8.2.7

EE25BTECH11043 - Nishid Khandagre

October 5 , 2025

Question

Find the coordinates of the focus, vertex, eccentricity, axis of the conic section, the equation of the directrix and the length of the latus rectum.
$16x^2 + y^2 = 16$

Theoretical Solution

We use an affine transformation to convert the conic equation to its standard form.

$$\mathbf{x}^\top \mathbf{V}\mathbf{x} + 2\mathbf{u}^\top \mathbf{x} + f = 0 \tag{1}$$

The symmetric matrix $\mathbf{V}$ is spectrally decomposed to align axes with eigenvectors.

$$\mathbf{V} = \mathbf{P}\mathbf{D}\mathbf{P}^\top, \; \mathbf{D} = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}, \; \mathbf{P}^\top \mathbf{P} = \mathbf{I} \tag{2}$$

Substituting the decomposition into the conic equation.

$$\mathbf{x}^\top \mathbf{P}\mathbf{D}\mathbf{P}^\top \mathbf{x} + 2\mathbf{u}^\top \mathbf{x} + f = 0 \tag{3}$$

A rotation

$$\mathbf{x_r} = \mathbf{P}^\top \mathbf{x} \tag{4}$$

aligns the conic with the coordinate axes.

$$\mathbf{x} = \mathbf{P}\mathbf{x_r} \tag{5}$$

Theoretical Solution

Applying the rotation to the conic equation.

$$(\mathbf{P}\mathbf{x_r})^\top \mathbf{P}\mathbf{D}\mathbf{P}^\top (\mathbf{P}\mathbf{x_r}) + 2\mathbf{u}^\top (\mathbf{P}\mathbf{x_r}) + f = 0 \tag{6}$$

$$\mathbf{x_r}^\top \mathbf{P}^\top \mathbf{P}\mathbf{D}\mathbf{P}^\top \mathbf{P}\mathbf{x_r} + 2\left(\mathbf{P}^\top \mathbf{u}\right)^\top \mathbf{x_r} + f = 0 \tag{7}$$

$$\mathbf{x_r}^\top \mathbf{D}\mathbf{x_r} + 2\mathbf{u_r}^\top \mathbf{x_r} + f = 0 \tag{8}$$

Theoretical Solution

A translation

$$\mathbf{x_c} = \mathbf{x_r} + \mathbf{D}^{-1}\mathbf{u_r} \tag{9}$$

moves the conic's center to the origin.

$$f_c = f - \mathbf{u_r}^\top \mathbf{D}^{-1}\mathbf{u_r} \tag{10}$$

The center of the conic in the original coordinates is

$$\mathbf{c} = -\mathbf{V}^{-1}\mathbf{u} \tag{11}$$

$$\mathbf{c} = -\left(\mathbf{P}\mathbf{D}\mathbf{P}^\top\right)^{-1}\mathbf{u} = -\mathbf{P}\mathbf{D}^{-1}\mathbf{P}^\top\mathbf{u} = -\mathbf{P}\mathbf{D}^{-1}\mathbf{u_r} \tag{12}$$

The complete transformation from original to centered coordinates is

$$\mathbf{x_c} = \mathbf{P}^\top (\mathbf{x} - \mathbf{c}) \tag{13}$$

$$\begin{aligned} \mathbf{x_c} &= \mathbf{P}^\top \mathbf{x} + \mathbf{D}^{-1}\mathbf{u_r} = \mathbf{P}^\top \mathbf{x} - \mathbf{P}^\top \mathbf{c} = \mathbf{P}^\top (\mathbf{x} - \mathbf{c}) && (14) \\ \implies \mathbf{x} &= \mathbf{P}\mathbf{x_c} + \mathbf{c} && (15) \end{aligned}$$

Theoretical Solution

The given conic equation

$$16x^2 + y^2 = 16 \tag{16}$$

$$\frac{16x^2}{16} + \frac{y^2}{16} = \frac{16}{16} \tag{17}$$

$$\frac{x^2}{1} + \frac{y^2}{16} = 1 \tag{18}$$

This is an ellipse centered at (0,0) with major axis along the y-axis.

$$\mathbf{V} = \begin{pmatrix} 1 & 0 \\ 0 & \frac{1}{16} \end{pmatrix}, \ \mathbf{u} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \ f = -1 \tag{19}$$

Theoretical Solution

The major axis corresponds to smaller eigenvalue.

$$\lambda_1 = \frac{1}{16},\ \lambda_2 = 1,\ \mathbf{P} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix},\ \mathbf{c} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \tag{20}$$

