9.5.11

Sai Krishna Bakki - EE25BTECH11049

Question

Two pipes running together can fill a tank in $100/9$ minutes. If one pipe takes 5 minutes more than the other to fill the tank separately, find the time in which each pipe would fill the tank separately.

Theoretical Solution

Given:

Let the time taken by the faster pipe to fill the tank be **'x'** minutes and the time taken by the slower pipe to fill the tank be **'x+5'** minutes.

The amount of the tank each pipe fills in one minute is its work rate.

- Work rate of the first pipe $= \frac{1}{x}$
- Work rate of the second pipe $= \frac{1}{x+5}$

When working together, they fill the tank in $\frac{100}{9}$ minutes. Therefore, their combined work rate is the reciprocal, $\frac{9}{100}$ of the tank per minute.

$$\frac{1}{x} + \frac{1}{x+5} = \frac{9}{100} \tag{1}$$

$$\frac{2x+5}{x^2+5x} = \frac{9}{100} \tag{2}$$

$$\implies y = 9x^2 - 155x - 500 = 0 \tag{3}$$

which can be expressed as the conic

$$\mathbf{x}^T\mathbf{V}\mathbf{x} + 2\mathbf{u}^T\mathbf{x} + f = 0 \tag{4}$$

$$\mathbf{V} = \begin{pmatrix} 9 & 0 \\ 0 & 0 \end{pmatrix}, \mathbf{u} = \begin{pmatrix} \frac{-155}{2} \\ \frac{-1}{2} \end{pmatrix}, f = -500 \tag{5}$$

To find the roots of (3), we find the points of intersection of the conic with the x-axis

$$\mathbf{x} = \mathbf{h} + \kappa\mathbf{m} \tag{6}$$

$$\mathbf{h} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \mathbf{m} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \tag{7}$$

The parameter κ for the points of intersection is found using the formula:

$$\kappa = \frac{1}{\mathbf{m}^\top\mathbf{V}\mathbf{m}}\left(-\mathbf{m}^\top(\mathbf{V}\mathbf{h}+\mathbf{u}) \pm \sqrt{\left[\mathbf{m}^\top(\mathbf{V}\mathbf{h}+\mathbf{u})\right]^2 - g(\mathbf{h})(\mathbf{m}^\top\mathbf{V}\mathbf{m})}\right) \tag{8}$$

where $g(\mathbf{h}) = \mathbf{h}^\top\mathbf{V}\mathbf{h} + 2\mathbf{u}^\top\mathbf{h} + f$.

using (8). The values of κ are given by

Theoretical Solution

$$\kappa_i = \frac{1}{9}\left(\frac{155}{2} \pm \sqrt{\left(\frac{-155}{2}\right)^2 + 4500}\right) \tag{9}$$

$$\implies \kappa_1 = 20, \kappa_2 = \frac{-25}{9} \tag{10}$$

Hence the points of intersection are

$$\mathbf{h} + \kappa\mathbf{m} = \begin{pmatrix} 20 \\ 0 \end{pmatrix} \begin{pmatrix} \frac{-25}{9} \\ 0 \end{pmatrix} \tag{11}$$

Hence the solutions of (3) are x=20 and x=$\frac{-25}{9}$.

C Code

```c
    #include <math.h>
/*
 * Calculates the real roots of a standard quadratic equation ax^2 + bx + c = 0.
 * Kept for reference.
 */
void solve_quadratic(double a, double b, double c, double* root1, double* root2) {
    double discriminant = b*b - 4*a*c;
    if (discriminant >= 0) {
        *root1 = (-b + sqrt(discriminant)) / (2 * a);
        *root2 = (-b - sqrt(discriminant)) / (2 * a);
    } else {
        *root1 = NAN;
        *root2 = NAN;
    }
}
```

C Code

```
/*
 * Calculates the intersection parameter 'kappa' for a line and a
     conic.
 * The conic is defined by x'Vx + 2u'x + f = 0.
 * The line is defined by x = h + kappa*m.
 * V is a 2x2 matrix (passed as a flat array [V11, V12, V21, V22
    ]).
 * u, h, m are 2x1 vectors (passed as flat arrays).
 */
void solve_conic_intersection(double* V, double* u, double f,
    double* h, double* m, double* kappa1, double* kappa2) {
    // Unpack vectors for clarity
    double h1 = h[0], h2 = h[1];
    double m1 = m[0], m2 = m[1];
    double u1 = u[0], u2 = u[1];

    // Unpack matrix (assuming row-major: V[0]=V11, V[1]=V12, V
        [2]=V21, V[3]=V22)
    double V11 = V[0], V12 = V[1], V21 = V[2], V22 = V[3];
```

