5.2.45

EE25BTECH11043 - Nishid Khandagre

September 30, 2025

Question

Solve the system of linear equations:

$$\begin{aligned} x + 2y &= 2 \\ 2x + 3y &= 3 \end{aligned}$$

Theoretical Solution

Given:

$$x + 2y = 2 \tag{1}$$
$$2x + 3y = 3 \tag{2}$$

$$\begin{pmatrix} 1 & 2 \\ 2 & 3 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 2 \\ 3 \end{pmatrix} \tag{3}$$

Theoretical Solution

Write the augmented matrix:

$$\left(\begin{array}{cc|c} 1 & 2 & 2 \\ 2 & 3 & 3 \end{array}\right) \tag{4}$$

Eliminate the first column using the operation $R_2 \rightarrow R_2 - 2R_1$:

$$\left(\begin{array}{cc|c} 1 & 2 & 2 \\ 0 & -1 & -1 \end{array}\right) \tag{5}$$

Theoretical Solution

Next, apply the operation $R_2 \to -R_2$:

$$\left(\begin{array}{cc|c} 1 & 2 & 2 \\ 0 & 1 & 1 \end{array}\right) \tag{6}$$

Then, apply $R_1 \to R_1 - 2R_2$

$$\left(\begin{array}{cc|c} 1 & 0 & 0 \\ 0 & 1 & 1 \end{array}\right) \tag{7}$$

Theoretical Solution

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{8}$$

Therefore, the solution to the system of linear equations is:

$$x = 0 \tag{9}$$
$$y = 1 \tag{10}$$

C Code

```c
#include <stdio.h>

// Function to solve a 2x2 system of linear equations
// a1, b1, c1: coefficients for the first equation (a1x + b1y = c1)
// a2, b2, c2: coefficients for the second equation (a2x + b2y = c2)
// x_sol, y_sol: pointers to store the solutions for x and y
void solveLinearSystem(double a1, double b1, double c1,
                       double a2, double b2, double c2,
                       double *x_sol, double *y_sol) {
    double determinant = a1 * b2 - a2 * b1;

    *x_sol = (c1 * b2 - c2 * b1) / determinant;
    *y_sol = (a1 * c2 - a2 * c1) / determinant;
}
```

Python Code using C shared library

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

# Load the shared library
lib_solver = ctypes.CDLL(./code10.so)

# Define the argument types and return type for the C function
lib_solver.solveLinearSystem.argtypes = [
    ctypes.c_double, # a1
    ctypes.c_double, # b1
    ctypes.c_double, # c1
    ctypes.c_double, # a2
    ctypes.c_double, # b2
    ctypes.c_double, # c2
    ctypes.POINTER(ctypes.c_double), # x_sol
    ctypes.POINTER(ctypes.c_double) # y_sol
]
```

Python Code using C shared library

```
lib_solver.solveLinearSystem.restype = None
# Given system of linear equations:
# x + 2y = 2
# 2x + 3y = 3
a1_given, b1_given, c1_given = 1.0, 2.0, 2.0
a2_given, b2_given, c2_given = 2.0, 3.0, 3.0

# Create ctypes doubles to hold the results
x_solution = ctypes.c_double()
y_solution = ctypes.c_double()

# Call the C function to solve the system
lib_solver.solveLinearSystem(
    a1_given, b1_given, c1_given,
    a2_given, b2_given, c2_given,
    ctypes.byref(x_solution),
    ctypes.byref(y_solution)
)
```

Python Code using C shared library

```
x_found = x_solution.value
y_found = y_solution.value

print(fThe solution to the system is: x = {x_found:.2f}, y = {y_found:.2f})
# Optional: Plotting the lines to visualize the intersection
# Define a range for x values
x_vals = np.linspace(-5, 5, 400)

# Calculate y values for the first equation: y = (2 - x) / 2
y1_vals = (c1_given - a1_given * x_vals) / b1_given

# Calculate y values for the second equation: y = (3 - 2x) / 3
y2_vals = (c2_given - a2_given * x_vals) / b2_given
```

Python Code using C shared library

```
plt.figure(figsize=(8, 6))
plt.plot(x_vals, y1_vals, label=f'{a1_given}x + {b1_given}y = {
    c1_given}')
plt.plot(x_vals, y2_vals, label=f'{a2_given}x + {b2_given}y = {
    c2_given}')

