4.13.47

EE25BTECH11043 - Nishid Khandagre

September 30, 2025

Question

The ends A, B of a straight line segment of constant length c slide upon the fixed rectangular axes OX, OY respectively. If the rectangle $OAPB$ be completed, then show that the locus of the foot of perpendicular drawn from P to AB is $x^{\frac{2}{3}} + y^{\frac{2}{3}} = c^{\frac{2}{3}}$.

Theoretical Solution

Given

$$\mathbf{A} = \begin{pmatrix} a \\ 0 \end{pmatrix} \tag{1}$$

$$\mathbf{B} = \begin{pmatrix} 0 \\ b \end{pmatrix} \tag{2}$$

Since $OAPB$ is a rectangle, the opposite corner P is:

$$\mathbf{P} = \mathbf{A} + \mathbf{B} \tag{3}$$

$$= \begin{pmatrix} a \\ b \end{pmatrix} \tag{4}$$

Theoretical Solution

$\mathbf{B} - \mathbf{A}$ has fixed length of c

$$\left\|\mathbf{B} - \mathbf{A}\right\|^2 = \left(\mathbf{B} - \mathbf{A}\right)^\top \left(\mathbf{B} - \mathbf{A}\right) \tag{5}$$

$$c^2 = a^2 + b^2 \tag{6}$$

Let $\mathbf{H}$ be the foot of the perpendicular from $\mathbf{P}$ to the line through $\mathbf{A}$ in the direction $\mathbf{B} - \mathbf{A}$.

$$\mathbf{H} = \mathbf{A} + \lambda \left(\mathbf{B} - \mathbf{A}\right) \tag{7}$$

$$\lambda = \frac{\left(\mathbf{P} - \mathbf{A}\right)^\top \left(\mathbf{B} - \mathbf{A}\right)}{\left(\mathbf{B} - \mathbf{A}\right)^\top \left(\mathbf{B} - \mathbf{A}\right)} \tag{8}$$

Theoretical Solution

$$\mathbf{P} - \mathbf{A} = \left(\mathbf{A} + \mathbf{B}\right) - \mathbf{A} \tag{9}$$

$$= \mathbf{B} \tag{10}$$

So,

$$\lambda = \frac{\mathbf{B}^\top \left(\mathbf{B} - \mathbf{A}\right)}{\left(\mathbf{B} - \mathbf{A}\right)^\top \left(\mathbf{B} - \mathbf{A}\right)} \tag{11}$$

$$= \frac{\mathbf{B}^\top \mathbf{B} - \mathbf{B}^\top \mathbf{A}}{a^2 + b^2} \tag{12}$$

Theoretical Solution

We know

$$\mathbf{B}^\top \mathbf{A} = 0 \tag{13}$$
$$\mathbf{B}^\top \mathbf{B} = b^2 \tag{14}$$

$$\lambda = \frac{b^2}{a^2 + b^2} \tag{15}$$

Theoretical Solution

Now compute **H**:

$$\mathbf{H} = \mathbf{A} + \frac{b^2}{a^2+b^2}\left(\mathbf{B}-\mathbf{A}\right) \tag{16}$$

$$= \begin{pmatrix} a \\ 0 \end{pmatrix} + \frac{b^2}{a^2+b^2}\begin{pmatrix} -a \\ b \end{pmatrix} \tag{17}$$

$$= \begin{pmatrix} a - \frac{ab^2}{a^2+b^2} \\ \frac{b^3}{a^2+b^2} \end{pmatrix} \tag{18}$$

$$= \begin{pmatrix} \frac{a(a^2+b^2)-ab^2}{a^2+b^2} \\ \frac{b^3}{a^2+b^2} \end{pmatrix} \tag{19}$$

$$= \begin{pmatrix} \frac{a^3}{a^2+b^2} \\ \frac{b^3}{a^2+b^2} \end{pmatrix} \tag{20}$$

Theoretical Solution

Let $\mathbf{H} = \begin{pmatrix} x \\ y \end{pmatrix}$. Then,

$$x = \frac{a^3}{a^2 + b^2} \tag{21}$$

$$y = \frac{b^3}{a^2 + b^2} \tag{22}$$

Using the constraint $a^2 + b^2 = c^2$:

$$a^3 = x(a^2 + b^2) = xc^2 \tag{23}$$

$$b^3 = y(a^2 + b^2) = yc^2 \tag{24}$$

Theoretical Solution

Thus,

$$a = (xc^2)^{1/3} = c^{2/3}x^{1/3} \tag{25}$$

$$b = (yc^2)^{1/3} = c^{2/3}y^{1/3} \tag{26}$$

Substitute these into $a^2 + b^2 = c^2$:

