1. **Optimal operation of a two-state chemical reactor.** Consider a chemical reactor containing \( n \) compounds, labeled \( 1, \ldots, n \). Let \( x_i(t) \) be the amount of compound \( i \) in the reactor at time \( t \). The chemical reactor has two modes of operation, labeled 1 and 2. (For example, the first mode may be operating the reactor at a low temperature, and the second mode may be operating the reactor at a high temperature.) For simplicity we assume that the mode of operation can be changed instantaneously. When we operate the reactor in mode \( j \), the vector of compound amounts evolves according to the equation

\[
\dot{x}(t) = A_j x(t).
\]

We are given the vector \( x(0) \in \mathbb{R}^n \) of initial compound amounts, and the dynamics matrices \( A_1 \) and \( A_2 \). Our objective is to maximize the amount of compound \( k \) at time \( T \), where \( k \in \{1, \ldots, n\} \) and \( T > 0 \) are given.

a) Suppose the reactor operates in mode 1 for \( 0 \leq t \leq T_0 \), and mode 2 for \( T_0 < t \leq T \). Explain how to choose the time \( T_0 \) in order to maximize the amount of compound \( k \) at time \( T \). Your answer only needs to be accurate to two decimal digits.

b) Apply your method to the data given in `chemical_reactor_data.m`. Report the optimal value of \( T_0 \) and the corresponding amount of compound \( k \) at time \( T \); submit a plot showing all of the components of \( x(t) \) as functions of time on a single set of axes.

c) Suppose the reactor operates in mode 1 for \( 0 \leq t \leq T_1 \) and \( T_2 < t \leq T \), and mode 2 for \( T_1 < t \leq T_2 \). Explain how to choose the times \( T_1 \) and \( T_2 \) in order to maximize the amount of compound \( k \) at time \( T \). Your answers for \( T_1 \) and \( T_2 \) only need to be accurate to two decimal digits.

d) Apply your method to the data given in `chemical_reactor_data.m`. Report the optimal values of \( T_1 \) and \( T_2 \) and the corresponding amount of compound \( k \) at time \( T \); submit a plot showing all of the components of \( x(t) \) as functions of time on a single set of axes.

**Solution.**

a) Since the reactor operates in mode 1 for \( 0 \leq t \leq T_0 \), we have that

\[
x(T_0) = \exp(T_0A_1)x(0).
\]

Similarly, because the reactor operates in mode 2 for \( T_0 < t \leq T \), we have that

\[
x(T) = \exp((T - T_0)A_2)x(T_0) = \exp((T - T_0)A_2)\exp(T_0A_1)x(0).
\]
We want to maximize the amount of compound $k$ at time $T$:

$$x_k(T) = e_k^T x(T) = e_k^T e_k e^T \exp((T - T_0)A_2) \exp(T_0 A_1) x(0).$$

Since our answer for $T_0$ only needs to be accurate to two decimal digits, we simply use the formula above to compute $x_k(T)$ for all $T_0 \in \{0.00, 0.01, 0.02, \ldots, T\}$, and then choose the value that maximizes $x_k(T)$.

b) We find that the optimal time to change the mode of operation is $T_0 = 0.61$, and the corresponding amount of compound $k$ at time $T$ is 0.3655. A plot showing all of the components of $x(t)$ as functions of time is given in figure 1.

c) Since the reactor operates in mode 1 for $0 \leq t \leq T_1$, we have that

$$x(T_1) = \exp(T_1 A_1) x(0).$$

Similarly, because the reactor operates in mode 2 for $T_1 < t \leq T_2$, we have that

$$x(T_2) = \exp((T_2 - T_1)A_2) x(T_1) = \exp((T_2 - T_1)A_2) \exp(T_1 A_1) x(0).$$

Finally, since the reactor operates in mode 1 for $T_2 < t \leq T$, we have that

$$x(T) = \exp((T - T_2)A_1) x(T_2) = \exp((T - T_2)A_1) \exp((T_2 - T_1)A_2) \exp(T_1 A_1) x(0).$$

We want to maximize

$$x_k(T) = e_k^T x(T) = e_k^T e_k e^T \exp((T - T_2)A_1) \exp((T_2 - T_1)A_2) \exp(T_1 A_1) x(0).$$

Since our answers for $T_1$ and $T_2$ only need to be accurate to two decimal digits, we simply use the formula above to compute $x_k(T)$ for all $T_1, T_2 \in \{0.00, 0.01, 0.02, \ldots, T\}$ such that $T_1 < T_2$, and then choose the values that maximize $x_k(T)$.

d) We find that the optimal times to change the mode of operation are $T_1 = 0.49$ and $T_2 = 0.87$, and the corresponding amount of compound $k$ at time $T$ is 0.3763. A plot showing all of the components of $x(t)$ as functions of time is given in figure 2.
\[ \text{xk}(i) = \text{sparse}(1,k,1,1,n) \dots \]
\[ \quad \times \text{expm}((T - t(i)) \times A2) \dots \]
\[ \quad \times \text{expm}(t(i) \times A1) \times x0; \]
end
\[ [~, \text{idx}] = \max(\text{xk}); \]
\[ T0_{\text{opt}} = t(\text{idx}) \]
\[ \text{xk}_{\text{opt}} = \text{xk}(\text{idx}) \]

