1. Drawing a graph. We consider the problem of drawing (in two dimensions) a graph with $n$ vertices (or nodes) and $m$ undirected edges (or links). This just means assigning an $x$- and a $y$-coordinate to each node. We let $x \in \mathbb{R}^n$ be the vector of $x$- coordinates of the nodes, and $y \in \mathbb{R}^n$ be the vector of $y$- coordinates of the nodes. When we draw the graph, we draw node $i$ at the location $(x_i, y_i) \in \mathbb{R}^2$. The problem, of course, is to make the drawn graph look good.

One goal is that neighboring nodes on the graph (i.e., ones connected by an edge) should not be too far apart as drawn. To take this into account, we will choose the $x$- and $y$-coordinates so as to minimize the objective

$$ J = \sum_{i<j, i \sim j} \left( (x_i - x_j)^2 + (y_i - y_j)^2 \right), $$

where $i \sim j$ means that nodes $i$ and $j$ are connected by an edge. The objective $J$ is precisely the sum of the squares of the lengths (in $\mathbb{R}^2$) of the drawn edges of the graph. We have to introduce some other constraints into our problem to get a sensible solution. First of all, the objective $J$ is not affected if we shift all the coordinates by some fixed amount (since $J$ only depends on differences of the $x$- and $y$-coordinates). So we can assume that

$$ \sum_{i=1}^n x_i = 0, \quad \sum_{i=1}^n y_i = 0, $$

i.e., the sum (or mean value) of the $x$- and $y$-coordinates is zero. These two equations ‘center’ our drawn graph. Another problem is that we can minimize $J$ by putting all the nodes at $x_i = 0$, $y_i = 0$, which results in $J = 0$. To force the nodes to spread out, we impose the constraints

$$ \sum_{i=1}^n x_i^2 = 1, \quad \sum_{i=1}^n y_i^2 = 1, \quad \sum_{i=1}^n x_i y_i = 0. $$

The first two say that the variance of the $x$- and $y$-coordinates is one; the last says that the $x$- and $y$-coordinates are uncorrelated. (You don’t have to know what variance or uncorrelated mean; these are just names for the equations given above.) The three equations above enforce ‘spreading’ of the drawn graph. Even with these constraints, the coordinates that minimize $J$ are not unique. For example, if $x$ and $y$ are any set of coordinates, and $Q \in \mathbb{R}^{2 \times 2}$ is any orthogonal matrix, then the coordinates given by

$$ \left[ \tilde{x}_i \right] = Q \left[ x_i \right], \quad \tilde{y}_i = y_i, \quad i = 1, \ldots, n $$

would also minimize $J$. However, we can choose $Q$ so that $\tilde{x}_i$ and $\tilde{y}_i$ are not all positive or all negative, which would make the drawn graph look better.
satisfy the centering and spreading constraints, and have the same value of $J$. This means
that if you have a proposed set of coordinates for the nodes, then by rotating or reflecting
them, you get another set of coordinates that is just as good, according to our objective. We’ll
just live with this ambiguity. Here’s the question:

a) Explain how to solve this problem, i.e., how to find $x$ and $y$ that minimize $J$ subject
to the centering and spreading constraints, given the graph topology. You can use any
method or ideas we’ve encountered in the course. Be clear as to whether your approach
solves the problem exactly (i.e., finds a set of coordinates with $J$ as small as it can
possibly be), or whether it’s just a good heuristic (i.e., a choice of coordinates that
achieves a reasonably small value of $J$, but perhaps not the absolute best). In describing
your method, you may not refer to any programming commands or operators; your
description must be entirely in mathematical terms.

b) Implement your method, and carry it out for the graph given in \texttt{dg_data.json}. This
JSON file contains the node adjacency matrix of the graph, denoted $A$, and defined
as $A_{ij} = 1$ if nodes $i$ and $j$ are connected by an edge, and $A_{ij} = 0$ otherwise. (The
graph is undirected, so $A$ is symmetric. Also, we do not have self-loops, so $A_{ii} = 0$, for
$i = 1, \ldots, n$.) Give the value of $J$ achieved by your choice of $x$ and $y$, and verify that
your $x$ and $y$ satisfy the centering and spreading conditions, at least approximately. If
your method is iterative, plot the value of $J$ versus iteration. Draw the corresponding
down

graph by plotting nodes as small circles and edges as lines. For comparison, the JSON
file also contains the vectors $x_{circ}$ and $y_{circ}$. These coordinates were obtained using
a standard technique for drawing a graph, by placing the nodes in order on a circle. The
radius of the circle has been chosen so that $x_{circ}$ and $y_{circ}$ satisfy the centering and
spread constraints. Draw this graph on a separate plot.

**Hint.** You are welcome to use the results described below, without proving them. Let $A \in \mathbb{R}^{n \times n}$ be symmetric, with eigenvalue decomposition
$A = \sum_{i=1}^{n} \lambda_i q_i q_i^T$, with $\lambda_1 \geq \cdots \geq \lambda_n$, and \{$q_1, \ldots, q_n$\} orthonormal. You know that a solution of the
problem

\[
\begin{align*}
\text{minimize} \quad & x^T A x \\
\text{subject to} \quad & x^T x = 1,
\end{align*}
\]

where the variable is $x \in \mathbb{R}^n$, is $x = q_n$. The related maximization problem is

\[
\begin{align*}
\text{maximize} \quad & x^T A x \\
\text{subject to} \quad & x^T x = 1
\end{align*}
\]

with variable $x \in \mathbb{R}^n$. A solution to this problem is $x = q_1$. Now consider the following
generalization of the first problem:

\[
\begin{align*}
\text{minimize} \quad & \text{trace}(X^T A X) \\
\text{subject to} \quad & X^T X = I_k
\end{align*}
\]

where the variable is $X \in \mathbb{R}^{n \times k}$, and $I_k$ denotes the $k \times k$ identity matrix, and we assume
$k \leq n$. The constraint means that the columns of $X$, say, $x_1, \ldots, x_k$, are orthonormal; the
objective can be written in terms of the columns of $X$ as $\text{trace}(X^TAX) = \sum_{i=1}^{k} x_i^T A x_i$. A solution of this problem is $X = [q_{n-k+1} \cdots q_n]$. Note that when $k = 1$, this reduces to the first problem above. The related maximization problem is

$$\begin{align*}
\text{maximize} & \quad \text{trace}(X^TAX) \\
\text{subject to} & \quad X^T X = I_k
\end{align*}$$

with variable $X \in \mathbb{R}^{n \times k}$. A solution of this problem is $X = [q_1 \cdots q_k]$. 

2. A heuristic for MAXCUT. Consider a graph with $n$ nodes and $m$ edges, with the nodes labeled $1, \ldots, n$ and the edges labeled $1, \ldots, m$. We partition the nodes into two groups, $B$ and $C$, i.e., $B \cap C = \emptyset$, $B \cup C = \{1, \ldots, n\}$. We define the number of cuts associated with this partition as the number of edges between pairs of nodes when one of the nodes is in $B$ and the other is in $C$. A famous problem, called the MAXCUT problem, involves choosing a partition (i.e., $B$ and $C$) that maximizes the number of cuts for a given graph. For any partition, the number of cuts can be no more than $m$. If the number of cuts is $m$, nodes in group $B$ connect only to nodes in group $C$ and the graph is bipartite.

The MAXCUT problem has many applications. We describe one here, although you do not need it to solve this problem. Suppose we have a communication system that operates with a two-phase clock. During periods $t = 0, 2, 4, \ldots$, each node in group $B$ transmits data to nodes in group $C$ that it is connected to; during periods $t = 1, 3, 5, \ldots$, each node in group $C$ transmits to the nodes in group $B$ that it is connected to. The number of cuts, then, is exactly the number of successful transmissions that can occur in a two-period cycle. The MAXCUT problem is to assign nodes to the two groups so as to maximize the overall efficiency of communication.

It turns out that the MAXCUT problem is hard to solve exactly, at least if we don’t want to resort to an exhaustive search over all, or most of, the $2^{n-1}$ possible partitions. In this problem we explore a sophisticated heuristic method for finding a good (if not the best) partition in a way that scales to large graphs.

We will encode the partition as a vector $x \in \mathbb{R}^n$, with $x_i \in \{-1, 1\}$. The associated partition has $x_i = 1$ for $i \in B$ and $x_i = -1$ for $i \in C$. We describe the graph by its node adjacency matrix $A \in \mathbb{R}^{n \times n}$, with

$$A_{ij} = \begin{cases} 
1 & \text{there is an edge between node } i \text{ and node } j \\
0 & \text{otherwise}
\end{cases}$$

Note that $A$ is symmetric and $A_{ii} = 0$ (since we do not have self-loops in our graph).

a) Find a symmetric matrix $P$, with $P_{ii} = 0$ for $i = 1, \ldots, n$, and a constant $d$, for which $x^T P x + d$ is the number of cuts encoded by any partitioning vector $x$. Explain how to calculate $P$ and $d$ from $A$. Of course, $P$ and $d$ cannot depend on $x$.

The MAXCUT problem can now be stated as the optimization problem

$$\begin{align*}
\text{maximize} & \quad x^T P x + d \\
\text{subject to} & \quad x_i^2 = 1, \quad i = 1, \ldots, n,
\end{align*}$$

with variable $x \in \mathbb{R}^n$. 

b) A famous heuristic for approximately solving MAXCUT is to replace the $n$ constraints $x_i^2 = 1$, $i = 1, \ldots, n$, with a single constraint $\sum_{i=1}^{n} x_i^2 = n$, creating the so-called relaxed problem

$$\begin{align*}
\text{maximize} & \quad x^T P x + d \\
\text{subject to} & \quad \sum_{i=1}^{n} x_i^2 = n.
\end{align*}$$

Explain how to solve this relaxed problem (even if you could not solve part (a)).

Let $x^*$ be a solution to the relaxed problem. We generate our candidate partition with $x_i = \text{sign}(x_i^*)$. (This means that $x_i = 1$ if $x_i^* \geq 0$, and $x_i = -1$ if $x_i^* < 0$.)

Remark: We can give a geometric interpretation of the relaxed problem, which will also explain why it’s called relaxed. The constraints in the problem in part (a), that $x_i^2 = 1$, require $x$ to lie on the vertices of the unit hypercube. In the relaxed problem, the constraint set is the unit ball of unit radius. Because this constraint set is larger than the original constraint set (i.e., it includes it), we say the constraints have been relaxed.

c) Run the MAXCUT heuristic described in part (b) on the data given in `max_cut_data.json`. How many cuts does your partition yield?

A simple alternative to MAXCUT is to generate a large number of random partitions, using the random partition that maximizes the number of cuts as an approximate solution. Carry out this method with 1000 random partitions generated by $x = \text{sign}(\text{rand}(n, 1) - 0.5)$. What is the largest number of cuts obtained by these random partitions?

Note: There are many other heuristics for approximately solving the MAXCUT problem. However, we are not interested in them. In particular, please do not submit any other method for approximately solving MAXCUT.

3. Simultaneously estimating student ability and exercise difficulty. Each of $n$ students takes an exam that contains $m$ questions. Student $j$ receives (nonnegative) grade $G_{ij}$ on question $i$. One simple model for predicting the grades is to estimate $G_{ij} \approx \hat{G}_{ij} = a_j/d_i$, where $a_j$ is a (nonnegative) number that gives the ability of student $j$, and $d_i$ is a (positive) number that gives the difficulty of exam question $i$. Given a particular model, we could simultaneously scale the student abilities and the exam difficulties by any positive number, without affecting $\hat{G}_{ij}$. Thus, to ensure a unique model, we will normalize the exam question difficulties $d_i$, so that the mean exam question difficulty across the $m$ questions is 1.

In this problem, you are given a complete set of grades (i.e., the matrix $G \in \mathbb{R}^{m \times n}$). Your task is to find a set of nonnegative student abilities, and a set of positive, normalized question difficulties, so that $G_{ij} \approx \hat{G}_{ij}$. In particular, choose your model to minimize the RMS error, $J$,

$$J = \left( \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} (G_{ij} - \hat{G}_{ij})^2 \right)^{1/2}.$$
This can be compared to the RMS value of the grades,

\[ \left( \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} G_{ij}^2 \right)^{1/2} \]

a) Explain how to solve this problem, using any concepts from EE263. If your method is approximate, or not guaranteed to find the global minimum value of \( J \), say so. If carrying out your method requires some rank or other conditions to hold, say so.

