9.5.6

EE25BTECH11043 - Nishid Khandagre

October 7, 2025

Question

Find the sum and product of the roots of the quadratic equation $2x^2 - 9x + 4 = 0$.

Theoretical Solution

Given quadratic equation:

$$y = 2x^2 - 9x + 4 \tag{1}$$

Representing this equation as a conic section

$$\mathbf{x}^\top \mathbf{V}\mathbf{x} + 2\mathbf{u}^\top \mathbf{x} + f = 0 \ , \ \mathbf{V} = \begin{pmatrix} 2 & 0 \\ 0 & 0 \end{pmatrix} \ , \ \mathbf{u} = \begin{pmatrix} -9/2 \\ -1/2 \end{pmatrix} \ , \ f = 4 \tag{2}$$

We need to find intersection points with $y = 0$, that is, the X-axis.

$$\mathbf{x} = \mathbf{h} + k\mathbf{m} \ , \ \mathbf{h} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \ , \ \mathbf{m} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \tag{3}$$

Substituting $\mathbf{x} = k\mathbf{m}$

Theoretical Solution

$$k^2\mathbf{m}^\top\mathbf{V}\mathbf{m} + 2k\mathbf{u}^\top\mathbf{m} + f = 0 \tag{4}$$

$$\implies k = \frac{1}{2\mathbf{m}^\top\mathbf{V}\mathbf{m}}\left[-2\mathbf{u}^\top\mathbf{m} \pm \sqrt{\left(2\mathbf{u}^\top\mathbf{m}\right)^2 - 4f\mathbf{m}^\top\mathbf{V}\mathbf{m}}\right] \tag{5}$$

$$\implies k = \frac{1}{\mathbf{m}^\top\mathbf{V}\mathbf{m}}\left[-\mathbf{u}^\top\mathbf{m} \pm \sqrt{\left(\mathbf{u}^\top\mathbf{m}\right)^2 - f\mathbf{m}^\top\mathbf{V}\mathbf{m}}\right] \tag{6}$$

Now,

$$\mathbf{u}^\top\mathbf{m} = \begin{pmatrix} -9/2 & -1/2 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = -9/2 \tag{7}$$

$$\mathbf{m}^\top\mathbf{V}\mathbf{m} = \begin{pmatrix} 1 & 0 \end{pmatrix}\begin{pmatrix} 2 & 0 \\ 0 & 0 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = 2 \tag{8}$$

Theoretical Solution

$$k = \frac{1}{2}\left[-(-9/2) \ \pm \ \sqrt{(-9/2)^2 - 4 \cdot 2}\right] \tag{9}$$

$$k = \frac{1}{2}\left[\frac{9}{2} \ \pm \ \frac{7}{2}\right] \tag{10}$$

This gives us two values for k:

$$\implies k_1 = 4 \tag{11}$$

$$\implies k_2 = \frac{1}{2} \tag{12}$$

Substituting k into $\mathbf{x}$, we get the roots:

$$\mathbf{x} = \begin{pmatrix} 4 \\ 0 \end{pmatrix} \text{ OR } \mathbf{x} = \begin{pmatrix} 1/2 \\ 0 \end{pmatrix} \tag{13}$$

Theoretical Solution

This implies that the roots of $2x^2 - 9x + 4 = 0$ are 4 and $\frac{1}{2}$. Now, calculate the sum and product of these roots:
Sum of the roots:

$$\text{Sum} = 4 + \frac{1}{2} = \frac{9}{2} \tag{14}$$

Product of the roots:

$$\text{Product} = 4 \times \frac{1}{2} = 2 \tag{15}$$

C Code

```
#include <stdio.h>

// Function to find the sum and product of the roots of a
    quadratic equation
// For a quadratic equation ax^2 + bx + c = 0
// Sum of roots = -b/a
// Product of roots = c/a
void calculateRootsInfo(double a, double b, double c, double *
    sum_of_roots, double *product_of_roots) {
    *sum_of_roots = -b / a;
    *product_of_roots = c / a;
}
```

Python Code (using C shared library)

