1. convex functions

$$f(x) = \inf \{ \|z\|_p \mid Az = x \} \quad \text{dom} f = \mathbb{R}^n$$

$$A \in \mathbb{R}^{n \times m}, \; p \geq 1$$

ref. Ex. 3.17 pg. 84

Let $h(z) = \|z\|_p$ for $p \geq 1$ which is convex by definition of a norm. Then the function f defined above is also convex. To see this, we define f by

$$f(x,z) = \begin{cases} h(z) & \text{if } Az = x \\ \infty & \text{otherwise,} \end{cases}$$

which is convex in (x,z). Then f is the minimum of h over z, and hence is convex.

(b) Is the function $f(x) = \dfrac{\|Ax+b\|_2^2}{c^T x + d}$ where

$\text{dom} f = \{ x \mid c^T x + d > 0 \}$ quasiconvex? Is it convex?

<u>def</u> $f: \mathbb{R}^n \to \mathbb{R}$ is quasiconvex if $\text{dom} f$ is convex and the sublevel sets

$$S_\alpha = \{ x \in \text{dom} f \mid f(x) \leq \alpha \}$$

are convex for all α.

- We have $\text{dom} f = \{ x \mid c^T x + d > 0 \}$ which is clearly convex.

- $f(x) = \frac{\|Ax+b\|_2^2}{c^Tx+d}$

$$S_\alpha = \left\{ x \mid \frac{\|Ax+b\|_2^2}{c^Tx+d} \leq \alpha, \; c^Tx+d > 0 \right\}$$

$$\Rightarrow \|Ax+b\|_2^2 \leq \alpha (c^Tx+d)$$

$$\Rightarrow \|Ax+b\|_2^2 - \alpha (c^Tx+d) \leq 0$$

→ $\|Ax+b\|_2^2$ convex quadratic

→ halfspace, convex constraint

Therefore, $f(x)$ is quasiconvex.

convex? yes.

The function is the composition of the function $g(y,t) = y^Ty/t$ with an affine transformation $(y,t) = (Ax+b, c^Tx+d)$. Therefore convexity of f follows from the fact that g is convex on $\{(y,t) \mid t \leq 0\}$.

2. Duality. Consider the optimization problem

$$\begin{aligned} &\text{minimize} && \log\det X^{-1} \\ &\text{s.t.} && F_i^T X F_i \preceq G_i, \quad i = 1, \dots, m, \end{aligned}$$

with variable $X \in S^n$ and problem data $F_i \in \mathbb{R}^{n \times k_i}$, $G_i \in S^{k_i}_{++}$, $i = 1, \dots, m$. The constraint $X \succ 0$ is implicit.

(a) Derive the Lagrange dual and dual function. Assume that $\sum_{i=1}^m F_i F_i^T \succ 0$.

This is a determinant maximization problem with linear matrix inequalities (LMIs). Lets define a dual variable (standard when taking duals of SDPs) as $Z_i \in S^{F_i}$ and let $Z_i \succeq 0$. Then,

$$L(X, Z_1, \dots, Z_m) = -\log\det X + \sum_{i=1}^m \mathrm{Tr}((F_i^T X F_i - G_i) Z_i).$$

So the dual function is

$$g(Z_1, \dots, Z_m) = \inf_X \{ L(X, Z_1, \dots, Z_m) \}$$

$$= \inf_X \left\{ -\log\det X + \sum_{i=1}^m \mathrm{Tr}((F_i^T X F_i - G_i) Z_i) \right\}$$

$$= \inf_X \left\{ -\log\det X + \sum_{i=1}^m \mathrm{Tr}(F_i^T X F_i) Z - \mathrm{Tr}(G_i Z) \right\}$$

take the derivative with respect to X since X minimizes the Lagrangian iff the gradient of the Lagrangian with respect to X is zero,

