Princeton University

ORF 363/COS 323
Final Exam, Fall 2024

December 14, 2024

Instructor:
A.A. Ahmadi

AIs:
Aydin, Budway, Hua, Tziampazis

1. Please write your name on the first page of your solutions. Next to it, please write out and sign the following pledge: "I pledge my honor that I have not violated the Honor Code or the rules specified by the instructor during this examination."
2. The exam is not to be discussed with *anyone* except possibly the instructor and the AIs. You can only ask *clarification questions*, and only as *public* (and preferably non-anonymous) questions on Ed Discussion. No emails.
3. You are allowed to consult the lecture notes, your own notes, the reference books of the course as indicated on the syllabus, the problem sets and their solutions (yours and ours), the midterm and its solutions (yours and ours), the practice midterm and final exams and their solutions, all Ed Discussion posts, but *nothing else*. You can only use the Internet in case you run into problems related to software.
4. You may refer to facts proven in the notes or problem sets without reproving them.
5. For computational problems, include your code. The output you present should come from your code. Report requested numerical values to 4 digits after the decimal point.
6. You have 48 hours from the time of download to submit this exam on Gradescope as a *single PDF file*. The latest submission time is Thursday (December 19, 2024) at 11:59PM EST. You are free to write your solutions on paper or on a tablet, or to type them up. Only the latest version submitted before your deadline will be graded.
7. Each question has 25 points. You need to justify your answers to receive full credit.

Problem 1: Lazy Newton

On his lazy days, Newton considers modifying his algorithm for minimizing a twice continuously differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ as follows: Instead of inverting the full Hessian matrix at each iteration, he inverts the diagonal matrix consisting of the diagonal entries of the Hessian. This leads to the following iterative method:

$$x_{k+1} = x_k - \alpha D^{-1}(x_k)\nabla f(x_k). \tag{1}$$

Here, $x_k \in \mathbb{R}^n$ is the k-th iterate of the algorithm, $\alpha > 0$ is a fixed stepsize, $\nabla f(x_k)$ denotes the gradient of f at x_k, and $D(x_k)$ is the diagonal matrix whose diagonal entries are the same as those of the Hessian matrix at x_k, i.e. $\nabla^2 f(x_k)$. In other words, $D_{ii}(x_k) = \frac{\partial^2 f}{\partial x_i^2}(x_k)$ for $i = 1, \ldots, n$. In this problem, we analyze the global convergence of this algorithm on a particular quadratic function (the results can be generalized to any quadratic function but this is not required for the problem).

Consider the function $f : \mathbb{R}^3 \to \mathbb{R}$ defined as $f(x) = \frac{1}{2}x^T Q x$, with

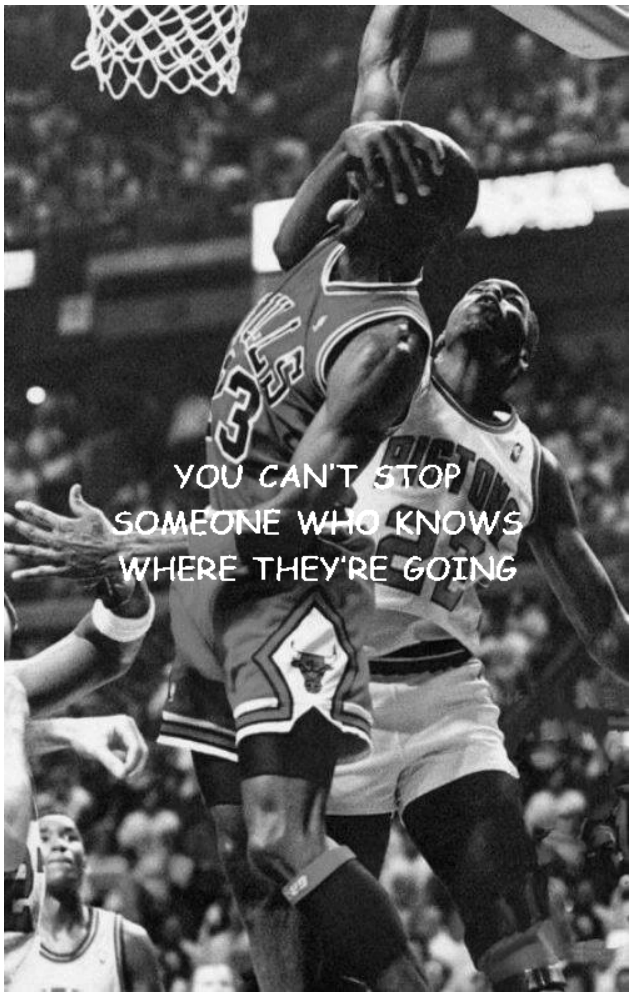
$$Q = \begin{pmatrix} 5 & 2 & 1 \\ 2 & 6 & 3 \\ 1 & 3 & 4 \end{pmatrix}.$$