Applying the rotation to find the canonical coordinates.

$$\mathbf{x_c} = \mathbf{P}^\top \mathbf{x} \implies \begin{pmatrix} x_c \\ y_c \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} y \\ x \end{pmatrix} \tag{21}$$

Theoretical Solution

The standard form of the ellipse in canonical coordinates.

$$\frac{x_c^2}{-f/\lambda_1} + \frac{y_c^2}{-f/\lambda_2} = 1 \tag{22}$$

From this, $a^2 = -f/\lambda_1 = -(-1)/(1/16) = 16 \implies a = 4$ (major semi-axis) and $b^2 = -f/\lambda_2 = -(-1)/1 = 1 \implies b = 1$ (minor semi-axis).

Theoretical Solution

Now we can calculate the properties:

$$e = \sqrt{1 - \frac{b^2}{a^2}} = \sqrt{1 - \frac{1}{16}} = \frac{\sqrt{15}}{4} \tag{23}$$

$$\mathbf{f_c} = \pm ae\mathbf{e_1} \text{ (if major axis is x-axis)} \tag{24}$$

$$= \pm\sqrt{a^2 - b^2}\mathbf{e_1} = \pm\sqrt{16 - 1}\mathbf{e_1} = \pm\sqrt{15}\mathbf{e_1} \text{ (along canonical y-axis)} \tag{25}$$

$$\mathbf{v_c} = \pm a\mathbf{e_1} = \pm 4\mathbf{e_1} \text{ (along canonical y-axis)} \tag{26}$$

$$\mathbf{d_c} : \mathbf{e_1}^\top \mathbf{x_c} = \pm\frac{a}{e} = \pm\frac{4}{\sqrt{15}/4} = \pm\frac{16}{\sqrt{15}} \tag{27}$$

$$L = \frac{2b^2}{a} = \frac{2(1)^2}{4} = \frac{1}{2} \tag{28}$$

Transforming properties back to the original coordinate system using (15).

$$\mathbf{f} = \mathbf{P}\left(\pm\sqrt{15}\mathbf{e_1}\right) = \pm\sqrt{15}\mathbf{e_2} = \begin{pmatrix} 0 \\ \pm\sqrt{15} \end{pmatrix} \tag{29}$$

$$\mathbf{v} = \mathbf{P}\left(\pm 4\mathbf{e_1}\right) = \pm 4\mathbf{e_2} = \begin{pmatrix} 0 \\ \pm 4 \end{pmatrix} \tag{30}$$

$$\mathbf{d} : \mathbf{e_2}^\top \mathbf{x} = \pm\frac{16}{\sqrt{15}} \implies \begin{pmatrix} 0 & 1 \end{pmatrix}\mathbf{x} = \pm\frac{16}{\sqrt{15}} \tag{31}$$

Theoretical Solution Summary

Property	Value
Eccentricity	$\frac{\sqrt{15}}{4}$
Axis	$x = 0$ (Y-axis, major axis)
Vertices	$(0, \pm 4)$
Foci	$\left(0, \pm\sqrt{15}\right)$
Directrices	$y = \pm\frac{16}{\sqrt{15}}$
Latus Rectum	$\frac{1}{2}$

C Code

```c
#include <math.h>

// Function to calculate ellipse properties and pass them back
    via pointers
void calculateEllipseProperties(
    double a_val,
    double b_val,
    double* focus_y_ptr,
    double* vertex_y_ptr,
    double* eccentricity_ptr,
    double* directrix_y_ptr,
    double* latus_rectum_ptr
) {
```

C Code

```
    double c_val = sqrt(a_val * a_val - b_val * b_val);

    *focus_y_ptr = c_val;
    *vertex_y_ptr = a_val;
    *eccentricity_ptr = c_val / a_val;
    *directrix_y_ptr = a_val / (*eccentricity_ptr); // Use the
        calculated eccentricity
    *latus_rectum_ptr = (2 * b_val * b_val) / a_val;
}
```

Python Code (using C shared output)

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

# Load the shared library
lib_conic = ctypes.CDLL(./code14.so)

# Define the argument types and return type for the C function
lib_conic.calculateEllipseProperties.argtypes = [
    ctypes.c_double, # a_val
    ctypes.c_double, # b_val
    ctypes.POINTER(ctypes.c_double), # focus_y_ptr
    ctypes.POINTER(ctypes.c_double), # vertex_y_ptr
    ctypes.POINTER(ctypes.c_double), # eccentricity_ptr
    ctypes.POINTER(ctypes.c_double), # directrix_y_ptr
    ctypes.POINTER(ctypes.c_double) # latus_rectum_ptr
]
lib_conic.calculateEllipseProperties.restype = None
```

Python Code (using C shared output)