$$\nabla_X L = -X^{-1} + \sum_{i=1}^{m} F_i^T Z F_i = 0.$$

This only has a solution if

$$\sum_{i=1}^{m} F_i^T Z F_i > 0$$

otherwise the Lagrangian is unbounded. Therefore the optimal value is

$$X^* = \left(\sum_{i=1}^{m} F_i^T Z F_i\right)^{-1}.$$

So plugging into the dual function we have

$$g(z_1, \ldots, z_m) = L(X^*, z_1, \ldots, z_m)$$

$$= -\log\det(X^*)^{-1} + \sum_{i=1}^{m} \mathrm{Tr}(F_i^T X^* F_i - G_i) z_i$$

$$= -\log\det\left(\sum_{i=1}^{m} F_i z_i F_i^T\right)^{-1} + \sum_{i=1}^{m} \mathrm{Tr}\left(F_i^T \left(\sum_{i=1}^{m} F_i z_i F_i^T\right) F_i - G_i\right) z_i$$

$$= \log\det\left(\sum_{i=1}^{m} F_i z F_i^T\right)^{-1} - \sum_{i=1}^{m} \mathrm{Tr}(G_i z_i) + \mathrm{Tr}\left(\sum_{i=1}^{m} F_i z_i F_i^T\right)\left(\sum_{i=1}^{m} F_i z_i F_i^T\right)^{-1}$$

pg. 92
Ex. 3.23

$f = \log\det X^{-1} \Rightarrow f^*(Y) = \log\det(-Y)^{-1} - n$

$$= \log\det\left(\sum_{i=1}^{m} F_i Z F_i^T\right) - \sum_{i=1}^{m} \mathrm{Tr}(G_i Z_i) + n \; .$$

So the dual problem is then (to help define variables / constraints)

$$\text{maximize} \quad \log\det\left(\sum_{i=1}^{m} F_i Z_i F_i^T\right) - \sum_{i=1}^{m} \mathrm{Tr}(G_i Z_i)$$

$$\text{s.t.} \quad Z_i \succeq 0, \; i = 1, \ldots, m$$
$$\sum_{i=1}^{m} F_i Z_i F_i \succ 0$$

where $Z_i \in S^{k_i}$ and the domain would be all Z_i such that the constraints hold (or else its unbounded).

(b) What can we say about the duality gap?

Slater's constraint qualification ie strict feasibility:

$$\exists X \in \mathrm{int}\, D : \; f_i(X) < 0, \; i = 1, \ldots m \quad AX = b$$

For our problem finding and arbitrarily large $\varepsilon > 0$ we can see that $X = \varepsilon I$ is strictly feasible for the primal problem. Hence by Slater's condition, strong duality will always hold.

3. (a) Derive the conjugate function f^*.

$$f(z) = \frac{1}{4}\|z\|_2^4$$

$$f^*(y) = \sup_x \left(x^T y - \frac{1}{4}\|x\|_2^4 \right)$$

From Holders inequality

$$x^T y \leq \|x\| \|y\|_* \text{ where } \|y\|_* \text{ is the dual norm.}$$

$$\Rightarrow \sup_x \left(x^T y - \frac{1}{4}\|x\|_2^4 \right) \leq \sup_x \left(\|x\| \|y\|_* - \frac{1}{4}\|x\|_2^4 \right)$$

$$= \sup_{\|x\|} \left(\|x\| \|y\|_* - \frac{1}{4}\|x\|_2^4 \right)$$ $\|x\|_*$ of L2 is $\|x\|_2$

$$= \sup_{\|x\|} \left(\|x\|_2 \|y\|_2 - \frac{1}{4}\|x\|_2^4 \right)$$

hence we get $\frac{3}{4}\|x\|_2^{4/3}$.

b) Prove that $\hat{x}(\beta) = x + \nabla f^*(\beta)$

$$\hat{x}(\beta) = \underset{x' \in \mathbb{R}^D}{\operatorname{argmax}} \; \beta^T x' - f(x' - x)$$

c) Show that the function $\ell(\beta) = \log(1 + \exp(\beta^T \hat{x}(\beta)))$ is convex in β.