$$(c^{2/3}x^{1/3})^2 + (c^{2/3}y^{1/3})^2 = c^2 \tag{27}$$

$$c^{4/3}x^{2/3} + c^{4/3}y^{2/3} = c^2 \tag{28}$$

$$c^{4/3}(x^{2/3} + y^{2/3}) = c^2 \tag{29}$$

The locus is:

$$x^{2/3} + y^{2/3} = c^{2/3} \tag{30}$$

C Code

```
include <math.h>
// Function to calculate the foot of the perpendicular from
↪  point P to line segment AB
// P_x, P_y: coordinates of point P
// A_x, A_y: coordinates of point A
// B_x, B_y: coordinates of point B
// foot_x, foot_y: pointers to store the calculated
↪  coordinates of the foot of the perpendicular
void calculateFootOfPerpendicular(double P_x, double P_y,
double A_x, double A_y,
double B_x, double B_y,
double *foot_x, double *foot_y) {
```

C Code

```c
/ Vector AB
double BA_x = B_x - A_x;
double BA_y = B_y - A_y;

// Vector AP
double AP_x = P_x - A_x;
double AP_y = P_y - A_y;

// Calculate lambda using the projection formula:
// lambda = (AP . AB) / |AB|^2
double dot_product_AP_BA = AP_x * BA_x + AP_y * BA_y;
double length_sq_BA = BA_x * BA_x + BA_y * BA_y;
```

C Code

```c
ouble lambda = dot_product_AP_BA / length_sq_BA;

// The foot of the perpendicular F lies on the line AB:
// F = A + lambda * (B - A)
*foot_x = A_x + lambda * BA_x;
*foot_y = A_y + lambda * BA_y;
}
```

Python Code using C Shared Library

```
mport ctypes
import numpy as np
import matplotlib.pyplot as plt
lib_geometry = ctypes.CDLL("./code9.so")
# Define the argument types and return type for the C function
lib_geometry.calculateFootOfPerpendicular.argtypes = [
ctypes.c_double,  # P_x
ctypes.c_double,  # P_y
ctypes.c_double,  # A_x
ctypes.c_double,  # A_y
ctypes.c_double,  # B_x
ctypes.c_double,  # B_y
ctypes.POINTER(ctypes.c_double), # foot_x
ctypes.POINTER(ctypes.c_double)  # foot_y
]
```

Python Code using C Shared Library

```
ib_geometry.calculateFootOfPerpendicular.restype = None

def generate_locus_image():
"""
Generates an image showing the locus of the foot of the
↪ perpendicular
from P to AB, using a C function for calculation.
"""
# Define the length of the line segment
c = 5.0  # Let's choose a value for c, e.g., 5.0
# Create a range of angles for the line segment AB
# These angles will determine the positions of A and B
# Avoid 0 and pi/2 to prevent division by zero for some
↪ calculations or degenerate cases
theta_vals = np.linspace(0.01, np.pi/2 - 0.01, 100)
```

Python Code using C Shared Library

```
Initialize lists to store the coordinates of the foot of the
↪  perpendicular
locus_x = []
locus_y = []
# Ctypes variables to hold the results from the C function
foot_x_result = ctypes.c_double()
foot_y_result = ctypes.c_double()
for theta in theta_vals:
    # Coordinates of A and B
    # A lies on OY (x=0), B lies on OX (y=0)
    # Length AB = c
    A_x = 0.0
    A_y = c * np.sin(theta)
    B_x = c * np.cos(theta)
    B_y = 0.0
```

Python Code using C Shared Library

```
Complete the rectangle OAPB
    # P will have coordinates (B_x, A_y)
    P_x = B_x
    P_y = A_y
    # Call the C function to find the foot of the
    ↪ perpendicular
    lib_geometry.calculateFootOfPerpendicular(
        P_x, P_y,
        A_x, A_y,
        B_x, B_y,
        ctypes.byref(foot_x_result),
        ctypes.byref(foot_y_result)
    )
    locus_x.append(foot_x_result.value)
    locus_y.append(foot_y_result.value)
```

Python Code using C Shared Library

```
--- Plotting ---
plt.figure(figsize=(8, 8))
plt.plot(locus_x, locus_y, color='blue', linewidth=2,
↪  label='Locus from C calculation')

# For illustrative purposes, let's plot one instance of the
↪  rectangle and the foot of the perpendicular
# Choose a specific angle for demonstration
demo_t = np.pi/4
A_y_demo = c * np.sin(demo_t)
B_x_demo = c * np.cos(demo_t)
A_demo = np.array([0, A_y_demo])
B_demo = np.array([B_x_demo, 0])
P_demo = np.array([B_x_demo, A_y_demo])
```

