variables to store the results
x_intersect_c = ctypes.c_double()
y_intersect_c = ctypes.c_double()

# Call the C function
solve_func(A, b, ctypes.byref(x_intersect_c), ctypes.byref(
    y_intersect_c))

# Get the Python values from the C types
x_intersect = x_intersect_c.value
y_intersect = y_intersect_c.value
```

Python Code

```
# --- 2. Plot the graph ---

# Generate a range of x values for plotting the lines
x_vals = np.linspace(x_intersect - 10, x_intersect + 10, 400)

# Calculate y values for each equation
# Eq1: 2x - y = 10 => y = 2x - 10
y1_vals = x_vals
# Eq2: 3x + y = 5 => y = 5 - 3x
y2_vals = - x_vals -2

# Create the plot
plt.figure(figsize=(10, 10))
plt.plot(x_vals, y1_vals, color='blue')
plt.plot(x_vals, y2_vals, color='green')
```

Python Code

```python
# Mark and label the intersection point
plt.plot(x_intersect, y_intersect, 'ro', markersize=8)
plt.text(x_intersect + 1.0, y_intersect, f'({x_intersect:.2f}, {
    y_intersect:.2f})', fontsize=12, va='center')

# --- 3. Add non-overlapping labels directly to the lines ---

# Position the labels on the lines at specific points for clarity
plt.text(4, 2.5, 'x-y=0', color='blue', va='center', ha='left',
    fontsize=11)
plt.text(-6, 2.5, 'x+y=-2', color='green', va='center', ha='
    center', fontsize=11)
```

Python Code

```
# --- 4. Style the plot ---

plt.title('Solution of the System of Linear Equations for a=1,b
    =-1')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.axhline(0, color='black', linewidth=0.5)
plt.axvline(0, color='black', linewidth=0.5)
plt.grid(True, which='both', linestyle='--', linewidth=0.5)
# Set axis limits to better match the example image
plt.xlim(-15, 15)
plt.ylim(-20, 20)
plt.axis('equal') # Ensure aspect ratio is equal
plt.show()
```

Plot

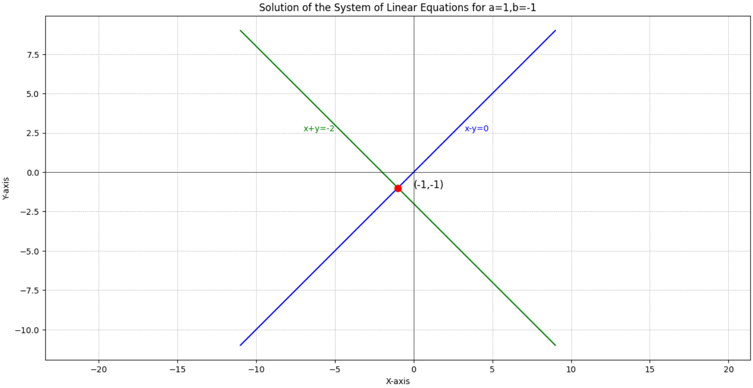

Figure: Visualization of the solution