```
# --- Analyze the Ellipse: 16x^2 + y^2 = 16 ---
# From 16x^2 + y^2 = 16, divide by 16: x^2/1 + y^2/16 = 1
# This is an ellipse centered at (0,0) with major axis along y.
# a^2 = 16 => a = 4 (major semi-axis)
# b^2 = 1 => b = 1 (minor semi-axis)
a_val = 4.0
b_val = 1.0
center = np.array([0.0, 0.0]) # Center is (0,0)

# Create ctypes doubles to hold the results from the C function
focus_y_result = ctypes.c_double()
vertex_y_result = ctypes.c_double()
eccentricity_result = ctypes.c_double()
directrix_y_result = ctypes.c_double()
latus_rectum_result = ctypes.c_double()
```

Python Code (using C shared output)

```
# Call the C function to get the ellipse properties
lib_conic.calculateEllipseProperties(
    a_val, b_val,
    ctypes.byref(focus_y_result),
    ctypes.byref(vertex_y_result),
    ctypes.byref(eccentricity_result),
    ctypes.byref(directrix_y_result),
    ctypes.byref(latus_rectum_result)
)

# Extract the values from the ctypes doubles
focus_y = focus_y_result.value
vertex_y = vertex_y_result.value
eccentricity = eccentricity_result.value
directrix_y = directrix_y_result.value
latus_rectum = latus_rectum_result.value
```

Python Code (using C shared output)

```
# Calculate the other points needed for plotting and printing
# Vertices (along major axis, y-axis)
vertex1 = np.array([0.0, vertex_y])
vertex2 = np.array([0.0, -vertex_y])

# Foci (along major axis, y-axis)
focus1 = np.array([0.0, focus_y])
focus2 = np.array([0.0, -focus_y])

print(f--- Conic Section Properties (Ellipse: 16x^2 + y^2 = 16) ---)
print(fCenter: ({center[0]:.0f}, {center[1]:.0f}))
print(fVertices: ({vertex1[0]:.0f}, {vertex1[1]:.0f}) and ({vertex2[0]:.0f}, {vertex2[1]:.0f}))
print(fFoci: ({focus1[0]:.2f}, {focus1[1]:.2f}) and ({focus2[0]:.2f}, {focus2[1]:.2f}))
print(fEccentricity: {eccentricity:.4f})
```

Python Code (using C shared output)

```
print(fAxis of the conic section: y-axis (x=0) is the major axis)
print(fEquation of Directrices: y = {directrix_y:.2f} and y = {-directrix_y:.2f})
print(fLength of Latus Rectum: {latus_rectum:.2f})

# --- Plotting the Ellipse with improved aesthetics ---
plt.figure(figsize=(10, 10))
ax = plt.gca()

# Generate points for the ellipse
theta = np.linspace(0, 2 * np.pi, 200)
x_ellipse = b_val * np.cos(theta)
y_ellipse = a_val * np.sin(theta)
plt.plot(x_ellipse, y_ellipse, blue, linewidth=2, label='Ellipse $16x^2 + y^2 = 16$')
```

Python Code (using C shared output)

```python
# Plot Center (Black dot)
plt.scatter(0, 0, color='black', s=30, zorder=5, label='Center
    (0,0)')

# Plot Vertices (Red dots)
plt.scatter(0, vertex_y, color='red', s=30, zorder=5, label=f'
    Vertices (0, $\\pm${vertex_y:.0f})')
plt.scatter(0, -vertex_y, color='red', s=30, zorder=5)
# Annotations for vertices
plt.annotate(f'(0, {vertex_y:.0f})', (0, vertex_y), textcoords=
    offset points, xytext=(5, 5), ha='left', color='red',
    fontsize=10)
plt.annotate(f'(0, {-vertex_y:.0f})', (0, -vertex_y), textcoords=
    offset points, xytext=(5, 5), ha='left', color='red',
    fontsize=10)
```

Python Code (using C shared output)

