Princeton University

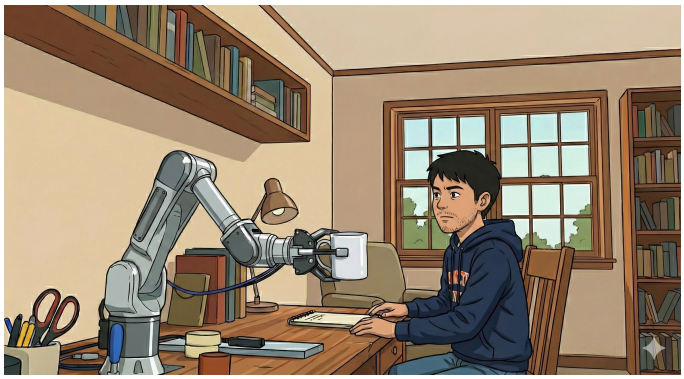
ORF 363/COS 323
Final Exam, Fall 2025

December 13, 2025

Instructor:
A.A. Ahmadi

TAs:
Budway, Hua, Shi, Tang, Yang

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, including the restrictions on the use of web resources and AI tools."
2. The exam is not to be discussed with *anyone* except possibly the instructors and the TAs. 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 or an AI tool 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 Friday (December 19, 2025) 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.
8. All data files needed for this exam can be downloaded from the following link: `https://www.princeton.edu/~aaa/Public/Teaching/ORF363_COS323/F25/Final_Exam_Data_Files`.



Problem 1: Hand-eye coordination of a robot

In robotic manipulation, a key perception task is to estimate an object's pose (i.e., its position and orientation). A robot arm may have a pre-programmed grasp for a cup in its *standard pose* (upright at the origin), but in practice the cup's pose is unknown. The robot must therefore estimate that pose and adjust its joints so the gripper aligns with the cup.

Objects are often represented by 3D point clouds. Let $a_1, \ldots, a_N \in \mathbb{R}^3$ be the cup's point cloud in the standard pose. A depth camera captures the cup in its real-time pose (*observed pose*), yielding $b_1, \ldots, b_N \in \mathbb{R}^3$. These are the same physical points as $a_1, \ldots, a_N$, but transformed by an unknown rotation and translation and corrupted by measurement noise. The goal is to estimate the rotation and translation that best align the standard and observed point clouds, enabling the robot to adjust its joints to grasp the cup.

Throughout the problem, let I denote the 3×3 identity matrix. In 3D geometry, any translation can be represented by a vector $t \in \mathbb{R}^3$, and any rotation by a matrix $Q \in \mathbb{R}^{3\times 3}$ such that $QQ^T = I$ and $\det(Q) = 1$.[1] For the robot to find the proper rotation and translation, it needs to solve the following optimization problem:

$$\begin{aligned} \min_{Q\in\mathbb{R}^{3\times 3}, t\in\mathbb{R}^3} \quad & \sum_{i=1}^{N} \|Qa_i + t - b_i\|_2^2 \\ \text{subject to} \quad & QQ^T = I, \quad \det(Q) = 1. \end{aligned} \tag{1}$$

(a) Is problem (1) a convex optimization problem? Justify your answer.

(b) Define the following 3×3 matrix that can be constructed from the two point clouds:

$$M = \sum_{i=1}^{N} (a_i - \frac{1}{N}\sum_{k=1}^{N} a_k)(b_i - \frac{1}{N}\sum_{k=1}^{N} b_k)^T.$$

Consider the following optimization problem:

$$\begin{aligned} OPT = \max_{Q\in\mathbb{R}^{3\times 3}} \quad & Tr(QM) \\ \text{subject to} \quad & QQ^T = I, \quad \det(Q) = 1. \end{aligned} \tag{2}$$

[1] A matrix Q satisfying $QQ^T = I$ is called an orthogonal matrix and has determinant ± 1.

Show that problem (1) is equivalent to problem (2) in the following sense: For any optimal solution $(\hat{Q}^\star, \hat{t}^\star)$ to (1), $\hat{Q}^\star$ is optimal to (2); conversely, for any optimal solution $Q^\star$ to (2), there exists $t^\star$ such that $(Q^\star, t^\star)$ is optimal to (1).

Hint: If we fix any feasible matrix Q in problem (1), what choice of the vector t would make the objective function of problem (1) as small as possible? (You can also use the fact that since Q is a square matrix, $QQ^T = I$ also implies $Q^TQ = I$.)

(c) Let the matrix M be defined as above and consider the following SDP

$$\begin{aligned} OPT_{SDP} = \max_{Q \in \mathbb{R}^{3\times 3}} \quad & Tr(QM) \\ \text{subject to} \quad & \begin{bmatrix} I & Q^T \\ Q & I \end{bmatrix} \succeq 0, \end{aligned} \tag{3}$$

where the positive semidefiniteness constraint is on a 6×6 block matrix. Show that $OPT_{SDP} \geq OPT$.

