1. Orthonormalization

The idea of orthonormalization is that we take a list of vectors \( \vec{a}_1, \vec{a}_2, \ldots, \vec{a}_n \) and get a new list of vectors \( \vec{q}_1, \vec{q}_2, \ldots, \vec{q}_n \) such that the following properties are satisfied:

- **Spans are preserved:** For every \( 1 \leq \ell \leq n \), we know that \( \text{span}(\vec{a}_1, \ldots, \vec{a}_\ell) = \text{span}(\vec{q}_1, \ldots, \vec{q}_\ell) \).
- **The inner products of the \( \vec{q}_i \) with each other are zero — they are orthogonal.** That is: if \( i \neq j \), we know \( \vec{q}_i^T \vec{q}_j = 0 \).
- **The \( \vec{q}_i \) have unit norm whenever they are nonzero.** That is, if \( \vec{q}_i \neq \vec{0} \), then \( \vec{q}_i^T \vec{q}_i = 1 \).

An algorithm for doing this was derived naturally in lecture building on what you learned in 16A about the nature of projections. This problem is about making sure that you understand it within the context of mathematical induction. Mathematical induction is a basic proof technique that is critical to understand as we build mathematical maturity. Our follow-on course CS70 assumes exposure to mathematical induction as a prerequisite and expects students to be grow to be able to craft reasonably intricate inductive proofs from scratch. Here in 16B, our goal is simply for you to be able to follow through with an induction that we set up for you, and to follow inductive arguments.

Anyway, first, let us explicitly state the iterative algorithm:

```
1: for \( i = 1 \) up to \( n \) do
  \( \vec{r}_i = \vec{a}_i - \sum_{j < i} \vec{q}_j (\vec{q}_j^T \vec{a}_i) \) \hspace{1cm} \triangleright \text{Iterate through the vectors}
  2: if \( \vec{r}_i = \vec{0} \) then
  3:     \( \vec{q}_i = \vec{0} \)
  4: else
  5:     \( \vec{q}_i = \frac{\vec{r}_i}{||\vec{r}_i||} \) \hspace{1cm} \triangleright \text{Find the amount of } \vec{a}_i \text{ that remains after we project}
  6: end if
  7: end for
```

(a) From the If/Then/Else statement in the algorithm above, it is almost completely clear that the third desired property holds by construction, at least for the case that \( \vec{q}_i = \vec{0} \). **Show that** \( ||\vec{q}_i|| = 1 \) if \( \vec{q}_i \neq \vec{0} \), i.e. Why does the “normalize the vector” line actually result in something whose norm is 1?

**Solution:** When \( \vec{r}_i = \vec{0} \), then line 4 of the algorithm will make it terminate with \( \vec{q}_i = \vec{0} \). This means that the nonzero case must come from the else statement and so from line 6 of the algorithm, we have \( \vec{q}_i = \frac{\vec{r}_i}{||\vec{r}_i||} \). We know the norm of \( ||\vec{q}_i|| = ||\vec{r}_i|| \). Expanding this, we see

\[
||\vec{q}_i||^2 = \vec{q}_i^T \vec{q}_i = (\frac{\vec{r}_i}{||\vec{r}_i||})^T (\frac{\vec{r}_i}{||\vec{r}_i||}).
\]

Grouping terms and using the definition of the norm, we get

\[
||\vec{q}_i||^2 = \frac{1}{||\vec{r}_i||^2} \vec{r}_i^T \vec{r}_i = \frac{1}{||\vec{r}_i||^2} ||\vec{r}_i||^2 = 1.
\]
(b) To establish that the spans are the same, we need to proceed by induction over \( \ell \). This is a classic proof by induction. (You should always strongly suspect an inductive proof lurking when you see a for loop or a recursive construction in an algorithm.)

The statement is clearly true in the base case of \( \ell = 1 \) since the \( \vec{q}_1 \) is just a scaled version of \( \vec{a}_1 \). Now assume that it is true for \( \ell = k - 1 \). What is true? We need to translate the spans being the same into math. Namely that whenever we have an \( \vec{a}_k \) so that \( \vec{y} = \sum_{j=1}^{k-1} \vec{a}[j] \vec{a}_j \), we know there exists \( \vec{\beta} \) such that

\[
\vec{y} = \sum_{j=1}^{k-1} \vec{\beta}[j] \vec{q}_j. \quad \text{And vice-versa: from } \vec{\beta} \text{ to } \vec{\alpha}.
\]

**Show that the spans are the same for \( \ell = k \) as well.**

*(HINT: First write out what you need to show in one direction. Then just write \( \vec{a}_k \) in terms of \( \vec{q}_k \) and earlier \( \vec{q}_j \) and then proceed. Don’t forget the case that \( \vec{q}_k = \vec{0} \). Then make sure you do the reverse direction as well.)*

This establishes the induction step, and since we have the base case, we know that “all dominos must fall” and the statement is true for all \( \ell \). This follows the “dominos” picture for induction. Establishing the inductive step shows that each domino will knock over the next domino. The base case establishes that the first domino falls. And thus, they all must fall.

**Solution:** Assume for all \( \ell \leq k - 1 \), for any \( \alpha[1], \ldots, \alpha[k - 1] \) there exists \( \beta[1], \ldots, \beta[k - 1] \) such that

\[
\sum_{j=1}^{k-1} \beta[j] \vec{q}_j = \sum_{j=1}^{k-1} \alpha[j] \vec{a}_j,
\]

Then for \( \ell = k \), consider a generic set of \( \alpha \)'s, \( \alpha[1], \ldots, \alpha[k - 1], \alpha[k] \).

\[
\sum_{j=1}^{k} \alpha[j] \vec{a}_j = \sum_{j=1}^{k} \alpha[j] \vec{a}_j + \sum_{j=1}^{k-1} \alpha[j] \vec{a}_j
\]

\[
= \alpha[k] \left( \sum_{i<k} \vec{q}_i + \sum_{i<k} \vec{q}_i^T \vec{a}_i \right) + \sum_{j=1}^{k-1} \beta[j] \vec{q}_j
\]

\[
= \alpha[k] \left( \sum_{i<k} \vec{q}_i + \sum_{j=1}^{k-1} \beta[j] + \sum_{j=1}^{k-1} \alpha[k] \left( \sum_{i<k} \vec{q}_i^T \vec{a}_i \right) \right) \vec{q}_j
\]

The second equality is from line 2 of the algorithm where it states \( \vec{r}_k = \vec{a}_k - \sum_{i<k} \vec{q}_i (\vec{q}_i^T \vec{a}_k) \). This can be rearranged as

\[
\vec{a}_k = \vec{r}_k + \sum_{i<k} \vec{q}_i (\vec{q}_i^T \vec{a}_k).
\]

This finishes the first direction of the proof, as we can now write the vector \( \vec{y} \) as

\[
\vec{y} = \sum_{j=1}^{k} \beta[j] \vec{q}_j.
\]

Thus, it holds for \( \ell = k \). The opposite direction follows similarly, which is shown below.

Once again, for purposes of induction, we assume that the statement holds for \( \ell = k - 1 \). For any \( \beta[1], \ldots, \beta[k - 1] \) there exists \( \alpha[1], \ldots, \alpha[k - 1] \) such that

\[
\sum_{j=1}^{k-1} \beta[j] \vec{q}_j = \sum_{j=1}^{k-1} \alpha[j] \vec{a}_j
\]
Then for $\ell = k$, consider a generic set of $\vec{\beta}$’s, $\vec{\beta}[1], \ldots, \vec{\beta}[k-1]$.

$$\vec{y} = \sum_{j=1}^{k} \vec{\beta}[j] \vec{a}_j = \vec{\beta}[k] \vec{a}_k + \sum_{j=1}^{k-1} \vec{\beta}[j] \vec{a}_j$$

$$= \vec{\beta}[k] \frac{\vec{r}_k}{\|\vec{r}_k\|} + \sum_{j=1}^{k-1} \vec{\beta}[j] \vec{a}_j$$

$$= \vec{\beta}[k] \frac{\vec{r}_k}{\|\vec{r}_k\|} \left( \vec{a}_k - \sum_{j=k}^{\infty} \sum_{j=1}^{k-1} \vec{\beta}[j] (\vec{a}_j^T \vec{a}_k) \right) + \sum_{j=1}^{k-1} \vec{\beta}[j] \vec{a}_j$$

$$= \vec{\beta}[k] \frac{\vec{r}_k}{\|\vec{r}_k\|} \vec{a}_k + \sum_{j=1}^{k-1} \vec{\beta}[j] \left( \vec{a}_j - \frac{\vec{r}_k}{\|\vec{r}_k\|} (\vec{a}_j^T \vec{a}_k) \right).$$

Notice that the second sum is only up through $k - 1$ and involves a linear combination of the $\vec{a}_j$. So we can invoke our induction hypothesis and summon an appropriate set of $\alpha$ that would weight $\vec{a}_j$ to equal that second sum.

$$\vec{y} = \frac{\vec{\beta}[k]}{\|\vec{r}_k\|} \vec{a}_k + \sum_{j=1}^{k-1} \alpha[j] \vec{a}_j$$

Here, $\alpha[k]$ can just be defined to be $\frac{\vec{\beta}[k]}{\|\vec{r}_k\|}$ and we have proved the result for $\ell = k$. Thus, we have the new form of

$$\vec{y} = \sum_{j=1}^{k} \alpha[j] \vec{a}_j.$$

(c) To establish orthogonality, we also need to do another little proof by induction, where we again do induction over $\ell$. The statement we want to prove is that for all $j < \ell$, it must be that $\vec{a}_j^T \vec{a}_\ell = 0$. The base case here of $\ell = 1$ is trivially true since there are no $j < 1$. So, we can focus on the induction part of the proof.

Here, it is convenient to use what is sometimes called “strong induction” where we assume that we know for some $k - 1$ that for all $i \leq k - 1$, we have that for all $j < i$, that $\vec{a}_j^T \vec{a}_i = 0$. (i.e. We don’t just assume that the statement is true for $\ell = k - 1$, we assume it is true for all $\ell$ up to and including $k - 1$.)

(In the dominos analogy for induction, strong induction is just the fancy name for assuming that all the dominos have fallen before this one. And then showing that this one also falls. This is spiritually not that different from assuming that the previous domino has fallen and then showing that this one also falls.)

Based on this strong induction hypothesis, show by direct calculation that for all $i \leq k$ for all $j < i$, that $\vec{a}_j^T \vec{a}_i = 0$.

(HINT: The cases $i \leq k - 1$ are already covered by the induction hypothesis. So you can just focus on $i = k$. Next, notice that the case $\vec{q}_k = 0$ is also easily true. So, focus on the case $\vec{q}_k \neq 0$ and just expand what you know about $\vec{q}_k$. The strong induction hypothesis will then let you zero out a bunch of terms.)

This establishes the induction step, and since we have the base case, we know that “all dominos must fall” and the statement is true for all $\ell$.

Solution: The cases $i \leq k - 1$ are given by the induction hypothesis, as stated in the hint. Then, for all $i \leq k - 1$, for all $j < i$, $\vec{a}_j^T \vec{a}_i = 0$. Remember that $\vec{a}_j^T \vec{a}_j = 1$. All that remains is to deal with the case of $\vec{q}_k$ itself. We need to verify that it is orthogonal to all the previous $\vec{q}_j$. If $\|\vec{r}_k\| = 0$, then $\vec{q}_k = 0$ and so
it definitely holds since the zero vector is trivially orthogonal to every vector. So now we can assume \( \| \vec{r}_k \| > 0 \) and check for \( i = k \).

For any \( j < k \),

\[
\vec{q}_j^T \vec{q}_k = \frac{1}{\| \vec{r}_k \|} \left( \vec{a}_k - \sum_{n<k} \vec{a}_n (\vec{q}_n^T \vec{a}_k) \right) \tag{9}
\]

\[
= \frac{1}{\| \vec{r}_k \|} \left( \vec{q}_j^T \vec{a}_k - \vec{q}_j^T \sum_{n<k} \vec{a}_n (\vec{q}_n^T \vec{a}_k) \right) \tag{10}
\]

\[
= \frac{1}{\| \vec{r}_k \|} \left( \vec{q}_j^T \vec{a}_k - \sum_{n<k} \vec{a}_n (\vec{q}_n^T \vec{a}_k) \right) \tag{11}
\]

\[
= \frac{1}{\| \vec{r}_k \|} \left( \vec{q}_j^T \vec{a}_k - \vec{q}_j^T \vec{a}_k \right) = 0 \tag{12}
\]

The final step is a cancellation of the cross terms, since they are all zero. Only when the index \( n = j \) in the sum, the \( \vec{q}_j^T \vec{a}_j = 1 \) and \( \vec{q}_j^T \vec{a}_k \) survives. So it is proven.

(d) It turns out that the fact that the spans are the same can be summarized in matrix form. Let \( A = [\vec{a}_1, \ldots, \vec{a}_n] \) and \( Q = [\vec{q}_1, \ldots, \vec{q}_n] \). If \( A \) and \( Q \) have the same column span then it must be the case that \( A = QU \) where \( U = [\vec{u}_1, \ldots, \vec{u}_n] \) is a square matrix. After all, this \( U \) tells us how we can find the \( \vec{\beta} \) that correspond to a particular \( \vec{\alpha} \) — namely \( \vec{\beta} = U \vec{\alpha} \).

Show that a \( U \) can be found that is upper-triangular — that is that the \( i \)-th column \( \vec{u}_i \) of \( U \) has zero entries in it for every row after the \( i \)-th position.

(Hint: Matrix multiplication tells you that \( \vec{a}_1 = \sum_{j=1}^n \vec{u}_i[j] \vec{q}_i \). What does the algorithm tell you about this relationship? Can you figure out what \( \vec{u}_i[j] \) should be?)

Notice that this explicit construction of \( U \) can serve as part of an alternative proof of the fact that the spans are the same. The fact that the span of \( Q \) is contained within the span of \( A \) is immediate from the fact that by construction, the columns of \( Q \) are linear combinations of the columns of \( A \). The interesting part is the other direction — that the columns of \( A \) are also all linear combinations of the columns of \( Q \).

**Solution:** This is the kind of question that many students might have gotten stuck on. It is important to know how to start working on such things. As shown repeatedly in lectures and exemplified in discussions, the way to start is small. Our procedure is iterative, and our proofs have been inductive. So we should see what happens and discover the pattern.

While we are looking to find the pattern of interest, we can not worry about the case when vectors are linearly dependent, or zero. We are just trying to understand the basic story at this point.

We know that \( \vec{q}_1 = \frac{1}{\| \vec{q}_1 \|} \vec{a}_1 \), which in turn implies \( \vec{a}_1 = \vec{q}_1 \| \vec{a}_1 \| = \vec{q}_1 (\vec{q}_1^T \vec{a}_1) \). From there, \( \vec{q}_2 = \frac{1}{\| \vec{q}_2 \|} (\vec{a}_2 - \vec{q}_1 (\vec{q}_1^T \vec{a}_2)) \). Reversing this equation, we see that again

\[
\vec{a}_2 = \vec{q}_2 (\vec{q}_2^T \vec{a}_2) + \vec{q}_1 (\vec{q}_1^T \vec{a}_2).
\]

In other words, to get the coordinates for \( \vec{a}_2 \) in the orthonormal basis given by \( \vec{q}_1, \vec{q}_2 \), we just take the inner products of \( \vec{q}_i \) with \( \vec{a}_2 \).

\[
\begin{bmatrix}
\vec{a}_1 & \vec{a}_2
\end{bmatrix} =
\begin{bmatrix}
\vec{q}_1 & \vec{q}_2
\end{bmatrix} \cdot
\begin{bmatrix}
(\vec{q}_1^T \vec{a}_1) & (\vec{q}_1^T \vec{a}_2) \\
0 & (\vec{q}_2^T \vec{a}_2)
\end{bmatrix} \tag{13}
\]
Then, for $\bar{a}_3$ we get the same pattern again and can express them all using matrix multiplications:

$$
\begin{bmatrix}
\bar{a}_1 & \bar{a}_2 & \bar{a}_3
\end{bmatrix} =
\begin{bmatrix}
\bar{q}_1 & \bar{q}_2 & \bar{q}_3
\end{bmatrix}
\begin{bmatrix}
(\bar{q}_1^T \bar{a}_1) & (\bar{q}_1^T \bar{a}_2) & (\bar{q}_1^T \bar{a}_3) \\
0 & (\bar{q}_2^T \bar{a}_2) & (\bar{q}_2^T \bar{a}_3) \\
0 & 0 & (\bar{q}_3^T \bar{a}_3)
\end{bmatrix}
$$

(14)

This pattern allows us to guess that $\bar{u}_i[j] = \bar{q}_j^T \bar{a}_i$. And furthermore, we know why the terms below the diagonal are all zero — they are zero because we don’t need those $\bar{q}_j$ to express the relevant $\bar{a}_i$. Having a clear understanding makes doing the proof much easier. Let us consider the $i^{th}$ column of $A$, $\bar{a}_i$. We want to understand the weights $\bar{u}_i$ required to satisfy

$$
\bar{a}_i = \sum_{j=1}^{n} \bar{u}_i[j] \bar{q}_i.
$$

The relevant term in the algorithm is where $\bar{r}_i$ is being computed. We know $\bar{r}_i = \bar{a}_i - \sum_{j<i} \bar{q}_j (\bar{q}_j^T \bar{a}_i)$ and hence $\bar{a}_i = \bar{r}_i + \sum_{j<i} \bar{q}_j (\bar{q}_j^T \bar{a}_i)$. If $\bar{r}_i = \bar{0}$, we are already done since we have expressed $\bar{a}_i$ in terms of $\bar{q}_j$ with $j < i$. Otherwise, we know that $\bar{q}_i = \frac{\bar{r}_i}{||\bar{r}_i||}$ and hence $\bar{r}_i = ||\bar{r}_i|| \bar{q}_i$. So $\bar{a}_i = ||\bar{r}_i|| \bar{q}_i + \sum_{j<i} \bar{q}_j (\bar{q}_j^T \bar{a}_i)$ and in a sense, we are already done since we have expressed $\bar{a}_i$ in terms of $\bar{q}_j$ with $j < i$. However, it is nice to complete the story and notice that $\bar{q}_j^T \bar{a}_i = ||\bar{r}_i|| \bar{q}_j^T \bar{q}_i + \sum_{j<i} \bar{q}_j^T \bar{q}_j \bar{q}_j^T \bar{a}_i = ||\bar{r}_i||$ since all the cross terms cancel by the orthogonality proved in the previous part. So indeed $\bar{a}_i = \sum_{j=1}^{i} \bar{q}_j^T \bar{q}_i \bar{a}_i$. So the pattern we conjectured is actually proved.

2. BIBO Stability

(a) Consider the circuit below with $R = 1\Omega$, $C = 0.5F$, and $u(t) = \cos(t)$. Furthermore assume that $v(0) = 0$ (that the capacitor is initially discharged).

```
\begin{center}
\begin{tikzpicture}
  \node at (0,0) (u) {\textcolor{red}{+}};
  \node at (0,-1) (v) {\textcolor{blue}{-}};
  \draw[thick] (u) -- node[pos=0.5, above] {$R$} (v);
  \draw[thick] (u) -- node[pos=0.5, right] {$C$} (v);
  \node at (u) {$u(t)$};
  \node at (v) {$v$};
\end{tikzpicture}
\end{center}
```

This circuit can be modeled by the differential equation

$$
\frac{dv(t)}{dt} = -2v(t) + 2u(t)
$$

(15)

Show that $v(t)$ remains bounded for all time.

Thinking about this helps you understand what bounded-input-bounded-output stability means in a physical circuit.

Solution:

First, let’s follow the approach that was done in discussion.

Let’s try to make the differential equation $\frac{dv(t)}{dt} = -2v(t) + 2u(t)$ ‘blow up’ (ie have $v(t)$ grow towards + inf).
This means we want the right hand side to be positive:

\[-2v(t) + 2u(t) > 0 \quad \Rightarrow \quad u(t) > v(t)\]

But at some point, after \(v(t)\) has grown to 1, we can no longer have \(u > v\), as \(u(t) = \cos(t)\) is bounded by \(-1 < u(t) < 1\).

Thus at some point, the right-hand-side will not be positive, so \(v(t)\) cannot grow indefinitely. The system output is thus bounded in this case of a bounded input.

For the physical system, we can interpret this as: the voltage on the capacitor cannot ever exceed the voltage from the voltage source.

We can also try to understand this as a differential equation and see why it must be bounded directly. The input here is a specific cosine and so we could just calculate the solution and see that it is bounded.

Let us assume that |\(u(t)\)| ≤ \(k\). We know that the solution to the scalar differential equation is given by

\[x(t) = e^{-2t}x(0) + \int_0^t e^{-2(t-\tau)}2u(\tau)d\tau\]

Then we can bound \(x(t)\) for \(t \geq 0\) to get

\[|x(t)| \leq |e^{-2t}x(0)| + \int_0^t e^{-2(t-\tau)}2u(\tau)d\tau \quad (16)\]

\[\leq |e^{-2t}x(0)| + \int_0^t |e^{-2(t-\tau)}2u(\tau)|d\tau \quad (17)\]

\[= e^{-2t}|x(0)| + \int_0^t e^{-2(t-\tau)}2|u(\tau)|d\tau \quad (18)\]

\[\leq e^{-2t}|x(0)| + \int_0^t e^{-2(t-\tau)}2k\tau \quad (19)\]

\[= e^{-2t}|x(0)| + 2ke^{-2t}\int_0^t e^{2\tau}\tau \quad (20)\]

\[= e^{-2t}|x(0)| + 2ke^{-2t}\frac{1}{2}(e^{2t} - 1) \quad (21)\]

\[= e^{-2t}|x(0)| + k(1 - e^{-2t}) \quad (22)\]

\[= |x(0)| + k \quad (23)\]

The negative exponential is what makes this have to stay bounded.

(b) Consider the discrete-time system

\[x(t + 1) = -2x(t) + 2u(t) \quad (24)\]

with \(x(0) = 0\).

Is this system stable or unstable? If stable, prove it. If unstable, find a bounded input sequence \(u(t)\) that causes the system to ‘blow up’.

**Solution:** The system is unstable. This can be seen by considering the input

\[u(t) = 1, 0, 0, 0, 0, \ldots\]

This results in the state at time \(t \geq 1\),

\[x(t) = 2^t \cdot (-1)^{t+1} \]

And so

\[|x(t)| = 2^t\]

This is clearly exploding exponentially with \(t\), not staying bounded.
(c) For the example in the previous part, give an explicit sequence of inputs that are not zero but for which the state $x(t)$ will always stay bounded.

**Solution:**
There is a case for which a non-zero bounded input results in a bounded output:

$$u(t) = 1, 2, 1, 2, 1, 2, \ldots$$

In this case, we get $x(t) = 0$ when $t$ is even, and $x(t) = 2$ when $t$ is odd. In fact, there are an infinite number of input sequences that would result in bounded outputs. But because we can find a single example of a bounded input sequence that leads to an unbounded output, the system is deemed unstable. We can’t trust that we will only get nice inputs in engineering contexts.

(d) Consider a continuous-time scalar real differential equation with known solution

$$\frac{d}{dt}x(t) = ax(t) + bu(t) \quad x(t) = e^{at}x(0) + \int_0^t e^{a(t-\tau)}bu(\tau)d\tau.$$  

Show that if the system is unstable (has $a \geq 0$), then a bounded input can result in an unbounded output even if the initial condition was zero.

**Solution:** To start, let’s consider the case when $x(0) = 0$. Now, we are left with the integral term to show that a bounded input can result in an unbounded output. A bounded input $u$ implies

$$u(t) \leq k \forall t$$

Let’s consider the case when $u(t) = k \forall t$, giving us

$$x(t) = \int_0^t e^{a(t-\tau)}bk d\tau.$$  

Rearranging the integral, we can focus on the exponential term:

$$x(t) = bk \int_0^t e^{a(t-\tau)}d\tau.$$  

If $a = 0$, this is just $x(t) = bkt$ which is clearly growing without bound. For other $a \neq 0$, with a change of variables, we can evaluate this integral

$$\int_0^t e^{a(t-\tau)}d\tau = -\frac{1 - e^{at}}{a}.$$  

So, when $t \to \infty$, this will be unbounded since $e^{at}$ will grow exponentially for $a > 0$.  

© UCB EECS 16B, Fall 2019. All Rights Reserved. This may not be publicly shared without explicit permission.
(e) Repeat the previous part for the specific case of complex \( a = r + j2\pi \) where \( r > 0 \) and zero initial condition \( x(0) = 0 \).

All the other truly complex unstable cases are the same way for the same essential reason.

Solution: Following on from the integral form above,

\[
x(t) = \int_0^t e^{(r+j2\pi)(t-\tau)} bkd\tau.
\]

Using the solution to the integral above, we are left with

\[
\int_0^t e^{a(t-\tau)} d\tau = -\frac{1-e^{at}}{a} = -\frac{1-e^{(r+j2\pi)t}}{r+j2\pi}.
\]

It is less clear in the phasor case that this will diverge, but let’s consider the magnitude of this number.

\[
|x(t)| = \left| \frac{1-e^{(r+j2\pi)t}}{r+j2\pi} \right| = \frac{\sqrt{1+e^{2r\pi}}}{\sqrt{r^2+4\pi^2}}.
\]

As \( t \to \infty \), this magnitude is unbounded since \( r > 0 \).

(f) Repeat the previous part for the specific case of purely imaginary \( a = j2\pi \) with zero initial condition \( x(0) = 0 \).

All the other purely imaginary unstable cases are the same way for the same essential reason.

Solution: This part is a little different than the last two parts, that both diverged due to a positive exponential term. Here if you try the same input equal to all 1, it won’t work to show that the state grows without bound. Recall the solution of \( x(t) \) with the initial condition at zero

\[
x(t) = e^{at}x(0) + \int_0^t e^{a(t-\tau)} bu(\tau)d\tau.
\]

Consider this case of bounded input \( u(t) = e^{j2\pi t} \), which is bounded for all \( t \). This proof will follow by counterexample (remember the question asks to show that some bounded input exists that will make the state grow without bound). Plugging this input and a value in, we see

\[
x(t) = \int_0^t e^{j2\pi(t-\tau)} b e^{j2\pi\tau} d\tau = \int_0^t e^{j2\pi} b d\tau.
\]

Factoring out the terms that do not depend on \( \tau \), we are left with

\[
x(t) = be^{j2\pi} \int_0^t d\tau.
\]

Solving this integral, we get

\[
x(t) = bte^{j2\pi}.
\]

Which clearly diverges as \( t \to \infty \) since the magnitude \( |x(t)| = |bt| \).

(g) Consider the discrete-time real system with known solution:

\[
x(t+1) = ax(t) + bu(t) \quad x(0) = a^t x(0) + \sum_{\ell=0}^{t-1} a^{t-1-\ell} bu(\ell)
\]

Show that if the system is quite unstable (has \( |a| > 1 \)), then a bounded input can result in an unbounded output. Assume a zero initial condition here.
**Solution:** For simplicity, let’s say that $u(t) = 1 \forall t$. This gives us a new expression of

$$x(t + 1) = ax(t) + bu(t) \quad x(t) = a' x(0) + \sum_{\ell=0}^{t-1} a'^{-\ell} b$$

At $t = 0$, we start with

$$x(0) = 0.$$

Then, for larger $t$:

$$x(1) = \sum_{\ell=0}^{0} a'^{\ell} b = b.$$  

But this quickly becomes problematic, consider

$$x(2) = \sum_{\ell=0}^{1} a'^{-\ell} b = ab + b.$$  

Then,

$$x(t) = \sum_{\ell=0}^{t-1} a'^{-\ell} b = a'^{-1} b + \cdots + b$$

For $|a| > 1$, these terms quickly diverge. We can see this by looking at the geometric series sum for $|a| \neq 1$,

$$x(t) = \sum_{\ell=0}^{t-1} a'^{-\ell} b = b \frac{a'^{-1} - 1}{a - 1}$$

This is diverging since $a'$ has magnitude that grows without bound if $|a| > 1$.

(h) Repeat the previous part for the specific case of $a = -1$.

**Solution:** This part follows very closely to the previous part. Let’s start in considering the expanded sum form of the recurrence relation, with a non-specified input $u(t)$

$$x(t) = \sum_{\ell=0}^{t-1} a'^{-\ell} bu(t) = a'^{-1} bu(t-1) + a'^{-2} bu(t-2) + \cdots + abu(1) + bu(0)$$

Now, if we consider the bounded input of $u(t) = 1$ when $t$ is even, and $u(t) = -1$ when $t$ is odd. Namely, $u(t) = -1'$. Observe that

$$x(t) = \sum_{\ell=0}^{t-1} (-1)'^{-\ell} b (-1)' = \sum_{\ell=0}^{t-1} (-1)'^{-\ell} b.$$

Then, because the sum no longer depends on $\ell$, we are left with

$$x(t) = \sum_{\ell=0}^{t-1} (-1)'^{-\ell} b = (t-1)(-1)'^{-1} b.$$ 

This solution diverges over time with a bounded input.

(i) Now consider the discrete-time stable case where $a$ is complex and has $|a| < 1$. Show that as long as $|u(t)| < k$ for some $k$, that the solution $x(t)$ will be bounded for all time $t$.

(HINT: There are a few helpful facts about absolute values and inequalities that, while obvious, are helpful in such proofs. First: $|\sum_j a_j| \leq \sum_j |a_j|$. Second $|ab| = |a| \cdot |b|$. Third: $|e^{i\theta}| = 1$ no matter what real number $\theta$ is. And fourth, if $a_i > 0$ and $b_i > 0$, and $b_i \leq B$, then $\sum_i a_i b_i \leq \sum_i a_i B = B \sum a_i$.)
**Solution:** We want to show that all bounded inputs will result in a bounded output. First, we need to think about whether we want to consider the initial condition. Any reasonable definition of stability must let the bound on the output depend on the initial condition, otherwise there would always be a large enough initial condition that would violate the bound even if the system were obviously stable.

Let’s say that $0 \leq |x(0)| \leq \infty$ (a finite initial condition with a value that may be nonzero) and that $u(t) \leq \epsilon$ for all $t$ (the input is bounded at all timesteps). If we can find a bound for $|x(t)|$ for all timesteps (i.e. $|x(t)| \leq \alpha$ where $\alpha$ is some positive value) then we have shown that this system is stable. The $\alpha$ is allowed to depend on both $\epsilon$ and $x(0)$.

Once again, it is good to first understand why this is true before setting out to prove it.

Following this pattern we find:

$$x(1) = ax(0) + bu(0)$$

$$x(2) = ax(1) + bu(1) \rightarrow x(2) = a^2x(0) + abu(0) + bu(1)$$

$$x(3) = ax(2) + bu(2) \rightarrow x(3) = a^3x(0) + a^2bu(0) + abu(1) + bu(2)$$

Following this pattern we find:

$$x(t) = a^t x(0) + bu(t - 1) + abu(t - 2) + \cdots + a^{-1}bu(0) = a^t x(0) + \sum_{i=0}^{t-1} bu(i)a^{t-i-1}$$