(a) Show that f has a unique minimizer at $x^* = 0$.

(b) Show that if $\alpha \in (0, 1]$, the lazy Newton method for minimizing the function f given above is *globally convergent*; i.e., for any initial iterate $x_0 \in \mathbb{R}^3$, the sequence of vectors $\{x_k\}$ produced by the lazy Newton method in (1) converges to x^*.

Hint: Consider the function f itself as a Lyapunov function. If needed, you may use the following fact without proof: If A is a symmetric $n \times n$ matrix with smallest eigenvalue $\lambda_{\min}$, then $x^T Ax \geq \lambda_{\min}||x||^2$ for all $x \in \mathbb{R}^n$.

(c) Generate and report a random initial condition $x_0 \in \mathbb{R}^3$ (using `randn(3,1)` in MATLAB or `numpy.random.randn(3)` in Python). With step size $\alpha = 1$, run both the lazy Newton method and the steepest descent method (i.e., the same iterations as (1) but with $D^{-1}(x_k)$ replaced with the identity matrix) for 10 iterations. Report the function value after 10 iterations (i.e., $f(x_{10})$) for both methods. Briefly comment on what you observe.



Problem 2: You can't stop someone who knows where they are going

You love the image of Michael Jordan that you see here and want to set it as your desktop background. However, you think the text makes the message too obvious, and you would rather keep it more subtle. Therefore, you wish to restore the image to its original, text-free version. The image is grayscale and is given by the file `MJ.mat`.[1] You can load this file to MATLAB by running

```
load('MJ.mat')
V = double(MJ)
```

or to Python by running

```
from scipy.io import loadmat
V = loadmat("MJ.mat")["MJ"]
```

The matrix $V \in \mathbb{R}^{m \times n}$, with $m = 790, n = 500$, has entries in $[0, 1]$, where 0 representing

[1] Download at `https://www.princeton.edu/~aaa/Public/Teaching/ORF363_COS323/F24/MJ.mat`

pure black, 1 representing pure white, and numbers in between representing different shades of gray. The text pixels are the only pixels in pure white. To automatically remove the text, consider solving the following optimization problem:

$$\begin{aligned} \min_{X \in \mathbb{R}^{m \times n}} \quad & f(X) := \sum_{i=1}^{m-1} \sum_{j=1}^{n-1} \left\| \begin{bmatrix} X_{i+1,j} - X_{ij} \\ X_{i,j+1} - X_{ij} \end{bmatrix} \right\|_1 \\ \text{s.t.} \quad & X_{ij} = V_{ij} \quad \text{if } V_{ij} \neq 1, \end{aligned} \tag{2}$$

where $\|\cdot\|_1$ denotes the 1-norm. The constraints ensure that we leave the pixels that are not related to text unchanged. The objective is to minimize the deviation of the remaining pixel values (i.e., the ones corresponding to text) from their neighbors.