Hint: You may first want to show that for any matrix $Q \in \mathbb{R}^{3\times 3}$,

$$\begin{bmatrix} I & Q^T \\ Q & I \end{bmatrix} \succeq 0 \Leftrightarrow I - QQ^T \succeq 0.$$

Hint on the hint: Consider the following identity (which you can use without proof but can easily verify if you wish by expanding both sides):

$$\forall x, y \in \mathbb{R}^3, Q \in \mathbb{R}^{3\times 3}, \quad \begin{bmatrix} x^T & y^T \end{bmatrix} \begin{bmatrix} I & Q^T \\ Q & I \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \|x + Q^T y\|_2^2 \; + \; y^T(I - QQ^T)y.$$

(d) In the file `point_cloud.mat`, the matrices $A, B \in \mathbb{R}^{3\times N}$, with $N = 10000$, store two point clouds, $a_1, \ldots, a_N$ and $b_1, \ldots, b_N$, of a cup in its standard pose and observed pose, respectively. In MATLAB the data can be loaded with

```
load('point_cloud.mat')
```

And in Python the data can be loaded with

```
import scipy.io
data = scipy.io.loadmat("point_cloud.mat")
A = data["A"]; B = data["B"]
```

Solve problem (3) numerically and show that in this example we have $OPT_{SDP} = OPT$. Then use your solution of (3) to construct an optimal pair $(Q^\star, t^\star)$ for the original problem (1). Finally, produce a plot that displays the following three point clouds: (1) the standard pose $a_1, \ldots, a_N$, (2) the observed pose $b_1, \ldots, b_N$, and (3) the aligned pose $Q^\star a_1 + t^\star, \ldots, Q^\star a_N + t^\star$.

Problem 2: The risk of copying Nancy Pelosi's portfolio

There are apps that allow you to invest exactly as certain public figures do (for instance, by matching the stock portfolio of Nancy Pelosi). In this problem, we examine how risky Nancy Pelosi's portfolio could be despite some uncertainty about its covariance matrix.

Let $\mathbb{S}_+^n$ denote the set of $n \times n$ (symmetric) positive semidefinite matrices. For a portfolio in n assets with weights $w \in \mathbb{R}^n$ and covariance matrix $\Sigma^* \in \mathbb{S}_+^n$ of the asset returns over a fixed time period, we define the risk of the portfolio to be $\sqrt{w^T \Sigma^* w}$. This represents the standard deviation of the portfolio's return (viewed as a random quantity). In practice, the true covariance matrix Σ^* is unknown and only an estimate $\hat{\Sigma} \in \mathbb{S}_+^n$ may be available. One would like to account for the possibility that Σ^* belongs to an uncertainty set $\mathcal{S}$ containing plausible covariance matrices around the nominal estimate. The goal is then to calculate the *worst-case risk* of a given portfolio w:

$$R^*(w) := \max_{\Sigma \in \mathcal{S}} \sqrt{w^T \Sigma w}.$$

(a) Let $\delta > 0$ and consider the uncertainty set

$$\mathcal{S} = \left\{ \Sigma \in \mathbb{S}_+^n \;\middle|\; \sum_{i,j=1}^{n} |\Sigma_{ij} - \hat{\Sigma}_{ij}| \leq \delta \right\}.$$

Prove that the worst-case risk of a portfolio w is given by

$$R^*(w) = \sqrt{w^T \hat{\Sigma} w \;+\; \delta \left(\max_{1 \leq k \leq n} |w_k| \right)^2}.$$

(b) The file `pelosi_portfolio.mat` contains Nancy Pelosi's portfolio extracted from a live-tracking website, including:

- the tickers of $n = 11$ stocks,
- the corresponding portfolio weights $w \in \mathbb{R}^n$,
- an estimated nominal covariance matrix $\hat{\Sigma} \in \mathbb{S}^n_+$ based on historical return data.

In MATLAB, the data can be loaded with

```
load('pelosi_portfolio.mat')
```

In Python, the data can be loaded with

```
from scipy.io import loadmat
data = loadmat('pelosi_portfolio.mat')
w = data['w']
Sigma_hat = data['Sigma_hat']
```

You believe that the estimated variances of individual stocks (the diagonal entries of $\hat{\Sigma}$) are accurate, but you are less confident about the correlations (the off-diagonal entries of $\hat{\Sigma}$). To model this, consider the uncertainty set

$$\mathcal{S} = \left\{ \Sigma \in \mathbb{S}^n_+ \;\middle|\; |\Sigma_{ij} - \hat{\Sigma}_{ij}| \leq \delta\, |\hat{\Sigma}_{ij}|,\ i \neq j, \quad \Sigma_{ii} = \hat{\Sigma}_{ii},\ i = 1, \ldots, n \right\}.$$

Use CVX or CVXPY to compute the worst-case risk of Nancy Pelosi's portfolio with $\delta = 0.5$ and report the value you obtain.[2]

[2]For context, since the weights w_i correspond to fractions of your money invested in each asset, if you invest k dollars using these weights, the worst-case risk of the portfolio in dollar value would be $kR^*(w)$.