# Plot the intersection point
plt.scatter(x_found, y_found, color='red', s=100, zorder=5,
           label=f'Solution ({x_found:.2f}, {y_found:.2f})')
plt.annotate(f'({x_found:.2f},{y_found:.2f})', (x_found, y_found)
    ,
            textcoords=offset points, xytext=(5,5), ha='left',
                color='red')
```

Python Code using C shared library

```
plt.xlabel('x')
plt.ylabel('y')
plt.title('Solution of Linear Equations (using C shared library)'
    )
plt.grid(True)
plt.axhline(0, color='black', linewidth=0.5)
plt.axvline(0, color='black', linewidth=0.5)
plt.legend()
plt.savefig(fig1.png)
plt.show()
```

Pure Python Code

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

# Function to solve a 2x2 system of linear equations
def solve_2x2_system(a1, b1, c1, a2, b2, c2):
    det = a1 * b2 - a2 * b1
    if det == 0:
        return None, None # No unique solution
    x = (c1 * b2 - c2 * b1) / det
    y = (a1 * c2 - a2 * c1) / det
    return x, y

# Given equations
# x + 2y = 2
# 2x + 3y = 3
a1, b1, c1 = 1, 2, 2
a2, b2, c2 = 2, 3, 3
```

Pure Python Code

```python
# Solve the system
x_solution, y_solution = solve_2x2_system(a1, b1, c1, a2, b2, c2)

if x_solution is not None and y_solution is not None:
    print(fSolution: x = {x_solution}, y = {y_solution})

    # Generate points for plotting the lines
    x_vals = np.linspace(-5, 5, 400)

    # First equation: x + 2y = 2 => y = (2 - x) / 2
    y1_vals = (2 - x_vals) / 2

    # Second equation: 2x + 3y = 3 => y = (3 - 2x) / 3
    y2_vals = (3 - 2 * x_vals) / 3
```

Pure Python Code Cont.

```
    # Plotting
    plt.figure(figsize=(8, 6))
    plt.plot(x_vals, y1_vals, label='x + 2y = 2')
    plt.plot(x_vals, y2_vals, label='2x + 3y = 3')

    # Plot the intersection point
    plt.plot(x_solution, y_solution, 'ro', markersize=8,
             label=f'Solution: ({x_solution:.2f}, {y_solution:.2f})')
    plt.annotate(f'({x_solution:.2f}, {y_solution:.2f})', (x_solution, y_solution),
                 textcoords=offset points, xytext=(10,10), ha='center')

    plt.xlabel('x')
    plt.ylabel('y')
    plt.title('Solution of a System of Linear Equations (Pure Python)')
```

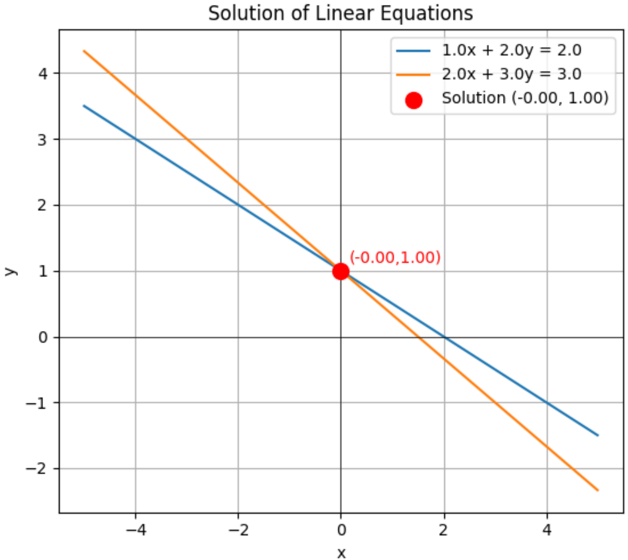
Pure Python Code

```
    plt.axhline(0, color='gray', linestyle='--', linewidth=0.7)
    plt.axvline(0, color='gray', linestyle='--', linewidth=0.7)
    plt.grid(True)
    plt.legend()
    plt.axis('equal') # Ensures equal scaling for x and y axes
    plt.xlim(-5, 5)
    plt.ylim(-5, 5)
    plt.savefig(fig2.png)
    plt.show()

    print(Figure saved as fig2.png)

else:
    print(The system has no unique solution (lines are parallel
        or coincident))
```

Plot by Python using shared output from C



Plot by Python only