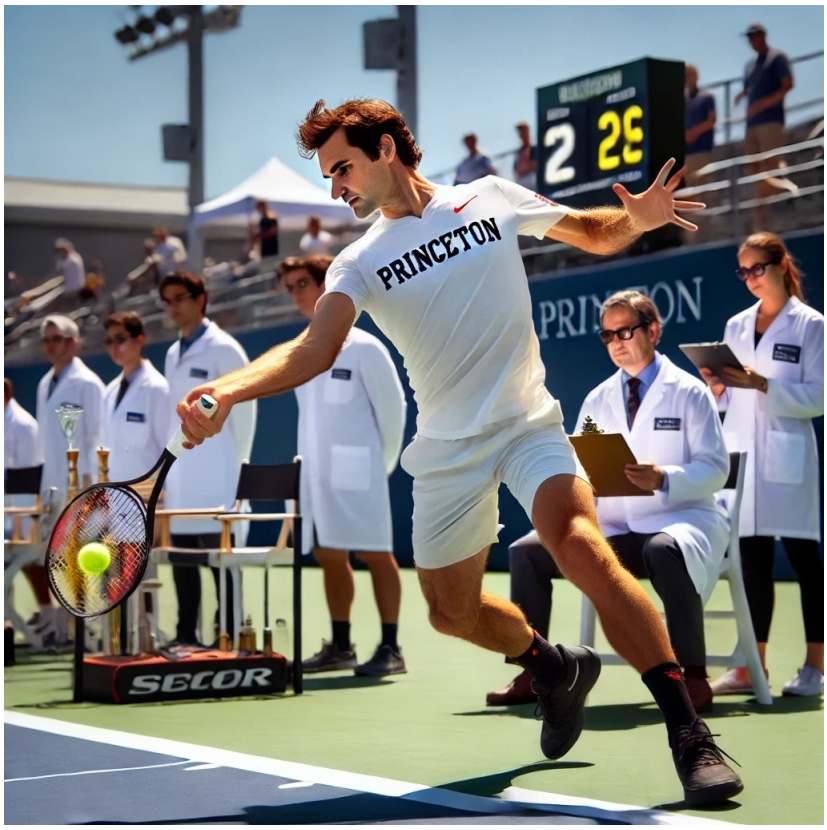
(a) Show that problem (2) is a convex optimization problem.

(b) Solve problem (2), report the optimal value, and display your optimal matrix. Assuming your decision matrix is called X, you can display your image in MATLAB using the command `imshow(X)`, and in Python by running `import matplotlib.pyplot as plt` and `plt.imshow(X.value, cmap='gray')`.

Implementation hint: For an efficient implementation of this problem, code up the objective function in the following equivalent form

$$f(X) = g(X_{2:m,1:n-1} - X_{1:m-1,1:n-1}) + g(X_{1:m-1,2:n} - X_{1:m-1,1:n-1}).$$

Here, the notation $X_{2:m,1:n}$ for example refers to a submatrix of X which has rows 2 through m and columns 1 through n of X (both ends inclusive). For a general matrix variable A, the function g is defined as $g(A) := \sum_i \sum_j |A_{ij}|$ and can be implemented in MATLAB as `sum(sum(abs(A)))` and in Python as `cp.sum(cp.abs(A))`.
To efficiently implement the constraints of (2), consider defining an $m \times n$ indicator matrix which has $(i,j)^{\text{th}}$ element equal to 1 if $V_{ij} \neq 1$ and zero otherwise. Then consider entrywise multiplying this matrix with V and with your decision matrix.



Problem 3: For your own safety, please do not play to my forehand

In *blackbox optimization*, we would like to solve an optimization problem without having an explicit representation of the objective function. Instead, we can evaluate the objective function at specific points, though each function evaluation is typically expensive. In the model-based approach to blackbox optimization, one fits a model to a current set of points and function values and picks the optimal solution to this model as the next point at which to evaluate the function. The model is then updated based on how close the new function evaluation is to the prediction of the model.

In this question, we apply the first step of this approach to the problem of customizing a tennis racket with the goal of improving the forehand shot of a particular player. The variables under our control are the weight of the racket (which we change by adding lead tape to a specific spot) and the tension of the racket string. We would like to find the combination of weight/tension that maximizes the so-called "Power-Accuracy Composite Score (PACS)". For any weight/tension combination, this metric is recorded by asking the player to hit 100 forehands towards a specific target and measuring the average speed of the shots divided by the average distance to the target. Note that each evaluation of PACS is expensive, as it requires changing the lead tape, restringing the racket, and asking the player

to hit 100 shots under similar conditions. In the table below, we show ten evaluations of PACS as a function of racket weight and string tension for a particular player.