\% simulate the system using the optimal value of T0
\[ t\text{plot} = 0.00:0.01:T; \]
\[ dt = t\text{plot}(2) - t\text{plot}(1); \]
\[ x\text{plot} = [x0 \text{nan}(n,\text{length}(t\text{plot})-1)]; \]
for i = 1:(\text{length}(t\text{plot})-1)
    if t\text{plot}(i) \leq T0_{\text{opt}}
        x\text{plot}(:,i+1) = \text{expm}(dt \times A1) \times x\text{plot}(:,i);
    else
        x\text{plot}(:,i+1) = \text{expm}(dt \times A2) \times x\text{plot}(:,i);
    end
end

figure();
plot(t , x\text{plot} );
xlabel('t');
ylabel('x(t)');
legend('x1(t)' , 'x2(t)' , 'x3(t)');
grid on;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\% find the optimal times to change the mode of the system
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\[ \text{xk} = -\text{Inf}(\text{length}(t) , \text{length}(t)); \]
for i = 1:(\text{length}(t)-1)
    for j = (i+1):\text{length}(t)
        xk(i,j) = \text{sparse}(1,k,1,1,n) \dots
            \times \text{expm}((T - t(j)) \times A1) \dots
            \times \text{expm}((t(j) - t(i)) \times A2) \dots
            \times \text{expm}(t(i) \times A1) \times x0; \]
    end
end
\[ [~, \text{idx}] = \max(xk(:)); \]
\[ [i\text{opt} , j\text{opt}] = \text{ind2sub}(\text{size}(xk) , \text{idx}); \]
\[ T1_{\text{opt}} = t(i\text{opt}) \]
\[ T2_{\text{opt}} = t(j\text{opt}) \]
\[ \text{xk}_{\text{opt}} = \text{xk}(\text{idx}) \]
Figure 1: the trajectory of $x(t)$ with a single optimal mode change

```matlab
% simulate the system using the optimal values of T1 and T2
xplot = [x0 nan(n,length(tplot)-1)];
for i = 1:(length(tplot)-1)
    if tplot(i) <= T1_opt || tplot(i) > T2_opt
        xplot(:,i+1) = expm(dt * A1) * xplot(:,i);
    else
        xplot(:,i+1) = expm(dt * A2) * xplot(:,i);
    end
end
figure();
plot(t , xplot);
xlabel('t');
ylabel('x(t)');
legend('x1(t)' , 'x2(t)' , 'x3(t)');
grid on;
```
Figure 2: the trajectory of $x(t)$ with two optimal mode changes
2. The smoothest input that takes the state to zero. We consider the discrete-time linear dynamical system \( x(t+1) = Ax(t) + Bu(t) \), with
\[
A = \begin{bmatrix}
1.0 & 0.5 & 0.25 \\
0.25 & 0 & 1.0 \\
1.0 & -0.5 & 0
\end{bmatrix}, \quad B = \begin{bmatrix}
1.0 \\
0.1 \\
0.5
\end{bmatrix}, \quad x(0) = \begin{bmatrix}
25 \\
0 \\
-25
\end{bmatrix}.
\]
The goal is to choose an input sequence \( u(0), u(1), \ldots, u(19) \) that yields \( x(20) = 0 \). Among the input sequences that yield \( x(20) = 0 \), we want the one that is smoothest, i.e., that minimizes
\[
J_{\text{smooth}} = \left( \frac{1}{20} \sum_{t=0}^{19} (u(t) - u(t-1))^2 \right)^{1/2},
\]
where we take \( u(-1) = 0 \) in this formula. Explain how to solve this problem. Plot the smoothest input \( u_{\text{smooth}} \), and give the associated value of \( J_{\text{smooth}} \).

**Solution.** We first express \( x(20) \) in terms of \( u(0), \ldots, u(19) \) and \( x(0) \). Using the linear recursion \( x(t+1) = Ax(t) + Bu(t) \), we get \( x(1) = Ax(0) + Bu(0) \),
\[
x(2) = Ax(1) + Bu(1) \\
= A(Ax(0) + Bu(0)) + Bu(1) \\
= A^2 x(0) + ABu(0) + Bu(1)
\]
Continuing in this way we get
\[
x(20) = A^{20} x(0) + A^{19} Bu(0) + \cdots + ABu(18) + Bu(19).
\]
In matrix form, and substituting \( x(20) = 0 \), we have
\[
\begin{bmatrix}
A^{19} B & \cdots & AB & B
\end{bmatrix}
\begin{bmatrix}
u(0) \\
u(1) \\
u(2) \\
u(18) \\
u(19)
\end{bmatrix}
= -A^{20} x(0).
\]
(1)
Thus, we have an underdetermined set of linear equations, with 3 equations and 20 variables. We know how to find the smallest (or least-norm) solution of this set of equations, but we’re asked here to find the smoothest solution, so we have to do a little more work. We define the difference between two consecutive inputs as \( \delta(t) = u(t) - u(t-1) \), with \( \delta(0) = u(0) \) (consistent with \( u(-1) = 0 \)). These are sometimes called *input increments* or *input differences*. We can express the inputs in terms of these increments as
\[
u(t) = \delta(0) + \cdots + \delta(t),
\]
which can be expressed in matrix form as \( u = F \delta \), where \( u = (u(0), \ldots, u(19)) \), \( \delta = (\delta(0), \ldots, \delta(19)) \), and
\[
F = \begin{bmatrix}
1 & 0 & 0 & \cdots & 0 & 0 \\
1 & 1 & 0 & \cdots & 0 & 0 \\
& \vdots & \ddots & & \vdots & \vdots \\
1 & 1 & 1 & \cdots & 1 & 0 \\
1 & 1 & 1 & \cdots & 1 & 1
\end{bmatrix}.
\]
Now, we substitute $u = F\delta$ into (1) to obtain
\[
\begin{bmatrix}
  A^{19}B & \cdots & AB & B
\end{bmatrix} F\delta = -A^{20}x(0).
\]
Defining $G$ as $G = \begin{bmatrix} A^{19}B & \cdots & AB & B \end{bmatrix} F$, we can express this as
\[
G\delta = -A^{20}x(0).
\]

The original smoothness measure can be expressed in terms of $\delta$ as
\[
J_{\text{smooth}} = \left( \frac{1}{20} \sum_{t=0}^{19} (u(t) - u(t-1))^2 \right)^{1/2}
\]
\[
= \left( \frac{1}{20} \sum_{t=0}^{19} \delta(t)^2 \right)^{1/2} = \frac{1}{\sqrt{20}} \|\delta\|.
\]

Thus, we have reduced our problem to one we know how to solve: find the least norm vector $\delta_{\text{ln}}$ that satisfies $G\delta = -A^{20}x(0)$. The solution is given by
\[
\delta_{\text{ln}} = G^T(GG^T)^{-1}(-A^{20}x(0)).
\]

Then the smoothest input is given by
\[
u_{\text{smooth}} = F\delta_{\text{ln}} = -FG^T(GG^T)^{-1}A^{20}x(0).
\]

Its RMS value is $J_{\text{smooth}} = 1/\sqrt{20}\|\delta_{\text{ln}}\|$. Below is the matlab code used to compute the smoothest input.

```matlab
A = [1 .5 .25; .25 0 1; 1 -0.5 0];
B = [1; .1; .5];
x0 = [25; 0; -25];
N = 20;
Ck = [];
for k = 0:N-1
    Ck = [A^k*B Ck];
end
F = tril(ones(N,N)); % lower triagonal of ones
del_smooth = pinv(Ck*F)*(-A^N*x0);
u_smooth = F*del_smooth
plot([0:N-1],u_smooth); xlabel('time'); ylabel('usmooth');
Jsmooth = 1/sqrt(N)*norm(del_smooth)
xN = A^N*x0 + Ct*del_smooth
```

```
>> xN =
1.0e-10 *
   -0.0546
   -0.2910
   0.0909
```
The plot of the smoothest input is given below. Its RMS smoothness measure is \( J_{\text{smooth}} = 1.1246 \).

It is interesting to note some of the alternative methods people used to solve this problem. One method worked as follows: we have three equality constraints, \( i.e. \), the constraint that the three components of \( x(20) \) be zero. We can use these to eliminate three of our variables, say, \( u(17), u(18), \) and \( u(19) \). Thus, we have an \textit{unconstrained} problem, with 17 variables, \( \tilde{u} = (u(0), \ldots, u(16)) \). We can choose any values we like for these, and then \( u(17), u(18), \) and \( u(19) \) are determined (to ensure \( x(20) = 0 \)). Finally, we express the objective, the smoothness measure, as the norm of a linear function of \( \tilde{u} \) minus a constant vector. Then we apply least-squares. (That's the idea — the details and equations are not too pretty!) This method is correct, \( i.e. \), it gets the exact solution, so it got full credit. Some people formulated the the problem as a mixed quadratic problem with equality constraints, and derived the general solution of such problems. This also works. The most popular alternative approach involved a regularization method. In this method, the input \( u \) is considered unconstrained. We identify two objectives, \( J_{\text{smooth}} \) and \( J_{\text{miss}} \), where \( J_{\text{miss}} \) gives the final state error:

\[
J_{\text{miss}} = \| x(20) \|.
\]

We then form a regularized problem: we minimize

\[
J_{\text{smooth}}^2 + \lambda J_{\text{miss}}^2
\]

where \( \lambda > 0 \) is a parameter. Now we let \( \lambda \) get really big, which drives the final state error to be really small. This method works, at least in the limit. Even though the solution with a large value of \( \lambda \) is numerically very close to the exact solution, we did deduct some points for this. The reason is simple: this method is only approximate, or involves a limit. The other methods described above, in contrast, are exact.
3. Portfolio selection with sector neutrality constraints. We consider the problem of selecting a portfolio composed of \( n \) assets. We let \( x_i \in \mathbb{R} \) denote the investment (say, in dollars) in asset \( i \), with \( x_i < 0 \) meaning that we hold a short position in asset \( i \). We normalize our total portfolio as \( \mathbf{1}^T x = 1 \), where \( \mathbf{1} \) is the vector with all entries 1. (With normalization, the \( x_i \) are sometimes called portfolio weights.)

The portfolio (mean) return is given by \( r = \mu^T x \), where \( \mu \in \mathbb{R}^n \) is a vector of asset (mean) returns. We want to choose \( x \) so that \( r \) is large, while avoiding risk exposure, which we explain next.

First we explain the idea of sector exposure. We have a list of \( k \) economic sectors (such as manufacturing, energy, transportation, defense, ...). A matrix \( F \in \mathbb{R}^{k \times n} \), called the factor loading matrix, relates the portfolio \( x \) to the factor exposures, given as \( \mathbf{R}_{\text{fact}} = Fx \in \mathbb{R}^k \). The number \( \mathbf{R}_{\text{fact}}^i \) is the portfolio risk exposure to the \( i \)th economic sector. If \( \mathbf{R}_{\text{fact}}^i \) is large (in magnitude) our portfolio is exposed to risk from changes in that sector; if it is small, we are less exposed to risk from that sector. If \( \mathbf{R}_{\text{fact}}^i = 0 \), we say that the portfolio is neutral with respect to sector \( i \).

Another type of risk exposure is due to fluctuations in the returns of the individual assets. The idiosyncratic risk is given by

\[
\mathbf{R}_{\text{id}} = \sum_{i=1}^{n} \sigma_i^2 x_i^2,
\]

where \( \sigma_i > 0 \) are the standard deviations of the asset returns. (You can take the formula above as a definition; you do not need to understand the statistical interpretation.)

We will choose the portfolio weights \( x \) so as to maximize \( r - \lambda \mathbf{R}_{\text{id}} \), which is called the risk-adjusted return, subject to neutrality with respect to all sectors, i.e., \( \mathbf{R}_{\text{fact}} = 0 \). Of course we also have the normalization constraint \( \mathbf{1}^T x = 1 \). The parameter \( \lambda \), which is positive, is called the risk aversion parameter. The (known) data in this problem are \( \mu \in \mathbb{R}^n, F \in \mathbb{R}^{k \times n}, \sigma = (\sigma_1, \ldots, \sigma_n) \in \mathbb{R}^n, \) and \( \lambda \in \mathbb{R} \).

a) Explain how to find \( x \), using methods from the course. You are welcome (even encouraged) to express your solution in terms of block matrices, formed from the given data.

b) Using the data given in sector_neutral_portfolio_data.m, find the optimal portfolio. Report the associated values of \( r \) (the return), and \( \mathbf{R}_{\text{id}} \) (the idiosyncratic risk). Verify that \( \mathbf{1}^T x = 1 \) (or very close) and \( \mathbf{R}_{\text{fact}} = 0 \) (or very small).

Solution.

a) We define \( \Sigma \in \mathbb{R}^{n \times n} \) to be a diagonal matrix with \( \Sigma_{ii} = \sigma_i^2 \), so \( \mathbf{R}_{\text{id}} = x^T \Sigma x \). The problem we are trying to solve is

\[
\begin{align*}
\text{maximize} & \quad \mu^T x - \lambda x^T \Sigma x \\
\text{subject to} & \quad \mathbf{1}^T x = 1, \quad Fx = 0
\end{align*}
\]

with variable \( x \in \mathbb{R}^n \). Maximizing an objective is equivalent to minimizing the negative of the objective, so we can rewrite this as

\[
\begin{align*}
\text{minimize} & \quad -\mu^T x + \lambda x^T \Sigma x \\
\text{subject to} & \quad \mathbf{1}^T x = 1, \quad Fx = 0
\end{align*}
\]

9
We introduce Lagrange multipliers $\kappa \in \mathbb{R}$ and $\nu \in \mathbb{R}^k$ for the two constraints, and write the Lagrangian of this problem as

$$L(x, \nu, \kappa) = -\mu^T x + \lambda^T \Sigma x + \nu^T (Fx) + \kappa (1^T x - 1).$$

The optimality conditions are then given by

$$\nabla_x L = -\mu + 2\lambda \Sigma x + F^T \nu + \kappa 1 = 0,$$
$$\nabla_\nu L = F x = 0,$$
$$\nabla_\kappa L = 1^T x - 1 = 0,$$

which we can write in block matrix form as

$$\begin{bmatrix}
2\lambda \Sigma & F^T & 1 \\
F & 0 & 0 \\
1^T & 0 & 0 \\
\end{bmatrix}
\begin{bmatrix}
x \\
\nu \\
\kappa \\
\end{bmatrix}
= \begin{bmatrix}
\mu \\
0 \\
1 \\
\end{bmatrix}.$$

To find the optimal $x$ we solve this set of $n + k + 1$ linear equations in $n + k + 1$ variables.

b) Using the data provided we arrive at optimal values $r = 26.71$, and $R_{id} = 133.6$, with the optimal objective value of the original problem being $13.3$. The following matlab code implements the method of part (a).

```matlab
sector_neutral_portfolio_data

%M=
\begin{bmatrix}
2*lambda*diag(sigmas.^2) & F' & ones(n,1); \\
F & zeros(k,k) & zeros(k,1); \\
ones(1,n) & zeros(1,k) & 0; \\
\end{bmatrix};

v=[mu;zeros(k,1);1];

M\% this solves the set of linear equations M*all_vars = v
all_vars = M\v;

% extract the portfolio weights
x=all_vars(1:n)
% plug in to find the optimal objective value
objective = -(-mu'*x + lambda*x'*diag(sigmas.^2)*x)
r = mu'*x
R_id = x'*diag(sigmas.^2)*x
```

4. Controlling a system using the initial conditions. Consider the mechanical system shown below:

![Mechanical System Diagram]
Here $q_i$ give the displacements of the masses, $m_i$ are the values of the masses, and $k_i$ are the spring stiffnesses, respectively. The dynamics of this system are

$$\dot{x} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -\frac{k_1+k_2}{m_1} & \frac{k_2}{m_1} & 0 & 0 \\ \frac{k_2}{m_2} & -\frac{k_2}{m_2} & 0 & 0 \end{bmatrix} x$$

where the state is given by

$$x = \begin{bmatrix} q_1 \\ q_2 \\ \dot{q}_1 \\ \dot{q}_2 \end{bmatrix}.$$ 

Immediately before $t = 0$, you are able to apply a strong impulsive force $\alpha_i$ to mass $i$, which results in initial condition

$$x(0) = \begin{bmatrix} 0 \\ 0 \\ \alpha_1/m_1 \\ \alpha_2/m_2 \end{bmatrix}.$$ 

(i.e., each mass starts with zero position and a velocity determined by the impulsive forces.) This problem concerns selection of the impulsive forces $\alpha_1$ and $\alpha_2$. For parts a–c below, the parameter values are

$$m_1 = m_2 = 1, \quad k_1 = k_2 = 1.$$

Consider the following specifications:

a) $q_2(10) = 2$

b) $q_1(10) = 1$, $q_2(10) = 2$

c) $q_1(10) = 1$, $q_2(10) = 2$, $\dot{q}_1(10) = 0$, $\dot{q}_2(10) = 0$

d) $q_2(10) = 2$ when the parameters have the values used above (i.e., $m_1 = m_2 = 1$, $k_1 = k_2 = 1$), and also, $q_2(10) = 2$ when the parameters have the values $m_1 = 1$, $m_2 = 1.3$, $k_1 = k_2 = 1$.

Determine whether each of these specifications is feasible or not (i.e., whether there exist $\alpha_1$, $\alpha_2 \in \mathbb{R}$ that make the specification hold). If the specification is feasible, find the particular $\alpha_1$, $\alpha_2$ that satisfy the specification and minimize $\alpha_1^2 + \alpha_2^2$. If the specification is infeasible, find the particular $\alpha_1$, $\alpha_2$ that come closest, in a least-squares sense, to satisfying the specification. (For example, if you cannot find $\alpha_1$, $\alpha_2$ that satisfy $q_1(10) = 1$, $q_2(10) = 2$, then find $\alpha_i$ that minimize $(q_1(10) - 1)^2 + (q_2(10) - 2)^2$.) Be sure to be very clear about which alternative holds for each specification.

**Solution.** The dynamics of the system is given by $\dot{x} = Ax$ where (for $m_1 = m_2 = 1$ and $k_1 = k_2 = 1$)

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -2 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \end{bmatrix}$$
\[
A =
\begin{pmatrix}
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
-2 & 1 & 0 & 0 \\
1 & -1 & 0 & 0 \\
\end{pmatrix}
\]

\[
x(10) \text{ is related to } x(0) \text{ through } x(10) = \Phi x(0) \text{ where } \Phi = e^{10A} \text{ is}
\]

\[
\text{>> phi=expm(10*A)}
\]

\[
\phi =
\begin{pmatrix}
-0.3694 & 0.8431 & -0.2494 & 0.0515 \\
0.8431 & 0.4737 & 0.0515 & -0.1979 \\
0.5503 & -0.3009 & -0.3694 & 0.8431 \\
-0.3009 & 0.2494 & 0.8431 & 0.4737 \\
\end{pmatrix}
\]

\[
\text{>> The values of } q_1(0) \text{ and } q_2(0) \text{ are taken as zero and therefore}
\]

\[
x(10) = \begin{bmatrix}
-0.2494 & 0.0515 \\
0.0515 & -0.1979 \\
-0.3694 & 0.8431 \\
0.8431 & 0.4737 \\
\end{bmatrix}
\begin{bmatrix}
\alpha_1 \\
\alpha_2 \\
\end{bmatrix},
\]

in other words, the first two columns of \( \Phi \) are irrelevant. Take

\[
\Psi =
\begin{pmatrix}
-0.2494 & 0.0515 \\
0.0515 & -0.1979 \\
-0.3694 & 0.8431 \\
0.8431 & 0.4737 \\
\end{pmatrix}
\]

so that \( x(10) = \Psi \alpha \) where \( \alpha = [\alpha_1 \alpha_2]^T \). We define the matrix \( \psi \) in the matlab environment:

\[
\text{>> psi=phi(:,3:4)}
\]

\[
\psi =
\begin{pmatrix}
-0.2494 & 0.0515 \\
0.0515 & -0.1979 \\
-0.3694 & 0.8431 \\
0.8431 & 0.4737 \\
\end{pmatrix}
\]

a) We have \( q_2(10) = e_2^T x(10) \) where \( e_2 \) is the 2nd unit vector in \( \mathbb{R}^4 \). Therefore, \( \alpha_1 \) and \( \alpha_2 \) should satisfy the linear equation \( 2 = e_2^T \Psi \alpha \). There are two variables \( \alpha_1 \) and \( \alpha_2 \) but only one equation. Therefore the choice of \( \alpha_1 \) and \( \alpha_2 \) is not unique and we pick the minimum norm solution. In matlab:

\[
\text{>> alpha=pinv(psi(2,:))*2}
\]

\[
\alpha =
\begin{pmatrix}
2.4614 \\
-9.4647 \\
\end{pmatrix}
\]
So here we can meet the spec, and even have one extra degree of freedom, which we use to minimize the norm of $\alpha$.

b) The requirements in this part are more stringent than in the previous one. Here we have the additional requirement $q_1(10) = 1$ and we get two linear equations in two unknowns, i.e., $[e_1^T \ e_2^T]x(10) = \Psi\alpha$ which has a unique solution since the resulting matrix is invertible. In matlab:

$$\text{>> alpha=psi(1:2,:)}\backslash[1;2]$$
\begin{align*}
\text{alpha} &= \\
-6.4408 \\
-11.7798
\end{align*}$$

> In this case there is only one $\alpha$ that meets the specs; there is no extra freedom.

c) Here we have two more requirements $\dot{q}_1(10) = 0$ and $\dot{q}_2(10) = 0$ and therefore we get an overdetermined system of linear equations (four equations in two unknowns) $x(10) = \Psi\alpha$. We solve for $\alpha$ in a least-squares sense, which will show us if we are lucky and can find an exact solution. Using matlab:

$$\text{>> alpha=psi\backslash[1;2;0;0]}$$
\begin{align*}
\text{alpha} &= \\
-0.1361 \\
-0.3435
\end{align*}$$

> It can be checked that this $\alpha$ does not meet the spec, but it comes closest in the sense that $\|x(10) - [1 \ 2 \ 0 \ 0]^T\|$ is minimized.

d) With the new set of parameters $m_1 = 1$, $m_2 = 1.3$, $k_1 = k_2 = 1$ we get a new system $\dot{x} = \tilde{A}x$ with

$$\text{>> tilde_A=[0 \ 0 \ 1 \ 0;0 \ 0 \ 0 \ 1;-2 \ 1 \ 0 \ 0;1/1.3 -1/1.3 0 \ 0]}$$
\begin{align*}
\text{tilde_A} &= \\
0 & 0 & 1.0000 & 0 \\
0 & 0 & 0 & 1.0000 \\
-2.0000 & 1.0000 & 0 & 0 \\
0.7692 & -0.7692 & 0 & 0
\end{align*}$$

> Similarly we define $\tilde{\Phi}$ and $\tilde{\Psi}$:

$$\text{>> tilde_phi=expm(10*tilde_A)}$$
\begin{align*}
\text{tilde_phi} &= \\
-0.6221 & 0.8272 & -0.2229 & -0.5395 \\
0.6363 & 0.3961 & -0.4150 & -0.8869
\end{align*}$$
The requirements $q_2(10) = 2$ for $m_1 = m_2 = 1$, $k_1 = k_2 = 1$ and $q_2(10) = 2$ for $m_1 = 1$, $m_2 = 1.3$, $k_1 = k_2 = 1$ can be written as

$$2 = e_2^T \Psi \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix}, \quad 2 = e_2^T \tilde{\Psi} \begin{bmatrix} \alpha_1 \\ \alpha_2/1.3 \end{bmatrix} = e_2^T \tilde{\Psi} \begin{bmatrix} 1 & 0 \\ 0 & 1/1.3 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix},$$

or in matrix form

$$\begin{bmatrix} 2 \\ 2 \end{bmatrix} = \begin{bmatrix} e_2^T \Psi \\ e_2^T \tilde{\Psi} \begin{bmatrix} 1 & 0 \\ 0 & 1/1.3 \end{bmatrix} \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix}.$$ 

Using matlab:

```matlab
>> alpha=[psi(2,:);tilde_psi(2,:)*diag([1;1/1.3])]
alpha =
 8.2608
-7.9566
```

Here we can meet the spec, but there is only one solution.

5. Linear system with one-bit quantized output. We consider the system

$$\dot{x} = Ax, \quad y(t) = \text{sign}(c x(t))$$

where

$$A = \begin{bmatrix} -0.1 & 1 \\ -1 & 0.1 \end{bmatrix}, \quad c = \begin{bmatrix} 1 & -1 \end{bmatrix},$$

and the sign function is defined as

$$\text{sign}(a) = \begin{cases} +1 & \text{if } a > 0 \\ -1 & \text{if } a < 0 \\ 0 & \text{if } a = 0 \end{cases}$$

Roughly speaking, the output of this autonomous linear system is quantized to one-bit precision. The following outputs are observed:

$$y(0.4) = +1, \quad y(1.2) = -1, \quad y(2.3) = -1, \quad y(3.8) = +1$$
What can you say (if anything) about the following:

\[ y(0.7), \quad y(1.8), \quad \text{and} \quad y(3.7), \]

Your response might be, for example: “\( y(0.7) \) is definitely +1, and \( y(1.8) \) is definitely −1, but \( y(3.7) \) can be anything (i.e., −1, 0, or 1)”. Of course you must fully explain how you arrive at your conclusions. (What we mean by “\( y(0.7) \) is definitely +1” is: for any trajectory of the system for which \( y(0.4) = +1, \ y(1.2) = -1, \ y(2.3) = -1, \) and \( y(3.8) = +1, \) we also have \( y(0.7) = +1. \))

**Solution.** We know that \( x(t) = e^{At}x(0) \), so we have

\[ y(t) = \text{sign}(cx(t)) = \text{sign}(ce^{At}x(0)). \]

What’s unknown here is the exact value of \( x(0) \). The given output data is

\[ ce^{A0.4}x(0) > 0, \quad ce^{A1.2}x(0) < 0, \quad ce^{A2.3}x(0) < 0, \quad ce^{A3.8}x(0) > 0. \]

These data give us information about \( x(0) \). From that information about \( x(0) \), we can (maybe) determine information about

\[ y(0.7) = \text{sign} \left( ce^{A0.7}x(0) \right), \quad y(1.8) = \text{sign} \left( ce^{A1.8}x(0) \right), \quad y(3.7) = \text{sign} \left( ce^{A3.7}x(0) \right). \]

Geometrically, a constraint of the form \( fx(0) > 0 \), where \( f \in \mathbb{R}^{1 \times 2} \) is a row vector, defines a halfspace in \( \mathbb{R}^2 \), with inward normal vector \( f^T \). Therefore each of our four measurements of \( y \) gives us a halfspace in \( \mathbb{R}^2 \) that \( x(0) \) must lie in. Taking the intersection, we see that \( x(0) \) must lie in a cone or sector in \( \mathbb{R}^2 \). This cone is the set of all possible values of \( x(0) \) that are consistent with the measured data. This is shown below.
To determine the sign of $y(t)$ for $t = 0.7, 1.8, \text{ and } 3.7$, we have to check the sign of $ce^{At}x(0)$, for $t = 0.7, 1.8, 3.7$. Geometrically, this means we are checking whether the cone of possible values of $x(0)$ lies on one side, or both sides, of the halfspaces determine by $ce^{At}$, for $t = 0.7, 1.8, 3.7$. This results in $y(0.7) = -1$, and $y(1.8) = -1$, whereas $y(3.7)$ cannot be determined, since the
boundary of the halfplanes lies within the uncertainty region.

\[\begin{align*}
\text{Checking for } & y(0.7) \\
\text{Checking for } & y(1.8) \\
\text{Checking for } & y(3.7)
\end{align*}\]

The matlab code below solves the problem using the approach just described.

\[\begin{align*}
A &= \begin{bmatrix} -0.1 & 1 \\ -1 & 0.1 \end{bmatrix}; \quad c = \begin{bmatrix} 1 & -1 \end{bmatrix}; \\
M_1 &= c*\expm(A*0.4); \quad M_2 = c*\expm(A*1.2); \quad M_3 = c*\expm(A*2.3); \quad M_4 = c*\expm(A*3.8); \\
x &= \text{linspace}(-10,10); \quad \text{coeff1} = -M_1(1)/M_1(2); \quad y_1 = \text{coeff1}*x; \\
\text{coeff2} &= -M_2(1)/M_2(2); \quad y_2 = \text{coeff2}*x; \quad \text{coeff3} = -M_3(1)/M_3(2); \\
\text{coeff3} &= -M_3(1)/M_3(2); \quad y_3 = \text{coeff3}*x; \quad \text{coeff4} = -M_4(1)/M_4(2); \quad y_4 = \text{coeff4}*x; \\
\text{coeff5} &= -M_5(1)/M_5(2); \quad y_5 = \text{coeff5}*x; \quad \text{coeff6} = -M_6(1)/M_6(2); \quad y_6 = \text{coeff6}*x; \quad \text{coeff7} = -M_7(1)/M_7(2); \quad y_7 = \text{coeff7}*x; \\
[a,b] &= \text{meshgrid}(-10:.5:10,-10:.5:10); \quad \text{figure}(1); \quad \text{clf}; \\
\text{subplot}(2,2,1); \quad \text{plot}(x,y_1); \quad \text{grid on}; \quad \text{hold on}; \\
\text{axis}([-10 10 -10 10]); \\
\text{for } i=1:(\text{size}(a,1)^2); \\
\text{if } (b(i)<\text{coeff1}*a(i)); \quad \text{plot}(a(i),b(i),'.'); \quad \text{end}; \quad \text{end}; \\
\text{title}('Halfplanes for } y(0.4)'); \\
\text{subplot}(2,2,2); \quad \text{plot}(x,y_1); \quad \text{grid on}; \quad \text{hold on}; \\
\text{axis}([-10 10 -10 10]); \\
\text{for } i=1:(\text{size}(a,1)^2); \\
\text{if } (b(i)<\text{coeff1}*a(i)); \quad \text{plot}(a(i),b(i),'.'); \quad \text{end}; \quad \text{end};
\end{align*}\]
plot(x,y2,'--'); for i=1:(size(a,1))^2; if (b(i)<coeff2*a(i)); plot(a(i),b(i),'+'); end; end; title('Adding y(1.2)'); subplot(2,2,3); plot(x(1:50),y1(1:50)); grid on; hold on; axis([-10 10 -10 10]); for i=1:(size(a,1))^2; if ((b(i)<coeff1*a(i)) & (b(i)<coeff2*a(i))); plot(a(i),b(i),'.'); end; if (b(i)<coeff3*a(i)); plot(a(i),b(i),'+'); end; end; plot(x(51:100),y2(51:100)); plot(x,y3,'--'); title('Adding y(2.3)'); subplot(2,2,4); plot(x(1:50),y1(1:50)); grid on; hold on; axis([-10 10 -10 10]); for i=1:(size(a,1))^2; if ((b(i)<coeff1*a(i)) & (b(i)<coeff2*a(i))); plot(a(i),b(i),'.'); end; if (b(i)>coeff4*a(i)); plot(a(i),b(i),'+'); end; end; plot(x(51:100),y2(51:100)); plot(x,y4,'--'); title('Adding y(3.8)'); figure(2); clf plot(x(1:50),y1(1:50)); grid on; hold on; axis([-10 10 -10 10]); for i=1:(size(a,1))^2; if ((b(i)<coeff1*a(i)) & (b(i)>coeff4*a(i))); plot(a(i),b(i),'.'); end; end; plot(x(1:50),y4(1:50)); plot(x,y5,'--'); title('Resulting uncertainty region for x(0)'); figure(3); clf; subplot(2,2,1); plot(x(1:50),y1(1:50)); grid on; hold on; axis([-10 10 -10 10]); for i=1:(size(a,1))^2; if ((b(i)<coeff1*a(i)) & (b(i)>coeff4*a(i))); plot(a(i),b(i),'.'); end; if (b(i)<coeff5*a(i)); plot(a(i),b(i),'+'); end; end; plot(x(1:50),y4(1:50)); plot(x,y5,'--');
Some of you used another method, which was less geometric but perfectly correct. First we observe that the eigenvalues are $\pm 0.995 j$, so all solutions are periodic, and we have

$$ce^{At}x(0) = \alpha \cos(0.995t + \theta),$$

where $\alpha \geq 0$ and $\theta$ give another parametrization of all possible trajectories. In this problem all that matters in the sign of this signal, which simply changes every $\pi/0.995 = 3.16$ seconds. What we need to do is figure out bounds on when the signal zero crossings can be. Evidently the positive to negative crossing takes place between $t = 0.4$ and $t = 1.2$, and the negative to positive transition occurs between $2.3$ and $3.6$. Putting this together, using the fact that the two crossings are $3.16$ seconds apart, we find the original crossing occurs between $0.4 + 3.16 = 3.56$ and $3.8$. From these facts we see that $y(0.7) = -1$, $y(1.8) = -1$, whereas $y(3.7)$ cannot be determined.

6. Some basic properties of eigenvalues. Show the following:

a) The eigenvalues of $A$ and $A^T$ are the same.

b) $A$ is invertible if and only if $A$ does not have a zero eigenvalue.

c) If the eigenvalues of $A$ are $\lambda_1, \ldots, \lambda_n$ and $A$ is invertible, then the eigenvalues of $A^{-1}$ are $1/\lambda_1, \ldots, 1/\lambda_n$.

d) The eigenvalues of $A$ and $T^{-1}AT$ are the same.
Hint: you’ll need to use the facts that $\det A = \det(A^T)$, $\det(AB) = \det A \det B$, and, if $A$ is invertible, $\det A^{-1} = 1/\det A$.

Solution.

a) The eigenvalues of a matrix $A$ are given by the roots of the polynomial $\det(sI - A)$. From determinant properties we know that $\det(sI - A) = \det (sI - A)^T = \det (sI - A^T)$. We conclude that the eigenvalues of $A$ and $A^T$ are the same.

b) First we recall that $A$ is invertible if and only if $\det(A) \neq 0$. But $\det(A) \neq 0 \iff \det(-A) \neq 0$.
   
i. If 0 is an eigenvalue of $A$, then $\det(sI - A) = 0$ when $s = 0$. It follows that $\det(-A) = 0$ and thus $\det(A) = 0$, and $A$ is not invertible. From this fact we conclude that if $A$ is invertible, then 0 is not an eigenvalue of $A$.
   
ii. If $A$ is not invertible, then $\det(A) = \det(-A) = 0$. This means that, for $s = 0$, $\det(sI - A) = 0$, and we conclude that in this case 0 must be an eigenvalue of $A$. From this fact it follows that if 0 is not an eigenvalue of $A$, then $A$ is invertible.

c) From the results of the last item we see that 0 is not an eigenvalue of $A$. Now consider the eigenvalue/eigenvector pair $(\lambda_i, x_i)$ of $A$. This pair satisfies $Ax_i = \lambda_i x_i$. Now, since $A$ is invertible, $\lambda_i$ is invertible. Multiplying both sides by $A^{-1}$ and $\lambda_i^{-1}$ we have $\lambda_i^{-1} x_i = A^{-1} x_i$, and from this we conclude that the eigenvalues of the inverse are the inverse of the eigenvalues.

d) First we note that $\det(sI - A) = \det(I(sI - A)) = \det(T^{-1}T(sI - A))$. Now, from determinant properties, we have $\det(T^{-1}T(sI - A)) = \det(T^{-1}(sI - A)T)$. But this is also equal to $\det(sI - T^{-1}AT)$, and the conclusion is that the eigenvalues of $A$ and $T^{-1}AT$ are the same.

7. Characteristic polynomial. Consider the characteristic polynomial $\chi(s) = \det(sI - A)$ of the matrix $A \in \mathbb{R}^{n \times n}$.

   a) Show that $\chi$ is monic, which means that its leading coefficient is one: $\chi(s) = s^n + \cdots$.

   b) Show that the $s^{n-1}$ coefficient of $\chi$ is given by $-\text{trace} A$. (trace $X$ is the trace of a matrix: $\text{trace} X = \sum_{i=1}^n X_{ii}$)

   c) Show that the constant coefficient of $\chi$ is given by $\det(-A)$.

   d) Let $\lambda_1, \ldots, \lambda_n$ denote the eigenvalues of $A$, so that
   $$\chi(s) = s^n + a_{n-1}s^{n-1} + \cdots + a_1 s + a_0 = (s - \lambda_1)(s - \lambda_2)\cdots(s - \lambda_n).$$
   By equating coefficients show that $a_{n-1} = -\sum_{i=1}^n \lambda_i$ and $a_0 = \prod_{i=1}^n (-\lambda_i)$. 

20
Solution.

a) Expand the determinant expression to get

$$\det(sI - A) = (s - a_{11}) \det \tilde{A} + \text{other terms},$$

where $\tilde{A}$ is the $A$ matrix without the first row and first column. The other terms are similar, except for the fact that the determinant is multiplied by a scalar. Expanding $\det \tilde{A}$ we will reach a similar equation, and after expanding all terms you will reach something like

$$\det(sI - A) = \prod_{i=1}^{n} (s - a_{ii}) + \text{other terms}.$$

The other terms contribute with polynomials whose order is less than $n$, and since the first term is a monic polynomial with order $n$ it follows that $\det(sI - A)$ is also monic.

b) Let’s take a closer look at the relation

$$\det(sI - A) = \prod_{i=1}^{n} (s - a_{ii}) + \text{other terms}.$$

A little reasoning will show us that the other terms in fact are polynomials whose degree is less than $n - 1$ (provided that $n > 1$, and for $n = 1$ we have the trivial case). This is so because in the first expression of item (a) we have that $\tilde{A}$ is the only matrix that has $n - 1$ entries with $s$, and the same applies to other expansions of the expression. Then it follows that the $s^{n-1}$ term of $X$ is the $s^{n-1}$ term of $\prod (s - a_{ii})$. But this term is $\sum -a_{ii}$, which is equal to $-\text{trace} A$.

c) The constant coefficient is given by $X'(0)$. But $X$ is simply $\det(sI - A)$. By taking $s = 0$ it follows that $X'(0) = \det(-A)$.

d) First we note that, if $n = 1$, the relations are valid for the polynomial $s - \lambda_1$. Now suppose the relations are valid for a monic polynomial $P(s)$. Multiply $P(s)$ by $s - \lambda_i$ and expand as

$$P(s)(s - \lambda_i) = sP(s) - \lambda_i P(s).$$

Suppose $P(s)$ has degree $n$. Then $sP(s)$ is monic with degree $n + 1$ and the constant coefficient is zero. The polynomial $-\lambda_i P(s)$ has degree $n$, the $s^n$ coefficient is $-\lambda_i$ and the constant coefficient is $\prod (-\lambda_j)$. Since the constant coefficient of $sP(s)$ is zero we conclude by induction that $a_0 = \prod_{i=1}^{n} (-\lambda_i)$. Since $P(s)$ satisfies the properties, the $s^n$ term of $P(s)$ is $\sum \lambda_j$ and we conclude, again by induction, that $a_{n-1} = -\sum_{i=1}^{n} \lambda_i$.