C Code

```c
// 1. Calculate m_T_V_m
double m_T_V_m = m1*(V11*m1 + V12*m2) + m2*(V21*m1 + V22*m2);

// 2. Calculate g(h) = h'Vh + 2u'h + f
double h_T_V_h = h1*(V11*h1 + V12*h2) + h2*(V21*h1 + V22*h2);
double two_u_T_h = 2 * (u1*h1 + u2*h2);
double g_h = h_T_V_h + two_u_T_h + f;

// 3. Calculate m_T * (V*h + u)
double Vh1 = V11*h1 + V12*h2;
double Vh2 = V21*h1 + V22*h2;
double m_T_Vh_plus_u = m1 * (Vh1 + u1) + m2 * (Vh2 + u2);
```

C Code

```
    // 4. Calculate the term under the square root
    double discriminant_term = m_T_Vh_plus_u * m_T_Vh_plus_u -
        g_h * m_T_V_m;

    if (discriminant_term >= 0 && m_T_V_m != 0) {
        double sqrt_discriminant = sqrt(discriminant_term);
        *kappa1 = (-m_T_Vh_plus_u + sqrt_discriminant) / m_T_V_m;
        *kappa2 = (-m_T_Vh_plus_u - sqrt_discriminant) / m_T_V_m;
    } else {
        *kappa1 = NAN;
        *kappa2 = NAN;
    }
}
```

Python Code uses ctypes

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

# --- 1. SETUP CTYPES TO INTERFACE WITH THE C LIBRARY ---

# Define the name of the shared library based on the OS
if os.name == 'nt': # Windows
    lib_name = 'roots.dll'
else: # Linux, macOS, etc.
    lib_name = 'roots.so'

# Find the full path to the library in the current directory
lib_path = os.path.join(os.path.dirname(os.path.abspath(__file__)
    ), lib_name)

# Load the shared library
try:
```

Python Code uses ctypes

```
    solver_lib = ctypes.CDLL(lib_path)
except OSError as e:
    print(fError loading shared library: {e})
    print(Please make sure you have compiled solver.c into a shared library.)
    exit()
# Define the argument types and return type for the NEW C function
# void solve_conic_intersection(double* V, double* u, double f, double* h, double* m, double* kappa1, double* kappa2)
solve_conic_c = solver_lib.solve_conic_intersection
solve_conic_c.argtypes = [ctypes.POINTER(ctypes.c_double), ctypes.POINTER(ctypes.c_double),
                          ctypes.c_double,
                          ctypes.POINTER(ctypes.c_double), ctypes.POINTER(ctypes.c_double),
                          ctypes.POINTER(ctypes.c_double), ctypes.POINTER(ctypes.c_double)]
solve_conic_c.restype = None
```

Python Code uses ctypes

```
# --- 2. SOLVE FOR THE ROOTS USING THE C MATRIX FUNCTION ---

# Define conic parameters for 9x^2 - 155x - y - 500 = 0
V = np.array([[9, 0], [0, 0]], dtype=np.float64)
u = np.array([-155/2, -1/2], dtype=np.float64)
f = -500.0

# Define line parameters for the x-axis (y=0)
h = np.array([0, 0], dtype=np.float64)
m = np.array([1, 0], dtype=np.float64)

# Create ctypes variables to hold the results
root1_c = ctypes.c_double()
root2_c = ctypes.c_double()

# Convert numpy arrays to the ctypes pointers C expects
V_p = V.ctypes.data_as(ctypes.POINTER(ctypes.c_double))
u_p = u.ctypes.data_as(ctypes.POINTER(ctypes.c_double))
```

Python Code uses ctypes

```
h_p = h.ctypes.data_as(ctypes.POINTER(ctypes.c_double))
m_p = m.ctypes.data_as(ctypes.POINTER(ctypes.c_double))

# Call the C function
solve_conic_c(V_p, u_p, f, h_p, m_p, ctypes.byref(root1_c),
    ctypes.byref(root2_c))

# Extract the Python float values
roots = np.array([root1_c.value, root2_c.value])

print(--- Root Calculation (from C using Matrix Theory) ---)
print(fThe roots (kappa values) are: x1 = {roots[0]:.4f} and x2 =
     {roots[1]:.4f}\n)


# --- 3. GENERATE THE PLOT ---
# This part remains the same, as it just visualizes the results.
a, b, c = 9.0, -155.0, -500.0
```

Python Code uses ctypes

