1. Minimum energy control. Consider the discrete-time linear dynamical system

\[ x(t + 1) = Ax(t) + Bu(t), \quad t = 0, 1, 2, \ldots \]

where \( x(t) \in \mathbb{R}^n \), and the input \( u(t) \) is a scalar (hence, \( A \in \mathbb{R}^{n \times n} \) and \( B \in \mathbb{R}^{n \times 1} \)). The initial state is \( x(0) = 0 \).

a) Find the matrix \( C_T \) such that

\[ x(T) = C_T \begin{bmatrix} u(T - 1) \\ \vdots \\ u(1) \\ u(0) \end{bmatrix}. \]

b) For the remainder of this problem, we consider a specific system with \( n = 4 \). The dynamics and input matrices are

\[
A = \begin{bmatrix}
0.5 & 0.7 & -0.9 & -0.5 \\
0.4 & -0.7 & 0.1 & 0.3 \\
0.7 & 0.0 & -0.6 & 0.1 \\
0.4 & -0.1 & 0.8 & -0.5
\end{bmatrix}, \quad B = \begin{bmatrix}
1 \\
1 \\
0 \\
0
\end{bmatrix}.
\]

Suppose we want the state to be \( x_{\text{des}} \) at time \( T \). Consider the desired state

\[ x_{\text{des}} = \begin{bmatrix}
0.8 \\
2.3 \\
-0.7 \\
-0.3
\end{bmatrix}. \]

What is the smallest \( T \) for which we can find inputs \( u(0), \ldots, u(T - 1) \), such that \( x(T) = x_{\text{des}} \)? What are the corresponding inputs that achieve \( x_{\text{des}} \) at this minimum time? What is the smallest \( T \) for which we can find inputs \( u(0), \ldots, u(T - 1) \), such that \( x(T) = x_{\text{des}} \) for any \( x_{\text{des}} \in \mathbb{R}^4 \)? We’ll denote this \( T \) by \( T_{\text{min}} \).

c) Suppose the energy expended in applying inputs \( u(0), \ldots, u(T - 1) \) is

\[ E(T) = \sum_{t=0}^{T-1} (u(t))^2. \]
For a given \( T \) (greater than \( T_{\min} \)) and \( x_{\text{des}} \), how can you compute the inputs which achieve \( x(T) = x_{\text{des}} \) with the minimum expense of energy? Consider now the desired state

\[
x_{\text{des}} = \begin{bmatrix} -1 \\ 1 \\ 0 \\ 1 \end{bmatrix}.
\]

For each \( T \) ranging from \( T_{\min} \) to 30, find the minimum energy inputs that achieve \( x(T) = x_{\text{des}} \). For each \( T \), evaluate the corresponding input energy, which we denote by \( E_{\text{min}}(T) \). Plot \( E_{\text{min}}(T) \) as a function of \( T \). (You should include in your solution a description of how you computed the minimum-energy inputs, and the plot of the minimum energy as a function of \( T \). But you don’t need to list the actual inputs you computed!)

d) You should observe that \( E_{\text{min}}(T) \) is non-increasing in \( T \). Show that this is the case in general (i.e., for any \( A, B \), and \( x_{\text{des}} \)).

**Note:** There is a direct way of computing the asymptotic limit of the minimum energy as \( T \to \infty \). We’ll cover these ideas in more detail in **ee363**.

**Solution.**

a) Since \( x(0) = 0 \),

\[
x(1) = Ax(0) + Bu(0)
= Bu(0),
\]

\[
x(2) = Ax(1) + Bu(1) = A (Bu(0)) + Bu(1)
= ABu(0) + Bu(1),
\]

\[
x(3) = Ax(2) + Bu(2) = A (ABu(0) + Bu(1)) + Bu(2)
= A^2Bu(0) + A Bu(1) + Bu(2),
\]

\[
\vdots
\]

\[
x(T) = A^{T-1}Bu(0) + \cdots + A^2Bu(T-3) + ABu(T-2) + Bu(T-1).
\]

Therefore,

\[
x(T) = Bu(T-1) + ABu(T-2) + A^2Bu(T-3) + \cdots + A^{T-1}Bu(0)
\]

\[
= \begin{bmatrix} B & AB & A^2B & \cdots & A^{T-1}B \end{bmatrix} \begin{bmatrix} u(T-1) \\ u(T-2) \\ u(T-3) \\ \vdots \\ u(0) \end{bmatrix}
\]

Hence,

\[
C_T = \begin{bmatrix} B & AB & A^2B & \cdots & A^{T-1}B \end{bmatrix}
\]

2
b) To find the inputs $u(0), \ldots, u(T-1)$, such that $x(T) = x_{\text{des}}$, $x_{\text{des}}$ has to be in the range of $C_T$. A quick way to check this is to see whether

$$\text{rank}([C_T \ x_{\text{des}}]) = \text{rank}(C_T).$$

i. For $T = 1$, $C_T = B = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}$. Clearly $x_{\text{des}}$ is not in the range of $C_T$.

ii. For $T = 2$, $C_T = [B \ AB] = \begin{bmatrix} 1 & 1.2 \\ 1 & -0.3 \\ 0 & 0.7 \\ 0 & 0.3 \end{bmatrix}$, which has rank two. Since

$$\begin{bmatrix} 0.8 & 1 & 1.2 \\ 2.3 & 1 & -0.3 \\ -0.7 & 0 & 0.7 \\ -0.3 & 0 & 0.3 \end{bmatrix}$$

also has rank two, there are $u(0)$ and $u(1)$, such that $x_{\text{des}} = C_T \begin{bmatrix} u(1) \\ u(0) \end{bmatrix}$. These inputs can easily be found (either by hand or with matlab, e.g., $u=C\backslash x_{\text{des}}$). The result is $\begin{bmatrix} 2 \\ -1 \end{bmatrix}$.

Therefore the smallest $T$ is 2 and the corresponding inputs are $u(0) = -1$ and $u(1) = 2$. And indeed, $x_{\text{des}} = \begin{bmatrix} 0.8 \\ 2.3 \\ -0.7 \\ -0.3 \end{bmatrix} = \begin{bmatrix} 1 & 1.2 \\ 1 & -0.3 \\ 0 & 0.7 \\ 0 & 0.3 \end{bmatrix} \begin{bmatrix} 2 \\ -1 \end{bmatrix}$.

To find the inputs $u(0), \ldots, u(T-1)$, such that $x(T) = x_{\text{des}}$, for any $x_{\text{des}} \in \mathbb{R}^4$, the range of $C_T$ has to be $\mathbb{R}^4$. If $\mathcal{R}(C_T)$ is $\mathbb{R}^4$, $C_T$ has to have four independent columns. So we can start by checking $C_T$ for $T = 4$.

$$C_T = \begin{bmatrix} 1.000 & 1.200 & -0.390 & -0.465 \\ 1.000 & -0.300 & 0.850 & -0.430 \\ 0.000 & 0.700 & 0.450 & -0.451 \\ 0.000 & 0.300 & 0.920 & -0.341 \end{bmatrix}$$

We check the rank of $C_T$ using the following matlab code, and obtain $\text{rank}(C_T) = 4$. Therefore the four column vectors $\in \mathbb{R}^4$ of $C_T$ are independent, so $C_T$ is invertible and $\mathcal{R}(C_T)$ is $\mathbb{R}^4$. Therefore $T_{\text{min}} = 4$. Now you can find $[u(3) \ u(2) \ u(1) \ u(0)]^T = C_T^{-1} x_{\text{des}}$ for any $x_{\text{des}} \in \mathbb{R}^4$.

$$A=[ 0.5 \ 0.7 \ -0.9 \ -0.5; \\ 0.4 \ -0.7 \ 0.1 \ 0.3; \\ 0.7 \ 0.0 \ -0.6 \ 0.1; \\ 0.4 \ -0.1 \ 0.8 \ -0.5 ];$$

$$B=[ 1 \ 1 \ 0 \ 0 ];$$

$$C4=[ B \ A*B \ A^{-2}*B \ A^{-3}*B ];$$

$$\text{rank}(C4) = 4.$$
Note: We can check whether the vector $x_{des}$ is in the range of $C_T$ with the following matlab code.

\[
\text{rank}([ B ]) \quad \text{rank}([ B \ x_{des} ]) \quad \text{rank}([ B \ AB ]) \quad \text{rank}([ B \ AB \ x_{des} ])
\]

The results are $\text{rank}(B) = 1$, $\text{rank}([ B \ x_{des} ]) = 2$, $\text{rank}([ B \ AB ]) = 2$, and $\text{rank}([ B \ AB \ x_{des} ]) = 2$. Therefore $x_{des}$ is in the range of $[ B \ AB ]$ while it is not in the range of $B$.

c) Let

\[
u_T = \begin{bmatrix} u(T - 1) \\ \vdots \\ u(1) \\ u(0) \end{bmatrix}
\]

Then $\|u_T\|^2 = E(T) = \sum_{t=0}^{T-1} (u(t))^2$. We want to find the $u_T$ with minimum $\|u_T\|^2$ from among the solutions of $x_{des} = C_T u_T$. Therefore the minimum energy input, $u_T$, is the minimum-norm solution for the underdetermined linear equations since $T \geq T_{\text{min}}$ so $C_T$ is full rank and fat. Hence the minimum energy inputs are the minimum-norm solution, $u_{T,\text{min}} = C_T^{-1} C_T^T x_{des}$.

For each $T$ ranging from $T_{\text{min}}$ to 30, we obtain $C_T$ as shown in part (a), obtain the minimum energy inputs by $u_{T,\text{min}} = C_T^{-1} C_T^T x_{des}$, and evaluate the minimum energy, $E_{\text{min}}(T) = \|u_{T,\text{min}}\|^2$. Finally we can plot $E_{\text{min}}(T)$ as a function of $T$. We can do this easily by the following matlab code.

```matlab
clear;
A=[ 0.5 0.7 -0.9 -0.5;
    0.4 -0.7 0.1 0.3;
    0.7 0.0 -0.6 0.1;
    0.4 -0.1 0.8 -0.5 ];
B=[ 1 1 0 0 ]';
x_des=[-1 1 0 1]';
Tmin=4; Tmax=30; E_min=[]; C_T=[B A*B A^2*B];
for T = Tmin:Tmax
    C_T=[B A*C_T];
    u_min = pinv(C_T)*x_des; % minimum energy inputs
    E_min = [E_min norm(u_min)^2]; % minimum energy
end
figure(1);
plot(Tmin:Tmax,E_min,'-',Tmin:Tmax,E_min,'o');
title('Minimum Energy Control');
xlabel('T');
ylabel('E_{\text{min}}(T)');
```
The following plot shows $E_{\text{min}}(T)$ as a function of $T$.

d) (We’ll use $\tau$ for $T$, to avoid confusion with the transpose operation!) Let $u_{\text{min}}(0), u_{\text{min}}(1), \ldots, u_{\text{min}}(\tau - 1)$ be the minimum energy inputs that achieve $x(\tau) = x_{\text{des}}$, with $\tau \geq T_{\text{min}}$. Then the inputs $u(0) = 0, u(1) = u_{\text{min}}(0), u(2) = u_{\text{min}}(1), \ldots, u(\tau) = u_{\text{min}}(\tau - 1)$ achieve the same $x(\tau + 1) = x_{\text{des}}$ since

$$x(\tau + 1) = C_{\tau + 1} \begin{bmatrix} u(\tau) \\ \vdots \\ u(1) \\ u(0) \end{bmatrix} = \begin{bmatrix} C_{\tau} & A^T B \end{bmatrix} \begin{bmatrix} u_{\text{min}}(\tau - 1) \\ \vdots \\ u_{\text{min}}(0) \\ 0 \end{bmatrix} = C_{\tau} u_{\tau,\text{min}} = x(\tau) = x_{\text{des}},$$

and have the same input energy since $\sum_{t=0}^{\tau} (u(t))^2 = \sum_{t=0}^{\tau-1} (u_{\text{min}}(t))^2$. Then $E_{\text{min}}(\tau + 1) \leq \sum_{t=0}^{\tau} (u(t))^2 = \sum_{t=0}^{\tau-1} (u_{\text{min}}(t))^2 = E_{\text{min}}(\tau)$. Therefore $E_{\text{min}}(\tau)$ is non-decreasing in $\tau$ for any $A, B$, and $x_{\text{des}}$ as long as $x_{\text{des}}$ is achievable. Note: Many people tried to show this by saying that $C_{\tau + 1} C_{\tau + 1}^T$ is “bigger than” $C_{\tau} C_{\tau}^T$, hence $(C_{\tau} C_{\tau}^T)^{-1}$ “decreases” with $\tau$. The problem is, what does it mean for a matrix to be bigger than another? For instance, between

$$\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$
which is bigger? The result can in fact be proved this way, but you’d have to know about positive-definite matrices and matrix inequalities... (And, actually, you’ll see these ideas later on in the course.)

2. Gram matrices. Given functions $f_i : [a, b] \to \mathbb{R}$, $i = 1, \ldots, n$, the Gram matrix $G \in \mathbb{R}^{n \times n}$ associated with them is defined by

$$G_{ij} = \int_a^b f_i(t) f_j(t) \, dt.$$ 

a) Show that $G = G^T \geq 0$. Note: $G \geq 0$ means that $G$ is a positive semidefinite matrix. A matrix $G \in \mathbb{R}^{n \times n}$ is positive semidefinite if we have $x^T G x \geq 0$ for any $x \in \mathbb{R}^n$. Equivalently, matrix $G$ is positive semidefinite if all of its eigenvalues are non-negative.

b) Show that $G$ is singular if and only if the functions $f_1, \ldots, f_n$ are linearly dependent.

Solution.

a) First of all it is obvious that $G = G^T$ because

$$G_{ij} = \int_a^b f_i(t) f_j(t) \, dt = \int_a^b f_j(t) f_i(t) \, dt = G_{ji}.$$ 

Define the vector function $f : [a, b] \to \mathbb{R}^n$ as $f(t) = [f_1(t) \ f_2(t) \ \cdots \ f_n(t)]^T$. Hence

$$G = \int_a^b f(t) f(t)^T \, dt$$ 

since

$$f(t) f(t)^T = \begin{bmatrix} f_1(t) \\ f_2(t) \\ \vdots \\ f_n(t) \end{bmatrix} \begin{bmatrix} f_1(t) & f_2(t) & \cdots & f_n(t) \end{bmatrix} = \begin{bmatrix} f_1(t) f_1(t) & f_1(t) f_2(t) & \cdots & f_1(t) f_n(t) \\ f_2(t) f_1(t) & f_2(t) f_2(t) & \cdots & f_2(t) f_n(t) \\ \vdots & \vdots & \ddots & \vdots \\ f_n(t) f_1(t) & f_n(t) f_2(t) & \cdots & f_n(t) f_n(t) \end{bmatrix}.$$ 

Now to show that $G \geq 0$, we consider any $x = [x_1 \ \cdots \ x_n]^T \in \mathbb{R}^n$ and verify that $x^T G x \geq 0$. We have

$$x^T G x = x^T (\int_a^b f(t) f(t)^T \, dt) x = \int_a^b x^T f(t) f(t)^T x \, dt = \int_a^b (x^T f(t))^2 \, dt,$$

and therefore $x^T G x \geq 0$ since it is the area under the nonnegative function $(x^T f(t))^2$ from $a$ to $b$. 

6
b) Note that if $G \geq 0$ is nonsingular it does not have any zero eigenvalues (in fact all eigenvalues should be strictly positive) and therefore $x^T G x > 0$ for all $x$ or $G > 0$. We show that if $f_1, \ldots, f_n$ are linearly independent then $G > 0$ and vice versa. In the previous part we showed that

$$x^T G x = \int_a^b (x^T f(t))^2 \, dt = \int_a^b (x_1 f_1(t) + \cdots + x_n f_n(t))^2 \, dt.$$  

Therefore, $x^T G x > 0$ for all $x$ as long as no linear combination of the functions $f_1, \ldots, f_n$ is zero, i.e., $x_1 f_1(t) + \cdots + x_n f_n(t) \neq 0$. In other words, $G > 0$ if and only if the functions $f_1, \ldots, f_n$ are linearly independent and we are done.

3. Properties of symmetric matrices. In this problem $P$ and $Q$ are symmetric matrices. For each statement below, either give a proof or a specific counterexample. Note: We say $A \geq B$ if $A - B \geq 0$ and $A \leq B$ if $-A \geq 0$.

a) If $P \geq 0$ then $P + Q \geq Q$.

b) If $P \geq Q$ then $-P \leq -Q$.

c) If $P > 0$ then $P^{-1} > 0$.

d) If $P \geq Q$ then $P^2 \geq Q^2$.

Solution.

a) By definition, $A \geq B$ if and only if $A - B \geq 0$. So, if $P \geq 0$, then $P + Q - Q \geq 0$ and therefore $P + Q \geq Q$.

b) If $P \geq Q$ then $P - Q \geq 0$, and by definition $-(P - Q) \leq 0$ or $-P + Q \leq 0$ so finally $-Q \geq -P$.

c) If $P > 0$ then all eigenvalues of $P$ are strictly positive and $P^{-1}$ exists. If $\lambda_1, \ldots, \lambda_n > 0$ are the eigenvalues of $P$ then the eigenvalues of $P^{-1}$ are $1/\lambda_1, \ldots, 1/\lambda_n$. Since $\lambda_i > 0$ then $1/\lambda_i > 0$ so the eigenvalues of $P^{-1}$ are all positive and therefore $P^{-1} > 0$.

d) The statement is false. A simple counterexample is $P = -1$ and $Q = -2$.

4. Frobenius norm of a matrix. The Frobenius norm of a matrix $A \in \mathbb{R}^{n \times n}$ is defined as

$$\|A\|_F = \sqrt{\text{trace} A^T A}.$$  

(Recall trace is the trace of a matrix, i.e., the sum of the diagonal entries.)

a) Show that

$$\|A\|_F = \left( \sum_{i,j} |A_{ij}|^2 \right)^{1/2}.$$  

Thus the Frobenius norm is simply the Euclidean norm of the matrix when it is considered as an element of $\mathbb{R}^{n^2}$. Note also that it is much easier to compute the Frobenius norm of a matrix than the (spectral) norm (i.e., maximum singular value).
b) Show that if $U$ and $V$ are orthogonal, then $\|UA\|_F = \|AV\|_F = \|A\|_F$. Thus the Frobenius norm is not changed by a pre- or post- orthogonal transformation.

c) Show that $\|A\|_F = \sqrt{\sigma_1^2 + \cdots + \sigma_r^2}$, where $\sigma_1, \ldots, \sigma_r$ are the singular values of $A$. Then show that $\max(A) \leq \|A\|_F \leq \sqrt{r} \max(A)$. In particular, $\|Ax\| \leq \|A\|_F \|x\|$ for all $x$.

Solution.

a) Simply by definition

$$\|A\|_F^2 = \text{trace}(A^T A) = \sum_i [A^T A]_{ii} = \sum_i \left( \sum_j A_{ij} A_{ji} \right) = \sum_{i,j} A_{ij}^2.$$  

b) First note that $\|UA\|_F = \|A\|_F$ because

$$\|UA\|_F^2 = \text{trace}(UA)^T (UA) = \text{trace}(A^T U^T U A) = \text{trace}(A^T A) = \|A\|_F^2,$$

and $\|AV\|_F = \|A\|_F$ since

$$\|AV\|_F^2 = \text{trace}(AV)^T (AV) = \text{trace}(AV)(AV)^T = \text{trace}(AVV^T A^T) = \text{trace}(AA^T) = \text{trace}(A^T A) = \|A\|_F^2,$$

where we have used the fact that $\text{trace}(XY) = \text{trace}(YX)$.

c) We start with the full SVD of $A = U\Sigma V^T$. By the previous problem,

$$\|A\|_F = \|U^T \Sigma V\|_F = \|\Sigma V\|_F = \|\Sigma\|_F = \sqrt{\sigma_1^2 + \cdots + \sigma_r^2}.$$  

Since $\sigma_2^2, \ldots, \sigma_r^2 \geq 0$, we have $\sigma_1 \leq \sqrt{\sigma_1^2 + \cdots + \sigma_r^2} = \|A\|_F$. Next, since $\sigma_2, \ldots, \sigma_r \leq \sigma_1$, we have $\|A\|_F = \sqrt{\sigma_1^2 + \cdots + \sigma_r^2} \leq \sqrt{\sigma_1^2 + \cdots + \sigma_1^2} = \sqrt{r} \sigma_1$.

5. 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

\[
\begin{bmatrix}
\tilde{x}_i \\
\tilde{y}_i
\end{bmatrix} = Q \begin{bmatrix}
x_i \\
y_i
\end{bmatrix}, \quad i = 1, \ldots, n
\]

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 graph by plotting nodes as small circles and edges as lines. For comparison, the JSON file also contains the vectors \( x_{\text{circ}} \) and \( y_{\text{circ}} \). 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 \( x_{\text{circ}} \) and \( y_{\text{circ}} \) 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 $\text{trace}(X^T A X) = \sum_{i=1}^{k} x_i^T A x_i$. 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

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

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

**Solution.** We first note that the objective function of this problem is just a sum the same quadratic form of $x$ and $y$:

$$J = x^T L x + y^T L y,$$

where

$$x^T L x = \sum_{i<j, \ i \sim j} (x_i - x_j)^2 = \sum_{i<j, \ i \sim j} (x_i^2 - 2x_i x_j + x_j^2) = \sum_{i,j, \ i \sim j} (x_i^2 - x_i x_j),$$

(1)

and similarly for $y$. We can express $L$ as follows. If node $i$ has $d_i$ edges emanating from it (i.e., it has degree $d_i$), then the coefficient corresponding to $x_i^2$ in the above sum is $d_i$. Now, if nodes $i$ and $j$ are connected, then the sum contains a $-x_i x_j$ component. Thus the elements of $L$ are:

$$L_{ij} = \begin{cases} 
  d_i & i = j \\
  -1 & i \sim j \\
  0 & \text{otherwise}. 
\end{cases}$$
Now, in terms of $A$, $d_i$ is just the sum of the elements of $A$ along its $i$th row or column. Furthermore, if $i \sim j$ then $A_{ij} = 1$, otherwise $A_{ij} = 0$. We can thus express the coefficients of $L$ in terms of the coefficients of $A$ as:

$$L_{ij} = \begin{cases} \sum_{i=1}^{m} A_{ij} & i = j \\ -A_{ij} & \text{otherwise} \end{cases}$$

The matrix $L$ is called the Laplacian of the graph, and shows up in many different problems involving graphs. We can therefore write down the problem as

$$\min x^T L x + y^T L y$$

subject to

$$1^T x = 0, \quad 1^T y = 0$$

$$\|x\|_2 = 1, \quad \|y\|_2 = 1, \quad x^T y = 0.$$  

(2)

First we note that $L$ is positive semidefinite: $x^T L x$ is a sum of squared terms, hence nonnegative. Therefore all the eigenvalues of $L$ are nonnegative. Second, we have $L 1 = 0$, since the sum of the rows of $L$ are all zero. This means that $1$ is an eigenvector of $L$ with eigenvalue 0 (i.e., it’s in the nullspace of $L$). Of course, 0 must the smallest eigenvalue, since all eigenvalues are nonnegative, i.e., we have $\lambda_1 = 0$. Now, since $L 1 = 0$, we have $v^T L v = 0$, where we take $v$ to be $v = n^{-1/2} 1$. ($v$ is the normalized eigenvector of $L$ corresponding to eigenvalue 0.) Thus we can write the objective function as

$$J = x^T L x + y^T L y + v^T L v,$$

or, using matrix notation,

$$J = \text{trace} \left( [x y v]^T L [x y v] \right).$$

We can also gather all the equality constraints in problem (2) into a single compact matrix equality constraint:

$$[x y v]^T [x y v] = I,$$

where $I$ is the $3 \times 3$ identity matrix. Thus our problem is

$$\min \text{trace} \left( [x y v]^T L [x y v] \right)$$

subject to

$$[x y v]^T [x y v] = I.$$  

(3)

Here the variables are $x$ and $y$; $v$ is given by $v = n^{-1/2} 1$. We know how to solve a problem very similar to this problem. For $C = C^T \in \mathbb{R}^{n \times n}$, a solution $Q \in \mathbb{R}^{n \times r}$ of

$$\min \text{trace} \left( Q^T C Q \right)$$

subject to

$$Q^T Q = I,$$  

(4)

is to take the $r$ columns of the (orthonormal) eigenvectors of $C$ corresponding to its $r$ smallest eigenvalues. In fact $v = n^{-1/2} 1$ is already fixed as a normalized eigenvector of $L$ corresponding to its smallest eigenvalue, $\lambda_n = 0$. So to get the optimal values of $x$ and $y$, we just take them to be the eigenvectors of $L$ corresponding to the eigenvalues $\lambda_{n-1}$ and $\lambda_{n-2}$: We take $x = v_{n-1}$ and $y = v_{n-2}$. A very simple solution to a problem that looks pretty complicated. Some
people came up with this solution, without a clear argument as to why it’s the solution, especially, why it is that the constraints are satisfied. You pretty much have to mention the nullspace of $L$, (the span of the vector $1$), to show you really understand it. Believe it or not, several people invented their own iterative methods for solving the problem, and got the correct answer. All they were doing was re-inventing famous methods for computing the eigenvectors corresponding to the smallest eigenvalues of a symmetric matrix. By the way, there’s a really interesting field, called spectral graph theory, that studies the relationship between graphs and the eigenvalues of associated matrices (such as the Laplacian). It’s not only very interesting, but extremely useful in practice too. It’s a critical element in web search (like Google’s Pagerank), and also partitioning large graphs (such as in circuit design), and lots of other problems, too (like image segmentation). We also mention that the problem can be solved in terms of the singular value decomposition. Number the edges of the graph $1, \ldots, m$, and assign an arbitrary orientation to each one. Define the incidence matrix $B \in \mathbb{R}^{n \times n}$ as $B_{ij} = 1$ if edge $j$ points in to node $i$, $B_{ij} = -1$ if edge $j$ comes out of node $i$, and zero otherwise. Then we have the formula $L = BB^T$. The matrix $B$ has rank exactly $n - 1$, by a famous theorem of graph theory, since the graph is connected. (Of course, you could just check this numerically.) The solution to our problem is then to take $x$ and $y$ to be the right singular vectors associated with the two smallest (positive) singular values. The figure below shows the graph generated by placing all nodes on a circle, whose radius is such that $x$ and $y$ satisfy the problem constraints. In this case we have $J = 5.328$.

The following figure shows a graph which minimizes $J$ by the proposed method. This graph
achieves $J = 0.107$.

6. Uncovering a hidden linear explanation. Consider a set of vectors $y_1, \ldots, y_N \in \mathbb{R}^n$, which might represent a collection of measurements or other data. Suppose we have

$$y_i \approx Ax_i + b, \quad i = 1, \ldots, N,$$

where $A \in \mathbb{R}^{n \times m}$, $x_i \in \mathbb{R}^m$, and $b \in \mathbb{R}^n$, with $m < n$. (Our main interest is in the case when $N$ is much larger than $n$, and $m$ is smaller than $n$.) Then we say that $y = Ax + b$ is a linear explanation of the data $y$. We refer to $x$ as the vector of factors or underlying causes of the data $y$. For example, suppose $N = 500$, $n = 30$, and $m = 5$. In this case we have 500 vectors; each vector $y_i$ consists of 30 scalar measurements or data points. But these 30-dimensional data points can be ‘explained’ by a much smaller set of 5 ‘factors’ (the components of $x_i$). This problem is about uncovering, or discovering, a linear explanation of a set of data, given only the data. In other words, we are given $y_1, \ldots, y_N$, and the goal is to find $m$, $A$, $b$, and $x_1, \ldots, x_N$ so that $y_i \approx Ax_i + b$. To judge the accuracy of a proposed explanation, we’ll use the RMS explanation error, i.e.,

$$J = \left( \frac{1}{N} \sum_{i=1}^{N} \|y_i - Ax_i - b\|^2 \right)^{1/2}.$$

One rather simple linear explanation of the data is obtained with $x_i = y_i$, $A = I$, and $b = 0$. In other words, the data explains itself! In this case, of course, we have $y_i = Ax_i + b$, so the RMS explanation error is zero. But this is not what we’re after. To be a useful explanation, we want to have $m$ substantially smaller than $n$, i.e., substantially fewer factors than the dimension of the original data (and for this smaller dimension, we’ll accept a nonzero, but hopefully small, value of $J$.) Generally, we want $m$, the number of factors in the explanation, to be as small as possible, subject to the constraint that $J$ is not too large. Even if we fix the number of factors as $m$, a linear explanation of a set of data is not unique. Suppose $A$, $b$, and $x_1, \ldots, x_N$ is a linear explanation of our data, with $x_i \in \mathbb{R}^m$. Then we can multiply the matrix $A$ by two
(say), and divide each vector $x_i$ by two, and we have another linear explanation of the original data. More generally, let $F \in \mathbb{R}^{m \times m}$ be invertible, and $g \in \mathbb{R}^m$. Then we have

$$y_i \approx Ax_i + b = (AF^{-1})(Fx_i + g) + (b - AF^{-1}g).$$

Thus,

$$\tilde{A} = AF^{-1}, \quad \tilde{b} = b - AF^{-1}g, \quad \tilde{x}_1 = Fx_1 + g, \quad \ldots, \quad \tilde{x}_N = Fx_N + g$$

is another equally good linear explanation of the data. In other words, we can apply any affine (i.e., linear plus constant) mapping to the underlying factors $x_i$, and generate another equally good explanation of the original data by appropriately adjusting $A$ and $b$. To standardize or normalize the linear explanation, it is usually assumed that

$$\frac{1}{N} \sum_{i=1}^{N} x_i = 0, \quad \frac{1}{N} \sum_{i=1}^{N} x_i x_i^T = I.$$

In other words, the underlying factors have an average value zero, and unit sample covariance. (You don’t need to know what covariance is — it’s just a definition here.) Finally, the problem.

**a)** Explain clearly how you would find a hidden linear explanation for a set of data $y_1, \ldots, y_N$.

Be sure to say how you find $m$, the dimension of the underlying causes, the matrix $A$, the vector $b$, and the vectors $x_1, \ldots, x_N$. Explain clearly why the vectors $x_1, \ldots, x_N$ have the required properties.

**b)** Carry out your method on the data in the file `linexp_data.m` available on the course web site. The file gives the matrix $Y = [y_1 \cdots y_N]$. Give your final $A$, $b$, and $x_1, \ldots, x_N$, and verify that $y_i \approx Ax_i + b$ by calculating the norm of the error vector, $\|y_i - Ax_i - b\|$, for $i = 1, \ldots, N$. Sort these norms in descending order and plot them. (This gives a good picture of the distribution of explanation errors.) By explicit computation verify that the vectors $x_1, \ldots, x_N$ obtained, have the required properties.

**Solution.**

**a)** We have to find $b$ and $x_1, \ldots, x_N$ that minimize

$$\tilde{J} = NJ^2 = \sum_{i=1}^{N} (y_i - Ax_i - b)^T (y_i - Ax_i - b).$$

Taking the gradient with respect to $b$ and setting it to zero gives,

$$\nabla_b(\tilde{J}) = \sum_{i=1}^{N} 2(y_i - Ax_i - b)(-1) = 0.$$  

Since we want $\sum_{i=1}^{N} x_i = 0$, we get

$$b = \frac{1}{N} \sum_{i=1}^{N} y_i.$$
Let $X$ be the matrix $[x_1 \cdots x_N] \in \mathbb{R}^{m \times N}$. Let $z_i = y_i - b$, $i = 1, \ldots, N$, and $Z = [z_1 \cdots z_N] \in \mathbb{R}^{n \times N}$. Note that $Z$ is known from the data after $b$ is calculated as shown above. The matrix $(AX) \in \mathbb{R}^{n \times N}$ and has rank at most $m$. Then

$$J = \sum_{i=1}^{N} \|z_i - Ax_i\|^2 = \sum_{j=1}^{n} \sum_{k=1}^{N} (Z_{kj} - (AX)_{kj})^2 = \|Z - AX\|_F^2.$$

Thus minimizing $\tilde{J}$ is minimizing the Frobenius norm of the matrix $(Z - AX)$ where $AX$ is at most rank $m$. Let the SVD of $Z$ be $Z = U \Sigma V^T$, where $U \in \mathbb{R}^{n \times r}$, $\Sigma \in \mathbb{R}^{r \times r}$, $V \in \mathbb{R}^{N \times r}$ and $r$ is the rank of $Z$. The choice of $m$ depends on the singular values $\sigma_1, \ldots, \sigma_r$ obtained for the particular data. A good choice of $m$ would be when there is a significant jump in the singular values, i.e., $\sigma_m \gg \sigma_{m+1}$; or when the singular value becomes small enough ($\sigma_{m+1}$ is negligible). Thus we pick a value for $m$. Then the $m$ rank approximation to $Z$ is

$$(AX) = \sum_{i=1}^{m} \sigma_i u_i v_i^T = U_m \Sigma_m V_m^T,$$

where $U_m \in \mathbb{R}^{n \times m}$, $\Sigma_m \in \mathbb{R}^{m \times m}$, $V_m \in \mathbb{R}^{N \times m}$. We pick $A = \frac{1}{\sqrt{N}} U_m \Sigma_m$, and $x_i$ as $\sqrt{N}$ times the $i$th column of $V_m^T$. Then

$$\frac{1}{N} \sum_{i=1}^{N} x_i x_i^T = \frac{1}{N} (\sqrt{N} V_m)^T (\sqrt{N} V_m) = I.$$

In order to show that $\frac{1}{N} \sum_{i=1}^{N} x_i = 0$, consider

$$Z 1 = \sum_{i=1}^{N} z_i = \sum_{i=1}^{N} (y_i - b) = 0,$$

where $1$ is the vector of ones of size $N$. The vector $1$ is in the nullspace of $Z$ which we can write as $U \Sigma V^T 1 = 0$. The matrix $U \Sigma$ is full rank, therefore $V^T 1 = 0$. Hence $V_m^T 1 = 0$ as $V_m$ are the first $m$ columns of the matrix $V$. This means

$$\frac{1}{\sqrt{N}} V_m^T 1 = \frac{1}{N} [x_1 \cdots x_N] 1 = \frac{1}{N} \sum_{i=1}^{N} x_i = 0.$$

Thus we have found

$$b = \frac{1}{N} \sum_{i=1}^{N} y_i, \quad A = \frac{1}{\sqrt{N}} U_m \Sigma_m, \quad [x_1 \cdots x_N] = \sqrt{N} V_m^T,$$

with the required properties.

b) The following matlab code implements solution method described in the part (a). We observe that there are 3 significant singular values, and therefore we take $m = 3$.

```matlab
linexp_data;
```
\[ n, N = \text{size}(Y); \]
\[ b = \frac{\text{sum}(Y')'}{N}; \]
\[ \text{for } i = 1:N \]
\[ Z(:, i) = Y(:, i) - b; \]
\[ \text{end} \]
\[ [U \ S \ V] = \text{svd}(Z); \]
\[ A = \frac{1}{\sqrt{N}} \* U(:, 1:3) \* S(1:3, 1:3); \]
\[ X = \sqrt{N} \* V(:, 1:3)'; \]
\[ \text{error} = Z - A \* X; \]
\[ \text{for } i = 1:N \]
\[ \text{errNorm}(i) = \text{norm}(	ext{error}(:, i)); \]
\[ \text{end} \]
\[ \text{plot}(\text{sort(errNorm, 2, 'descend'));} \]
\[ \text{xlabel('index'); ylabel('errors');} \]

Here is explicit check shows that \( X \) satisfies given properties.

\[
\begin{align*}
\gg \text{sum}(X, 2)/N &= \\
&= 1.0e-15 * \\
&= -0.0333 \\
&= -0.1064 \\
&= 0.2331 \\
\gg X \* X'/N &= \\
&= \begin{bmatrix} 1.0000 & 0.0000 & -0.0000 \\
0.0000 & 1.0000 & 0.0000 \\
-0.0000 & 0.0000 & 1.0000 \end{bmatrix}
\end{align*}
\]
7. **Blind signal detection.** A binary signal $s_1, \ldots, s_T$, with $s_t \in \{-1, 1\}$ is transmitted to a receiver, which receives the (vector) signal $y_t = a s_t + v_t \in \mathbb{R}^n$, $t = 1, \ldots, T$, where $a \in \mathbb{R}^n$ and $v_t \in \mathbb{R}^n$ is a noise signal. We’ll assume that $a \neq 0$, and that the noise signal is centered around zero, but is otherwise unknown. (This last statement is vague, but it will not matter.)

The receiver will form an approximation of the transmitted signal as

$$\hat{s}_t = w^T y_t, \quad t = 1, \ldots, T,$$

where $w \in \mathbb{R}^n$ is a weight vector. Your job is to choose the weight vector $w$ so that $\hat{s}_t \approx s_t$. If you knew the vector $a$, then a reasonable choice for $w$ would be $w = a^\dagger = a/\|a\|^2$. This choice is the smallest (in norm) vector $w$ for which $w^T a = 1$.

Here’s the catch: You don’t know the vector $a$. Estimating the transmitted signal, given the received signal, when you don’t know the mapping from transmitted to received signal (in this case, the vector $a$) is called *blind signal estimation* or *blind signal detection*.

Here is one approach. Ignoring the noise signal, and assuming that we have chosen $w$ so that $w^T y_t \approx s_t$, we would have

$$(1/T) \sum_{t=1}^T (w^T y_t)^2 \approx 1.$$

Since $w^T v_t$ gives the noise contribution to $\hat{s}_t$, we want $w$ to be as small as possible. This leads us to choose $w$ to minimize $\|w\|$ subject to $(1/T) \sum_{t=1}^T (w^T y_t)^2 = 1$. This doesn’t determine $w$ uniquely; we can multiply it by $-1$ and it still minimizes $\|w\|$ subject to $(1/T) \sum_{t=1}^T (w^T y_t)^2 = 1$. So we can only hope to recover either an approximation of $s_t$ or of $-s_t$; if we don’t know $a$
we really can’t do any better. (In practice we’d use other methods to determine whether we have recovered \( s_t \) or \(-s_t\).)

a) Explain how to find \( w \), given the received vector signal \( y_1, \ldots, y_T \), using concepts from the class.

b) Apply the method to the signal in the file `bs_det_data.m`, which contains a matrix \( Y \), whose columns are \( y_t \). Give the weight vector \( w \) that you find. Plot a histogram of the values of \( w^T y_t \) using `hist(w^T y, 50)`. You’ll know you’re doing well if the result has two peaks, one negative and one positive. Once you’ve chosen \( w \), a reasonable guess of \( s_t \) (or, possibly, its negative \(-s_t\)) is given by

\[
\tilde{s}_t = \text{sign}(w^T y_t), \quad t = 1, \ldots, T,
\]

where \( \text{sign}(u) \) is +1 for \( u \geq 0 \) and −1 for \( u < 0 \). The file `bs_det_data.m` contains the original signal, as a row vector \( s \). Give your error rate, i.e., the fraction of times for which \( \tilde{s}_t \neq s_t \). (If this is more than 50%, you are welcome to flip the sign on \( w \).)

**Solution.** We can write

\[
\frac{1}{T} \sum_{t=1}^T (w^T y_t)^2 = \frac{1}{T} \|Y^T w\|^2,
\]

where \( Y = [y_1 \cdots y_T] \). We must minimize \( \|w\| \) subject to \( \|Y^T w\| = \sqrt{T} \). Both of these are homogeneous in \( w \), so we could just as well maximize \( \|Y^T w\| \), subject to \( \|w\| = 1 \), and then scale the solution so that \( \|Y^T w\| = \sqrt{T} \). To maximize \( \|Y^T w\| \) subject to \( \|w\| = 1 \) is easy: we take \( w \) to be \( v_1 \), the right singular vector associated with the largest singular value of \( Y^T \). (This is also, by the way, the left singular vector associated with the largest singular value of \( Y \).) We then scale \( v_1 \) by \( \alpha \), so that \( \|Y^T(\alpha v_1)\| = \alpha \|v_1\| = \sqrt{T} \), where \( \sigma_1 \) is the largest singular value (i.e., the norm) of \( Y \) (or \( Y^T \)). This yields

\[
w = \frac{\sqrt{T}}{\sigma_1} v_1.
\]

Note that we could just as well take the negative of this vector, which would also minimize \( \|w\| \) subject to \( \|Y^T w\| = \sqrt{T} \).

Here is the code to do this:

```matlab
% blind signal detection exercise
% solution

bs_det_data;
[U,S,V]=svd(Y');
w = (sqrt(T)/S(1,1))*V(:,1)

% now form estimate of original binary signal
shat = (Y'*w)';
hist(shat,50);
print -deps bs_det_hist
```
% error rate
stilde = sign(shat);
error_rate = sum(s~=stilde)/T

The error rate is 2.9%. The resulting histogram is shown below.