We know $\hat{x}(\beta) = x + \nabla f^*(\beta)$

$$\Rightarrow \log(1 + \exp(\beta^T(x + \nabla f^*(x))$$

$$= \log(1 + \exp(\beta^T x + \beta^T \nabla f^*(x))$$

$$= \log(1 + \exp(\beta^T x + \beta^T (\tfrac{3}{4} \|\beta\|_2^{4/3})$$

define $h(\beta) = \beta^T x + \beta^T \left(\frac{3}{4} \|\beta\|_2^{4/3}\right)$

$g(t) = \log(1 + \exp(t))$

g is convex by log-sum-exp with $1 = e^0$.

$h(\beta)$ is convex.

Therefore by the log-sum-exp composition we showed $\ell(\beta)$ is convex in β.

4. Given maximize $Tr(G)$

Subject to $1^T G 1 = 0$

$G = G^T \succeq 0$

$G_{ii} + G_{jj} - 2G_{ij} = d_{ij} \quad \{i,j\} \in E$

(a) Derive the dual of the convex problem above. For convenience, the last set of equality constraints can be written as

$$G_{ii} + G_{jj} - 2G_{ij} = Tr(GI)^{\{i,j\}} = d_{ij} \quad \{i,j\} \in E$$

where $I^{\{i,j\}}$ has all zero entries except for

$I_{ii}^{\{i,j\}} = I_{jj}^{\{i,j\}} = 1$, $I_{ij}^{\{i,j\}} = I_{ji}^{\{i,j\}} = -1$.

Lets write of the Lagrangian

$$L(G,Z) = Tr(G) + Tr((1^T G 1)Z) + Tr(GZ) + Tr(Tr(GI^{\{i,j\}} - d_{ij})Z)$$

the Lagrangian is unbounded unless $Tr(GI^{\{i,j\}}) = d_{ij}$

so the dual SDP is

maximize $Tr(GZ)$

s.t. $Tr(GI^{\{i,j\}}) = d_{ij}$

$1^T G 1 = 0, \quad G \succeq 0.$

(b) In the graph G we can assign weights $W_{ij} = W_{ji} > 0$ to each edge $\{i,j\} \in E$, then the weighted Laplacian $L \in S_+^n$ of the graph G is defined as

$$L_{ij} = \begin{cases} -W_{ij} & \text{if } i \neq j,\ \{i,j\} \in E \\ \sum W_{ik} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

Note that $L = \sum_{ij \in E} W_{ij} I^{\{ij\}}$. Let $\lambda_1, \ldots, \lambda_n$ be the eigenvalues of L where $\lambda_1 \geq \ldots \geq \lambda_n$. Using the fact that $L \geq 0$ show that

i) $\lambda_n(L) = 0$ and $\lambda_{n-1}(L)$ is concave in W

Since $L \geq 0$ we know the largest eigenvalue is the largest absolute row sum of column sum and in this case would be zero. It directly follows that the second largest eigenvalue of the Laplacian is then concave in W.

ii) The second largest eigenvalue is ≥ 1 iff $L \geq I - (1/n)11^T$

c) We want to find the optimal transition rates W_{ij}, $\{i,j\} \in E$ that give the fastest convergence for the Markov chain. Formulate a convex optimization problem.

objective: $\lambda_{n-1}(L)$, the second largest eigenvalue determines the rate of convergence. The smaller the second eigenvalue is the faster the Markov chain converges. Define a function to find the second largest eigenvalue as

$$\gamma(L) = \max_{i=2,\dots,n} |\lambda_i(L)|$$

and define the following given constraints

$$W_{ij} \geq 0$$

$$\sum_{\{i,j\} \in E} c_{ij} W_{ij} \leq c$$

then we can write our optimization problem as

$$\text{minimize } \gamma(L)$$

$$\text{s.t. } W_{ij} \geq 0$$

$$\sum_{\{ij\} \in E} C_{ij} W_{ij} \leq c$$

$$L = L^T \succeq 0$$

$$L\mathbf{1}^T = \mathbf{1}$$

$$L_{ij} = 0 \quad \{ij\} \notin E$$

where C_{ij} a given cost matrix and $c \geq 0$ is a constant.