```
fig = plt.figure(figsize=(8, 8))
ax = fig.add_subplot(111)
x_vals = np.linspace(-10, 30, 500)
y_vals = a*x_vals**2 + b*x_vals + c
ax.plot(x_vals, y_vals, label=f'$y = {int(a)}x^2 + {int(b)}x + {
    int(c)}$')
ax.axhline(0, color='orange', linewidth=1.5)

sorted_roots = np.sort(roots)
roots_y = np.zeros_like(sorted_roots)
point_labels = ['B', 'A']
colors = ['gold', '#9400D3']

ax.scatter(sorted_roots, roots_y, c=colors, s=50, zorder=5,
    edgecolor='black')

for i in range(len(sorted_roots)):
    label = f$\\mathbf{{{point_labels[i]}}}$\\n({sorted_roots[i
        ]:.2f}, {roots_y[i]:.0f})
```

Python Code uses ctypes

```
    ax.annotate(label, (sorted_roots[i], roots_y[i]), textcoords=
        offset points,
                xytext=(0, 15), ha='center', fontsize=10,
                    fontweight='bold')

ax.grid(True)
ax.legend(loc='lower left')
ax.set_title('Parabola with x-intercepts (solved with C Matrix)',
     fontsize=16)
ax.set_xlabel('$x$')
ax.set_ylabel('$y$')
ax.set_xlim(-15, 35)
ax.set_ylim(-1250, 250)

print(--- Plot Generation ---)
print(Displaying plot...)
plt.show()
```

Python Code

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

# --- 1. SOLVE FOR THE ROOTS ---

# Define the coefficients of the equation: 9x^2 - 155x - 500 = 0
coefficients = [9, -155, -500]

# Use numpy's `roots` function to find the solutions
roots = np.roots(coefficients)

# Print the calculated roots
print(--- Root Calculation ---)
print(fThe equation is: {coefficients[0]}x^2 + {coefficients[1]}x
     + {coefficients[2]} = 0)
print(fThe calculated roots are: x1 = {roots[0]:.4f} and x2 = {
    roots[1]:.4f}\n)
```

Python Code

```
# --- 2. GENERATE THE PLOT ---
# Setup the plot with a specific size
fig = plt.figure(figsize=(8, 8))
ax = fig.add_subplot(111)

# Define the parabola function and generate x and y points for
    plotting
x_vals = np.linspace(-10, 30, 500)
y_vals = 9*x_vals**2 - 155*x_vals - 500

# Plot the parabola curve
ax.plot(x_vals, y_vals, label='$y = 9x^2 - 155x - 500$')

# Plot the x-axis, styled like the example
ax.axhline(0, color='orange', linewidth=1.5)

# Define the intersection points using the roots we calculated
```

Python Code

```
# Sort the roots to consistently label the left one 'B' and the
    right one 'A'
sorted_roots = np.sort(roots)
roots_y = np.zeros_like(sorted_roots)
point_labels = ['B', 'A']
colors = ['gold', '#9400D3'] # Colors styled like your example

# Plot the intersection points as colored dots
ax.scatter(sorted_roots, roots_y, c=colors, s=50, zorder=5,
    edgecolor='black')

# Add labels for the intersection points (A and B)
for i in range(len(sorted_roots)):
    # --- CORRECTED CODE ---
    # Line 1: The bold letter (A or B) using LaTeX formatting
    line1 = f$\\mathbf{{{point_labels[i]}}}$

    # Line 2: The coordinates
    line2 = f({sorted_roots[i]:.2f}, {roots_y[i]:.0f})
```

Python Code

```
   # Join the two lines with a newline character
   label = f{line1}\n{line2}
ax.annotate(label,
              (sorted_roots[i], roots_y[i]),
              textcoords=offset points,
              xytext=(0, 15),
              ha='center',
              fontsize=10,
              fontweight='bold')
```

Python Code

```
# --- 3. FINALIZE AND DISPLAY THE PLOT ---

# Add a grid, legend, and labels
ax.grid(True)
ax.legend(loc='lower left')
ax.set_title('Parabola with x-intercepts', fontsize=16)
ax.set_xlabel('$x$')
ax.set_ylabel('$y$')

# Set the viewing window for the plot
ax.set_xlim(-15, 35)
ax.set_ylim(-1250, 250)

# Display the plot in a new window
print(--- Plot Generation ---)
print(Displaying plot...)
plt.show()
```

Plot By C code and Python Code