Python Code using C Shared Library

```
Recalculate foot for demo using C function
lib_geometry.calculateFootOfPerpendicular(
    P_demo[0], P_demo[1],
    A_demo[0], A_demo[1],
    B_demo[0], B_demo[1],
    ctypes.byref(foot_x_result),
    ctypes.byref(foot_y_result)
)
F_demo = np.array([foot_x_result.value,
 ↪  foot_y_result.value])

# Plot the axes
plt.axhline(0, color='gray', linewidth=0.8)
plt.axvline(0, color='gray', linewidth=0.8)
```

```
Plot the demo rectangle and points
plt.plot([0, B_x_demo], [0, 0], 'k--', linewidth=0.7) # OX
plt.plot([0, 0], [0, A_y_demo], 'k--', linewidth=0.7) # OY
plt.plot([0, B_x_demo], [A_y_demo, A_y_demo], 'k--',
    linewidth=0.7) # PA parallel to OX
plt.plot([B_x_demo, B_x_demo], [0, A_y_demo], 'k--',
    linewidth=0.7) # PB parallel to OY
plt.plot([A_demo[0], B_demo[0]], [A_demo[1], B_demo[1]],
    'k-', label='Line segment AB (demo)')
plt.plot([P_demo[0], F_demo[0]], [P_demo[1], F_demo[1]],
    'r--', label='Perpendicular PF (demo)')
```

Python Code using C Shared Library

```python
lt.scatter([0, B_x_demo, 0, B_x_demo, F_demo[0]], [0, 0, A_y_demo, A_y_demo, F_demo[1]],
            s=50, color='black', zorder=5)
plt.text(0.1, 0.1, 'O', fontsize=12)
plt.text(B_x_demo + 0.1, 0.1, 'B', fontsize=12)
plt.text(0.1, A_y_demo + 0.1, 'A', fontsize=12)
plt.text(P_demo[0] + 0.1, P_demo[1] + 0.1, 'P', fontsize=12)
plt.text(F_demo[0] + 0.1, F_demo[1] + 0.1, 'F', fontsize=12)
# Plot the analytical solution for comparison (Astroid: x^(2/3) + y^(2/3) = c^(2/3))
# Parametric form: x = c * cos^3(t), y = c * sin^3(t)
t_astroid = np.linspace(0, np.pi/2, 200) # Only first quadrant
x_analytic = c * np.cos(t_astroid)**3
y_analytic = c * np.sin(t_astroid)**3
```

Python Code using C Shared Library

```
lt.plot(x_analytic, y_analytic, 'g--', linewidth=1.5,
         label=f'Analytical Locus: $x^{{2/3}} + y^{{2/3}} =
         ↪  c^{{2/3}}$ (c={c})')
plt.xlabel('x')
plt.ylabel('y')
plt.title('Locus of the foot of perpendicular from P to AB')
plt.legend()
plt.grid(True)
plt.axis('equal')
plt.xlim(-0.1, c + 1)
plt.ylim(-0.1, c + 1)
plt.savefig("fig1.png")
plt.show()
generate_locus_image()
```

Python Code: Direct

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

def generate_locus_image():
# Define the length of the line segment
c = 5  # Let's choose a value for c, e.g., 5

# Create a range of angles for the line segment AB
# These angles will determine the positions of A and B
theta = np.linspace(0.01, np.pi/2 - 0.01, 100) # Avoid 0 and
↪  pi/2 to prevent division by zero

# Initialize lists to store the coordinates of the foot of
↪  the perpendicular
locus_x = []
locus_y = []
```

Python Code: Direct

```
or t in theta:
    # Coordinates of A and B
    # A lies on OY (x=0), B lies on OX (y=0)
    # Length AB = c
    A_y = c * np.sin(t)
    B_x = c * np.cos(t)

    A = np.array([0, A_y])
    B = np.array([B_x, 0])
    P = np.array([B_x, A_y])

    # Vector B-A
    BA = B - A # (B_x, -A_y)
```

Python Code: Direct

```
Vector A-P
    AP = A - P # (-B_x, 0)

    # Calculate lambda for projection
    lambda_val = -np.dot(AP, BA) / np.dot(BA, BA)

    # Coordinates of F (foot of the perpendicular)
    F = A + lambda_val * BA
    locus_x.append(F[0])
    locus_y.append(F[1])