(d)

5. Estimating a distance function from data. We are given (x_i, y_i), $i = 1, \ldots, m$ where $x_i \in \mathbb{R}^n$ and $y_i \in \mathbb{R}^n$, and a set of m corresponding (approximate) pairwise distances $d_1, \ldots, d_m$. We aim to estimate a distance function

$$d(x, y) = \sqrt{(x-y)^T A (x-y)}$$

so that the squared error objective $\sum_{i=1}^{m} (d(x_i, y_i) - d_i)^2$ is minimized.

(a) Express this as a convex optimization problem.

$$d(x, y) = \sqrt{(x-y)^T A (x-y)} = \|x - y\|_A$$

We want to minimize the squared error objective

$$\sum_{i=1}^{m} (z - d_i)^2, \text{ where } z = \sqrt{(x-y)^T A (x-y)}$$

but this objective is not DCP. Therefore we must reformulate it by realizing that we essentially want to minimize $\|x-y\|_A$ indexed over the x and y that are close in distance by some arbitrary ε. Then we can constrain $\|x-y\|_A$ to keep x and y that are far in distance above some threshold. We can write these as indexed sets as

$$C = \{(x_i, y_i) \mid x_i \text{ and } y_i \text{ are close (Euclidean distance)}\}$$

$$F = \{(x_i, y_i) \mid x_i \text{ and } y_i \text{ are far (Euclidean distance)}\}$$

then the optimization formulation is

$$\min_{A} \sum_{x_i, y_i \in C} \|x - y\|_A$$

$$\text{s.t.} \sum_{x_i, y_i \in F} \|x - y\|_A \geq 1.$$

$$A \succeq 0.$$

Another formulation for that is better for implementation may be epigraph form where we define an auxilary variable t and write this as

$$\sum_{i=1}^{m} (d(x_i, y_i) - d_i))^2 \leq t_i$$

$$= \sum_{i=1}^{m} (\sqrt{(x-y)^T A (x-y)} - d_i))^2 \leq t_i$$

$$= -\sqrt{t_i} \leq \sqrt{(x-y)^T A (x-y)} - d_i \leq +\sqrt{t_i}$$

$$= -\sqrt{t_i} - d_i \leq \sqrt{(x-y)^T A (x-y)} \leq \sqrt{t_i} + d_i$$

[42]:
```
# Given Data

import numpy as np
import cvxpy as cp
from scipy import linalg

np. random.seed (0)

n = 5 # dimension
N = 100 # number of distance samples
N_test = 10

X = np. random.normal(size =(n, N))
Y = np. random.normal(size =(n, N))

P = np.random.normal (size =(n, n))
P = P @ P.T + np.eye(n)
sqrtP = linalg.sqrtm (P)

d = np.linalg.norm (sqrtP @ (X - Y), axis =0) # exact distances
d = np.maximum(np.zeros (len(d)), d+np.random.normal( size =(1 , N))) # add␣
 ↪noise and make nonnegative

# create sets to determine if X and Y are similar to each other
distances = np.linalg.norm(X - Y, axis=0)

# for similarity
threshold = 1.0

similar_set = np.where(distances <= threshold)[0]
not_similar_set = np.where(distances > threshold)[0]
```

[47]:
```
import numpy as np
import cvxpy as cp

# problem 5

# Given data
n = 5  # dimension
N = 100  # number of distance samples
N_test = 10

X = np.random.normal(size=(n, N))
Y = np.random.normal(size=(n, N))

# Constructing the distance matrix
P = np.random.normal(size=(n, n))
```

```
P = P @ P.T + np.eye(n)
sqrtP = np.linalg.cholesky(P)
distances = np.linalg.norm(sqrtP @ (X - Y), axis=0)
d_true = np.maximum(np.zeros(len(distances)), distances + np.random.
 ↪normal(size=(1, N)))

# Sets for similarity and dissimilarity
similar_set = np.where(distances <= threshold)[0]
not_similar_set = np.where(distances > threshold)[0]

# Convex optimization problem
A = cp.Variable((n, n), symmetric=True)  # Positive semi-definite matrix
d_predicted = M @ (X - Y)


squared_error = cp.sum_squares(d_predicted - d_true)

prob = cp.Problem(cp.Minimize(squared_error))
prob.solve()


if prob.status == cp.OPTIMAL:
    # Optimal matrix M
    optimal_A = A.value
    print("Optimal A:\n", optimal_M)
else:
    print("Problem was not solved successfully.")
```

```
Optimal A:
 [[ 0.28332619  0.10190303  0.2609798   0.08383585 -0.08832993]
 [ 0.10190303 -0.06346669  0.07605456 -0.10060897 -0.23740941]
 [ 0.2609798   0.07605456  0.23565457  0.05064444 -0.11861157]
 [ 0.08383585 -0.10060897  0.05064444 -0.14886147 -0.29827173]
 [-0.08832993 -0.23740941 -0.11861157 -0.29827173 -0.39423258]]
```