Note: You do not have to concern yourself with the requirement that \( a_j \) are nonnegative and \( d_i \) are positive. You can just assume this works out, or is easily corrected.

b) Carry out your method on the data found in grade_data.json. Give the optimal value of \( J \), and also express it as a fraction of the RMS value of the grades. Give the difficulties of the 7 problems on the exam.

4. Square matrices and the SVD. Let \( A \) be an \( n \times n \) real matrix. State whether each of the following statements is true or false. Do not give any explanation or show any work.

\[ A \]

a) If \( x \) is an eigenvector of \( A \), then \( x \) is either a left or right singular vector of \( A \)

b) If \( \lambda \) is an eigenvalue of \( A \), then \( |\lambda| \) is a singular value

c) If \( A \) is symmetric, then every singular value of \( A \) is also an eigenvalue of \( A \)

d) If \( A \) is symmetric, then every singular vector of \( A \) is also an eigenvector of \( A \)

e) If \( A \) is symmetric with the following singular value decomposition

\[ A = U \Sigma V^T \]

then \( U = V \)

f) If \( A \) is invertible, then \( \sigma_i \neq 0 \) for all \( i = 1, \ldots, n \)

5. Identification of gene regulatory networks. Consider a gene regulatory network (GRN) with \( n \) genes labeled \( 1, \ldots, n \). The expression level of each gene is associated with the concentration of a specific mRNA sequence. Since we can measure the concentrations of mRNA sequences using oligonucleotide microarrays, we will use the mRNA concentrations as proxies for the expression levels of the corresponding genes.

Let \( x_i(t) \) denote the concentration of the mRNA sequence associated with the \( i \)th gene. A simple model for the dynamics of the GRN is

\[ \dot{x}_i(t) = -\alpha_i x_i(t) + \sum_{j \neq i} W_{ij} x_j(t) + \beta_i(t) + \epsilon_i(t), \quad i = 1, \ldots, n, \]

where \( x_i(t) \) is the concentration of the mRNA sequence associated with the expression level of the \( i \)th gene at time \( t \), \( W_{ij} \) is the coupling constant from the \( j \)th gene to the \( i \)th gene, \( \alpha_i \) is
the self-regulation rate of the $i$th gene, $\beta_i(t)$ is the external stimulus applied to the $i$th gene at time $t$, and $\epsilon_i(t)$ is an error term that includes measurement errors and unmodeled effects. (We assume that $\epsilon_i(t)$ is small.)

a) Suppose we apply a stimulus sequence to the GRN, and measure $\beta(1), \ldots, \beta(T), x(1), \ldots, x(T)$ and $\dot{x}(1), \ldots, \dot{x}(T)$. Explain how to use the measurements to estimate the model parameters $\alpha_i$ and $W_{ij}$.

b) Apply your method to the data given in grn_data.json. Report the gene with the highest self-regulation rate, as well as the corresponding self-regulation rate. We say that the genes $i$ and $j$ are strongly coupled if $|W_{ij}|$ is large. Which two genes are most strongly coupled? What is the corresponding coupling constant?

c) Suppose the external stimulus applied to the system is constant over time: that is, there exists a vector $\bar{\beta} \in \mathbb{R}^n$ such that $\beta(t) = \bar{\beta}$ for all $t$. Derive a closed-form expression for $x(t)$ in terms of $x(0)$, $\alpha$, $W$, and $\bar{\beta}$.

d) Simulate the system using $\bar{\beta} = \beta(1)$ (the value of $\beta(t)$ at time $t = 1$ in the experimental data), the value of $x(0)$ given in the data file, and the model parameters that you estimated above. Plot $x_j(t)$ versus time for $0 \leq t \leq T_{\text{sim}} = 10$ and $j = 1, 2, 3$. (Put all of your plots on the same set of axes.)