# Plotting
plt.figure(figsize=(8, 8))
plt.plot(locus_x, locus_y, color='blue', label='Locus of the
↪  foot of perpendicular')
```

Python Code: Direct

```
For illustrative purposes, let's plot one instance of the
↪  rectangle and the foot of the perpendicular
# Choose a specific angle for demonstration
demo_t = np.pi/4
A_y_demo = c * np.sin(demo_t)
B_x_demo = c * np.cos(demo_t)
A_demo = np.array([0, A_y_demo])
B_demo = np.array([B_x_demo, 0])
P_demo = np.array([B_x_demo, A_y_demo])

BA_demo = B_demo - A_demo
AP_demo = A_demo - P_demo
lambda_val_demo = -np.dot(AP_demo, BA_demo) /
↪  np.dot(BA_demo, BA_demo)
F_demo = A_demo + lambda_val_demo * BA_demo
```

Python Code: Direct

```
Plot the axes
plt.axhline(0, color='gray', linewidth=0.8)
plt.axvline(0, color='gray', linewidth=0.8)

# Plot the demo rectangle and points
plt.plot([0, B_x_demo], [0, 0], 'k--', linewidth=0.7) # OX
plt.plot([0, 0], [0, A_y_demo], 'k--', linewidth=0.7) # OY
plt.plot([0, B_x_demo], [A_y_demo, A_y_demo], 'k--',
    linewidth=0.7) # PA parallel to OX
plt.plot([B_x_demo, B_x_demo], [0, A_y_demo], 'k--',
    linewidth=0.7) # PB parallel to OY
plt.plot([A_demo[0], B_demo[0]], [A_demo[1], B_demo[1]],
    'k-', label='Line segment AB (demo)')
plt.plot([P_demo[0], F_demo[0]], [P_demo[1], F_demo[1]],
    'r--', label='Perpendicular PF (demo)')
```

Python Code: Direct

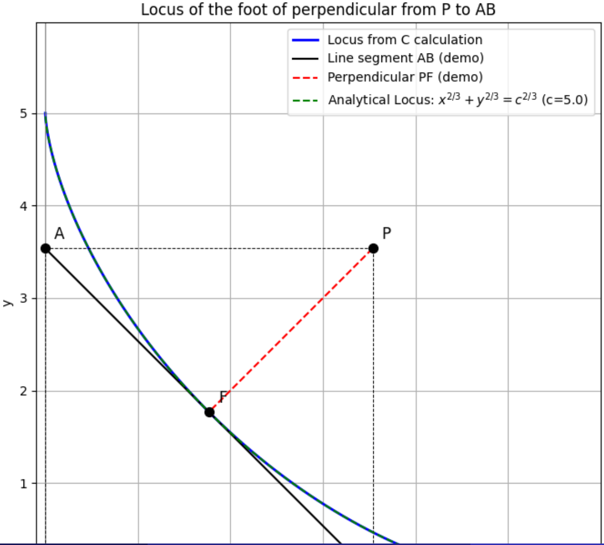
```
plt.scatter([0, B_x_demo, 0, B_x_demo, F_demo[0]], [0, 0,
    A_y_demo, A_y_demo, F_demo[1]],
            s=50, color='black', zorder=5)
plt.text(0.1, 0.1, 'O', fontsize=12)
plt.text(B_x_demo + 0.1, 0.1, 'B', fontsize=12)
plt.text(0.1, A_y_demo + 0.1, 'A', fontsize=12)
plt.text(B_x_demo + 0.1, A_y_demo + 0.1, 'P', fontsize=12)
plt.text(F_demo[0] + 0.1, F_demo[1] + 0.1, 'F', fontsize=12)

# Plot the analytical solution for comparison (x^(2/3) +
    y^(2/3) = c^(2/3))
# Parametrically: x = c * cos^3(theta), y = c * sin^3(theta)
x_analytic = (c * np.cos(theta)**3)
y_analytic = (c * np.sin(theta)**3)
```

Python Code: Direct

```
lt.plot(x_analytic, y_analytic, 'g--', label=f'Analytical
↪  Locus: $x^{{2/3}} + y^{{2/3}} = c^{{2/3}}$ (c={c})')
plt.xlabel('x')
plt.ylabel('y')
plt.title('Locus of the foot of perpendicular from P to AB')
plt.legend()
plt.grid(True)
plt.axis('equal')
plt.xlim(-0.1, c + 1)
plt.ylim(-0.1, c + 1)
plt.savefig("fig2.png")
plt.show()
generate_locus_image()
```

Plot by Python using shared output from C



Plot by Python only