[ ]:

6. Search and rescue. Tasked with developing a controller for the motion of the drone in the x,y plane. The kinematics are described by

$$p_{t+1} = p_t + \Delta t\, v_t$$
$$v_{t+1} = v_t + \Delta t\, a_t$$

where $p_k \in \mathbb{R}^2$ and $v_k \in \mathbb{R}^2$ are the drones position and velocity at time k, Δt is the time step between states, and $a_k \in \mathbb{R}^2$ is the acceleration at time k. The controller generates acceleration.

$$p^{ref} = \begin{bmatrix} 0.5\sin(\pi t) \\ 0.5\sin(\pi t)\cos(\pi t) \end{bmatrix}$$

and $p_0 = [0\ 0]^T$ with zero velocity. Mission should be completed in 20 seconds, minimize battery usage as the sum of 2-norm of the acceleration. Magnitudes of x and y components of acceleration must be less than 4 m/s^2.

Two metrics: tracking error $\left(\sum_{i=1}^{T} \|p_i - p_i^{ref}\|_2\right)$

acceleration: $\left(\sum_{i=1}^{T} \|a_i\|_2\right)$.

(a) Express the problem as a multi-objective convex optimization problem.

The objectives were given explicitly.

The constraints are:

drone dynamics: $p_{t+1} = p_t + \Delta t v_t$
$v_{t+1} = v_t + \Delta t a_t$

acceleration magnitude constraint: $\|a_t\| \leq 4$

initial conditions: $p_0 = v_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

final condition: $p_T = v_T = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$.

we can write out the multi-objective optimization problem as

$$\text{minimize} \quad \left(\sum_{i=1}^{T} \|p_i - p_i^{ref}\|_2 , \sum_{i=1}^{T} \|a_i\|_2 \right)$$

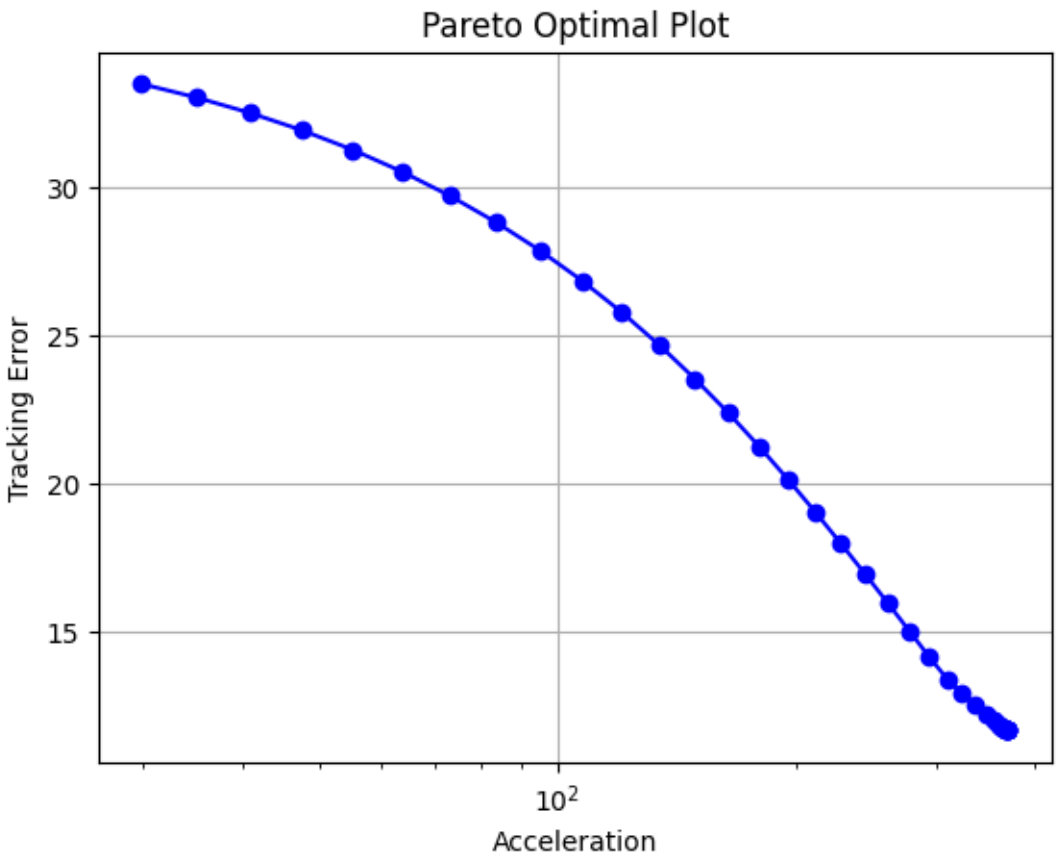
$$\text{subject to} \quad p_{t+1} = p_t + \Delta t v_t$$
$$v_{t+1} = v_t + \Delta t a_t \quad \text{for } t = 1, \dots, T$$
$$\|a_t\|_2 \leq 4$$
$$p_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, p_T = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$
$$v_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, v_T = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