```
# Plot Foci (Green dots)
plt.scatter(0, focus_y, color='green', s=30, zorder=5, label=f'Foci (0, $\\pm${focus_y:.2f})')
plt.scatter(0, -focus_y, color='green', s=30, zorder=5)
plt.annotate(f'(0, {focus_y:.2f})', (0, focus_y), textcoords=offset points, xytext=(5, -15), ha='left', color='green', fontsize=10)
plt.annotate(f'(0, {-focus_y:.2f})', (0, -focus_y), textcoords=offset points, xytext=(5, 5), ha='left', color='green', fontsize=10)
# Plot Directrices (Magenta dashed lines)
x_plot_limits = np.array([-b_val * 2.5, b_val * 2.5]) # Set x limits for directrix lines
plt.plot(x_plot_limits, [directrix_y, directrix_y], 'b--', linewidth=1.5, label=f'Directrices y = $\\pm${directrix_y:.2f}')
plt.plot(x_plot_limits, [-directrix_y, -directrix_y], 'b--', linewidth=1.5)
```

Python Code (using C shared output)

```
# Plot Latus Rectum (Cyan dotted lines)
lr_half = latus_rectum / 2
plt.plot([-lr_half, lr_half], [focus_y, focus_y], 'g-', linewidth
    =2, label=f'Latus Rectum Length={latus_rectum:.2f}')
plt.plot([-lr_half, lr_half], [-focus_y, -focus_y], 'g-',
    linewidth=2)

ax.set_aspect('equal', adjustable='box')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Properties of the Ellipse $16x^2 + y^2 = 16$')
plt.grid(True)
```

Python Code (using C shared output)

```
# Set explicit plot limits
plt.xlim(-2.5, 2.5)
plt.ylim(-5, 5)

# Use tight_layout to adjust plot parameters, leaving space at
    the bottom for the legend
plt.tight_layout(rect=[0, 0.2, 1, 1])

# Save the figure
plt.savefig(fig1.png)
plt.show()

print(\nFigure saved as fig1.png)
```

Python Code (Direct)

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

# Function to generate points for a line segment
def line_gen_num(A, B, num_points):
    A = np.array(A).flatten()
    B = np.array(B).flatten()
    t = np.linspace(0, 1, num_points)
    points = np.array([(1-t) * A[0] + t * B[0], (1-t) * A[1] + t * B[1]])
    return points
# Function to generate points for an ellipse
def ellipse_gen(center, a, b, num_points=100):
    center = np.array(center).flatten()
    theta = np.linspace(0, 2*np.pi, num_points)
    x = center[0] + b * np.cos(theta)
    y = center[1] + a * np.sin(theta)
    return np.array([x, y])
```

Python Code (Direct)

```
# --- Analyze the Ellipse: 16x^2 + y^2 = 16 ---
# Standard form: x^2/b^2 + y^2/a^2 = 1
# Divide by 16: x^2/1 + y^2/16 = 1
a_val = 4.0 # Major semi-axis along y
b_val = 1.0 # Minor semi-axis along x
center = np.array([0.0, 0.0])

# Calculate properties
c_val = np.sqrt(a_val**2 - b_val**2) # Distance from center to
    focus
eccentricity = c_val / a_val
latus_rectum_length = (2 * b_val**2) / a_val

# Vertices (along major axis, y-axis)
vertex1 = np.array([0.0, a_val])
vertex2 = np.array([0.0, -a_val])
```

Python Code (Direct)

```
# Foci (along major axis, y-axis)
focus1 = np.array([0.0, c_val])
focus2 = np.array([0.0, -c_val])

# Directrices (equations are y = +/- a/e)
directrix_y = a_val / eccentricity
print(f--- Conic Section Properties (Ellipse: 16x^2 + y^2 = 16)
    ---)
print(fCenter: ({center[0]:.0f}, {center[1]:.0f}))
print(fVertices: ({vertex1[0]:.0f}, {vertex1[1]:.0f}) and ({
    vertex2[0]:.0f}, {vertex2[1]:.0f}))
print(fFoci: ({focus1[0]:.2f}, {focus1[1]:.2f}) and ({focus2
    [0]:.2f}, {focus2[1]:.2f}))
print(fEccentricity: {eccentricity:.4f})
print(fAxis of the conic section: y-axis (x=0) is the major axis)
print(fEquation of Directrices: y = {directrix_y:.2f} and y = {-
    directrix_y:.2f})
print(fLength of Latus Rectum: {latus_rectum_length:.2f})
```

