Presentation - Matgeo

Aryansingh Sonaye
AI25BTECH11032
EE1030 - Matrix Theory

September 29, 2025

Problem Statement

Find the equation of the set of all points the sum of whose distances from the points $\begin{pmatrix} 3 \\ 0 \end{pmatrix}$ and $\begin{pmatrix} 9 \\ 0 \end{pmatrix}$ is 12.

Description of Variables used

Variable	Value
$\mathbf{F}_1$	$\begin{pmatrix} 3 \\ 0 \end{pmatrix}$
$\mathbf{F}_2$	$\begin{pmatrix} 9 \\ 0 \end{pmatrix}$
$2a$	12

Table

Theoretical Solution

Step 1: Center and axis data

$$\mathbf{c} = \frac{\mathbf{F}_1 + \mathbf{F}_2}{2} = \begin{pmatrix} 6 \\ 0 \end{pmatrix}, \quad \mathbf{v} = \frac{\mathbf{F}_2 - \mathbf{F}_1}{\|\mathbf{F}_2 - \mathbf{F}_1\|} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad c_f = \frac{\|\mathbf{F}_2 - \mathbf{F}_1\|}{2} = 3, \quad a = \tag{3.1}$$

Shift to the midpoint frame: $\mathbf{y} := \mathbf{x} - \mathbf{c}$.

Step 2: Start from the sum-of-distances definition

$$\|\mathbf{y} - c_f \mathbf{v}\| + \|\mathbf{y} + c_f \mathbf{v}\| = 2a. \tag{3.2}$$

Step 3: Eliminate square roots (squaring twice) Let $r_\pm := \|\mathbf{y} \pm c_f \mathbf{v}\|$. From (3.2), $r_+ + r_- = 2a$.

$$r_+ r_- = 2a^2 - \|\mathbf{y}\|^2 - c_f^2, \tag{3.3}$$

$$r_+ - r_- = \frac{2c_f}{a} \mathbf{v}^\top \mathbf{y} \ \Rightarrow \ r_+ r_- = a^2 - \frac{c_f^2}{a^2} (\mathbf{v}^\top \mathbf{y})^2. \tag{3.4}$$

Theoretical Solution

Equating the two expressions for $r_+ r_-$ yields

$$\|\mathbf{y}\|^2 - \frac{c_f^2}{a^2}(\mathbf{v}^\top \mathbf{y})^2 = a^2 - c_f^2 \ =: \ b^2. \tag{3.5}$$

Step 4: Principal directions and the matrix D
Choose an orthonormal basis of principal directions:

$$\mathbf{p}_1 = \mathbf{v}, \qquad \mathbf{p}_2 \perp \mathbf{p}_1, \qquad P := \begin{pmatrix} \mathbf{p}_1 & \mathbf{p}_2 \end{pmatrix} \ \text{(orthonormal)}. \tag{3.6}$$

Decompose $\mathbf{y}$ as $\mathbf{y} = \alpha\,\mathbf{p}_1 + \beta\,\mathbf{p}_2$, where $\alpha = \mathbf{p}_1^\top \mathbf{y} = \mathbf{v}^\top \mathbf{y}$ and $\beta = \mathbf{p}_2^\top \mathbf{y}$. Then $\|\mathbf{y}\|^2 = \alpha^2 + \beta^2$. Substituting into (5) gives

$$\frac{\alpha^2}{a^2} + \frac{\beta^2}{b^2} = 1. \tag{3.7}$$

In matrix form this is

Theoretical Solution

$$\mathbf{y}^\top \Big(P\,\mathrm{diag}\big(\tfrac{1}{a^2}, \tfrac{1}{b^2}\big)\,P^\top\Big)\mathbf{y} = 1. \tag{3.8}$$

Hence define

$$D := P\,\mathrm{diag}\Big(\tfrac{1}{a^2}, \tfrac{1}{b^2}\Big)\,P^\top, \qquad \text{so that} \qquad (\mathbf{x}-\mathbf{c})^\top D\,(\mathbf{x}-\mathbf{c}) = 1. \tag{3.9}$$

Step 5: Specialization to this data

Here $\mathbf{p}_1 = \begin{pmatrix}1\\0\end{pmatrix}$, $\mathbf{p}_2 = \begin{pmatrix}0\\1\end{pmatrix}$, so $P = I$ and

$$b^2 = a^2 - c_f^2 = 36 - 9 = 27, \tag{3.10}$$

$$D = \mathrm{diag}\Big(\tfrac{1}{a^2}, \tfrac{1}{b^2}\Big) = \begin{pmatrix}\frac{1}{36} & 0\\ 0 & \frac{1}{27}\end{pmatrix}. \tag{3.11}$$