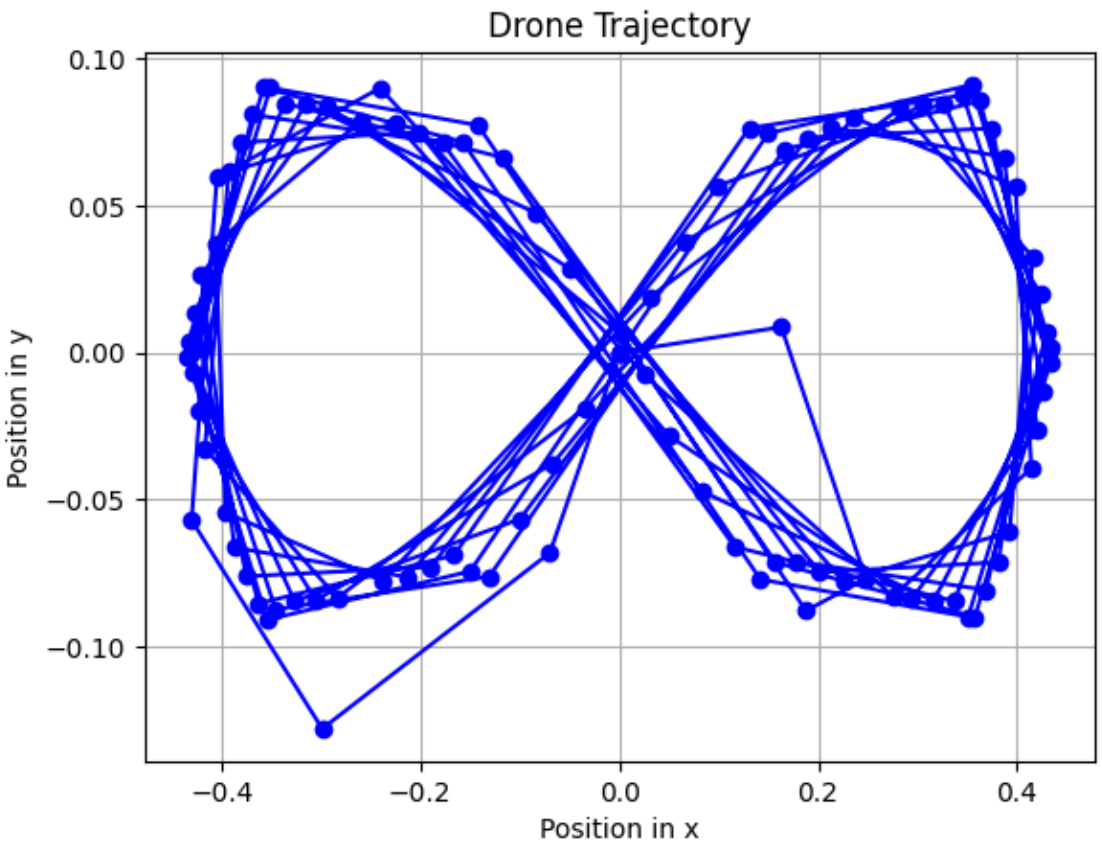
(b) code

[7]:
```
# problem 6

import numpy as np
import cvxpy as cp

K = 100
t = np.linspace(0,20,num=K)
dt = t[1] - t[0]
```

[8]:
```
import numpy as np
import cvxpy as cp
import matplotlib.pyplot as plt

K = 100
t = np.linspace(0, 20, num=K)
dt = t[1] - t[0]

pref_x = 0.5 * np.sin(np.pi * t)
pref_y = 0.5 * np.sin(np.pi * t) * np.cos(np.pi * t)

pref = np.vstack([pref_x, pref_y])

lambda_values = np.logspace(-4, -1, 40)

tracking_errors = np.zeros(len(lambda_values))
accelerations = np.zeros(len(lambda_values))

for i, lmbda in enumerate(lambda_values):
    p = cp.Variable((2, K + 1))
    v = cp.Variable((2, K + 1))
    a = cp.Variable((2, K))

    objective = cp.Minimize(cp.sum_squares(p[:, :-1] - pref) + lmbda * cp.
 ↪sum_squares(a))
    constraints = [
        p[:, 1:] == p[:, :-1] + dt * v[:, :-1],
        v[:, 1:] == v[:, :-1] + dt * a,
        cp.norm(a, 2, axis=0) <= 4,
        p[:, 0] == np.zeros(2),
        v[:, 0] == np.zeros(2),
        p[:, -1] == np.zeros(2),
        v[:, -1] == np.zeros(2)
    ]

    problem = cp.Problem(objective, constraints)
    problem.solve()
```

```
    tracking_errors[i] = np.sum(np.linalg.norm(p.value[:, :-1] - pref, axis=0))
    accelerations[i] = np.sum(np.linalg.norm(a.value, axis=0))