Problem 3: Demand shaping

Yonex offers $n = 10$ different tennis products and wants to nudge their prices slightly to reshape demand—for example, to better align it with their supply. We model the fractional change in demand by the linear relationship $\delta^{\text{dem}} = E\,\delta^{\text{price}}$, where $\delta^{\text{dem}} \in \mathbb{R}^n$ is the change in demand and $\delta^{\text{price}} \in \mathbb{R}^n$ is the change in price. The matrix $E \in \mathbb{R}^{n\times n}$ is called the *price-elasticity matrix* in economics and is often estimated from survey data. Yonex wants the change in demand to be close to a given target change $\delta^{\text{tar}} \in \mathbb{R}^n$, while also keeping price adjustments small. To acheieve this, it considers the following optimization problem, where $\lambda > 0$ is a given parameter:

$$\min_{\delta^{\text{price}}\in\mathbb{R}^n} \ \|E\delta^{\text{price}} - \delta^{\text{tar}}\|_2^2 \ + \ \lambda\|\delta^{\text{price}}\|_2^2. \tag{4}$$

(a) **Estimating the price-elasticity matrix:** In order to formulate the above problem, Yonex needs to first estimate the price-elasticity matrix from consumer behavior data. The file `demand_price_data.mat` contains two $n \times 15$ arrays, delta_price and delta_dem. The columns of the former contain proposed changes in price, and the columns of the latter contain the corresponding estimated changes in demand obtained from a consumer survey. (The data also contains a vector delta_tar which will be used in part (b).) In MATLAB the data can be loaded with

```
load('demand_price_data.mat')
```

In Python, the data can be loaded with

```
from scipy.io import loadmat
data = loadmat('demand_price_data.mat')
```

```
delta_price = data['delta_price']
delta_dem = data['delta_dem']
delta_tar = data['delta_tar']
```

To estimate the price-elasticity matrix, Yonex considers the following optimization problem:

$$\min_{E \in \mathbb{R}^{n \times n}} \sum_{j=1}^{15} \|E\delta^{\text{price},j} - \delta^{\text{dem},j}\|_2^2. \tag{5}$$

(i) Show that problem (5) has a unique optimal solution.

Hint: Note that we can write the objective function as

$$f(E) = \sum_{i=1}^{n} \left(\sum_{j=1}^{15} \left(r_i^T \delta^{\text{price},j} - \delta_i^{\text{dem},j} \right)^2 \right) = \sum_{i=1}^{n} f_i(r_i),$$

where $r_i \in \mathbb{R}^n$ is the i-th row of E.

(ii) Solve (5) (either by CVX/CVXPY or by explicitly deriving the solution) and report the optimal value. (You do not need to report the optimal matrix E, but you will need it part (b).)

(b) **Demand shaping using your price-elasticity matrix:**

(i) Yonex wants the change in demand to be close to a given target change $\delta^{\text{tar}} \in \mathbb{R}^n$. This vector can be found in the data file. Fix the matrix E as your optimal solution to (5) and let $\lambda = 10^{-2}$. Use CVX or CVXPY to solve (4) and report the optimal price change vector $\delta^{\text{price},*}$ for Yonex.

(ii) Instead of using CVX/CVXPY, implement steepest descent with exact line search to solve (4) with the same E and λ as in part (b). Start the algorithm at the origin and terminate when the 2-norm of the gradient of the objective function is less than 10^{-6}. What is the 2-norm of the difference between your final iterate and the vector $\delta^{\text{price},*}$ computed in the previous part?

(iii) Run the steepest descent algorithm with exact line search initialized at the origin to solve (4) with the same E matrix and δ^{tar} vector as before, but now with

$$\lambda = 10^{-4}, 2 * 10^{-4}, 3 * 10^{-4}, \ldots, 100 * 10^{-4}.$$

Plot the number of iterations required to make the 2-norm of the gradient of the objective function less than 10^{-6} as a function of λ. At an intuitive level, briefly explain the trend that you observe.

Problem 4: When the GOAT of game theory meets the GOATs of tennis

In this problem we introduce you to the notion of a Nash equilibrium and its connection to optimization.

