4.13.34

Jnanesh Sathisha karmar - EE25BTECH11029

September 30,2025

Solve the following system of linear equations

$x + 2y - 4 = 0$

$2x + 4y - 12 = 0$

Equation

Solution Given details

$$x + 2y - 4 = 0 \tag{1}$$

$$2x + 4y - 12 = 0 \tag{2}$$

$$\begin{pmatrix} 1 & 2 \\ 2 & 4 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 4 \\ 12 \end{pmatrix} \tag{3}$$

$$\mathbf{Ax} = \mathbf{B} \tag{4}$$

Theoretical Solution

To determine if a unique solution exists, we calculate the determinant of the coefficient matrix

$$\det(\mathbf{A}) = 4 - 4 = 0 \tag{5}$$

Since the determinant is zero, the matrix $\mathbf{A}$ is singular (it has no inverse). This means that the system does not have a unique solution. It will either have no solution or infinitely many solutions.
To find out which case it is, we use an augmented matrix $\left(\mathbf{A} \mid \mathbf{B}\right)$and apply row reduction.

$$\left(\begin{array}{cc|c} 1 & 2 & 4 \\ 2 & 4 & 12 \end{array}\right) \xrightarrow{R_2 \to R_2 - 2R_1} \left(\begin{array}{cc|c} 1 & 2 & 4 \\ 0 & 0 & 4 \end{array}\right)$$

Theoretical Solution

Since the second row of the reduced matrix corresponds to the equation $0x + 0y = 4$, which is a contradiction, the system is inconsistent and has no solution.

C Code (1) - Function to store the points

```c
#include <stdio.h>
double get_y_for_line1(double x) {
    return (4.0 - x) / 2.0;
}

double get_y_for_line2(double x) {
    return (12.0 - 2.0 * x) / 4.0;
}
```

Python Code - Using Shared Object

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

lib_path = './line_plotter.so'


line_lib = ctypes.CDLL(lib_path)


line_lib.get_y_for_line1.argtypes = [ctypes.c_double]
line_lib.get_y_for_line1.restype = ctypes.c_double

line_lib.get_y_for_line2.argtypes = [ctypes.c_double]
line_lib.get_y_for_line2.restype = ctypes.c_double
```

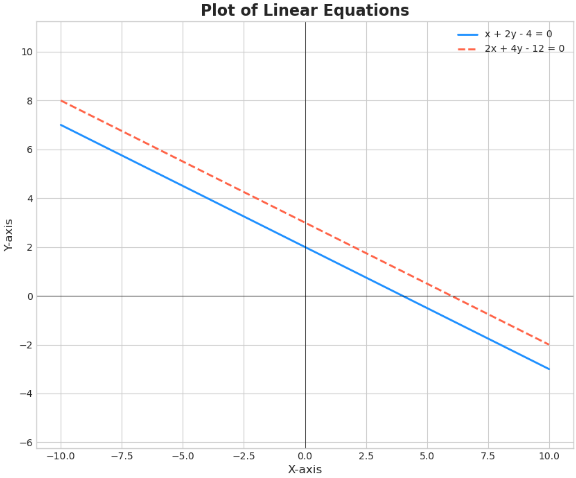
Python Code - Using Shared Object

```
x_values = np.linspace(-10, 10, 100)

y1_values = [line_lib.get_y_for_line1(x) for x in x_values]
y2_values = [line_lib.get_y_for_line2(x) for x in x_values]

plt.style.use('seaborn-v0_8-whitegrid')
plt.figure(figsize=(10, 8))

plt.plot(x_values, y1_values, label='x + 2y - 4 = 0', color='
    dodgerblue', linewidth=2)
plt.plot(x_values, y2_values, label='2x + 4y - 12 = 0', color='
    tomato', linewidth=2, linestyle='--')
```

Python Code - Using Shared Object

```
plt.title('Plot of Linear Equations', fontsize=16, fontweight='
    bold')
plt.xlabel('X-axis', fontsize=12)
plt.ylabel('Y-axis', fontsize=12)
plt.legend(fontsize=10)

plt.grid(True)
plt.axhline(0, color='black',linewidth=0.5)
plt.axvline(0, color='black',linewidth=0.5)
plt.axis('equal')
plt.savefig("./figs/lines.png")
subprocess.run(shlex.split('termux-open ../figs/lines.png'))
plt.show()
```

Plot-Using Both C and Python



Python Code

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

def get_y_for_line1(x):
    # Equation 1: x + 2y - 4 = 0 => y = (4 - x) / 2
    return (4 - x) / 2

def get_y_for_line2(x):
    # Equation 2: 2x + 4y - 12 = 0 => y = (12 - 2x) / 4
    return (12 - 2 * x) / 4

x_values = np.linspace(-10, 10, 100)

y1_values = get_y_for_line1(x_values)
y2_values = get_y_for_line2(x_values)
```

Python Code

```
plt.style.use('seaborn-v0_8-whitegrid')
plt.figure(figsize=(10, 8))
plt.plot(x_values, y1_values, label='x + 2y - 4 = 0', color='dodgerblue', linewidth=2)
plt.plot(x_values, y2_values, label='2x + 4y - 12 = 0', color='tomato', linewidth=2, linestyle='--')
plt.title('Plot of Linear Equations', fontsize=16, fontweight='bold')
plt.xlabel('X-axis', fontsize=12)
plt.ylabel('Y-axis', fontsize=12)
plt.legend(fontsize=10)
plt.savefig("./figs/lines2.png")
plt.grid(True)
plt.axhline(0, color='black',linewidth=0.5)
plt.axvline(0, color='black',linewidth=0.5)
plt.axis('equal')

plt.show()
```

Plot-Using only Python