# Pareto optimal plot
plt.plot(accelerations, tracking_errors, marker='o', linestyle='-', color='b')
plt.xlabel('Acceleration')
plt.ylabel('Tracking Error')
plt.title('Pareto Optimal Plot')
plt.xscale('log')
plt.grid(True)
plt.show()

# Select an appropriate value for lambda
selected_lambda = lambda_values[np.argmin(tracking_errors + lambda_values *␣
 ↪accelerations)]

# generate plot for xy position for the selected lambda
p = cp.Variable((2, K + 1))
v = cp.Variable((2, K + 1))
a = cp.Variable((2, K))

objective = cp.Minimize(cp.sum_squares(p[:, :-1] - pref) + selected_lambda * cp.
 ↪sum_squares(a))
constraints = [
    p[:, 1:] == p[:, :-1] + dt * v[:, :-1],
    v[:, 1:] == v[:, :-1] + dt * a,
    cp.norm(a, 2, axis=0) <= 4,
    p[:, 0] == np.zeros(2),
    v[:, 0] == np.zeros(2),
    p[:, -1] == np.zeros(2),
    v[:, -1] == np.zeros(2)
]

problem = cp.Problem(objective, constraints)
problem.solve()

# plot xy position for the selected lambda
plt.plot(p.value[0, :], p.value[1, :], marker='o', linestyle='-', color='b')
plt.xlabel('Position in x')
plt.ylabel('Position in y')
plt.title('Drone Trajectory')
plt.grid(True)
plt.show()
```


Pareto Optimal Plot
Tracking Error
30
25
20
15
10^2
Acceleration


Drone Trajectory
0.10
0.05
0.00
−0.05
−0.10
Position in y
−0.4
−0.2
0.0
0.2
0.4
Position in x

[ ]: