• This is a 24-hour take-home exam. Please turn it in on Gradescope. Be aware that you must turn it in within 24 hours of downloading it. After that, Gradescope will not let you turn it in and we cannot accept it.

• You may use any books, notes, or computer programs, including searching online. You may not discuss the exam or course material with others, or work in a group.

• The exam should not be discussed at all until 12/10 after everyone has taken the exam.

• If you have a question, please submit a private question on Ed, or email the staff mailing list. We have tried very hard to make the exam unambiguous and clear, so unless there is a mistake on the exam we’re unlikely to say much.

• We expect your solutions to be legible, neat, and clear. Do not hand in your rough notes, and please try to simplify your solutions as much as you can. We will deduct points from solutions that are technically correct, but much more complicated than they need to be.

• Please check your email during the exam, just in case we need to send out a clarification or other announcement.

• Start each solution on a new page.

• When a problem involves some computation (say, using Julia), we do not want just the final answers. We want a clear discussion and justification of exactly what you did as well as the final numerical result.

• Because this is an exam, you must turn in your code. Include the code in your pdf submission. We reserve the right to deduct points for missing code.

• In the portion of your solutions where you explain the mathematical approach, you cannot refer to Julia operators, such as the backslash operator. (You can, of course, refer to inverses of matrices, or any other standard mathematical constructs.)

• Some of the problems require you to download data or other files. These files can be found at the URL

    http://ee263.stanford.edu/goodbye21.html

• Good luck!
1. **Chasing a sea monster.** A sea monster is loose in the Pacific Ocean! Your monster-chasing colleague has been measuring the sea monster’s movements and has predicted it will surface at $m$ positions $p_i \in \mathbb{R}^2$ at times $s_i$. Here $p_i$ is the $i$th column of the matrix $P$ given by

$$P = \begin{bmatrix} 1 & 1.75 & 2.4 & 2 & 0.5 & 0 \\ 0.75 & 0.6 & 1.2 & 2.3 & 0.75 & 0 \end{bmatrix}$$

and the times $s = (2, 5, 8, 11, 17, 20)$. You plan to observe the monster with a drone. Unfortunately the sea monster ate the last two drones you sent and you are almost out of research funding so your drone’s sensors are not very good, and the drone must be exactly in the right position to observe the monster.

a) The dynamics of the drone are

$$\ddot{q} = u$$

where $q \in \mathbb{R}^2$ is the position of the drone, and $u \in \mathbb{R}^2$ is an input force. Write this as a linear dynamical system of the form

$$\dot{x} = Ax + Bu$$
$$y =Cx$$

where $y \in \mathbb{R}^2$ is the position of the drone.

b) We will use sample period $h$. Assume that the force input is piecewise constant on sample intervals, and construct the exact discretization

$$x_d(k+1) = A_dx_d(k) + B_d u_d(k)$$
$$y_d(k) = C_d x_d(k)$$

where $x_d(k) = x(kh)$, and similarly for $y_d$ and $u_d$.

c) The drone starts at the origin with zero velocity, and we would like to move the drone so that $y(s_i) = p_i$ for $i = 1, \ldots, m$. We will operate the drone on the time interval $[0, T]$ where $T = s_m$. For convenience, let $N = T/h$. Since drone batteries are limited, we would like to minimize

$$J = \sum_{k=0}^{N-1} ||u_d(k)||^2$$

Explain in detail how you would solve this problem.

d) Use your method to compute the optimal input $u$, and plot $u$ versus time. Use $h = 0.1$.

e) Report the optimal value of $J$ that you obtained.

f) Plot the trajectory of the drone. Use axes $q_1$ and $q_2$, so that the plot shows the path followed by the drone. Mark on your plot the points $p_i$ where the monster surfaces.

g) Draw a sea monster for 1 point of extra credit.
Solution.

a) The dynamics are
\[ \dot{x} = Ax + Bu \]
\[ y = Cx \]
where
\[
A = \begin{bmatrix}
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0
\end{bmatrix},
B = \begin{bmatrix}
0 & 0 \\
1 & 0 \\
0 & 0 \\
0 & 1
\end{bmatrix},
C = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0
\end{bmatrix}
\]

b) We have
\[
A_d = \begin{bmatrix}
1 & h & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & h \\
0 & 0 & 0 & 1
\end{bmatrix},
B_d = \begin{bmatrix}
h^2/2 & 0 \\
h & 0 \\
0 & h^2/2 \\
0 & h
\end{bmatrix},
C_d = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0
\end{bmatrix}
\]

