Presentation - Matgeo

Aryansingh Sonaye
AI25BTECH11032
EE1030 - Matrix Theory

September 28, 2025

Problem Statement

Problem 8.3.12 Find the equation of the set of all points the sum of whose distances from the points $(3, 0)$ and $(9, 0)$ 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 directions

$$\mathbf{c} = \frac{\mathbf{F_1} + \mathbf{F_2}}{2} = \begin{pmatrix} 6 \\ 0 \end{pmatrix} \tag{2.1}$$

$$\mathbf{p_1} = \frac{\mathbf{F_2} - \mathbf{F_1}}{\|\mathbf{F_2} - \mathbf{F_1}\|} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad \mathbf{p_2} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \quad P = I \tag{2.2}$$

Step 2: Semi-minor axis

$$c_f = \frac{\|\mathbf{F_2} - \mathbf{F_1}\|}{2} = 3 \tag{2.3}$$

$$a = 6 \tag{2.4}$$

$$b^2 = a^2 - c_f^2 = 36 - 9 = 27 \tag{2.5}$$

Theoretical Solution

Step 3: Standard ellipse form

$$(\mathbf{x}-\mathbf{c})^\top D(\mathbf{x}-\mathbf{c}) = 1 \tag{2.6}$$

$$D = \begin{pmatrix} 1/a^2 & 0 \\ 0 & 1/b^2 \end{pmatrix} = \begin{pmatrix} 1/36 & 0 \\ 0 & 1/27 \end{pmatrix} \tag{2.7}$$

$$V = PDP^\top = D \tag{2.8}$$

Step 4: Convert to general quadratic form

$$(\mathbf{x}-\mathbf{c})^\top V(\mathbf{x}-\mathbf{c}) = 1 \tag{2.9}$$

$$\mathbf{x}^\top V\mathbf{x} - 2\mathbf{c}^\top V\mathbf{x} + \mathbf{c}^\top V\mathbf{c} - 1 = 0 \tag{2.10}$$

Comparing with $\mathbf{x}^\top V\mathbf{x} + 2\mathbf{u}^\top\mathbf{x} + f = 0$:

$$\mathbf{u} = -V\mathbf{c} \tag{2.11}$$

$$f = \mathbf{c}^\top V\mathbf{c} - 1 \tag{2.12}$$

Theoretical Solution

Compute:

$$\mathbf{u} = -\begin{pmatrix} 1/36 & 0 \\ 0 & 1/27 \end{pmatrix} \begin{pmatrix} 6 \\ 0 \end{pmatrix} = \begin{pmatrix} -1/6 \\ 0 \end{pmatrix} \tag{2.13}$$

$$f = \begin{pmatrix} 6 & 0 \end{pmatrix} \begin{pmatrix} 1/36 & 0 \\ 0 & 1/27 \end{pmatrix} \begin{pmatrix} 6 \\ 0 \end{pmatrix} - 1 = 0 \tag{2.14}$$

Step 5: Clear denominators

$$V = \begin{pmatrix} 3 & 0 \\ 0 & 4 \end{pmatrix} \tag{2.15}$$

$$\mathbf{u} = \begin{pmatrix} -18 \\ 0 \end{pmatrix} \tag{2.16}$$

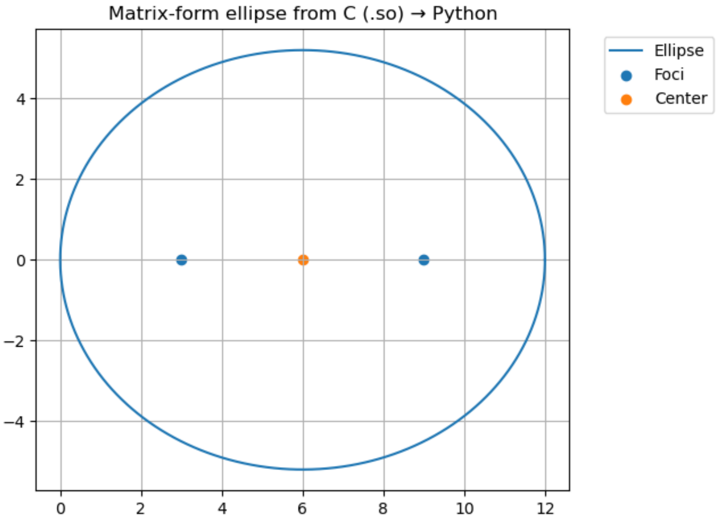
$$f = 0 \tag{2.17}$$

Theoretical Solution

Final Matrix Equation

$$\boxed{\mathbf{x}^\top V\mathbf{x} + 2\mathbf{u}^\top\mathbf{x} + f = 0, \quad V = \begin{pmatrix} 3 & 0 \\ 0 & 4 \end{pmatrix}, \ \mathbf{u} = \begin{pmatrix} -18 \\ 0 \end{pmatrix}, \ f = 0} \tag{2.18}$$

Plot


Matrix-form ellipse from C (.so) → Python
Ellipse
Foci
Center
4
2
0
−2
−4
0
2
4
6
8
10
12

Figure

Code - C

```c
#include <math.h>

// Compute V (2x2, row-major), u (2), f from foci F1,F2 and sum (=2a)
    .
// c = (F1+F2)/2
// a = sum/2
// c_f = ||F2-F1||/2, b^2 = a^2 - c_f^2
// D = diag(1/a^2, 1/b^2) [axis-aligned for this problem]
// V = D
// u = -V c
// f = c^T V c - 1
void ellipse_vuf(const double *F1, const double *F2, double sum,
                 double *V, double *u, double *f)
{
    // center
    double cx = (F1[0] + F2[0]) / 2.0;
    double cy = (F1[1] + F2[1]) / 2.0;
```

Code - C

```
// a, c_f, b
double a = sum / 2.0;
double dx = F2[0] - F1[0];
double dy = F2[1] - F1[1];
double cf = sqrt(dx*dx + dy*dy) / 2.0;
double b2 = a*a - cf*cf;
double b = sqrt(b2);

// D = diag(1/a^2, 1/b^2)
double D00 = 1.0/(a*a);
double D11 = 1.0/(b*b);

// V = D (axis-aligned for this question)
V[0] = D00; V[1] = 0.0;
V[2] = 0.0; V[3] = D11;
```

Code - C

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

    // f = c^T V c - 1
    *f = cx*(V[0]*cx + V[1]*cy) + cy*(V[2]*cx + V[3]*cy) - 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 shared lib ---
lib = ct.CDLL("./libellipse.so")
lib.ellipse_vuf.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)
]
lib.ellipse_vuf.restype = None
```

Code - Python(with shared C code)

```
# --- problem data ---
F1 = np.array([3.0, 0.0], dtype=np.float64)
F2 = np.array([9.0, 0.0], dtype=np.float64)
sum_dist = 12.0 # 2a
# --- outputs ---
V = np.zeros(4, dtype=np.float64) # row-major 2x2
u = np.zeros(2, dtype=np.float64)
f = np.zeros(1, dtype=np.float64)
# call C
lib.ellipse_vuf(
    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)),
)
```

Code - Python(with shared C code)

```
V2 = V.reshape(2,2)
print("V =\n", V2)
print("u =", u)
print("f =", f[0])

# --- derive c, f0, a, b from (V,u,f), exactly like theory ---
# c = -V^{-1} u
c = -np.linalg.solve(V2, u)

# f0 = u^T V^{-1} u - f
f0 = u @ np.linalg.solve(V2, u) - f[0]

# eigendecomposition V = P diag(lam) P^T
lam, P = np.linalg.eigh(V2) # lam[0]<=lam[1]
# axes: a^2 = f0/lam1, b^2 = f0/lam2
a = np.sqrt(f0 / lam[0])
b = np.sqrt(f0 / lam[1])
```

Code - Python(with shared C code)

```
print("center c =", c)
print("f0 =", f0)
print("semi-axes a, b =", a, b)

# --- plot from (V,u,f) ---
t = np.linspace(0, 2*np.pi, 600)
ellipse_local = np.vstack([a*np.cos(t), b*np.sin(t)]) # (2,N)
ellipse_global = (P @ ellipse_local).T + c # rotate+shift

plt.plot(ellipse_global[:,0], ellipse_global[:,1], label="Ellipse")
plt.scatter([F1[0], F2[0]], [F1[1], F2[1]], label="Foci")
plt.scatter([c[0]], [c[1]], label="Center")
```

Code - Python(with shared C code)

```
plt.gca().set_aspect("equal", adjustable="box")
plt.legend(loc="upper-left", bbox_to_anchor=(1.05, 1.0))
plt.grid(True)
plt.title("Matrix—form-ellipse-from-C-(.so)-—-Python")
plt.tight_layout()
plt.savefig("ellipse.png")
plt.show()
```

Code - Python only

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



def ellipse_vuf_from_foci(F1, F2, sum_dist):
    """
    Build (V, u, f) from foci and sum of distances (2a).
    Assumes axis-aligned ellipse when foci are collinear on x- or y-axis.
    """
    F1 = np.asarray(F1, dtype=float)
    F2 = np.asarray(F2, dtype=float)

    # Center and axes lengths
    c = (F1 + F2) / 2.0 # center
    a = sum_dist / 2.0 # semi-major
    cf = np.linalg.norm(F2 - F1) / 2.0 # half focal distance
    b2 = a * a - cf * cf
```

Code - Python only

```
if b2 <= 0:
    raise ValueError("Invalid inputs: b^2 <= 0 (no real ellipse).")
b = np.sqrt(b2)

# D = diag(1/a^2, 1/b^2), axis-aligned for this problem => V = D
V = np.diag([1.0 / (a * a), 1.0 / (b * b)])

# u = -V c, f = c^T V c - 1
u = -V @ c
f = float(c @ (V @ c) - 1.0)

return V, u, f, c, a, b
```

Code - Python only

```
def recover_from_Vuf(V, u, f):
    """
    From (V, u, f), recover:
      c = -V^{-1} u
      f0 = u^T V^{-1} u - f
      eigendecomposition V = P diag(lam) P^T
      semi-axes via theory: a^2 = f0 / lam_min, b^2 = f0 / lam_max
    """
    V = np.asarray(V, dtype=float)
    u = np.asarray(u, dtype=float)

    # center
    c = -np.linalg.solve(V, u)

    # f0
    Vinv_u = np.linalg.solve(V, u)
    f0 = float(u @ Vinv_u - f)
```

Code - Python only

```
    # eigen
    lam, P = np.linalg.eigh(V) # lam sorted ascending, columns of P are
        eigenvectors

    # semi-axes from theory
    if np.any(lam <= 0):
        raise ValueError("V must be positive definite for an ellipse.")
    a = np.sqrt(f0 / lam[0])
    b = np.sqrt(f0 / lam[1])

    return c, f0, lam, P, a, b

def sample_ellipse_points(c, P, a, b, num=600):
    """
    Parametric ellipse in principal coordinates, then rotate+shift:
      z(t) = [a cos t; b sin t],  x(t) = P z(t) + c
    """
```

Code - Python only

```
    t = np.linspace(0.0, 2.0 * np.pi, num)
    ellipse_local = np.vstack([a * np.cos(t), b * np.sin(t)]) # shape (2, N
        )
    ellipse_global = (P @ ellipse_local).T + c # shape (N, 2)
    return ellipse_global


def main():
    # ------ Problem data ------
    F1 = np.array([3.0, 0.0])
    F2 = np.array([9.0, 0.0])
    sum_dist = 12.0 # 2a

    # ------ Build V, u, f as per the align-derivation ------
    V, u, f, c_direct, a_direct, b_direct = ellipse_vuf_from_foci(F1, F2,
        sum_dist)
```

Code - Python only

```
print("From direct construction (D, V=D, u=-Vc, f=c^T V c - 1):")
print("V =\n", V)
print("u =", u)
print("f =", f)
print("center c (from foci) =", c_direct)
print("a (from foci,sum) =", a_direct, "  b =", b_direct)
print()
# ------ Recover via theory from (V, u, f) ------
c, f0, lam, P, a, b = recover_from_Vuf(V, u, f)

print("Recovered from (V,u,f) via theory:")
print("c = -V^{-1}u =", c)
print("f0 = u^T V^{-1} u - f =", f0)
print("Eigenvalues lam =", lam)
print("Eigenvectors P =\n", P)
print("Semi-axes from f0/lam: a =", a, " b =", b)
print()
```

Code - Python only

```
    # ------ Plot using principal-axis parametric form + transform
        ------
    pts = sample_ellipse_points(c, P, a, b, num=600)

    plt.plot(pts[:, 0], pts[:, 1], label="Ellipse")
    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.legend(loc="upper left", bbox_to_anchor=(1.05, 1.0))
    plt.grid(True)
    plt.title("Ellipse from (V,u,f) - theory-consistent build")
    plt.tight_layout()
    plt.savefig("newellipse.png")
    plt.show()

if __name__ == "__main__":
    main()
```