Index i	Weight w_i (g)	Tension t_i (lbs)	PACS y_i
1	309	61	72
2	329	62	85
3	304	53	70
4	315	54	88
5	300	45	56
6	317	55	91
7	327	53	77
8	328	49	62
9	325	64	86
10	329	61	85

Consider the task of fitting a quadratic function $f_c : \mathbb{R}^2 \to \mathbb{R}$ parameterized as

$$f_c(w,t) = c_1 w^2 + c_2 t^2 + c_3 wt + c_4 w + c_5 t + c_6$$

to the data points in the table by solving the optimization problem

$$\min_{c \in \mathbb{R}^6} \quad \sum_{i=1}^{10} (f_c(w_i, t_i) - y_i)^2 \ . \tag{3}$$

The optimal solution to this problem will be used as a model for the PACS function.

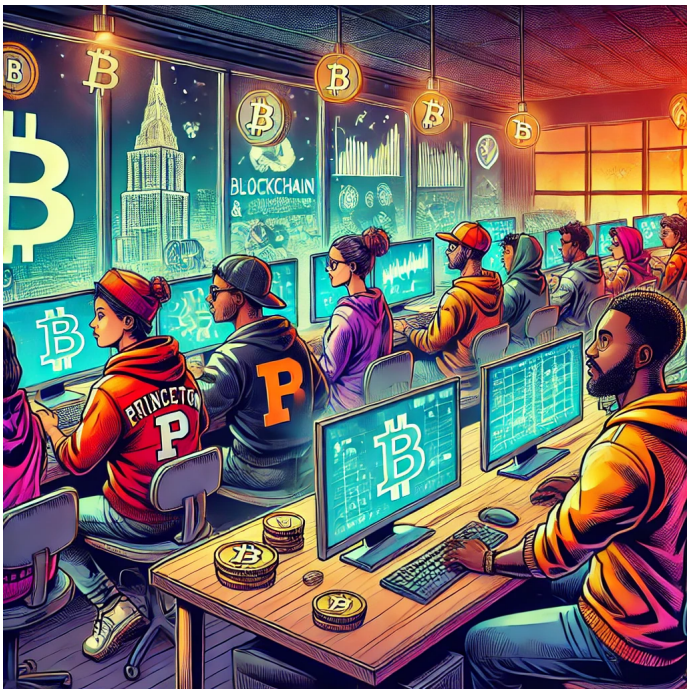
(a) Argue that an optimal solution to (3) exists and is unique.

(b) Solve problem (3) and report the optimal coefficients $(c_1^*, \ldots, c_6^*)$. Denote the optimal quadratic function by $f_{c^*}(w,t)$. Does this function have any of the following properties: convexity, strict convexity, concavity, or strict concavity? Justify.

(c) To find the optimal weight and tension for the racket, consider the following optimization problem (where the constraints encode realistic ranges for the racket weight and string tension):

$$\begin{array}{ll} \max\limits_{w,t \in \mathbb{R}} & f_{c^*}(w,t) \\ \text{s.t.} & w \in [300, 330] \\ & t \in [45, 65]. \end{array} \tag{4}$$

Argue that an optimal solution to this problem exists and is unique. Solve this optimization problem and report the optimal racket weight, string tension, and the PACS predicted by the model f_{c^*}.



Problem 4: Bitcoin is at $100,000 and the fever is back!

Decentralized exchanges (DEXs) enable users to trade cryptocurrencies without relying on a central authority. We consider a DEX which offers exchanges of n assets. Let $\mathbb{R}^n_+$ denote the set of entrywise nonnegative vectors in $\mathbb{R}^n$. The DEX has a reserve vector $r \in \mathbb{R}^n_+$, where entry r_i denotes the quantity of asset i in the reserves. A proposed trade by a user consists of two vectors $x \in \mathbb{R}^n_+, y \in \mathbb{R}^n_+$, where x_i denotes the quantity of asset i that the trader proposes to give to the DEX, and y_i denotes the quantity of asset i that the trader proposes to receive from the DEX. The DEX has a trading function $T : \mathbb{R}^n_+ \to \mathbb{R}$ that needs to remain constant through every exchange. More specifically, the DEX accepts a proposed trade (x, y) by a user if and only if

$$T(r + \gamma x - y) = T(r),$$

where $\gamma \in (0, 1)$ accounts for a small trading fee. The user wants to find a trade (x, y) that is accepted by the DEX and maximizes their utility function $U : \mathbb{R}^n \to \mathbb{R}$. This can be achieved by solving the following optimization problem:

$$\begin{aligned} \max_{x,y \in \mathbb{R}^n} \quad & U(y - x) \\ \text{s.t.} \quad & T(r + \gamma x - y) = T(r) \\ & r + \gamma x - y \geq 0 \\ & 0 \leq x_i \leq x_{max}, \quad i = 1, \ldots, n \\ & 0 \leq y_i \leq y_{max}, \quad i = 1, \ldots, n. \end{aligned} \tag{5}$$

Here, the scalars $x_{\max}$ and $y_{\max}$ denote bounds on the quantity of assets that the DEX is willing to receive and give out. In practice, the trading function T and the utility function U are both concave. Nevertheless, optimization problem (5) is generally nonconvex due to the equality constraint. Thus, we consider the following convex relaxation of (5):

$$\begin{array}{ll} \max\limits_{x,y\in\mathbb{R}^n} & U(y-x) \\ \text{s.t.} & T(r+\gamma x-y)\geq T(r) \\ & r+\gamma x-y\geq 0 \\ & 0\leq x_i\leq x_{max}, \quad i=1,\ldots,n \\ & 0\leq y_i\leq y_{max}, \quad i=1,\ldots,n. \end{array} \tag{6}$$

(a) Show that problem (6) is indeed a convex optimization problem.

(b) We say a function $f:\Omega\to\mathbb{R}$, where Ω is a subset of $\mathbb{R}^n$, is *non-decreasing* if for all vectors $u,v\in\Omega$, $u\leq v$ entrywise implies $f(u)\leq f(v)$. A non-decreasing function f is *increasing* if whenever $u\leq v$ entrywise and $u_i<v_i$ for some $i\in\{1,\ldots,n\}$, we have $f(u)<f(v)$. Suppose the functions U and T are both continuous, with T being non-decreasing and U being increasing. Show that the set of optimal solutions to (5) is the same as the set of optimal solutions to (6).

(c) Consider the setting of exchanging two assets, i.e., when $n=2$. Let the utility function $U:\mathbb{R}^2\to\mathbb{R}$ be given as $U(z)=c^Tz$, where $c\in\mathbb{R}^2$ is a given vector with positive entries. Let the trading function $T:\mathbb{R}^2_+\to\mathbb{R}$ be defined as $T(z)=\sqrt{z_1z_2}$. Show that in this setting, the set of optimal solutions to (5) is the same as the set of optimal solutions to the following semidefinite program:

$$\begin{array}{ll} \max\limits_{x,y\in\mathbb{R}^2} & U(y-x) \\ \text{s.t.} & \begin{bmatrix} r_1+\gamma x_1-y_1 & T(r) \\ T(r) & r_2+\gamma x_2-y_2 \end{bmatrix}\succeq 0 \\ & 0\leq x_1\leq x_{max}, \quad 0\leq x_2\leq x_{max} \\ & 0\leq y_1\leq y_{max}, \quad 0\leq y_2\leq y_{max}. \end{array} \tag{7}$$

(d) In the setting of part (c), let $\gamma=0.9$, $r=(1,2)^T$, $x_{max}=y_{max}=1$, and $c=(2t,1)^T$, where t is a parameter. Take $t=0.5,0.6,0.7,\ldots,2.0$ and for each value of t solve the optimization problem (7). For $i=1,2$, plot the net change y_i-x_i of the optimal trade versus t. Briefly comment on what you observe.