EN.553.662: Optimization for Data Science
Homework 1: Gradient Descent

Ha Manh Bui (CS Department)
hbui13@jhu.edu

Spring 2023

1 Problem 1

Define points $z_1, ..., z_4 \in \mathbb{R}^2$ by

$$z_1 = \begin{pmatrix} -1 \\ 0 \end{pmatrix}, z_2 = \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix}, z_3 = \begin{pmatrix} 0.25 \\ -0.5 \end{pmatrix}, z_4 = \begin{pmatrix} -0.5 \\ -0.25 \end{pmatrix}$$

and let

$$F(x) = \sum_{j=1}^{4} \left(1 - e^{-5|x-z_j|^2}\right).$$

1 *Give the expressions of ∇F and $\nabla^2 F$ as function of x.*
We have

$$\begin{aligned} dF(x)h &= \partial_t F(x+th)|_{t=0} \\ &= 10 \sum_{j=1}^{4} e^{-5|x-z_j|^2} \left(x^\top h - z_j^\top h\right). \end{aligned}$$

Since $\langle \nabla F(x), h \rangle = dF(x)h$, we obtain

$$\nabla F(x) = 10 \sum_{j=1}^{4} e^{-5|x-z_j|^2} (x - z_j).$$

We have

$$\begin{aligned} d^2F(x)(h,k) &= \partial_t (dF(x+tk)h)|_{t=0} \\ &= 100 \sum_{j=1}^{4} e^{-5|x-z_j|^2} h^\top \left(x z_j^\top - x x^\top - z_j z_j^\top + z_j x_t^\top + \frac{1}{10} I_2\right) k. \end{aligned}$$

Since $h^\top \nabla^2 F(x) k = d^2F(x)(h,k)$, we obtain

$$\nabla^2 F(x) = 100 \sum_{j=1}^{4} e^{-5|x-z_j|^2} \left(x z_j^\top - x x^\top - z_j z_j^\top + z_j x_t^\top + \frac{1}{10} I_2\right). \tag{1}$$

2 *Compute (numerically) $\nabla^2 F(x)$ at $x = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ and prove that F is not a convex function.*
Replace

$$x = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, z_1 = \begin{pmatrix} -1 \\ 0 \end{pmatrix}, z_2 = \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix}, z_3 = \begin{pmatrix} 0.25 \\ -0.5 \end{pmatrix}, z_4 = \begin{pmatrix} -0.5 \\ -0.25 \end{pmatrix}$$

to Equation 1, we obtain the following Hessian matrix:

$$\nabla^2 F\left(\begin{pmatrix}0\\0\end{pmatrix}\right) = \begin{pmatrix}-4.1958 & -2.0521\\ -2.0521 & -3.5220\end{pmatrix}. \tag{2}$$

F is not a convex function.

Proof. We have F is convex if and only if $\nabla^2 F(x) \succeq 0, \forall x$. However, Equation 2 shows $\nabla^2 F(x)$ is not positive semi-definite at $x = \begin{pmatrix}0\\0\end{pmatrix}$. Therefore, we obtain F is not a convex function. $\square$

3 *Provide a surface plot of the function F over $\Omega = [-1,1]^2$ (use the python code "surf plot example.py" as an example).*

```
import numpy as np
from matplotlib import pyplot as plt
import torch
import matplotlib.cm as cm

def func(x):
    zs = [torch.tensor([-1, 0], dtype=torch.float64), torch.tensor([0.5, 0.5], dtype=torch.float64),
        torch.tensor([0.25, -0.5], dtype=torch.float64), torch.tensor([-0.5, -0.25], dtype=torch.float64)]
    out = 0
    for z in zs:
        out += (1 - torch.exp(-5 * torch.square(torch.norm((x - z).float()))))
    return out

u, v = np.meshgrid(np.linspace(-1, 1, 100), np.linspace(-1, 1, 100))

f = []
for i in range(100):
    f.append([])
    for j in range(100):
        f[i].append(func(torch.Tensor([u[i][j], v[i][j]])))
f = torch.Tensor(f)

fig, ax = plt.subplots(subplot_kw={"projection": "3d"}, constrained_layout=True)
surf = ax.plot_surface(u, v, f, cmap=cm.coolwarm, linewidth=0, antialiased=False)
plt.savefig("1.3.pdf")
```

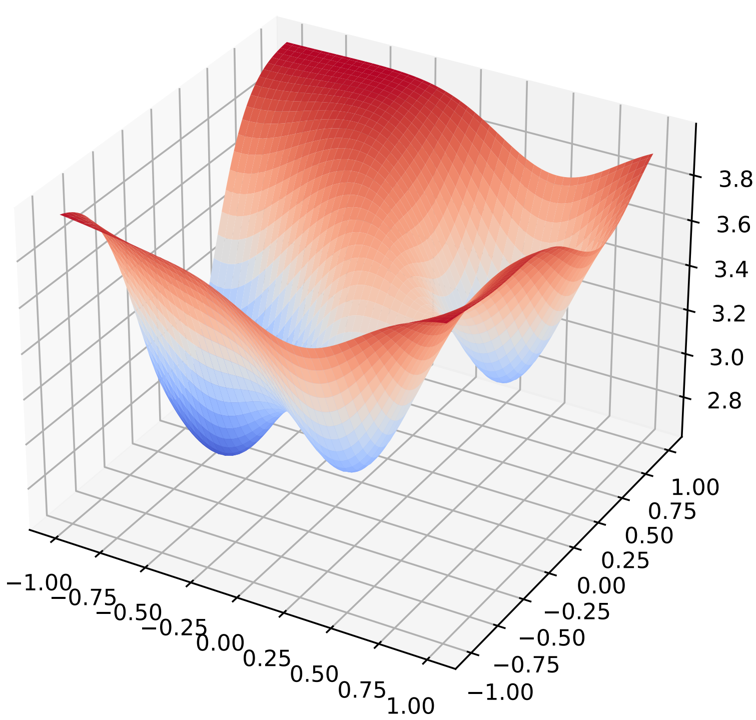

Figure 1: Surface plot of function F over $\Omega = [-1,1]^2$.

4 *Take* $x = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$. *Fix* $c_1 = 0.01$ *and* $c_2 = 0.9$. *Provide the following plots:*

- *The function* $t \mapsto F(x - t\nabla F(x))$ *discretized over* $t \in [-1, 1]$ *using two colors indicating whether* t *satisfies the condition*
$$F(x - t\nabla F(x)) \leq F(x) - c_1 t \left|\nabla F(x)\right|^2$$
or not.

- *The function* $t \mapsto F(x - t\nabla F(x))$ *discretized over* $t \in [-1, 1]$ *using two colors indicating whether* t *satisfies the condition*
$$F(x - t\nabla F(x)) \leq F(x) - c_1 t \left|\nabla F(x)\right|^2, \nabla F(x - t\nabla F(x))^\top \nabla F(x) \leq c_2 \left|\nabla F(x)\right|^2$$
or not.

- *The function* $t \mapsto F(x - t\nabla F(x))$ *discretized over* $t \in [-1, 1]$ *using two colors indicating whether* t *satisfies the condition*
$$F(x - t\nabla F(x)) \leq F(x) - c_1 t \left|\nabla F(x)\right|^2, \left|\nabla F(x - t\nabla F(x))\right| \leq c_2 \left|\nabla F(x)\right|^2$$
or not.