c) We want to constrain the drone’s position \( y_d(s_i) \) for \( i = 1, \ldots, m \). This is several constraints and we will have to concatenate them into one large equality constraint. We have
\[
y_d(t) = \begin{bmatrix}
C_dA_d^{i-1}B_d & \cdots & C_dA_dB_d & C_dB_d & 0 & \cdots & 0
\end{bmatrix}
\begin{bmatrix}
u_d(0) \\
\vdots \\
u_d(N-1)
\end{bmatrix}
= H_t u
\]
Define \( k_i = s_i/h \) and let the matrices \( H \) and \( z \) be
\[
H = \begin{bmatrix}
H_{k_1} \\
\vdots \\
H_{k_m}
\end{bmatrix},
z = \begin{bmatrix}
p_1 \\
\vdots \\
p_m
\end{bmatrix}
\]
Then this is a minimum norm problem, and so the optimal \( u \) is given by
\[ h = H^\dagger z \]

d) The plot is below.
e) The optimal $J$ is $J = 12.90$.

f) The trajectory is shown below.
using LinearAlgebra, Plots
include("readclassjson.jl")

# Part (a): Constructing A, B, C
A = 
\[
\begin{bmatrix}
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{bmatrix}
\]
B = 
\[
\begin{bmatrix}
0 & 0 \\
0 & 0 \\
1 & 0 \\
1 & 0
\end{bmatrix}
\]
C = 
\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0
\end{bmatrix}
\]
n = 4
m = 2

# Initial state is 0
x0 = zeros(n)

P = 
\[
\begin{bmatrix}
1 & 1.75 & 2.4 & 2.0 & 0.5 & 0.0 \\
0.75 & 0.6 & 1.2 & 2.3 & 0.75 & 0.0
\end{bmatrix}
\]
s = [2 5 8 11 17 20]
N = sh[end]

# Part (b) - exponentiate A, B
# exact value of Ad
Ad = exp(h*A)
# approximation of Bd
Bd = (h*I(n) + h^2/2*A + h^3/6*A^2 + h^4/24*A^3 + h^5/120*A^4)*B

# Part (c) - Construct the constraint matrix
constraints = []
for si in sh
    C_si = hcat([C*Ad^i*Bd for i=0:si-1]...)
    push!(constraints, [zeros(m,m*(N-si)) C_si])
end
C_bar = vcat(constraints...)
d = vec(P)

# solve problem
A_big = [2*I C_bar', C_bar zeros(6*m,6*m)]
b_big = [zeros(N*m); d]

u_lm = A_big \ b_big
u = u_lm[1:N*m]
# now we have control u
u = reshape(u,m,N)

# Part (d) - plot
xs = zeros(n,N+1)
ys = zeros(m,N+1)
xs[:,1] = x0
for t=1:N
    xs[:,t+1] = Ad*xs[:,t] + Bd*u[:,N-t+1]
    ys[:,t+1] = C*xs[:,t+1]
end
u = reverse(u, dims=2)

plot(h*(1:N), u[1,:], label="u1")
plot!(h*(1:N), u[2,:], label="u2")

# Part (e) - Minimum energy
norm(u)^2

# part (f)
x = zeros(n,N+1)
y = zeros(m,N+1)
x[:,1] = x0
for t=1:N
    x[:,t+1] = Ad*x[:,t] + Bd*y[:,t]
y[:,t+1] = C*x[:,t+1]
2. **System in a box.** You are given a mysterious box containing two unit masses connected via springs and dampers as follows.

![Diagram of a box with masses, springs, and dampers](image)

The equations of motion for this system are

\[
\ddot{q}_1 = -k_1 q_1 + k_2 (q_2 - q_1) - b_1 \dot{q}_1 + b_2 (\dot{q}_2 - \dot{q}_1) \\
\ddot{q}_2 = -k_2 (q_2 - q_1) - b_2 (\dot{q}_2 - \dot{q}_1)
\]

where \(k_i > 0\) are spring constants, \(b_i > 0\) are damping constants, and \(q_i\) is the displacement of mass \(i\). Both masses are \(m_i = 1\).

a) This mysterious box can be modeled as a continuous-time linear dynamical system

\[
\dot{x} = Ax
\]

Find \(A\) in terms of \(k_1, k_2, b_1, b_2\). Use state \(x = (q_1, q_2, \dot{q}_1, \dot{q}_2)\).

