3. Network design

In this problem, you will be playing the role of a network engineer. You are given a fully connected network and your job is to set the bandwidths of the links in the network such that the state of the network gets as close as possible to a desired state, at a given point in time. Before we setup the problem, let’s define some network terminology.

We consider a simple model of a communication network in which message packets of unit-length are carried between the nodes. Assume that you are given a 2-dimensional communication network formed by the set of nodes $N = \{(i, j) : i, j \in \mathbb{Z}, 1 \leq i \leq m, 1 \leq j \leq n\}$. Note that each node $(i, j)$ is indexed by two positive integers. It is your job to set up the links (i.e., connection wires) between the nodes. In this regard, for each node $(i, j)$ you need to allocate the bandwidth $b_{(i,j)\rightarrow(k,l)}$, $(k, l) \in N$, $(k, l) \neq (i, j)$, for the wires connecting node $(i, j)$ to other nodes $(k, l)$. Note that the link that goes from node $(i, j)$ to node $(k, l)$ is distinct from the one going from node $(k, l)$ to node $(i, j)$. Also note that what we mean by $(k, l) \neq (i, j)$ is that either $k \neq i$, or $l \neq j$ or both. The bandwidths $b_{(i,j)\rightarrow(k,l)}$ determine the out-flow dividing coefficients at each node, meaning that when we have some out-flow at node $(i, j)$, the following fraction of the out-flow will be carried to node $(k, l)$

$$
\phi_{(i,j)\rightarrow(k,l)} = \frac{b_{(i,j)\rightarrow(k,l)}}{\sum_{(p,q)\in N, (p,q)\neq(i,j)} b_{(i,j)\rightarrow(p,q)}}.
$$

(We assume that for all $(i, j) \in N$, $\sum_{(p,q)\in N, (p,q)\neq(i,j)} b_{(i,j)\rightarrow(p,q)} \neq 0$.) We study this network as a discrete-time system for $0 \leq t \leq 2$. Let $x_{(i,j)}(t)$ denote the total out-flow from node $(i, j)$ at time $t$. In addition, let $u_{(i,j)}(t)$ denote the known in-flow entering the node $(i, j)$ from outside the network at time $t$. One simple model for the network as it evolves toward a steady state through time is as follows:

$$
x_{(i,j)}(t + 1) = u_{(i,j)}(t) + \sum_{(k,l)\in N, (k,l)\neq(i,j)} \phi_{(k,l)\rightarrow(i,j)} x_{(k,l)}(t), \quad 0 \leq t \leq 2.
$$

Assume that at $t = 0$ there is no information out-flow in the network, which means that $x_{(i,j)}(0) = 0$ for all $(i, j) \in N$. Also, assume that at each time $t$, the in-flow entering the network is determined by known matrix $U(t) \in \mathbb{R}^{m \times n}$, such that $u_{(i,j)}(t) = U_{ij}(t)$.

We define the state of the network at time $t$ using a matrix $X(t) \in \mathbb{R}^{m \times n}$ for which $X_{ij}(t) = x_{(i,j)}(t)$. Finally, the problem of interest is as follows: as the network engineer, you want the state of the system at time $t = 2$ to be as close as possible to some desired state $Y \in \mathbb{R}^{m \times n}$ in the sense of minimizing the following cost function:

$$
\|X(2) - Y\|_F^2,
$$

where $\|X\|_F$ is the Frobenius norm of a matrix and is defined as:

$$
\|X\|_F = \left( \sum_{i,j} X_{ij}^2 \right)^{1/2}.
$$
Remark: In reality, the in-flows and out-flows of the network should all be non-negative quantities. However, to simplify things, you need not enforce this constraint when solving this problem. In other words, it is ok if your solution produces negative bandwidths for some of the links.

(a) Assume that you are given a set of bandwidths \( \{ b^*_\text{\(i,j\)}\rightarrow\text{\(k,l\)} : (i, j), (k, l) \in N \} \) that minimizes the cost function (2). Determine whether this set of bandwidths would be the unique minimizer of our cost function, or we can find other solutions as well?

Hint: What would happen if you scale the optimal bandwidths \( b^*_\text{\(i,j\)}\rightarrow\text{\(k,l\)} \)?

Solution
Assume that the bandwidths \( b^*_\text{\(i,j\)}\rightarrow\text{\(k,l\)} \)'s minimize (2). Now we define some new bandwidths as follows:

\[
b_{\text{\(i,j\)}\rightarrow\text{\(k,l\)}} = c_{\text{\(i,j\)}} b^*_\text{\(i,j\)}\rightarrow\text{\(k,l\)}; \quad c_{\text{\(i,j\)}} \neq 0, \quad \text{for all } (i, j), (k, l) \in N, (k, l) \neq (i, j).
\]

It is easy to check that the dividing coefficients \( \phi_{\text{\(i,j\)}\rightarrow\text{\(k,l\)}} \) do not change and as a result the value of (2) remains fixed. So we can conclude that there are infinite number of solutions to this minimization problem.

(b) Using the material covered in EE263 so far, propose a method to find the bandwidths \( b_{\text{\(i,j\)}\rightarrow\text{\(k,l\)}} \) that minimize the cost function (2). Explicitly mention any assumptions that you make about the matrices that you use in your solution. Comment on whether or not your method gives a unique solution for optimal bandwidths \( b_{\text{\(i,j\)}\rightarrow\text{\(k,l\)}} \) and why.

Hint: Note that your cost function (2) is not the standard least-squares cost function. However, there is a simple connection between the two. You can start by rearranging matrices \( X(t) \) and \( Y \) into vectors. You will also need to somehow rearrange the bandwidths \( b_{\text{\(i,j\)}\rightarrow\text{\(k,l\)}} \) into a vector. Now this is a little bit trickier. Think of how many entries this vector will have? To make this a little easier you can define \( b_{\text{\(i,j\)}\rightarrow\text{\(i,j\)}} = 0 \) for all \( (i, j) \in N \) and include them as part of the bandwidths \( b_{\text{\(i,j\)}\rightarrow\text{\(k,l\)}} \). Now try to rewrite you cost function (2) in terms of these new vectorized variables and see if it looks more like a standard least-squares cost function. Yes we know, the relationship between \( \phi_{\text{\(i,j\)}\rightarrow\text{\(k,l\)}} \) and \( b_{\text{\(i,j\)}\rightarrow\text{\(k,l\)}} \) is making the math kind of messy. That is why we added part (a) to this problem. It is an indirect hint on how to make the math simpler. Finally, remember you need to somehow ensure that in your final solution \( b_{\text{\(i,j\)}\rightarrow\text{\(i,j\)}} = 0 \) for all \( (i, j) \in N \). How about adding some constraints to your minimization problem?

Solution.
First, a matrix \( X \in \mathbb{R}^{m \times n} \) can be rearranged into a vector \( v(X) \in \mathbb{R}^{mn} \) in the following manner:

\[
X_{ij} = v_{i+(j-1)m} (X) \quad \text{for } 1 \leq i \leq m, \ 1 \leq j \leq n.
\]
In addition, let us define
\[ b_{(i,j)\rightarrow (i',j')} = 0 \quad \text{for all } (i, j) \in N. \tag{3} \]
Also, since multiplying all the bandwidths \( b_{(i,j)\rightarrow (k,l)} \) by any nonzero constant \( c_{(i,j)} \) does not change anything, without loss of generality, we can assume that:
\[ \sum_{(k,l)\in N} b_{(i,j)\rightarrow (k,l)} = 1 \quad \text{for all } (i, j) \in N. \tag{4} \]
With this added constraint we now have \( b_{(i,j)\rightarrow (k,l)} = \phi_{(i,j)\rightarrow (k,l)}. \)
Furthermore, let us define the vector \( b \in \mathbb{R}^{(mn)^2} \) as follows:
\[ b_{((l-1)m+(j-1))mn+(i-1)m+i} = b_{(i,j)\rightarrow (k,l)} \]
Finally we need to define the matrix \( A \in \mathbb{R}^{mn \times (mn)^2} \) as follows:
\[
A = \begin{bmatrix}
v^T(U_0) & 0 & \ldots & 0 & \ldots & 0 & \ldots & 0 \\
0 & v^T(U_0) & 0 & \ldots & 0 & \ldots & 0 & \ldots & 0 \\
0 & 0 & v^T(U_0) & 0 & \ldots & 0 & \ldots & 0 & \ldots & 0 \\
\vdots & \vdots & & \ddots & & \vdots & \ddots & \vdots & \ddots & \vdots \\
0 & \ldots & 0 & \ldots & 0 & \ldots & v^T(U_0)
\end{bmatrix},
\]
where \( U_{0,ij} = u_{(i,j)}(0). \) Now we are ready to change (2) into something familiar for us. It is easily seen that
\[ \|X(2) - Y\|_F^2 = \|Ab + v(U_1) - v(Y)\|_2^2. \tag{5} \]
As a result, from now on we try to minimize (5) instead of (2). But before we proceed, we note that there are two constraints (3) and (4) that have to be satisfied as well. So we need to write them in matrix notation. To do so, we define the matrices \( C_1, C_2 \in \mathbb{R}^{mn \times (mn)^2} \) as follows:
\[
C_{1,kl} = \begin{cases} 
1 & \text{if } k = (j-1)m + i \text{ and } l = k(mn+1) - mn \text{ for all } (i, j) \in N \\
0 & \text{o.w.}
\end{cases}
\]
\[ C_2 = \begin{bmatrix} I_{mn} & I_{mn} & \cdots & I_{mn} \end{bmatrix}. \]
In addition we define the matrix \( C \) as follows:
\[ C = \begin{bmatrix} C_1 \\ C_2 \end{bmatrix}, \]
and the vector $d \in \mathbb{R}^{2mn}$ as follows:

\[
d = \begin{bmatrix}
0 \\
\vdots \\
0 \\
1 \\
\vdots \\
1
\end{bmatrix}.
\]

Now we can write the constraints in terms of $C$ and $d$:

\[Cb = d.\]

In addition, for simplicity of notation let $y = v(Y) - v(U_1)$. We can wrap up everything as the following problem:

\[
\begin{align*}
\text{minimize} & \quad \|Ab - y\|_2^2 \\
\text{subject to} & \quad Cb = d
\end{align*}
\]

We know that $b \in \mathbb{R}^{(mn)^2}$ is optimal for this problem if and only if

\[
\begin{bmatrix}
A^T A & C^T \\
C & 0
\end{bmatrix}
\begin{bmatrix}
b \\
\lambda
\end{bmatrix} = \begin{bmatrix}
A^T y \\
d
\end{bmatrix},
\]

for some $\lambda \in \mathbb{R}^{2mn}$. Note that $A^T A$ and the block matrix are not invertible in this problem and hence, the formula in given pages 13 and 14 of lecture notes Least-norm solutions of underdetermined equations cannot be used. To combat this difficulty, we can have two different approaches:

i. **The first approach:** Since $C$ is fat and full-rank, $CC^T$ is invertible. Thus we can use the first block to solve and eliminate $\lambda$:

\[
\begin{bmatrix}
(I - C^T C)A^T A \\
C
\end{bmatrix}b = \begin{bmatrix}
(I - C^T C)A^T y \\
d
\end{bmatrix}
\]

It turns out that the coefficient matrix in this system is not full-rank (its rank is 26). However, because the normal equations are always consistent, we can still compute a solution using the pseudoinverse:

\[
b = \begin{bmatrix}
(I - C^T C)A^T A \\
C
\end{bmatrix}^\dagger \begin{bmatrix}
(I - C^T C)A^T y \\
d
\end{bmatrix}
\]

Since the coefficient matrix is singular, the optimal $b$ is not unique. Note that we cannot use the singularity of normal equations to immediately conclude that the optimal $b$ is not unique; the ambiguity could have been entirely in $\lambda$. 

12
ii. **The second approach:** Consider the optimization problem (6) again. Since the matrix $C$ is fat and full-rank any solution to this optimization problem has to be of the following form:

$$b = C^T (CC^T)^{-1} d + F z,$$

where the matrix $F$ constitutes some columns that are a basis for the $\mathcal{N}(C)$. If we plug this format into the objective function, the original optimization problem reduces to the following problem:

$$\text{minimize } \| A F z + A C^T (CC^T)^{-1} d - y \|_2^2,$$

The solution to which, based on the hint, is the following:

$$z = (A F)^\dagger (y - A C^T (CC^T)^{-1} d),$$

from which we can recover the vector $b$:

$$b = C^T (CC^T)^{-1} d + F (A F)^\dagger (y - A C^T (CC^T)^{-1} d).$$

(c) Implement your method on the data given in `network_design_data.m`. In this m-file, the matrices $U_0 \in \mathbb{R}^{m \times n}$ and $U_1 \in \mathbb{R}^{m \times n}$ are the in-flows at times $t = 0$, and $t = 1$, respectively. The matrix $Y \in \mathbb{R}^{m \times n}$ is the desired state at time $t = 2$. You only need to include your code and report the minimum value of cost function (2) for the given data.

**Hint:** The matlab function `pinv` can be used to find the pseudoinverse of a (not necessarily full-rank) matrix.

**Solution.**
The following matlab code solves the problem. The minimum value of the objective function is: 0.1520.

```matlab
%% network_design_sol
run network_design_data

% creating the required matrices and vectors
v_Y = zeros(m*n,1);
v_U0 = zeros(m*n,1);
v_U1 = zeros(m*n,1);
A = zeros(m*n,m^2*n^2);
C1 = zeros(m*n,m^2*n^2);
C2 = repmat(eye(m*n),1,m*n);
d = [zeros(m*n,1);ones(m*n,1)];

for i = 1:m
    for j=1:n
```
\[ k = (j-1)*m+i; \]
\[ l = k*(m*n+1)-m*n; \]

\[ v_Y(k) = Y(i,j); \]
\[ v_U0(k) = U0(i,j); \]
\[ v_U1(k) = U1(i,j); \]

\[ C1(k,l) = 1; \]
\[ \text{end} \]
\[ \text{end} \]

\[ \text{for } i = 1:m*n \]
\[ A(i,(i-1)*m*n+1:i*m*n) = v_U0'; \]
\[ \text{end} \]

\[ C = [C1;C2]; \]
\[ y = v_Y - v_U1; \]

\% implement the solution method

\% The first approach

\[ BB = [(\text{eye(size}(C'*C)) - \text{pinv}(C)*C)*A'*A]; \]
\[ yy = [(\text{eye(size}(C'*C)) - \text{pinv}(C)*C)*A'*y]; \]
\[ b = \text{pinv}(BB)*yy; \]

\% computing the optimal value
\[ \text{norm}(A*b-y)^2 \]

\% The second approach

\[ F = \text{null}(C); \]
\[ z = (A*F)\backslash(y-A*C'/(C*C')*d); \]
\[ \text{norm}(A*F*z-(y-A*C'/(C*C')*d))^2 \]
4. *Grab life by the Llama*

You are the owner of an alpine mountaineering company, who offer tours with guide llamas. You want to show your customers as much of the Llama Adventure WonderLand (L.A.W.L), but you don’t want to over exert your animals. We can model a llama as a first order linear dynamical system:

\[
x(t + 1) = Ax(t) + Bu(t)
\]

where \( x(t) \in \mathbb{R}^4 \) is the state vector consisting of the llama’s position and velocity, and the system input \( u(t) \in \mathbb{R}^2 \) is the velocity control effort.

\[
A = \begin{bmatrix}
1 & 0 & .1 & 0 \\
0 & 1 & 0 & .1 \\
0 & 0 & .5 & 0 \\
0 & 0 & 0 & .5 \\
\end{bmatrix}
\]

\[
B = \begin{bmatrix}
0 & 0 \\
0 & 0 \\
1 & 0 \\
0 & 1 \\
\end{bmatrix}
\]

Our goal is to minimize the overall effort put into controlling the llama between times 0 and \( T \) by minimizing the following cost function for a given \( T \).

\[
J = \sum_{t=0}^{T} \|u(t)\|^2
\]

We also want to hit a set of positional waypoints \( \{w_1, \ldots, w_n\} \), \( w_i \in \mathbb{R}^2 \) at target times \( \{k_1, \ldots, k_n\} \). Hence, the minimization problem has the following constraints:

\[
w_i = Hx(k_i) \quad i = 1, \ldots, n
\]

where \( H \) is a matrix that when left multiplied by \( x \) returns the first two entries, since the waypoints are only positions.

\[
H = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
\end{bmatrix}
\]

Finally we get to the problem.

(a) Determine a single matrix equation that relates the control inputs \( u(t) \) for \( t = 0, \ldots, T \) to the waypoints \( w_i \) for \( i = 1, \ldots, n \). Assume you are given an initial state \( x(0) = x_0 \) and that the final time is \( T = k_n - 1 \).

**Hint:** expand out the dynamics for \( x(1), x(2), \ldots \) until you find a pattern. Your final equation should express \( w_i \)'s as an affine function of \( u(t) \) for \( t = 0, \ldots, T \) (remember, an affine function is a linear function plus a constant).
(b) Using the matrix equation of part (a), formulate the problem as a standard least-norm problem and find the optimal sequence of control inputs \( u(t) \) that minimizes the cost function subject to the constraints. Leave your answer in terms of the problem data \((A, B, H, x_0, w_i, k_i)\). Feel free to introduce new variables to simplify your math.

(c) Implement your solution with the data in the file `llama_data.m`. Each column of \( w \) gives the \((\text{pos}_x, \text{pos}_y)\) position waypoint for the corresponding time in \( k \). Report the following plots:

- The trajectory of the llama with the waypoints clearly marked
- Each \( x \) and \( y \) velocity of the llama as a function of time

Also report your final minimized cost \( J \).

(d) What are the conditions on matrices \( A \) and \( B \) that must hold in order to guarantee that the llama passes through all the waypoints at the target times? We are looking for a mathematical argument using what you have learned in EE263 up until this point.

**Solution.**

(a) We can expand out the dynamics equation to obtain

\[
\begin{align*}
x(1) &= Ax(0) + Bu(0) \\
x(2) &= A^2x_0 + [AB \ B] \begin{bmatrix} u(0) \\ u(1) \end{bmatrix}
\end{align*}
\]

Continuing this for each way point yields

\[
\begin{bmatrix} w_1 \\ \vdots \\ w_n \end{bmatrix} = \begin{bmatrix} H \ A^{k_1} \\ \vdots \\ H \ A^{k_n} \end{bmatrix} \begin{bmatrix} u(0) \\ \vdots \end{bmatrix}
\]

where \( H \) is a matrix that when left multiplied returns the 1st two rows since the way points are only positions

\[
H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}
\]

(b) Let the above equation be

\[
w = Fx_0 + Gu
\]

Thus we want to solve the following optimization problem

\[
\begin{align*}
\text{minimize} & \quad \|u\| \\
\text{subject to} & \quad Gu = w - Fx_0
\end{align*}
\]
The result is

\[ u = G^T(GG^T)^{-1}(w - Fx_0) \]

which is the least norm solution.

(c) the code that solves this problem

```matlab
close all
clear

%% load data
run llama_data

%% build system
H=[1 0 0 0; 0 1 0 0];
G=zeros(length(k)*2, 2*k(end));
F=zeros(length(k)*2, 4);
L=zeros(length(k)*2, 1);
for i=1:length(k)
    %build G
    for j=1:k(i)
        G(2*i-1:2*i, 2*j-1:2*j)=H*A^(k(i)-j)*B;
    end
    %build F
    F(2*i-1:2*i, :)=H*A^(k(i));
    L(2*i-1:2*i)=w(:,i);
end

%% take least norm solution
U=G'*inv(G*G')*(L-F*x0);
U=reshape(U, 2, 50);

%% forward dynamics
T=k(end); % final time
X=zeros(4, T+1);
X(:,1)=x0;
for t=1:T
    X(:, t+1)=A*X(:,t)+B*U(:,t);
end

%% plotting
%trajectory
plot(w(1,:), w(2,:), 'ro')
hold on
plot(X(1,:), X(2,:), 'b')
xlabel('x'); ylabel('y')
title('Trajectory')

%velocity
figure()
```
```matlab
plot(0:T, X(3,:), 'r')
hold on
plot(0:T, X(4,:), 'b')
xlabel('time'); ylabel('velocity')
title('Velocity'); legend('X vel', 'Y vel')

J = norm(U, 'fro');
fprintf('The optimal cost is %.3f\n', J)
```
Optimal Cost is 23.458

(d) One condition that must hold is that \((w - Fx_0)\) is in the span of \(G\).

However, an even more general condition relates to what is called the Controllability Matrix. If the matrix

\[
C = \begin{bmatrix} A^{n-1}B & A^{n-2}B & \ldots & B \end{bmatrix}
\]

is full rank then it is possible to reach any final state from any initial state within \(n\) steps, where \(n\) is the size of the state.