Theoretical Solution

Therefore the **centered matrix equation of the locus** is exactly (9) with

$$\boxed{(\mathbf{x} - \begin{pmatrix} 6 \\ 0 \end{pmatrix})^\top \begin{pmatrix} \frac{1}{36} & 0 \\ 0 & \frac{1}{27} \end{pmatrix} (\mathbf{x} - \begin{pmatrix} 6 \\ 0 \end{pmatrix}) = 1} \tag{3.12}$$

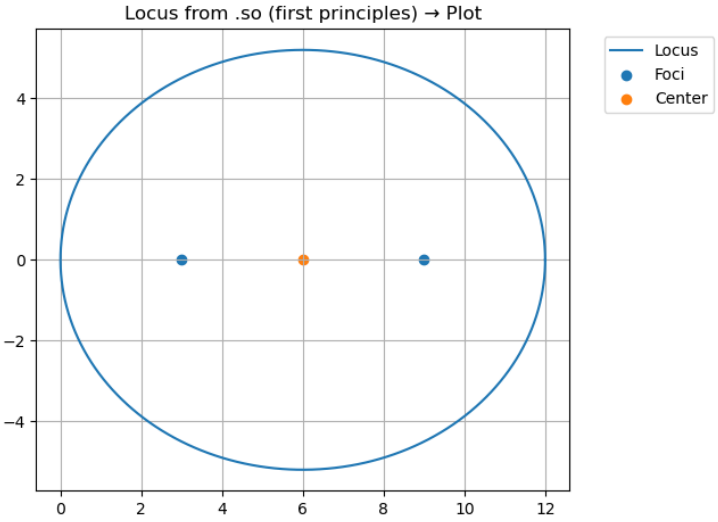
Step 6: General quadratic (matrix) form
Expanding (9) gives $\mathbf{x}^\top V\mathbf{x} + 2\mathbf{u}^\top\mathbf{x} + f = 0$ with

$$V = D, \qquad \mathbf{u} = -V\mathbf{c}, \qquad f = \mathbf{c}^\top V\mathbf{c} - 1. \tag{3.13}$$

Numerically,

$$\boxed{V = \begin{pmatrix} \frac{1}{36} & 0 \\ 0 & \frac{1}{27} \end{pmatrix}, \quad \mathbf{u} = \begin{pmatrix} -\frac{1}{6} \\ 0 \end{pmatrix}, \quad f = 0}. \tag{3.14}$$

Plot


Locus from .so (first principles) → Plot
Locus
Foci
Center
4
2
0
−2
−4
0
2
4
6
8
10
12

Figure

Code - C

```
#include <math.h>

void ellipse_params(const double *F1, const double *F2, double sum,
                    double *V, double *u, double *f,
                    double *c, double *a_out, double *b_out)
{
    // center
    c[0] = 0.5 * (F1[0] + F2[0]);
    c[1] = 0.5 * (F1[1] + F2[1]);

    // a, cf, b
    double a = 0.5 * sum;
    double dx = F2[0] - F1[0];
    double dy = F2[1] - F1[1];
    double cf = 0.5 * sqrt(dx*dx + dy*dy);
    double b2 = a*a - cf*cf;
    double b = sqrt(b2);
```

Code - C

```
    if (a_out) *a_out = a;
    if (b_out) *b_out = b;

    // V = diag(1/a^2, 1/b^2)
    V[0] = 1.0/(a*a); V[1] = 0.0;
    V[2] = 0.0; V[3] = 1.0/(b*b);

    // u = -V c
    u[0] = -(V[0]*c[0] + V[1]*c[1]);
    u[1] = -(V[2]*c[0] + V[3]*c[1]);

    // f = c^T V c - 1
    *f = c[0]*(V[0]*c[0] + V[1]*c[1]) + c[1]*(V[2]*c[0] + V[3]*c[1]) -
        1.0;
}
```

Code - Python(with shared C code)

The code to obtain the required plot is

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

# Load the shared library
# On macOS, use: lib = ct.CDLL("./libellipse_simple.dylib")
lib = ct.CDLL("./libellipse_simple.so")

# Set function signature
lib.ellipse_params.argtypes = [
    ct.POINTER(ct.c_double), # F1
    ct.POINTER(ct.c_double), # F2
    ct.c_double, # sum (2a)
    ct.POINTER(ct.c_double), # V (len 4, row-major)
    ct.POINTER(ct.c_double), # u (len 2)
    ct.POINTER(ct.c_double), # f (scalar)
```

Code - Python(with shared C code)

```
    ct.POINTER(ct.c_double), # c (len 2)
    ct.POINTER(ct.c_double), # a_out (scalar)
    ct.POINTER(ct.c_double), # b_out (scalar)
]
lib.ellipse_params.restype = None
# Inputs (your problem)
F1 = np.array([3.0, 0.0], dtype=np.float64)
F2 = np.array([9.0, 0.0], dtype=np.float64)
sum_dist = 12.0

# Outputs
V = np.zeros(4, dtype=np.float64)
u = np.zeros(2, dtype=np.float64)
f = np.zeros(1, dtype=np.float64)
c = np.zeros(2, dtype=np.float64)
a = np.zeros(1, dtype=np.float64)
b = np.zeros(1, dtype=np.float64)
```

Code - Python(with shared C code)

```
# Call the C function
lib.ellipse_params(
    F1.ctypes.data_as(ct.POINTER(ct.c_double)),
    F2.ctypes.data_as(ct.POINTER(ct.c_double)),
    ct.c_double(sum_dist),
    V.ctypes.data_as(ct.POINTER(ct.c_double)),
    u.ctypes.data_as(ct.POINTER(ct.c_double)),
    f.ctypes.data_as(ct.POINTER(ct.c_double)),
    c.ctypes.data_as(ct.POINTER(ct.c_double)),
    a.ctypes.data_as(ct.POINTER(ct.c_double)),
    b.ctypes.data_as(ct.POINTER(ct.c_double)),
)
print("V =\n", V.reshape(2,2))
print("u =", u)
print("f =", f[0])
print("center c =", c)
print("a, b =", a[0], b[0])
```

Code - Python(with shared C code)

```
# Parametric plot (simple & direct)
t = np.linspace(0, 2*np.pi, 600)
x = c[0] + a[0]*np.cos(t)
y = c[1] + b[0]*np.sin(t)

plt.plot(x, y, label="Locus")
plt.scatter([F1[0], F2[0]], [F1[1], F2[1]], label="Foci")
plt.scatter([c[0]], [c[1]], label="Center")
plt.gca().set_aspect("equal", adjustable="box")
plt.grid(True)
plt.legend(loc="upper left", bbox_to_anchor=(1.05, 1.0))
plt.title("Locus from .so (first principles) => Plot")
plt.tight_layout()
plt.savefig("ellipse.png")
plt.show()
```

Code - Python only

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

def ellipse_params_from_foci(F1, F2, s):
    F1 = np.asarray(F1, dtype=float)
    F2 = np.asarray(F2, dtype=float)
    c = 0.5 * (F1 + F2)
    a = 0.5 * s
    d = np.linalg.norm(F2 - F1)
    cf = 0.5 * d
    b2 = a*a - cf*cf
    if b2 <= 0:
        raise ValueError("Inputs do not define a real ellipse (b^2 <= 0).")
    b = np.sqrt(b2)
```

Code - Python only

```
    # Axis-aligned case (since foci are on a line here)
    V = np.diag([1.0/(a*a), 1.0/(b*b)])
    u = -V @ c
    f = float(c @ (V @ c) - 1.0)
    return V, u, f, c, a, b

def plot_ellipse(c, a, b, F1=None, F2=None, n=600):
    t = np.linspace(0, 2*np.pi, n)
    x = c[0] + a*np.cos(t)
    y = c[1] + b*np.sin(t)

    plt.plot(x, y, label="Locus")
    if F1 is not None and F2 is not None:
        plt.scatter([F1[0], F2[0]], [F1[1], F2[1]], label="Foci")
    plt.scatter([c[0]], [c[1]], label="Center")
    plt.gca().set_aspect("equal", adjustable="box")
    plt.grid(True)
```

Code - Python only

```
    plt.legend(loc="upper left", bbox_to_anchor=(1.05, 1.0))
    plt.title("Locus from first principles => (V,u,f) => Plot")
    plt.tight_layout()
    plt.savefig("newellipse.png")
    plt.show()

# --- Given data ---
F1 = np.array([3.0, 0.0])
F2 = np.array([9.0, 0.0])
s = 12.0 # sum of distances = 2a

V, u, f, c, a, b = ellipse_params_from_foci(F1, F2, s)
```

Code - Python only

```
# Show results
print("V =\n", V)
print("u =", u)
print("f =", f)
print("center c =", c)
print("semi-axes a, b =", a, b)

# Plot
plot_ellipse(c, a, b, F1=F1, F2=F2)
```