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

# Load the shared library
lib_quadratic = ctypes.CDLL(./code16.so)

# Define the argument types and return type for the C function
lib_quadratic.calculateRootsInfo.argtypes = [
    ctypes.c_double, # a
    ctypes.c_double, # b
    ctypes.c_double, # c
    ctypes.POINTER(ctypes.c_double), # sum_of_roots
    ctypes.POINTER(ctypes.c_double) # product_of_roots
]
lib_quadratic.calculateRootsInfo.restype = None
```

Python Code (using C shared library)

```python
# Given quadratic equation: 2x^2 - 9x + 4 = 0
a_given = 2.0
b_given = -9.0
c_given = 4.0

# Create ctypes doubles to hold the results
sum_result = ctypes.c_double()
product_result = ctypes.c_double()
# Call the C function
lib_quadratic.calculateRootsInfo(
    a_given, b_given, c_given,
    ctypes.byref(sum_result),
    ctypes.byref(product_result)
)

sum_of_roots = sum_result.value
product_of_roots = product_result.value
```

Python Code (using C shared library)

```
print(fFor the quadratic equation {a_given}x^2 + {b_given}x + {
    c_given} = 0:)
print(fThe sum of the roots is: {sum_of_roots:.2f})
print(fThe product of the roots is: {product_of_roots:.2f})

# --- Part 2: Plotting the parabola and its roots ---
# Calculate the discriminant
delta = b_given**2 - 4 * a_given * c_given

# Find the roots (if real)
roots = []
if delta >= 0:
    root1 = (-b_given + np.sqrt(delta)) / (2 * a_given)
    root2 = (-b_given - np.sqrt(delta)) / (2 * a_given)
    roots.append(root1)
    if root1 != root2: # Add distinct root2 if it exists
        roots.append(root2)
    roots.sort() # Sort for consistent labeling
```

Python Code (using C shared library)

```
# Determine plotting range based on roots or a default
if roots:
    min_root = min(roots)
    max_root = max(roots)
    # Expand the range a bit around the roots
    plot_min_x = min_root - (abs(max_root - min_root) * 0.5 + 1)
    plot_max_x = max_root + (abs(max_root - min_root) * 0.5 + 1)
    if plot_min_x == plot_max_x: # Case for a single root
        plot_min_x -= 5
        plot_max_x += 5
else: # If no real roots, use a reasonable default range
    plot_min_x = -5
    plot_max_x = 5

x_vals = np.linspace(plot_min_x, plot_max_x, 400)
y_vals = a_given * x_vals**2 + b_given * x_vals + c_given

plt.figure(figsize=(10, 6))
```

Python Code (using C shared library)

```
# Plot the parabola
plt.plot(x_vals, y_vals, label=f'${a_given}x^2 {b_given:+}x {c_given:+} = 0$', color='blue')

# Mark the roots
for i, root in enumerate(roots):
    plt.scatter(root, 0, color='red', s=100, zorder=5, label=f'Root {i+1}: {root:.2f}')
    plt.annotate(f'({root:.2f}, 0)', (root, 0), textcoords=offset points, xytext=(5,5), ha='left', color='red')

# Add x and y axes for reference
plt.axhline(0, color='black', linewidth=0.8, linestyle='--')
plt.axvline(0, color='black', linewidth=0.8, linestyle='--')
```

Python Code (using C shared library)

```
plt.xlabel('x')
plt.ylabel('y')
plt.title('Quadratic Parabola and its Real Roots')
plt.grid(True, linestyle=':', alpha=0.7)
plt.legend()
plt.ylim(min(y_vals)-abs(min(y_vals)*0.1)-1, max(y_vals)+abs(max(
    y_vals)*0.1)+1) # Adjust y-limits dynamically
plt.xlim(plot_min_x, plot_max_x) # Ensure x-limits are set
plt.show()
```

Python Code (Direct)

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

def solve_quadratic_and_plot(a, b, c):

    Calculates the sum and product of roots for a quadratic
        equation (ax^2 + bx + c = 0),
    and generates a plot of the parabola marking its real roots.

    # --- Part 1: Calculate sum and product of roots ---
    # Sum of roots = -b/a
    sum_of_roots = -b / a
    # Product of roots = c/a
    product_of_roots = c / a

    print(fFor the quadratic equation {a}x^2 + {b}x + {c} = 0:)
    print(fThe sum of the roots is: {sum_of_roots:.2f})
    print(fThe product of the roots is: {product_of_roots:.2f})
```