(Use at least 200 points in the discretization of $[-1, 1]$*.)*

```
def plot1(x, list_t):
    x = x.detach().requires_grad_()
    y = func(x)
    y.backward()
    t1, t2, f1, f2 = [], [], [], []
    for i in range(list_t[0].shape[0]):
        t = torch.Tensor([list_t[0][i]])
        low_bound = func(x - t * x.grad)
        up_bound = y - 0.01 * t * torch.square(torch.norm(x.grad))
        if low_bound <= up_bound:
            f1.append(low_bound)
            t1.append(t)
        else:
            f2.append(low_bound)
            t2.append(t)
    return torch.Tensor(f1), torch.Tensor(t1), torch.Tensor(f2), torch.Tensor(t2)

def plot2(x, list_t):
    x = x.detach().requires_grad_()
    y = func(x)
    y.backward()
    t1, t2, f1, f2 = [], [], [], []
    for i in range(list_t[0].shape[0]):
        t = torch.Tensor([list_t[0][i]])
        low_bound = func(x - t * x.grad)
        up_bound = y - 0.01 * t * torch.square(torch.norm(x.grad))

        x_2 = x - t * x.grad
        x_2 = torch.tensor([x_2[0], x_2[1]], requires_grad = True, dtype=torch.float64)
        y_2 = func(x_2)
        y_2.backward()

        low_bound_2 = torch.dot(x_2.grad, x.grad)
        up_bound_2 = 0.9 * torch.square(torch.norm(x.grad))
        if low_bound <= up_bound and low_bound_2 <= up_bound_2:
            f1.append(low_bound)
            t1.append(t)
        else:
            f2.append(low_bound)
            t2.append(t)
    return torch.Tensor(f1), torch.Tensor(t1), torch.Tensor(f2), torch.Tensor(t2)
```

```python
def plot3(x, list_t):
    x = x.detach().requires_grad_()
    y = func(x)
    y.backward()
    t1, t2, f1, f2 = [], [], [], []
    for i in range(list_t[0].shape[0]):
        t = torch.Tensor([list_t[0][i]])
        low_bound = func(x - t * x.grad)
        up_bound = y - 0.01 * t * torch.square(torch.norm(x.grad))

        x_2 = x - t * x.grad
        x_2 = torch.tensor([x_2[0], x_2[1]], requires_grad = True, dtype=torch.float64)
        y_2 = func(x_2)
        y_2.backward()

        low_bound_2 = torch.norm(x_2.grad)
        up_bound_2 = 0.9 * torch.square(torch.norm(x.grad))
        if low_bound <= up_bound and low_bound_2 <= up_bound_2:
            f1.append(low_bound)
            t1.append(t)
        else:
            f2.append(low_bound)
            t2.append(t)
    return torch.Tensor(f1), torch.Tensor(t1), torch.Tensor(f2), torch.Tensor(t2)

if __name__ == "__main__":
    x = torch.tensor([0, 0], dtype=torch.float64)
    list_t = np.meshgrid(np.linspace(-1, 1, 1000))
    f1_plt1, t1_plt1, f2_plt1, t2_plt1 = plot1(x, list_t)
    f1_plt2, t1_plt2, f2_plt2, t2_plt2 = plot2(x, list_t)
    f1_plt3, t1_plt3, f2_plt3, t2_plt3 = plot3(x, list_t)

    fig, axs = plt.subplots(1, 3, figsize = (12, 4), constrained_layout=True)

    axs[0].plot(t1_plt1, f1_plt1, 'o')
    axs[0].plot(t2_plt1, f2_plt1, 'o')
    axs[0].set_xlabel(r'$t$')
    axs[0].set_ylabel(r'$F(x-t\nabla F(x))$')
    axs[0].set_title(r'$F(x-t\nabla F(x)) \leq F(x) - c_1 t \left|\nabla F(x)\right|^2$')

    axs[1].plot(t1_plt2, f1_plt2, 'o')
    axs[1].plot(t2_plt2, f2_plt2, 'o')
    axs[1].set_xlabel(r'$t$')
    axs[1].set_ylabel(r'$F(x-t\nabla F(x))$')
    axs[1].set_title(r'$F(x-t\nabla F(x)) \leq F(x) - c_1 t \left|\nabla F(x)\right|^2,$'
    "\n" r'$\nabla F(x-t\nabla F(x))^\top \nabla F(x) \leq c_2 \left|\nabla F(x)\right|^2$')

    axs[2].plot(t1_plt3, f1_plt3, 'o')
    axs[2].plot(t2_plt3, f2_plt3, 'o')
    axs[2].set_xlabel(r'$t$')
    axs[2].set_ylabel(r'$F(x-t\nabla F(x))$')
    axs[2].set_title(r'$F(x-t\nabla F(x)) \leq F(x) - c_1 t \left|\nabla F(x)\right|^2,$'
    "\n" r'$\left|\nabla F(x-t\nabla F(x))\right|\leq c_2\left|\nabla F(x)\right|^2$')

    plt.savefig("1.4.pdf")
```

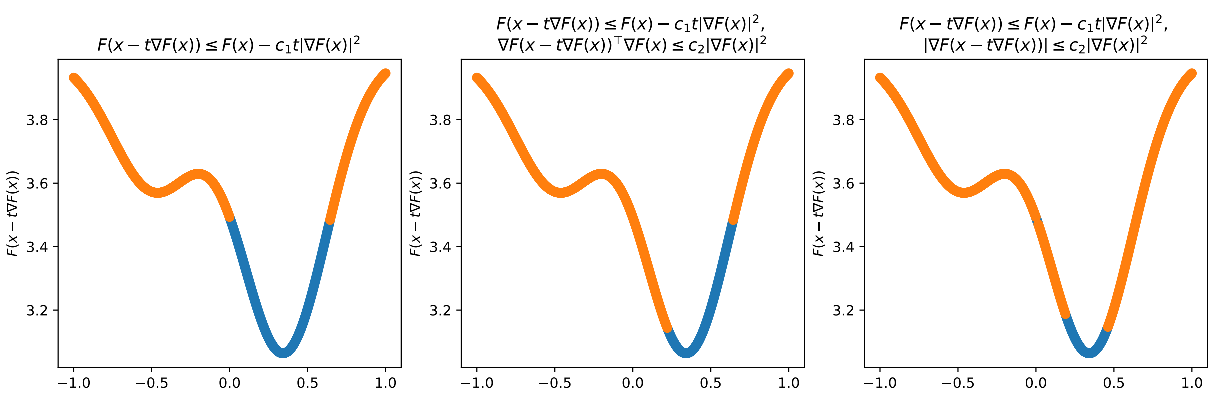

Figure 2: Visualization of the function $t \mapsto F(x - t\nabla F(x))$ discretized over $t \in [-1, 1]$ for 3 above conditions with 1000 points in the discretization of $[-1, 1]$ and $x = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$.

5 *Same question with* $x = \begin{pmatrix} -0.2 \\ 0.1 \end{pmatrix}$.

```
if __name__ == "__main__":
    x = torch.tensor([-0.2, 0.1], dtype=torch.float64)
    list_t = np.meshgrid(np.linspace(-1, 1, 1000))
    f1_plt1, t1_plt1, f2_plt1, t2_plt1 = plot1(x, list_t)
    f1_plt2, t1_plt2, f2_plt2, t2_plt2 = plot2(x, list_t)
    f1_plt3, t1_plt3, f2_plt3, t2_plt3 = plot3(x, list_t)

    fig, axs = plt.subplots(1, 3, figsize = (12, 4), constrained_layout=True)

    axs[0].plot(t1_plt1, f1_plt1, 'o')
    axs[0].plot(t2_plt1, f2_plt1, 'o')
    axs[0].set_xlabel(r'$t$')
    axs[0].set_ylabel(r'$F(x-t\nabla F(x))$')
    axs[0].set_title(r'$F(x-t\nabla F(x)) \leq F(x) - c_1 t \left|\nabla F(x)\right|^2$')

    axs[1].plot(t1_plt2, f1_plt2, 'o')
    axs[1].plot(t2_plt2, f2_plt2, 'o')
    axs[1].set_xlabel(r'$t$')
    axs[1].set_ylabel(r'$F(x-t\nabla F(x))$')
    axs[1].set_title(r'$F(x-t\nabla F(x)) \leq F(x) - c_1 t \left|\nabla F(x)\right|^2,$'
    "\n" r'$\nabla F(x-t\nabla F(x))^\top \nabla F(x) \leq c_2 \left|\nabla F(x)\right|^2$')

    axs[2].plot(t1_plt3, f1_plt3, 'o')
    axs[2].plot(t2_plt3, f2_plt3, 'o')
    axs[2].set_xlabel(r'$t$')
    axs[2].set_ylabel(r'$F(x-t\nabla F(x))$')
    axs[2].set_title(r'$F(x-t\nabla F(x)) \leq F(x) - c_1 t \left|\nabla F(x)\right|^2,$'
    "\n" r'$\left|\nabla F(x-t\nabla F(x))\right|\leq c_2\left|\nabla F(x)\right|^2$')

    plt.savefig("1.5.pdf")
```

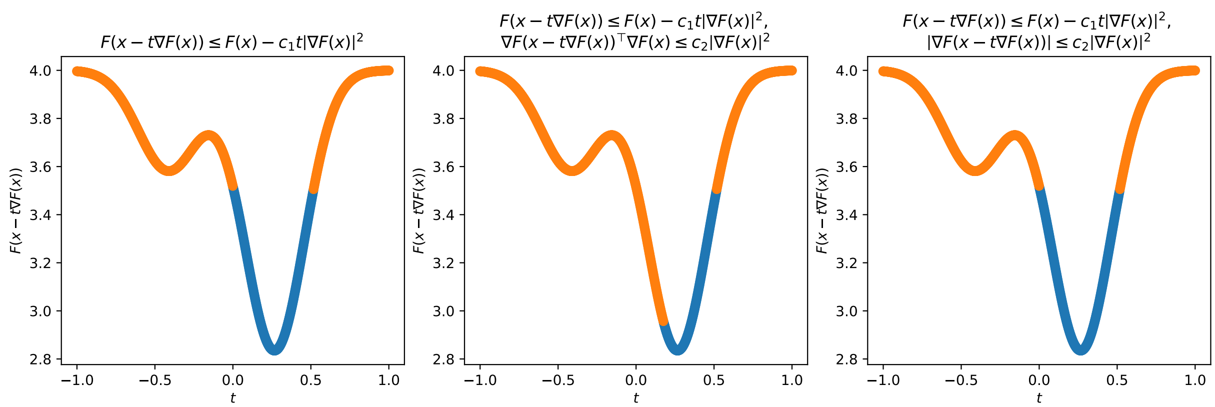

Figure 3: Visualization of the function $t \mapsto F(x - t\nabla F(x))$ discretized over $t \in [-1, 1]$ for 3 above conditions with 1000 points in the discretization of $[-1, 1]$ and $x = \begin{pmatrix} -0.2 \\ 0.1 \end{pmatrix}$.

2 Problem 2

1 *Define the function* $f : (\mathbb{R}^2)^3 \to \mathbb{R}$ *by*

$$f(x_1, x_2, x_3) = \det([x_2 - x_1, x_3 - x_1]).$$

Using the notation $\begin{pmatrix} u \\ v \end{pmatrix}^\perp = \begin{pmatrix} -v \\ u \end{pmatrix}$ *prove that*

$$df(x_1, x_2, x_3)(h_1, h_2, h_3) = h_1^\top (x_3 - x_2)^\perp + h_2^\top (x_1 - x_3)^\perp + h_3^\top (x_2 - x_1)^\perp$$

and

$$\nabla f(x_1, x_2, x_3) = \left[(x_3 - x_2)^\perp, (x_1 - x_3)^\perp, (x_2 - x_1)^\perp\right]$$

where elements $(h_1, h_2, h_3) \in (\mathbb{R}^2)^3$ *are identified with* 2×3 *matrices* $[h_1, h_2, h_3]$.

Proof. We have

$$\begin{aligned}
df(x_1,x_2,x_3)(h_1,h_2,h_3) &= \partial_t f(x_1+th_1,x_2+th_2,x_3+th_3)|_{t=0}\\
&= \partial_t det\left([x_2+th_2-x_1-th_1, x_3+th_3-x_1-th_1]\right)|_{t=0}\\
&= tr(adj([x_2+th_2-x_1-th_1, x_3+th_3-x_1-th_1][h_2-h_1,h_3-h_1]))|_{t=0}\\
&= tr(adj([x_2-x_1,x_3-x_1][h_2-h_1,h_3-h_1])).
\end{aligned}$$

Let $x_i = \begin{pmatrix} x_{i,1}\\ x_{i,2}\end{pmatrix} \in \mathbb{R}^2$, we obtain

$$\begin{aligned}
df(x_1,x_2,x_3)(h_1,h_2,h_3) &= tr(adj([x_2-x_1,x_3-x_1][h_2-h_1,h_3-h_1]))\\
&= (x_{3,2}-x_{1,2})(h_{2,1}-h_{1,1}) - (x_{3,1}-x_{1,1})(h_{2,2}-h_{1,2})\\
&- (x_{2,2}-x_{1,2})(h_{3,1}-h_{1,1}) + (x_{2,1}-x_{1,1})(h_{3,2}-h_{1,2})\\
&= h_{1,1}(-x_{3,2}+x_{2,2}) + h_{1,2}(x_{3,1}-x_{2,1}) + h_{2,1}(x_{3,2}-x_{1,2}) + h_{2,2}(x_{3,1}+x_{1,1})\\
&+ h_{3,1}(-x_{2,2}+x_{1,2}) + h_{3,2}(x_{2,1}-x_{1,1}\\
&= h_1^\top(x_3-x_2)^\perp + h_2^\top(x_1-x_3)^\perp + h_3^\top(x_2-x_1)^\perp \text{ with } \begin{pmatrix} u\\ v\end{pmatrix}^\perp = \begin{pmatrix} -v\\ u\end{pmatrix}.
\end{aligned}$$

Since $df(x_1,x_2,x_3)(h_1,h_2,h_3) = \langle [h_1,h_2,h_3], \nabla f(x_1,x_2,x_3)\rangle$, we obtain

$$\nabla f(x_1,x_2,x_3) = \left[(x_3-x_2)^\perp, (x_1-x_3)^\perp, (x_2-x_1)^\perp\right].$$

□

2 *Let* $\Omega_0 = \left\{[x_1,x_2,x_3] \in (\mathbb{R}^2)^3 : f(x_1,x_2,x_3) > 0\right\}$. *Define, on* Ω_0

$$g(x_1,x_2,x_3) = (\log f(x_1,x_2,x_3) - c)^2$$

where c *is a positive number. Compute* $\nabla g(x_1,x_2,x_3)$ *for* $(x_1,x_2,x_3) \in \Omega_0$.
We have

$$g(x_1,x_2,x_3) = (\log f(x_1,x_2,x_3))^2 - 2c\log f(x_1,x_2,x_3) + c^2.$$

Compute gradient, we obtain

$$\begin{aligned}
\nabla g(x_1,x_2,x_3) &= 2(\log f(x_1,x_2,x_3))\nabla \log f(x_1,x_2,x_3) - 2c\frac{\nabla f(x_1,x_2,x_3)}{f(x_1,x_2,x_3)\ln 10}\\
&= 2(\log f(x_1,x_2,x_3))\frac{\nabla f(x_1,x_2,x_3)}{f(x_1,x_2,x_3)\ln 10} - 2c\frac{\nabla f(x_1,x_2,x_3)}{f(x_1,x_2,x_3)\ln 10}\\
&= \frac{2(\log f(x_1,x_2,x_3)) - 2c}{f(x_1,x_2,x_3)\ln 10}\nabla f(x_1,x_2,x_3)\\
&= \frac{2}{\ln 10}\frac{\log\left(\det\left([x_2-x_1,x_3-x_1]\right)\right) - c}{\det\left([x_2-x_1,x_3-x_1]\right)}\left[(x_3-x_2)^\perp, (x_1-x_3)^\perp, (x_2-x_1)^\perp\right]. \qquad (3)
\end{aligned}$$

3 *Let* $\mathcal{F} = \{(i_1,j_1,k_1),\ldots,(i_m,j_m,k_m)\}$ *be a family of triples of integers with* $1 \le i_q, j_q, k_q \le n$, *where* m *and* n *are fixed integers. Let* $\Omega_\mathcal{F} \subset (\mathbb{R}^2)^n$ *contains all* $(x_1,\ldots,x_n)$ *such that* $(x_{i_q}, x_{j_q}, x_{k_q}) \in \Omega_0$ *for* $q = 1,\ldots,m$.
Given elements $(z_1,\ldots,z_n) \in \Omega_\mathcal{F}$ *and numbers* $(c_1,\ldots,c_n)$ *in* $(0,+\infty)^n$, *we define the function* F *on* $\Omega_\mathcal{F}$ *by*

$$F(x_1,\ldots,x_n) = \frac{1}{n}\sum_{l=1}^n |x_l - z_l|^2 + \frac{\lambda}{n}\sum_{q=1}^m \left(\log f(x_{i_q}, x_{j_q}, x_{k_q}) - c_q\right)^2$$

where λ *is a positive number. Prove that*

$$\nabla F(x) = \frac{1}{n}\begin{pmatrix} u_1\\ \vdots\\ u_n\end{pmatrix}$$

with

$$
\begin{aligned}
u_l = 2(x_l - z_l) &+ 2\lambda \sum_{q:i_q=l} \left(\log f(x_{i_q}, x_{j_q}, x_{k_q}) - c_q\right) \frac{(x_{k_q} - x_{j_q})^{\perp}}{f(x_{i_q}, x_{j_q}, x_{k_q})} \\
&+ 2\lambda \sum_{q:j_q=l} \left(\log f(x_{i_q}, x_{j_q}, x_{k_q}) - c_q\right) \frac{(x_{i_q} - x_{k_q})^{\perp}}{f(x_{i_q}, x_{j_q}, x_{k_q})} \\
&+ 2\lambda \sum_{q:k_q=l} \left(\log f(x_{i_q}, x_{j_q}, x_{k_q}) - c_q\right) \frac{(x_{j_q} - x_{i_q})^{\perp}}{f(x_{i_q}, x_{j_q}, x_{k_q})}
\end{aligned}
$$

Proof. From Equation 3, we have

$$
\begin{aligned}
\nabla g(x_1, x_2, x_3) &= \frac{2}{\ln 10} \frac{\log\left(\det\left([x_2 - x_1, x_3 - x_1]\right)\right) - c}{\det\left([x_2 - x_1, x_3 - x_1]\right)} \left[(x_3 - x_2)^{\perp}, (x_1 - x_3)^{\perp}, (x_2 - x_1)^{\perp}\right] \\
&= \frac{2}{\ln 10} (\log(f(x_1, x_2, x_3)) - c) \frac{\left[(x_3 - x_2)^{\perp}, (x_1 - x_3)^{\perp}, (x_2 - x_1)^{\perp}\right]}{f(x_1, x_2, x_3))}.
\end{aligned} \tag{4}
$$

On the other hand, we have

$$
\nabla F(x) = \begin{pmatrix} \frac{\partial F}{\partial x_1}(x) \\ \vdots \\ \frac{\partial F}{\partial x_n}(x) \end{pmatrix}
$$

and

$$
\begin{aligned}
F(x_1, \ldots, x_n) &= \frac{1}{n} \sum_{l=1}^{n} |x_l - z_l|^2 + \frac{\lambda}{n} \sum_{q=1}^{m} \left(\log f(x_{i_q}, x_{j_q}, x_{k_q}) - c_q\right)^2 \\
&= \frac{1}{n} \sum_{l=1}^{n} |x_l - z_l|^2 + \frac{\lambda}{n} \sum_{q=1}^{m} g(x_{i_q}, x_{j_q}, x_{k_q}).
\end{aligned}
$$

Using result from Equation 4 and consider $\frac{\partial F}{\partial x_l}(x)$, we obtain

$$
\begin{aligned}
\frac{\partial F}{\partial x_l}(x) &= \frac{1}{n} \left(2(x_l - z_l) + \lambda \sum_{q=1}^{m} \frac{\partial g(x_{i_q}, x_{j_q}, x_{k_q})}{\partial x_l} \right) \\
&= \frac{1}{n} (2(x_l - z_l) + \lambda \sum_{q:i_q=l} \frac{2}{\ln 10} \left(\log f(x_{i_q}, x_{j_q}, x_{k_q}) - c_q\right) \frac{(x_{k_q} - x_{j_q})^{\perp}}{f(x_{i_q}, x_{j_q}, x_{k_q})} \\
&+ \lambda \sum_{q:j_q=l} \frac{2}{\ln 10} \left(\log f(x_{i_q}, x_{j_q}, x_{k_q}) - c_q\right) \frac{(x_{i_q} - x_{k_q})^{\perp}}{f(x_{i_q}, x_{j_q}, x_{k_q})} \\
&+ \lambda \sum_{q:k_q=l} \frac{2}{\ln 10} \left(\log f(x_{i_q}, x_{j_q}, x_{k_q}) - c_q\right) \frac{(x_{j_q} - x_{i_q})^{\perp}}{f(x_{i_q}, x_{j_q}, x_{k_q})}).
\end{aligned}
$$

Therefore, with λ as an arbitrary positive number, we obtain

$$
\nabla F(x) = \frac{1}{n} \begin{pmatrix} u_1 \\ \vdots \\ u_n \end{pmatrix}
$$

with

$$u_l = 2(x_l - z_l) + 2\lambda \sum_{q:i_q=l} \left(\log f(x_{i_q}, x_{j_q}, x_{k_q}) - c_q\right) \frac{(x_{k_q} - x_{j_q})^\perp}{f(x_{i_q}, x_{j_q}, x_{k_q})}$$
$$+ 2\lambda \sum_{q:j_q=l} \left(\log f(x_{i_q}, x_{j_q}, x_{k_q}) - c_q\right) \frac{(x_{i_q} - x_{k_q})^\perp}{f(x_{i_q}, x_{j_q}, x_{k_q})}$$
$$+ 2\lambda \sum_{q:k_q=l} \left(\log f(x_{i_q}, x_{j_q}, x_{k_q}) - c_q\right) \frac{(x_{j_q} - x_{i_q})^\perp}{f(x_{i_q}, x_{j_q}, x_{k_q})}$$

□

4 *Take for this question $\lambda = 0.01$.*
Program a function (e.g., in Python) that takes as input a set of points x (as an $n \times 2$ array), the list of triples $\mathcal{F}$ (as an $m \times 3$ array of integers), the "target" sets of points z (as an $n \times 2$ array) and a collection of positive numbers $c_1, \ldots, c_n$ (as an $n \times 1$ array) and return $F(x)$ and $\nabla F(x)$. Your code should test whether $x \in \Omega_{\mathcal{F}}$ and return $F(x) = \infty$ and $\nabla F(x) =$ None if this is not true.
To assess the correctness of your function use the datasets "z.csv", which provides $(z_1, \ldots, z_n)$, "triples.csv" which provides the family $\mathcal{F}$ (with indexes starting at 0), and "y.csv", which provides a second family of points $y_1, \ldots, y_n$. You will use the latter family to compute $c_q = \log f(y_{i_q}, y_{j_q}, y_{k_q})$.
Provide (with six decimal values) the numerical values of $F(x)$ for $x = y + \rho(z - y)$ for $\rho = 0, 0.1, 0.25, 0.5$. (The value for $\rho = 0.05$ is 0.274074.)
To assess the correctness of your gradient, let $\epsilon = 10^{-8}$ and use the $n \times 2$ array h provided in the file "random direction.csv" to return, for $\rho = 0, 0.1, 0.25$, the values of

$$\frac{F(y + \rho(z - y) + \epsilon h) - F(y + \rho(z - y) - \epsilon h)}{2\epsilon}$$

and of $tr(h^\top \nabla F(y + \rho(z - y)))$ with 6 decimal digits. (The two values should coincide.)

```
import numpy as np
import pandas as pd
import torch
import matplotlib.pyplot as plt
import matplotlib.tri as mtri
torch.set_printoptions(precision=6)

def func(list_x, triples, list_z, list_c):
    list_x = list_x.detach().requires_grad_()
    if list_x.shape != (len(list_x), 2):
        return float('inf'), None
    out = func_F(list_x, triples, list_z, list_c)
    out.backward()
    return out, list_x.grad

def func_f(x1, x2, x3):
    X = torch.stack([x2-x1, x3-x1])
    return torch.linalg.det(X)

def func_F(list_x, triples, list_z, list_c, lda = 0.01):
    term_1, term_2 = 0, 0
    for l in range(len(list_x)):
        x_l = list_x[l]
        z_l = list_z[l]
        term_1 += torch.square(torch.norm(x_l - z_l))
    term_1 = term_1 * (1/len(list_x))

    for q in range(len(triples)):
        x_i = list_x[triples[q][0]]
        x_j = list_x[triples[q][1]]
        x_k = list_x[triples[q][2]]
        term_2 += torch.square(torch.log(func_f(x_i, x_j, x_k)) - list_c[q])
    term_2 = term_2 * (lda/len(list_x))

    return term_1 + term_2

def get_cq(list_y, triples, q):
    y_i = list_y[triples[q][0]]
    y_j = list_y[triples[q][1]]
    y_k = list_y[triples[q][2]]
    return torch.log(func_f(y_i, y_j, y_k))
```

```
if __name__ == "__main__":
    list_z = pd.read_csv('homework1_data/z.csv')
    list_z = list_z[['0', '1']].to_numpy()
    triples = pd.read_csv('homework1_data/triples.csv')
    triples = triples[['0', '1', '2']].to_numpy()
    list_y = pd.read_csv('homework1_data/y.csv')
    list_y = list_y[['0', '1']].to_numpy()
    list_h = pd.read_csv('homework1_data/random_direction.csv')
    list_h = list_h[['0', '1']].to_numpy()

    list_z = torch.tensor(list_z)
    list_y = torch.tensor(list_y)
    list_h = torch.tensor(list_h)
    list_c = [get_cq(list_y, triples, q) for q in range(len(triples))]
```

Print six decimal values of $F(x)$ for $x = y + \rho(z - y)$ for $\rho = 0, 0.1, 0.25, 0.5$.

```
print(func(list_y + 0. * (list_z - list_y), triples, list_z, list_c)[0])
tensor(0.303507, dtype=torch.float64, grad_fn=<AddBackward0>)

print(func(list_y + 0.1 * (list_z - list_y), triples, list_z, list_c)[0])
tensor(0.246393, dtype=torch.float64, grad_fn=<AddBackward0>)

print(func(list_y + 0.25 * (list_z - list_y), triples, list_z, list_c)[0])
tensor(0.173941, dtype=torch.float64, grad_fn=<AddBackward0>)

print(func(list_y + 0.5 * (list_z - list_y), triples, list_z, list_c)[0])
tensor(nan, dtype=torch.float64, grad_fn=<AddBackward0>)
```

Result: [0.303507, 0.246393, 0.173941, nan]

Access the correctness of gradient with $\rho = 0, 0.1, 0.25$.

```
p = 0
print((func((list_y + p * (list_z - list_y) + 1e-8 * list_h), triples, list_z, list_c)[0]
    - func((list_y + p * (list_z - list_y) - 1e-8 * list_h), triples, list_z, list_c)[0])/(2 * 1e-8))
print(torch.trace(torch.matmul(torch.transpose(list_h, 0, 1), func(list_y + p * (list_z - list_y), triples,
    list_z, list_c)[1])))
tensor(-0.260748, dtype=torch.float64, grad_fn=<DivBackward0>)
tensor(-0.260748, dtype=torch.float64)

p = 0.1
print((func((list_y + p * (list_z - list_y) + 1e-8 * list_h), triples, list_z, list_c)[0]
    - func((list_y + p * (list_z - list_y) - 1e-8 * list_h), triples, list_z, list_c)[0])/(2 * 1e-8))
print(torch.trace(torch.matmul(torch.transpose(list_h, 0, 1), func(list_y + p * (list_z - list_y), triples,
    list_z, list_c)[1])))
tensor(-0.448279, dtype=torch.float64, grad_fn=<DivBackward0>)
tensor(-0.448279, dtype=torch.float64)

p = 0.25
print((func((list_y + p * (list_z - list_y) + 1e-8 * list_h), triples, list_z, list_c)[0]
    - func((list_y + p * (list_z - list_y) - 1e-8 * list_h), triples, list_z, list_c)[0])/(2 * 1e-8))
print(torch.trace(torch.matmul(torch.transpose(list_h, 0, 1), func(list_y + p * (list_z - list_y), triples,
    list_z, list_c)[1])))
tensor(-1.086539, dtype=torch.float64, grad_fn=<DivBackward0>)
tensor(-1.086539, dtype=torch.float64)
```

Result: [-0.260748, -0.448279, -1.086539]

5 *Using $\alpha = 0.01$, program gradient descent iterations with constant steps*

$$x(t+1) = x(t) - \alpha \nabla F(x(t))$$

initialized at $x(0) = y$ and over $T = 10^5$ iterations. Provide the numerical value of $F(x(T))$ and a plot of $F(x(t))$ as a function of t for $t = 0, \dots, 10^5$.
Also provide a figure containing:

1. *A scatter plot of $z_1, \dots, z_n$.*
2. *A scatter plot of $y_1, \dots, y_n$ with all triangles $y_{i_q}, y_{j_q}, y_{k_q}$.*
3. *A scatter plot of $x_1(T), \dots, x_n(T)$ with all triangles $x_{i_q}(T), x_{j_q}(T), x_{k_q}(T)$.*

Make sure to label and use different colors for (1), (2) and (3).

```
x_t = list_y.clone()
alpha = 0.01
list_t = []
list_f = []
for t in range(int(1e5)):
  f, grad = func(x_t, triples, list_z, list_c)
  list_t.append(t)
  list_f.append(f.detach().numpy().item())
  x_t -= alpha * grad
plt.plot(list_t, list_f)
plt.xlabel("t")
plt.ylabel(r'$F(x(t))$')
plt.title("Visualiztion of " + r'$F(x(t))$' + " as a function of t")
plt.savefig("2.5.1.pdf")
```

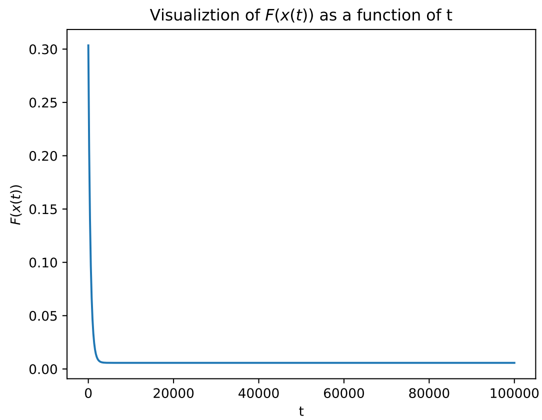

Figure 4: Visualization of the function $t \mapsto F(x(t))$ discretized over $t \in [0, 10^5]$.

```
list_triangles = []
for q in range(len(triples)):
  list_triangles.append([triples[q][0], triples[q][1], triples[q][2]])

triang_y = mtri.Triangulation(list_y[:,0], list_y[:,1], triangles=list_triangles)
triang_xt = mtri.Triangulation(x_t[:,0], x_t[:,1], triangles=list_triangles)

plt.plot(list_z[:,0], list_z[:,1], 'o', color = 'blue', label='z')
plt.triplot(triang_y, marker="o", color = 'red', label='y')
plt.triplot(triang_xt, marker="o", color = 'green', label='x(T)')
plt.title("Scatter plot of z, y, and x(T) with triangles from the triples list")
plt.legend()
plt.savefig("2.5.2.pdf")
```

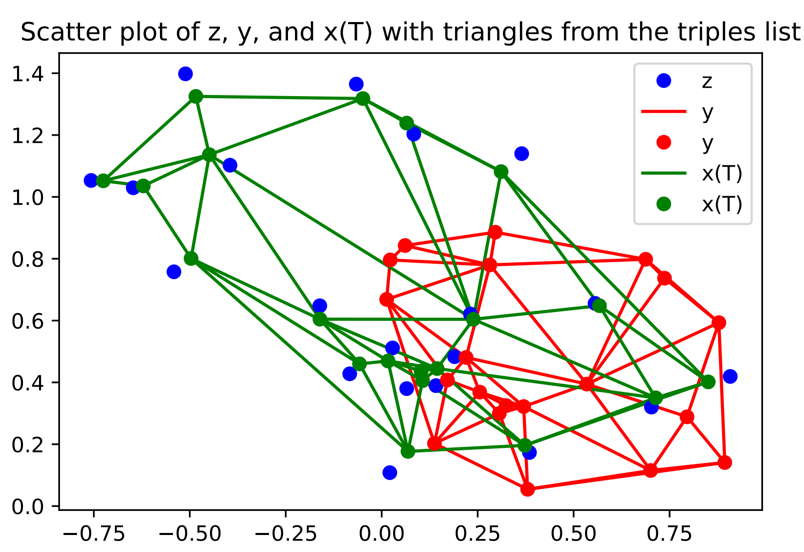

Figure 5: Scatter plot of $z_1, \ldots, z_n$, $y_1, \ldots, y_n$ with all triangles $y_{i_q}, y_{j_q}, y_{k_q}$, and $x_1(T), \ldots, x_n(T)$ with all triangles $x_{i_q}(T), x_{j_q}(T), x_{k_q}(T)$.

3 Problem 3

We use the notation $A \bullet B = trace(A^\top B)$. Let Ω denote the set of $n \times n$ matrices with positive determinant.

1 *Justify why Ω is an open subset of $\mathcal{M}_{n,n}$.*
Let $A \in \Omega$, then $\det(A) > 0$.
Let $r = I_n$, $B \in \mathcal{B}(A; r)$, we have

$$\det(A + r) \geq \det(A) + \det(r)$$

Since $\det(A) > 0$ and $\det(r) = \det(I_n) = 1$, then $\det(A + r) > 0$ and $\det(B) > 0$. Therefore, we obtain Ω is an open set.

2 *Let F is C^1 function defined on Ω. Given $A \in \Omega$ and $H \in \mathcal{M}_{n\times n}$ and consider the function $f : \epsilon \mapsto F(A + \epsilon HA)$.*

(1) Prove that f is well defined on an interval $(-r, r)$ for some $r > 0$.

Proof. Consider function $f : \epsilon \mapsto F(A+\epsilon HA)$, due to F is C^1 function defined on Ω, we obtain $f \subseteq \mathbb{R}\times\mathbb{R}$, $\det(A + \epsilon HA) > 0$, and the domain of f is $\mathbb{R}$. On the other hand, we have

$$A + \epsilon HA = A(I_n + \epsilon H),$$

therefore, $f : \epsilon \mapsto F(A + \epsilon HA)$ is equivalent to $f : \epsilon \mapsto F(A(I_n + \epsilon H))$ and

$$\begin{aligned}
&\det(A(I_n + \epsilon H)) > 0 \\
&\Rightarrow \det(A)\det(I_n + \epsilon H)) > 0 \\
&\Rightarrow \det(I_n + \epsilon H)) > 0 \text{ (since } A \in \Omega),
\end{aligned}$$

so $(I_n + \epsilon H) \in \Omega$. Let $\epsilon \in (-r, r)$ for some $r > 0$, then there is a unique $F(A(I_n + \epsilon H)) \in \mathbb{R}$ with $f(\epsilon) = F(A(I_n + \epsilon H))$. As a consequence, we obtain f is well defined on an interval $(-r, r)$ for some $r > 0$. □

(2) Prove that $\partial f(0) = (\nabla F(A)A^\top) \bullet H$ where ∇F is defined so that $dF(A)H = \nabla F(A) \bullet H$.

Proof. We have

$$\partial f(\epsilon) = \partial_\epsilon F(A + \epsilon HA)$$

so

$$\partial f(0) = \partial_\epsilon F(A + \epsilon HA)|_{\epsilon=0}. \tag{5}$$

On the other hand, if ∇F is defined so that $dF(A)H = \nabla F(A) \bullet H$ then

$$\begin{aligned}
dF(A)H &= \nabla F(A) \bullet H \\
&= \partial_\epsilon F(A + \epsilon H)|_{\epsilon=0},
\end{aligned}$$

therefore

$$(\nabla F(A)A^\top) \bullet H = \partial_\epsilon F(A + \epsilon HA)|_{\epsilon=0}. \tag{6}$$

From Equation 5 and Equation 6, we obtain $\partial f(0) = (\nabla F(A)A^\top) \bullet H$ where ∇F is defined so that $dF(A)H = \nabla F(A) \bullet H$. □

(3) Deduce from this that $-\nabla F(A)A^\top A$ is a direction of descent of F at A.
Due to $\partial f(0) = (\nabla F(A)A^\top)\bullet H$ where ∇F is defined so that $dF(A)H = \nabla F(A)\bullet H$, apply Cauchy–Schwarz inequality, we have

$$\begin{aligned}
&|\nabla F(A)A^\top||H| \geq (\nabla F(A)A^\top) \bullet H \\
\Rightarrow &|\nabla F(A)A^\top||H| \geq tr((\nabla F(A)A^\top)H^\top) > 0.
\end{aligned}$$

Therefore, we obtain

$$\nabla F(A)^\top(\nabla F(A)A^\top A)) > 0$$

equivalent to

$$\nabla F(A)^\top(-\nabla F(A)A^\top A)) < 0.$$

As a consequence, we obtain $-\nabla F(A)A^\top A$ is a direction of descent of F at A.

3 *Fix a matrix $B \in \mathcal{M}_{n\times n}$ and define the function F on Ω by*

$$F(A) = \frac{c}{2}(A - A^{-1}) \bullet (A - A^{-1}) + \frac{1}{2}(A - B) \bullet (A - B)$$

(a) Consider the mapping $\mathcal{I} : \Omega \to \Omega$ defined by $\mathcal{I}(A) = A^{-1}$. Using the fact that the differential of $A \mapsto \mathcal{I}(A)$ is zero, prove that

$$d\mathcal{I}(A)H = -A^{-1}HA^{-1}.$$

Proof. Since the differential of $A \mapsto \mathcal{I}(A)$ is zero, we obtain

$$\begin{aligned}
d\mathcal{I}(A)H &= \partial_\epsilon \mathcal{I}(A + \epsilon H)|_{\epsilon=0} \\
&= \partial_\epsilon (A + \epsilon H)^{-1}|_{\epsilon=0} \\
&= -(A + \epsilon H)^{-1}\partial_\epsilon(A + \epsilon H)(A + \epsilon H)^{-1}|_{\epsilon=0} \\
&= -(A + \epsilon H)^{-1}H(A + \epsilon H)^{-1}|_{\epsilon=0} \\
&= -A^{-1}HA^{-1}.
\end{aligned}$$

□

(b) Deduce from this that

$$dF(A)H = c(A - A^{-1}) \bullet (H + A^{-1}HA^{-1}) + (A - B) \bullet H$$

and

$$\nabla F(A) = c(A^{-\top}(A - A^{-1})A^{-\top} - A^{-1}) + (1 + c)A - B.$$

($A^{-\top}$ is the transpose of the inverse of A.)

We have

$$\begin{aligned}
dF(A)H &= \partial_\epsilon F(A + \epsilon H)|_{\epsilon=0} \\
&= \partial_\epsilon \left\{ \frac{c}{2}\left[(A + \epsilon H) - (A + \epsilon H)^{-1}\right] \bullet \left[(A + \epsilon H) - (A + \epsilon H)^{-1}\right] + \frac{1}{2}\left[(A + \epsilon H) - B\right] \bullet \left[(A + \epsilon H) - B\right] \right\} |_{\epsilon=0} \\
&= \underbrace{\partial_\epsilon \frac{c}{2}\left[(A + \epsilon H) - (A + \epsilon H)^{-1}\right] \bullet \left[(A + \epsilon H) - (A + \epsilon H)^{-1}\right] |_{\epsilon=0}}_{U} + \underbrace{\partial_\epsilon \frac{1}{2}\left[(A + \epsilon H) - B\right] \bullet \left[(A + \epsilon H) - B\right] |_{\epsilon=0}}_{V}.
\end{aligned}$$

Derive U with the fact that $dA^{-1}H = \partial_\epsilon(A + \epsilon H)^{-1}|_{\epsilon=0} = -A^{-1}HA^{-1}$, we obtain

$$\begin{aligned}
U &= \partial_\epsilon \frac{c}{2} tr\left(\left[(A + \epsilon H) - (A + \epsilon H)^{-1}\right]^\top \left[(A + \epsilon H) - (A + \epsilon H)^{-1}\right] \right) |_{\epsilon=0} \\
&= \frac{c}{2} tr\left(\partial_\epsilon \left\{ \left[(A + \epsilon H)^\top - \left[(A + \epsilon H)^{-1}\right]^\top\right] \left[(A + \epsilon H) - (A + \epsilon H)^{-1}\right] \right\} |_{\epsilon=0} \right) \\
&= \frac{c}{2} tr\left(A^\top H + H^\top A + A^\top(A^{-1}HA^{-1}) - H^\top A^{-1} + (A^{-1}HA^{-1})^\top A - HA^{-\top} - (A^{-1}HA^{-1})^\top A^{-1} - A^{-\top}(A^{-1}HA^{-1}) \right) \\
&= c\ tr\left(A^\top H + A^\top(A^{-1}HA^{-1}) - A^{-\top}H - A^{-\top}(A^{-1}HA^{-1}) \right) \\
&= c\ tr\left((A - A^{-1})^\top(H + A^{-1}HA^{-1}) \right) \\
&= c(A - A^{-1}) \bullet (H + A^{-1}HA^{-1}).
\end{aligned}$$

Derive V with the fact that $dA^{-1}H = \partial_\epsilon (A+\epsilon H)^{-1}|_{\epsilon=0} = -A^{-1}HA^{-1}$, we obtain

$$\begin{aligned} V &= \partial_\epsilon \frac{1}{2} tr \left([(A+\epsilon H) - B]^\top [(A+\epsilon H) - B] \right) |_{\epsilon=0} \\ &= \frac{1}{2} tr \left(\partial_\epsilon \left\{ \left[(A+\epsilon H)^\top - B^\top \right] [(A+\epsilon H) - B] \right\} |_{\epsilon=0} \right) \\ &= \frac{1}{2} tr \left(A^\top H + H^\top A - H^\top B - B^\top H \right) \\ &= tr \left((A-B)^\top H \right) \\ &= (A-B) \bullet H. \end{aligned}$$

Therefore, we obtain

$$dF(A)H = c(A - A^{-1}) \bullet (H + A^{-1}HA^{-1}) + (A-B) \bullet H.$$

Continue, we have

$$\begin{aligned} dF(A)H &= \nabla F(A) \bullet H \\ &= tr(\nabla F(A) H^\top). \end{aligned}$$

On the other hand, continue derive $dF(A)H$, we have

$$\begin{aligned} dF(A)H &= c\ tr \left((A - A^{-1})(H + A^{-1}HA^{-1})^\top \right) + tr \left((A-B)H^\top \right) \\ &= tr \left(c \left[(A - A^{-1}) \left(H^\top + A^{-\top} H A^{-\top} \right) \right] + (A-B)H^\top \right) \\ &= tr \left(\left\{ c \left[A^{-\top}(A - A^{-1})A^{-\top} - A^{-1} \right] + (1+c)A - B \right\} H^\top \right). \end{aligned}$$

Therefore, we obtain

$$\nabla F(A) = c(A^{-\top}(A - A^{-1})A^{-\top} - A^{-1}) + (1+c)A - B.$$

(c) Use $c = 5$ in this question. Let B be the matrix provided in "B.csv" (for which $n = 25$). Program a descent algorithm minimizing F using the iterations (with $\alpha = 0.01$)

$$A_{t+1} = A_t - \alpha \nabla F(A_t) A_t^\top A_t$$

initialized at $A_t = \mathbf{Id}_{\mathbb{R}^n}$. Run the algorithm until all entries of $\nabla F(A_t)$ are smaller, in absolute value, than 10^{-8}.
Provide the number of iterations required by the algorithm and the value of the objective function at convergence. Provide also a plot of $F(A_t)$ as a function of t and of $\log \det A_t$ as a function of t

```
import numpy as np
import pandas as pd
import torch
import matplotlib.pyplot as plt

def func_F(A, B, c):
    A_A_inv = A - torch.inverse(A)
    term_1 = (c/2) * torch.trace(torch.mm(torch.transpose(A_A_inv, 0, 1), A_A_inv))
    A_B = A - B
    term_2 = (1/2) * torch.trace(torch.mm(torch.transpose(A_B, 0, 1), A_B))
    return term_1 + term_2

def grad_F(A, B, c):
    A_A_inv = A - torch.inverse(A)
    A_tr_inv = torch.transpose(torch.inverse(A), 0, 1)
    return c*(torch.mm(torch.mm(A_tr_inv, A_A_inv), A_tr_inv) - torch.inverse(A)) + (1+c) * A - B

def check_tmn(grad_F):
    for row in grad_F:
        for entry in row:
            if torch.abs(entry) > 1e-8:
                return False
    return True
```

```python
if __name__ == "__main__":
    c, alpha = 5, 0.01
    B = pd.read_csv('homework1_data/B.csv')
    B = B.drop(['Unnamed: 0'], axis=1).to_numpy()
    B = torch.tensor(B)
    A = torch.eye(25).double()
    T = 0
    list_t, list_f, list_logdet = [], [], []
    while True:
        grad_tmp = grad_F(A, B, c)
        if check_tmn(grad_tmp) is True:
            break
        else:
            f_tmp = func_F(A, B, c)
            list_f.append(f_tmp)
            list_logdet.append(torch.log(torch.det(A)))
            list_t.append(T)
            A -= alpha * torch.mm(torch.mm(grad_tmp, torch.transpose(A, 0, 1)), A)
            T += 1

    print("The number of required iterations: " + str(T))
    print("The value of the objective function at convergence: " + str(list_f[T-1].item()))

    fig, axs = plt.subplots(1, 2, figsize = (8, 4), constrained_layout=True)

    axs[0].plot(list_t, list_f)
    axs[0].set_xlabel(r'$t$')
    axs[0].set_ylabel(r'$F(A_t)$')
    axs[0].set_title(r'$F(A)$')

    axs[1].plot(list_t, list_logdet)
    axs[1].set_xlabel(r'$t$')
    axs[1].set_ylabel(r'$\log(\det(A_t))$')
    axs[1].set_title(r'$\log(\det(A))$')

    plt.savefig("3.3.pdf")
```

Result:
The number of required iterations: 90
The value of the objective function at convergence: 310.93344862069233

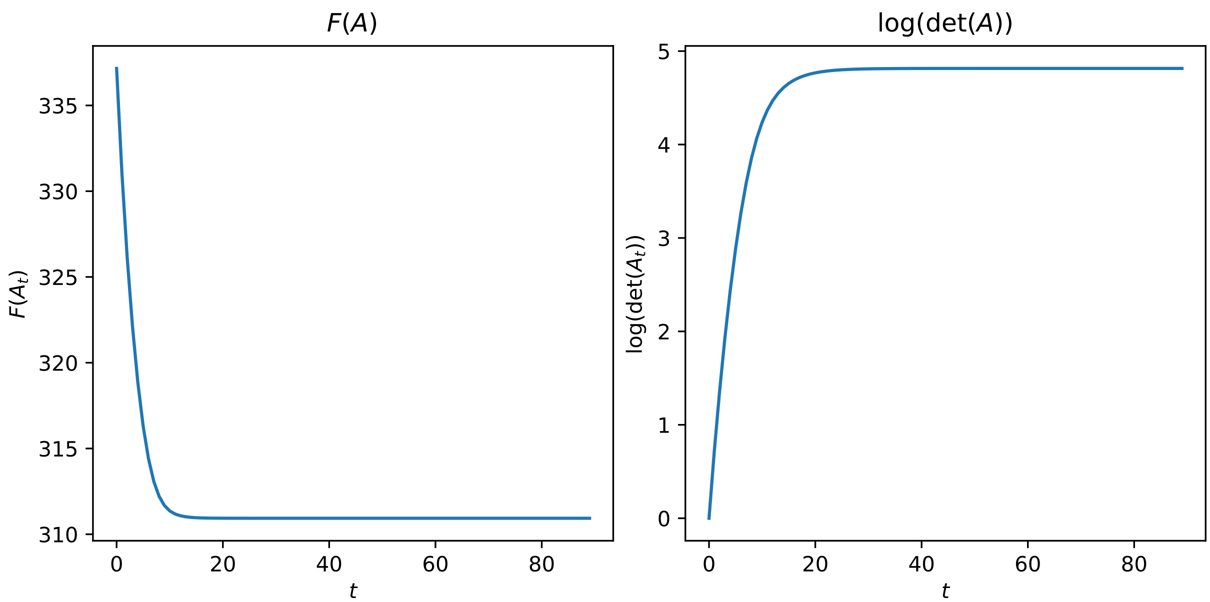

Figure 6: Visualization of $F(A_t)$ as a function of t and of $\log\det A_t$ as a function of t.