A *bimatrix game* is a game between two players each having a finite number of actions. If Player 1 has m actions and Player 2 has n actions, then the game is fully defined by two real $m \times n$ *payoff matrices* A and B. When Player 1 plays action i and Player 2 plays action j, then the payoffs that they receive are respectively A_{ij} and B_{ij}.

In the case where Player 1 chooses to only play action i, we represent her choice by a vector $x \in \mathbb{R}^m$ which consists of all zeros except for a 1 in the i^{th} position. This is called a *"pure strategy"*. However, Player 1 can also choose to play a *"mixed strategy"*, where she chooses to randomize between her pure strategies, i.e. she plays a *convex combination* of her m pure strategies. In either case, Player 1's strategy is always an element of the "unit simplex", which by definition is the following set:

$$\Delta_m = \left\{ x \in \mathbb{R}^m | \sum_{i=1}^{m} x_i = 1, x_i \geq 0 \right\}.$$

Likewise, Player 2's strategy will be an element of Δ_n. Under a pair of mixed strategies $(x, y) \in \Delta_m \times \Delta_n$, the payoff of Player 1 is $x^T Ay$ and the payoff of Player 2 is $x^T By$.

A pair of strategies (x^*, y^*) constitutes a Nash equilibrium if it satisfies:

$$x^* \in \Delta_m,\ y^* \in \Delta_n,\ x^{*^T} Ay^* \geq x^T Ay^*\ \forall x \in \Delta_m,\ x^{*^T} By^* \geq x^{*^T} By\ \forall y \in \Delta_n. \tag{6}$$

This means that if Player 2 sticks to his strategy y^*, then there is no incentive for Player 1 to change her strategy from x^* to some other strategy, and, conversely, if Player 1 sticks to her strategy x^*, then there is no incentive for Player 2 to change his strategy from y^* to

some other strategy. Hence, we are at "equilibrium" (the Minions will not ruin Gru's lab, Gru will not cancel banana time, and life will remain good).
John Nash's Nobel prize winning result states that independent of the payoff matrices A and B, a (Nash) equilibrium always exists.[3]

(a) Show that a pair of strategies (x^*, y^*) is a Nash equilibrium if and only if it satisfies

$$x^* \in \Delta_m, y^* \in \Delta_n, {x^*}^T A y^* \geq e_i^T A y^* \; \forall i \in \{1, \ldots, m\}, {x^*}^T B y^* \geq {x^*}^T B \tilde{e}_j \; \forall j \in \{1, \ldots, n\}$$

where $e_i \in \mathbb{R}^m$ (resp. $\tilde{e}_j \in \mathbb{R}^n$) has all zeros except for a 1 in the i^{th} (resp. j^{th}) position.

(b) An important class of bimatrix games is the class of *zero-sum* games. These are games where the payoff matrices $A, B \in \mathbb{R}^{m \times n}$ satisfy $B = -A$. This means that the two players are in direct competition (any win for Player 1 is a loss for Player 2 and vice versa). The classic "rock-paper-scissors" game is a zero-sum game. Show that in a zero-sum game, a pair of strategies (x^*, y^*) is a Nash equilibrium if and only if it satisfies the following affine constraints:

$$x^* \in \Delta_m, y^* \in \Delta_n, \; {x^*}^T A \tilde{e}_j \geq e_i^T A y^* \; \forall \{i, j\} \in \{1, \ldots, m\} \times \{1, \ldots, n\}.$$

Again, $e_i \in \mathbb{R}^m$ (resp. $\tilde{e}_j \in \mathbb{R}^n$) has all zeros except for a 1 in the i^{th} (resp. j^{th}) position.

(c) Consider a zero-sum game between *Roger Federer* (row player) and *Rafael Nadal* (column player). Let's assume that each player is considering 4 play-styles (actions): AB (aggressive baseline), AV (approach/volley), HB (heavy to the backhand), and S (slice).

Let $A \in \mathbb{R}^{4 \times 4}$ be the payoff matrix for Federer, where A_{ij} is his excess points won over Nadal per 100 points if he uses the action in row i and Nadal uses the action in column j, both following the ordering $\{\text{AB}, \text{AV}, \text{HB}, \text{S}\}$. Hence, Nadal's payoff matrix is $B = -A$.

$$A = \begin{pmatrix} 8 & 2 & -9 & 5 \\ -2 & 12 & -8 & 6 \\ 3 & 7 & -6 & -9 \\ -3 & -5 & 4 & 4 \end{pmatrix}, \qquad B = -A.$$

Find a Nash equilibrium (x^*, y^*) of this zero-sum game.

(d) Is the Nash equilibrium you found in part (c) unique? Why or why not?

[3] In fact, he proved the result more generally for games among a finite number of players.