Python Code (Direct)

```
# --- Part 2: Plotting the parabola and its roots ---
# Calculate the discriminant
delta = b**2 - 4 * a * c

# Find the roots (if real)
roots = []
if delta >= 0:
    root1 = (-b + np.sqrt(delta)) / (2 * a)
    root2 = (-b - np.sqrt(delta)) / (2 * a)
    roots.append(root1)
    if root1 != root2: # Add distinct root2 if it exists
        roots.append(root2)
    roots.sort() # Sort for consistent labeling
else:
    print(\nNo real roots for this equation (discriminant is
        negative).)
```

Python Code (Direct)

```python
    # Determine plotting range based on roots or a default
    if roots:
        min_root = min(roots)
        max_root = max(roots)
        # Expand the range a bit around the roots
        padding = abs(max_root - min_root) * 0.5 + 1
        if padding == 1: # Case where there's only one root or roots are identical
            padding = 2 # Ensure some sensible padding
        plot_min_x = min_root - padding
        plot_max_x = max_root + padding
    else: # If no real roots, use a reasonable default range
        plot_min_x = -5
        plot_max_x = 5

    x_vals = np.linspace(plot_min_x, plot_max_x, 400)
    y_vals = a * x_vals**2 + b * x_vals + c
```

Python Code (Direct)

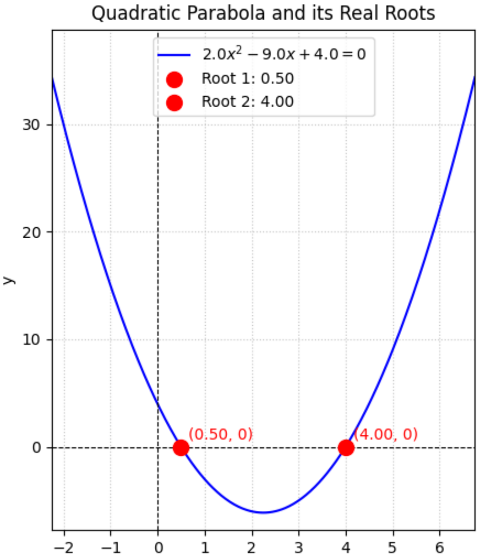
```
    plt.figure(figsize=(10, 6))
    # Plot the parabola
    plt.plot(x_vals, y_vals, label=f'${a}x^2 {b:+}x {c:+} = 0$',
        color='blue')
    # Mark the roots
    for i, root in enumerate(roots):
        plt.scatter(root, 0, color='red', s=100, zorder=5, label=
            f'Root {i+1}: {root:.2f}')
        plt.annotate(f'({root:.2f}, 0)', (root, 0), textcoords=
            offset points, xytext=(5,5), ha='left', color='red')
    # Add x and y axes for reference
    plt.axhline(0, color='black', linewidth=0.8, linestyle='--')
    plt.axvline(0, color='black', linewidth=0.8, linestyle='--')
    plt.xlabel('x')
    plt.ylabel('y')
    plt.title('Quadratic Parabola and its Real Roots')
    plt.grid(True, linestyle=':', alpha=0.7)
    plt.legend()
```

Python Code (Direct)

```python
    if len(y_vals) > 1 and np.std(y_vals) > 1e-6:
        y_min_plot = np.min(y_vals)
        y_max_plot = np.max(y_vals)
        y_padding = abs(y_max_plot - y_min_plot) * 0.1
        if y_padding == 0: y_padding = 1
        plt.ylim(y_min_plot - y_padding, y_max_plot + y_padding)
    else: # Fallback for cases with very flat or constant functions
        plt.ylim(min(y_vals)-2, max(y_vals)+2)
    plt.xlim(plot_min_x, plot_max_x) # Ensure x-limits are set
    plt.show()

# --- Main execution ---
# Given quadratic equation: 2x^2 - 9x + 4 = 0
a_given = 2.0
b_given = -9.0
c_given = 4.0
solve_quadratic_and_plot(a_given, b_given, c_given)
```

Plot by Python using shared output from C



Plot by Python only