Python Code (Direct)

```
# --- Plotting ---
plt.figure(figsize=(12, 10)) # Increased width from 10 to 12
ax = plt.gca()
# Generate points for the ellipse
x_ellipse = ellipse_gen(center, a_val, b_val)
plt.plot(x_ellipse[0,:], x_ellipse[1,:], g-, linewidth=2, label='
    Ellipse $16x^2 + y^2 = 16$')
# Plot Center
plt.scatter(center[0], center[1], color='green', s=50, zorder=5,
    label='Center (0,0)')
# Plot Vertices
plt.scatter(vertex1[0], vertex1[1], color='black', s=30, zorder
    =5, label=f'Vertices (0, $\\pm${a_val:.0f})')
plt.scatter(vertex2[0], vertex2[1], color='black', s=30, zorder
    =5)
plt.annotate(f'(0, {vertex1[1]:.0f})', (vertex1[0],vertex1[1]),
    textcoords=offset points, xytext=(3, 0), ha='left', color='
    black', weight='bold') # Adjusted xytext
```

Python Code (Direct)

```python
plt.annotate(f'(0, {vertex2[1]:.0f})', (vertex2[0],vertex2[1]), textcoords=offset points, xytext=(3, -4), ha='left', color='black', weight='bold') # Adjusted xytext
# Plot Foci
plt.scatter(focus1[0], focus1[1], color='blue', s=30, zorder=5, label=f'Foci (0, $\\pm${focus1[1]:.2f})')
plt.scatter(focus2[0], focus2[1], color='blue', s=30, zorder=5)
plt.annotate(f'(0, {focus1[1]:.2f})', (focus1[0],focus1[1]), textcoords=offset points, xytext=(0, -15), ha='left', color='blue', weight='bold') # Adjusted xytext
plt.annotate(f'(0, {focus2[1]:.2f})', (focus2[0],focus2[1]), textcoords=offset points, xytext=(0, 10), ha='left', color='blue', weight='bold') # Adjusted xytext
```

```
# Plot Directrices
x_lim = np.array([-b_val * 2.5, b_val * 2.5]) # Adjust x-limits
    for directrix lines, slightly wider
plt.plot(x_lim, [directrix_y, directrix_y], 'r', linewidth=1.5,
    label=f'Directrices y = $\\pm${directrix_y:.2f}')
plt.plot(x_lim, [-directrix_y, -directrix_y], 'r', linewidth=1.5)
# Plot Latus Rectum
lr_half = latus_rectum_length / 2
plt.plot([-lr_half, lr_half], [focus1[1], focus1[1]], 'g-',
    linewidth=2, label=f'Latus Rectum Length={latus_rectum_length
    :.2f}')
plt.plot([-lr_half, lr_half], [focus2[1], focus2[1]], 'g-',
    linewidth=2)
ax.set_aspect('equal', adjustable='box')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Properties of the Ellipse $16x^2 + y^2 = 16$')
plt.grid(True)
```

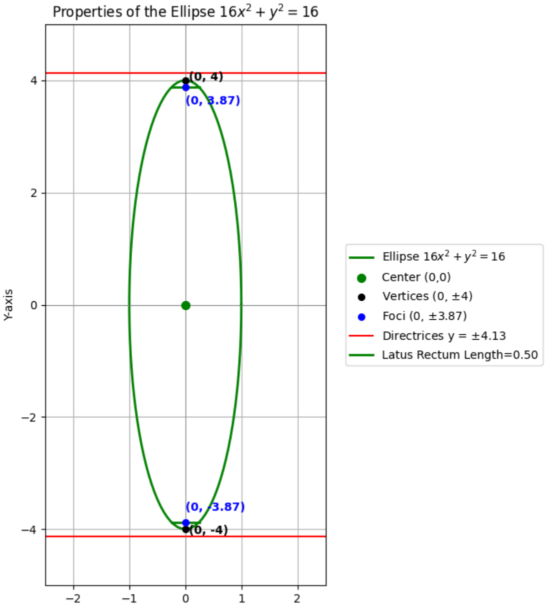
```
# Place the legend outside the plot area
plt.legend(loc='center left', bbox_to_anchor=(1.05, 0.5),
    fontsize='medium') # Moves legend to the right

plt.axhline(0, color='gray', linewidth=0.5)
plt.axvline(0, color='gray', linewidth=0.5)

# Adjust plot limits if necessary to ensure all annotations and
    elements are visible
plt.xlim(-2.5, 2.5) # Slightly wider X-axis to make space for
    annotations
plt.ylim(-5, 5) # Slightly taller Y-axis if needed

plt.savefig(fig2.png)
plt.show()
print(\nFigure saved as fig2.png)
```

Plot by Python using shared output from C



Plot by Python only