b) Use the forward Euler discretization

\[
x(t + h) = (I + hA)x(t)
\]

to simulate this system, with parameters \(k_1 = 1, k_2 = 2, b_1 = 1, b_2 = 3\). Set the initial conditions so that \(q_1(0) = 1, q_2(0) = 2\), and \(\dot{q}_1(0) = \dot{q}_2(0) = 0\). Use time step \(h = 0.1\), and simulate on time interval \([0, T]\) with \(T = 15\). Plot \(q_1, q_2, \dot{q}_1, \) and \(\dot{q}_2\) (on one plot) as functions of time.

c) Unfortunately your dog ate the documentation for this system in a box so you do not know the parameters \(k_1, k_2, b_1, b_2\). However, we have experimental data, consisting of measurements of \(x(t)\) with sample period \(h = 0.1\) on time interval \([0, T]\) where \(T = 15\). These may be found in the file `box.json`. We will use this data to estimate \(k_1, k_2, b_1, b_2\).

To do this, we will find the matrix \(A\) that minimizes

\[
\sum_{k=0}^{N-1} \|(I + hA)x(kh) - x(kh + h)\|^2
\]

where \(N = T/h\), and \(x\) is the given data. Explain how you would do this.
d) Apply your method from part (c) to estimate $A$, and report the estimate you find.

e) Using initial conditions of $x(0)$ in the dataset, and again with $h = 0.1$ and $T = 15$, simulate the system using your estimate of $A$ and plot $q_1$, $q_2$, $\dot{q}_1$, and $\dot{q}_2$ (on one plot) as functions of time.

Solution.

a) We have

\[
A = \begin{bmatrix}
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
-(k_1 + k_2) & k_2 & -(b_1 + b_2) & b_2 \\
k_2 & -k_2 & b_2 & -b_2
\end{bmatrix}
\]

b) The plot is

\[
\begin{array}{cccc}
0 & 20 & 40 & 60 & 80 & 100 & 120 & 140 & 160 \\
-1.0 & -0.5 & 0.0 & 0.5 & 1.0 & 1.5 & 2.0
\end{array}
\]

\[
\begin{array}{cccc}
0 & 20 & 40 & 60 & 80 & 100 & 120 & 140 & 160 \\
-1.0 & -0.5 & 0.0 & 0.5 & 1.0 & 1.5 & 2.0
\end{array}
\]

c) The problem is equivalent to

\[
\min_a \| W a - v \|
\]

where

\[
W = \begin{bmatrix}
h x(0) \otimes I \\
h x(h) \otimes I \\
\vdots \\
h x((N - 1)h) \otimes I
\end{bmatrix}
\]

\[
v = \begin{bmatrix}
x(h) - x(0) \\
x(2h) - x(h) \\
\vdots \\
x(Nh) - x((N - 1)h)
\end{bmatrix}
\]

and $a = \text{vec}(A)$.
d) We find
\[
A = \begin{bmatrix}
-0.0120 & 0.0229 & 1.0004 & 0.0081 \\
-0.0327 & -0.0157 & -0.0175 & 1.0054 \\
-6.5739 & 2.0700 & -1.9523 & 0.7532 \\
2.1230 & -2.0799 & 0.8136 & -0.7946
\end{bmatrix}
\]
e) The plot is below.

For comparison, the original data is here also.
using Plots, LinearAlgebra
include("readclassjson.jl")

# Part (a)
k1 = 1
k2 = 2
b1 = 1
b2 = 3
A = [0. 0. 1. 0.
     0. 0. 0. 1.
     -k1-k2 k2 -b1-b2 b2
     k2 -k2 b2 -b2]
x0 = [1. 2. 0. 0.]
h = 0.1
T = 15

# Part (b)
N = Integer(T/h)
x = zeros(4,N+1)
x[:,1] = x0
for t=1:N
    x[:,t+1] = (I + h*A)*x[:,t]
end
plot(h*(0:N), x[1,:], label="q1", color="blue")
plot!(h*(0:N), x[2,:], label="q2", color="green")
plot!(h*(0:N), x[3,:], label="q1 dot", color="red")
plot!(h*(0:N), x[4,:], label="q2 dot", color="orange")
xlabel!("time")

# part (c)
data = readclassjson("../data/box.json")
h = data["h"]
N = data["N"]
x = data["x"]
size(x)
x[:,1]
Xs = []
ys = []
for i=1:N
    for j=1:4
        v = zeros(1,16)
        v[(j-1)*4+1:j*4] = x[:,i]
        push!(Xs, v)
    end
    push!(ys,x[:,i+1]-x[:,i])
end
X = vcat(Xs...)
y = vcat(ys...)
println("size X ", size(X), " size y ", size(y))

A_vec = h*X \ y
A_est = reshape(A_vec,4,4)'

# part (e)
T = 150
h=0.1
xs = zeros(4, T+1)
x[:,1] = x[:,1]
for t=1:T
    xs[:,t+1] = (I + h*A_est)*xs[:,t]
end
plot(h*(0:N), xs[1,:], label="q1 est", color="blue")
plot!(h*(0:N), xs[2,:], label="q2 est",color="green")
plot!(h*(0:N), xs[3,:], label="q1 dot est",color="red")
plot!(h*(0:N), xs[4,:], label="q2 dot est",color="orange")
s Scatter!(h*(0:N), x[1,:], label="q1", color="blue")
s Scatter!(h*(0:N), x[2,:], label="q2", color="green")
s Scatter!(h*(0:N), x[3,:], label="q1 dot",color="red")
s Scatter!(h*(0:N), x[4,:], label="q2 dot",color="orange")
xlabel!("Time")

11
3. Measuring network latency. We have a network of three computers. There are six uni-
directional links connecting them, so any computer may send a message directly to any of the
others. If computer 1 sends as message to computer 2, it takes $l_{21}$ seconds to travel across the
link connecting them. The quantity $l_{21}$ is called the latency of the link from 1 to 2. For a real
network the link sends data at close to the speed of light, and so if the machines are in the
same datacenter then we would expect $l_{21}$ to be measured in nanoseconds.

We would like to determine the latencies of the six links $l_{12}, l_{21}, l_{13}, l_{31}, l_{23}, l_{32}$. At first
.glance the problem seems simple; simply subtract the time at which the message is sent from
the time at which the message is received. The difficulty is that this requires the two machines
to have synchronized clocks, because any offset in the clock will directly give an error in the
latency.

We can also use the computers to forward messages, so that for example computer 1 can
send a message to computer 2, who forwards it to computer 3. The total time taken for the
communication is then $l_{21} + l_{32}$, and we assume throughout that forwarding adds no additional
waiting time at the intermediate computer (which is called a relay.)

To address the synchronization problem, we avoid using two clocks by forwarding messages
in round trips. For example, we send a message from 1 to 2 to 3 to 1. We write this
as $1 \rightarrow 2 \rightarrow 3 \rightarrow 1$. Then computer 1 can measure the round-trip time using only its
clock. We assume that clocks can measure the duration of time accurately, even if they are
not synchronized with some absolute standard. For this round-trip, we therefore measure
$l_{21} + l_{32} + l_{13}$.

a) We measure the five round trips

$1 \rightarrow 2 \rightarrow 1 \quad 1 \rightarrow 3 \rightarrow 1 \quad 2 \rightarrow 3 \rightarrow 2 \quad 1 \rightarrow 2 \rightarrow 3 \rightarrow 1 \quad 3 \rightarrow 2 \rightarrow 1 \rightarrow 3$

which give measurements $y_1, \ldots, y_5$. Let $x$ be the vector of latencies

$x = (l_{12}, l_{21}, l_{13}, l_{31}, l_{23}, l_{32})$

Find the matrix $A$ such that $y = Ax$.

b) What is the rank of $A$?

c) It turns out that one cannot produce any vector $y \in \mathbb{R}^5$. There is a constraint that any
measurement $y$ must satisfy. This constraint can be expressed in the form $c^T y = 0$, for
some unit vector $c \in \mathbb{R}^5$. Find $c$. Give an interpretation of this constraint.

d) In general, we cannot determine the latencies $x$ given the measurements $y$. It is proposed
that, if we had exact knowledge of two of the latencies, we could determine the other
four. Your lab team sits down to discuss this, and comes up with three suggestions. We
say that the scheme works if it enables us to determine all of the unknown latencies.

i. Bob’s suggestion: knowledge of any two of the six latencies will work.

ii. Jeff’s suggestion: we need to choose the two latencies carefully. Some choices will
work, and some will not.

iii. Ruskin’s suggestion: there is no choice of two latencies that will work.

Who is right? If you think Jeff is right, give an example of two latencies that work, and
two that will not, and explain. If you think Bob or Ruskin is right, prove it.
Solution. Here is the solution.

a) The matrix $A$ is

$$A = \begin{bmatrix}
1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 \\
0 & 1 & 1 & 0 & 0 & 1 \\
1 & 0 & 0 & 1 & 1 & 0
\end{bmatrix}$$

b) $\text{rank}(A) = 4$

c) Pick $c = (1, 1, 1, -1, -1)$, normalized. The sum of the first three measurements is the sum of all the latencies, and this is the same as the sum of the last two measurements.

d) Jeff is right. The original matrix $A$ has rank 4, which means that we can add rows to it (i.e., make more measurements) to increase its rank. But you have to add the right ones.

If you know $l_{12}$ and $l_{13}$, then you have a measurement matrix

$$A_1 = \begin{bmatrix}
1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 \\
1 & 0 & 0 & 1 & 1 & 0 \\
0 & 1 & 1 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0
\end{bmatrix}$$

which has rank 6.

If on the other hand you know $l_{12}$ and $l_{21}$, then you have

$$A_2 = \begin{bmatrix}
1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 \\
1 & 0 & 0 & 1 & 1 & 0 \\
0 & 1 & 1 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0
\end{bmatrix}$$

which has rank 5. Knowing $l_{12}$ tells us $l_{21}$, since we have already measured $y_1 = l_{12} + l_{21}$. So there is no point measuring both $l_{12}$ and $l_{21}$.

4. Converging to the minimum-norm solution.

a) We have a discrete-time linear dynamical system

$$x(t + 1) = Ax(t) + b \quad x(0) = x_0$$

where $A \in \mathbb{R}^{n \times n}$ and $b \in \mathbb{R}^n$. Recall the spectral radius of a square matrix $A$ is

$$\rho(A) = \max_i |\lambda_i(A)|$$
Suppose $\rho(A) < 1$. Find an expression for

$$\lim_{t \to \infty} x(t)$$

in terms of $A$, $b$, and $x_0$.

b) There are some systems with $\rho(A) = 1$ for which $\lim_{t \to \infty} x(t)$ converges. Suppose $T \in \mathbb{R}^{n \times n}$ is invertible, and

$$A = T \begin{bmatrix} \hat{A} & 0 \\ 0 & I \end{bmatrix} T^{-1}$$

where $\hat{A} \in \mathbb{R}^{r \times r}$ and $\rho(\hat{A}) < 1$. Give conditions on $T, \hat{A}, b, x_0$ such that $\lim_{t \to \infty} x(t)$ converges.

c) Under your conditions of the previous part, what is the limit

$$\lim_{t \to \infty} x(t)$$

d) One system for which this condition is useful is as follows. We have a matrix $B \in \mathbb{R}^{m \times n}$. We emphasize here that $B$ is not known to be skinny or full rank, so the usual formula for the least-squares solution $(B^T B)^{-1} B^T y$ does not make sense. We would nonetheless like to solve the least-squares problem

$$\min_{x \in \mathbb{R}^n} \|Bx - y\|$$

Suppose $B$ has full singular value decomposition $B = U \Sigma V^T$. What are all the optimal solutions for this problem, in terms of the SVD?

e) When $B$ is very large, one commonly used method to solve this problem is the gradient method. Define the function

$$f(x) = \frac{1}{2} \|Bx - y\|^2$$

and consider the gradient update rule

$$x(t + 1) = x(t) - h \nabla f(x(t))$$

where $h > 0$ is a (small) step size. The update for the gradient method is a discrete-time linear dynamical system of the form

$$x(t + 1) = Ax(t) + b$$

Find $A$ and $b$ in terms of $h$, $B$ and $y$.

f) Show that, if $x(0) = 0$ and $h$ is sufficiently small, then

$$\lim_{t \to \infty} x(t) = B^\dagger y$$

The importance of this result is that it shows that the gradient algorithm for this problem converges to the minimum-norm solution.
Solution.

a) The solution to this system is

\[ x(t) = A^t x(0) + \sum_{\tau=0}^{t-1} A^{t-1-\tau} b \]

Using the geometric series formula

\[(I - A)(I + A + A^2 + \cdots + A^s) = I - A^{s+1}\]

gives

\[ x(t) = A^t x(0) + (I - A)^{-1}(I - A^t)b \]

Since \(\rho(A) < 1\), we have \(A^t \to 0\) as \(t \to \infty\), and so

\[ \lim_{t \to \infty} x(t) = (I - A)^{-1}b \]

b) Choose coordinates \(\tilde{x}(t) = T^{-1}x(t)\) and \(\tilde{b} = T^{-1}b\), then

\[ \tilde{x}(t+1) = \begin{bmatrix} \hat{A} & 0 \\ 0 & I \end{bmatrix} \tilde{x}(t) + \tilde{b} \]

and we partition \(\tilde{x}\) and \(\tilde{b}\) to write this as

\[ \tilde{x}_1(t+1) = \hat{A}\tilde{x}_1(t) + \tilde{b}_1 \]
\[ \tilde{x}_2(t+1) = \tilde{x}_2(t) + \tilde{b}_2 \]

Since \(\rho(\hat{A}) < 1\), this converges if and only if \(\tilde{b}_2 = 0\).

c) The limit is then

\[ \lim_{t \to \infty} \tilde{x}_1(t) = (I - \hat{A})^{-1}\tilde{b}_1 \]

and

\[ \lim_{t \to \infty} \tilde{x}_2(t) = \tilde{x}_2(0) \]

which we can write in the original coordinates as

\[ \lim_{t \to \infty} x(t) = T \begin{bmatrix} (I - \hat{A})^{-1} & 0 \\ 0 & 0 \end{bmatrix} T^{-1}b + T \begin{bmatrix} 0 & 0 \\ 0 & I \end{bmatrix} T^{-1}x(0) \]

d) All solutions are

\[ x = B^T y + V_2 z \]

for some \(z\).

e) We have

\[ \nabla f(x) = B^T Bx - B^T y \]

and so the linear dynamical system is

\[ x(t+1) = (I - hB^T B)x(t) + hB^T y \]

and

\[ A = I - hB^T B \quad b = hB^T y \]
f) Using the full SVD

\[
B = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} \Sigma_{11} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix}
\]

and so

\[
A = I - hB^TB = V \begin{bmatrix} I - h\Sigma_{11}^2 & 0 \\ 0 & 0 \end{bmatrix} V^T
\]

and

\[
b = V \begin{bmatrix} h\Sigma_{11} & 0 \\ 0 & 0 \end{bmatrix} U^T y
\]

Now we can use the previous result, with \( T = V, \hat{A} = I - h\Sigma_{11}^2 \). Notice that for \( h > 0 \) and small enough, \( \rho(\hat{A}) < 1 \) as required. Then, since \( x(0) = 0 \),

\[
\lim_{t \to \infty} x(t) = T \begin{bmatrix} (I - \hat{A})^{-1} & 0 \\ 0 & 0 \end{bmatrix} T^{-1} b
\]

\[
= V \begin{bmatrix} (h\Sigma_{11}^2)^{-1} & 0 \\ 0 & 0 \end{bmatrix} V^T V \begin{bmatrix} h\Sigma_{11} & 0 \\ 0 & 0 \end{bmatrix} U^T y
\]

\[
= V \begin{bmatrix} \Sigma_{11}^{-1} & 0 \\ 0 & 0 \end{bmatrix} U^T y
\]

\[
= B^T y
\]

5. **Filling-in missing data.** In this problem we have a signal, \( y_i \in \mathbb{R} \) for \( i = 1, \ldots, n \), which we view as \( y \in \mathbb{R}^n \). We will have \( n = 100 \). The signal \( y \) comes from measurements of a physical system, and so \( y_{i+1} \) is measured a short time interval after \( y_i \). Unfortunately, during the data acquisition process some of the data was lost and so the signal we have has gaps in it. Specifically, we have a known set \( K \subset \mathbb{Z} \) and we know \( y_i \) only for values \( i \in K \).

The data for this problem is in the file `missing.json`. The supplied vector `known` contains the list of known points \( K \), and the vector `yknown` is the list of values of \( y \) at the points in \( K \). The length of `yknown` is therefore \( |K| \).

a) For a signal \( z \in \mathbb{R}^n \), we define the discrete derivative \( z^{\text{der}} \in \mathbb{R}^{n-1} \) by

\[
z^{\text{der}}_i = z_{i+1} - z_i \quad \text{for} \quad i = 1, \ldots, n - 1
\]

Find the matrix \( G \) such that \( z^{\text{der}} = Gz \)

b) Our first approach will be to find the signal \( z \) which minimizes \( \|z^{\text{der}}\| \) and satisfies

\[
z_i = y_i \quad \text{if} \quad i \in K
\]

Give a method finding the optimal \( z \).

c) Find the optimal \( z \) in the previous part and plot \( z_i \) against \( i \). Be sure to plot the points \((i, z_i)\), not just a line joining them.
d) One way to do a better job at filling in the missing data is to put additional criteria on our estimate. Here we will do this by additionally penalizing the second derivative of $z$. Define the discrete second derivative $z^{hes} \in \mathbb{R}^{n-2}$ by

$$z^{hes}_i = z_{i+2} - 2z_{i+1} + z_i \quad \text{for } i = 1, \ldots, n - 2$$

Find the matrix $H$ such that $z^{hes} = Hz$

e) Define the two objective functions

$$J_1 = \|Gz\|^2 \quad J_2 = \|Hz\|^2$$

We would like to find the signal $z$ that minimizes

$$J_1 + \mu J_2$$

and satisfies

$$z_i = y_i \quad \text{if } i \in K$$

Give a method for finding the optimal $z$.

f) Plot the trade-off curve of $J_2$ (on the vertical axis) versus $J_1$ (on the horizontal axis). Give the interpretation of the endpoints of this curve.

g) Find the optimal $z$ for the three different cases $\mu = 5, 20, 100$.

**Solution.**

a) The matrix $G$ is

$$G_{ij} = \begin{cases} -1 & \text{if } i = j \\ 1 & \text{if } j = i + 1 \\ 0 & \text{otherwise} \end{cases}$$

b) The problem is in the form

$$\begin{align*}
\text{minimize} & \quad \|Az - b\| \\
\text{subject to} & \quad Cz = d
\end{align*}$$

which (from the lecture notes) has solution

$$\begin{bmatrix} x \\ \lambda \end{bmatrix} = \begin{bmatrix} A^T A & C^T \\ C & 0 \end{bmatrix}^{-1} \begin{bmatrix} A^T b \\ d \end{bmatrix}$$

Here we have $A = G$, $b = 0$. The matrix $C$ consists of the rows $i$ of the identity matrix for which $i \in K$, and $d = y$.

c) The optimal $z$ is below.
d) The matrix $H$ is
\[ H_{ij} = \begin{cases} 
1 & \text{if } i = j \\
-2 & \text{if } j = i + 1 \\
1 & \text{if } j = i + 2 \\
0 & \text{otherwise}
\end{cases} \]

e) This also has the same form as part b). In this case
\[ A = \begin{bmatrix} D \\ \sqrt{\pi H} \end{bmatrix} \]
f) The trade-off curve is
The bottom right corner is the solution when we minimize $J_2$, with $J_1$ unconstrained. It is the solution with smallest second derivative that interpolates the data. Similarly the upper left corner minimizes $J_1$, and is the solution with smallest first derivative that interpolates the data.

g) The optimal $z$ with $\mu = 5$ is below.

The optimal $z$ with $\mu = 20$ is below.
The optimal $z$ with $\mu = 100$ is below.

using LinearAlgebra, Plots
include("readclassjson.jl")

data = readclassjson("../data/missing.json")
known = data["known"]
K = size(known)[1]
y = data["y"]
n = maximum(known)
# part (a)
\[ G = [-I \ zeros(n-1)] + [\zeros(n-1) \ I] \]

# construct the matrix \( C \) for the constraint
\[
C = \zeros(K,n)
\]
for \( k=1:K \)
    \( C[k,\known[k]] = 1 \)
end

# parts (b), (c)
\[
A_{\text{big}} = [2.0*G'*G \ C' \\
C \ \zeros(K,K)]
\]
\[
b_{\text{big}} = [\zeros(n); \ y]
\]
z1 = \( (A_{\text{big}} \ b_{\text{big}})[1:n]\)
plot(z1)
scatter!(\known, y)

# part (d), (e)
\[
H = [I \ \zeros(n-2,2)] + [\zeros(n-2) \ -2.0*I \ \zeros(n-2)] + [\zeros(n-2) \ I]
\]

function find\_z(mu)
    \[
    A_{\text{big}} = [2.0*G'*G + mu*2.0*H'*H \ C' \\
C \ \zeros(K,K)]
\]
    \[
b_{\text{big}} = [\zeros(n); \ y]
\]
z = \( (A_{\text{big}} \ b_{\text{big}})[1:n]\)
    return z
end

# part (f) Plot the tradeoff curve
mus = [0.0 0.001 0.005 0.01 0.05 0.1 0.5 1.0 5.0 10.0 50.0 100.0]
J1 = []
J2 = []
for mu in mus
    z = find\_z(mu)
    push!(J1, norm(G*z)^2)
    push!(J2, norm(H*z)^2)
end
plot(J1, J2)
xlabel!("J1"); ylabel!("J2")
# The endpoints of the curve correspond to the places
# where one or the other objective dominates.
# the left endpoint is dominated by J2
# the right endpoint is dominated by J1

# part (g) find the optimal z for 3 cases
z_5 = find_z(5.0)
z_20 = find_z(20.0)
z_100 = find_z(100.0)
plot(z_5, label="mu=5")
plot!(z_20, label="mu=20")
plot!(z_100, label="mu=100")
scatter!(known, y, label="Known")

6. Some true/false questions. For each of the following statements, if it is true give a proof, and if it is false give a counterexample.

a) Suppose \( A = \begin{bmatrix} B & C \\ 0 & D \end{bmatrix} \) where \( B, C, D \) are matrices of compatible dimension. Then \( s \) is a singular value of \( A \) if and only if it is a singular value of \( B \) or \( D \).

b) Suppose \( A, B \in \mathbb{R}^{n \times n} \). There exists \( u \in \mathbb{R}^n \) such that
\[
A_{ij} = B_{ij} + u_i - u_j \quad \text{for all } i, j
\]
if and only if \( A + A^T = B + B^T \) (that is, \( A \) and \( B \) have the same symmetric part.)

c) Suppose \( Q \) is nonzero and \( Q + Q^T = 0 \). Then \( \text{rank}(Q) \geq 2 \).

d) Suppose \( A \in \mathbb{R}^{n \times n} \). Then \( \|A\| < 1 \) if and only if \( |\lambda_i(A)| < 1 \) for all \( i \).

e) Suppose \( A \in \mathbb{R}^{n \times n} \) and \( A \neq 0 \). Then \( A \) has at least one nonzero eigenvalue.

f) Suppose \( \|x\| = 1 \) and \( A = I - xx^T \). Then all eigenvalues of \( A \) are 0 or 1.

g) Suppose \( A \in \mathbb{R}^{m \times n} \) and \( \|A\| < 1 \). Then \( \|AB\| < \|B\| \) for all matrices \( B \).

h) Suppose \( A \) is not full rank. Then \( A^\dagger \) is neither a left-inverse nor a right-inverse of \( A \).

i) Suppose \( A \) is a matrix, and \( ABA = A \). Then \( \text{range}(I - BA) = \text{null}(A) \).

Solution.

a) False. Pick, for example
\[
B = \begin{bmatrix} 1 & 0 \\ \end{bmatrix} \quad C = 1 \quad D = 1
\]
Since \( A^T A = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix} \) we have \( \sigma_1(A) > \sqrt{2} \).

b) False. We can write this as
\[
A = B + 1u^T - u1^T
\]
which means \( A - B \) has rank 2. Pick
\[
A = \begin{bmatrix} 0 & -1 & -1 & -1 \\ 1 & 0 & -1 & -1 \\ 1 & 1 & 0 & -1 \\ 1 & 1 & 1 & 0 \end{bmatrix}
\]
and \( B = 0 \), then \( \text{rank}(A - B) = 4 \).
c) **True.** Suppose for a contradiction that \( \text{rank}(Q) = 1 \). Then we have

\[
Q = uv^T
\]

for some vectors \( u, v \), and so every column is some multiple of \( u \). But since \( Q = -Q^T = -vu^T \), every column is also some multiple of \( v \), and so \( u \) is a multiple of \( v \), say \( u = \alpha v \). Then \( Q = \alpha uu^T \) which is symmetric, and hence we have a contradiction.

d) **False.** For example,

\[
A = \begin{bmatrix} 0 & 2 \\ 0 & 0 \end{bmatrix}
\]

which has \( \|A\| = 2 \) but both eigenvalues are zero.

e) **False.** For example \( A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \).

f) **True.** Let \( U = \begin{bmatrix} x & V \end{bmatrix} \) where \( V \) is chosen to make \( U \) orthogonal. Then

\[
AU = U \begin{bmatrix} 0 & 0 \\ 0 & I \end{bmatrix}
\]

which gives the eigendecomposition of \( A \).

g) **True.** We know \( \|ABx\| < \|Bx\| \) since \( \|A\| < 1 \), and hence \( \|ABx\| < \|B\|\|x\| \) for all \( x \). Therefore \( \|AB\| < \|B\| \).

h) **True.** Since \( A \) does not have full rank, its SVD has the form

\[
A = U \begin{bmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{bmatrix} V^T
\]

and therefore

\[
A^\dagger A = V \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} V^T \neq I
\]

and a similar approach shows \( AA^\dagger \neq I \).

i) **True.** If \( x = (I - BA)z \) for some \( z \), then

\[
Ax = Az - ABAz = 0
\]

and so \( x \in \text{null}(A) \).

Conversely, if \( Ax = x \), then \( (I - BA)x = x - BAx = x \) and so \( x \in \text{range}(I - BA) \) as desired.