Finding the magnitude of $|x(t)|$ (we are trying to find an upper bound for this value), we get

$$|x(t)| = |a^t x(0) + bu(t - 1) + abu(t - 2) + \cdots + a^{-1}bu(0)|.$$

At this point, it is useful to use the summation form so we can apply the hints more effectively

$$|x(t)| = |a^t x(0) + \sum_{i=0}^{t-1} bu(i)a^{t-i-1}|$$

$$\leq |a^t x(0)| + \sum_{i=0}^{t-1} bu(i)a^{t-i-1}$$

$$|a^t x(0)| + \sum_{i=0}^{t-1} bu(i)a^{t-i-1}$$

$$\leq |a^t x(0)| + \sum_{i=0}^{t-1} |b||u(i)||a^{t-i-1}|$$

$$= |a^t||x(0)| + \sum_{i=0}^{t-1} |b||u(i)||a^{t-i-1}|$$

$$= |a^t||x(0)| + \sum_{i=0}^{t-1} |b||u(i)||a^{t-i-1}|$$

$$\leq |a^t||x(0)| + \sum_{i=0}^{t-1} |b||u(i)||a^{t-i-1}|$$

$$= |a^t||x(0)| + \sum_{i=0}^{t-1} |b||u(i)||a^{t-i-1}|$$

$$< |a^t||x(0)| + \sum_{i=0}^{t-1} |b||u(i)||a^{t-i-1}|$$

$$= |a^t||x(0)| + \frac{|b||u(0)||a^t|}{1-|a|}$$

$$(30)$$

$$(31)$$

$$(32)$$

$$(33)$$

$$(34)$$

$$(35)$$

$$(36)$$
At this point, we have used most of the hints that were given. We bounded an absolute value of a sum by the sum of absolute values in the first two inequalities, and then we used the fact that the absolute value of a product is the product of absolute values in the next two equalities. Then we used the fact that making every term in a sum bigger makes the sum as a whole bigger for the next inequality. Then we pulled out a constant from a sum. Then we used the fact that adding positive terms to a sum only makes it bigger to get the strict inequality, and finally we used the formula for an infinite geometric series to get the last equality. This was also the path taken in lecture.

But we still have the pesky $|a|^t$ term out in front. Since we are interested in $|a| < 1$, we know that $|a|^t < 1$ as well. So we know that $|x(t)| < |x(0)| + \frac{|b| \varepsilon}{1-|a|}$ and we are done.

As we have found a bound for $|x(t)|$ given any initial condition and bounded input, we have shown that this system must be BIBO stable.

These kind of manipulations of sums and inequalities are a part of basic mathematical maturity. You have seen some of this in your basic calculus courses, and we need to keep up the practice so that you get to the right level for later courses that touch probability (70 and then 126), optimization (127 and then 189), control (128 and then 221a), signal processing (120 and then 123), etc. The ideas of bounding are also critical for doing more advanced circuit analysis and design.

3. Eigenvalue Placement through State Feedback

Consider the following discrete-time linear system:

$$\ddot{x}(t + 1) = \begin{bmatrix} -2 & 2 \\ -2 & 3 \end{bmatrix} \ddot{x}(t) + \begin{bmatrix} 1 \\ 1 \end{bmatrix} u(t).$$

In standard language, we have $A = \begin{bmatrix} -2 & 2 \\ -2 & 3 \end{bmatrix}, B = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ in the form: $\ddot{x}(t + 1) = Ax(t) + Bu(t)$.

(a) Is this system controllable?

Solution: We calculate

$$C = [B, AB] = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$$

Observe that the $C$ matrix has linearly independent columns (we can’t get the 1 as a multiple of the 0) and hence our system is controllable.

(b) Is this discrete-time linear system stable on its own?

Solution: We have to calculate the eigenvalues of matrix $A$. Thus,

$$\det(\lambda I - A) = 0$$

$$\det \begin{bmatrix} \lambda + 2 & -2 \\ 2 & \lambda - 3 \end{bmatrix} = 0$$

$$\lambda^2 - \lambda - 2 = 0$$

$$\lambda_1 = 2, \lambda_2 = -1$$

Since the magnitude of eigenvalue $\lambda_1$ is greater than 1, and the magnitude of $\lambda_2 = 1$, the discrete-time system is unstable.
(c) Suppose we use state feedback of the form 
\[ u(t) = \begin{bmatrix} f_1 & f_2 \end{bmatrix} \vec{x}(t) \]

Find the appropriate state feedback constants, \( f_1, f_2 \) so that the state space representation of the resulting closed-loop system has eigenvalues at \( \lambda_1 = -\frac{1}{2}, \lambda_2 = \frac{1}{2} \).

**Solution:** The closed loop system using state feedback has the form
\[
\vec{x}[t + 1] = \begin{bmatrix} -2 & 2 \\ -2 & 3 \end{bmatrix} \vec{x}(t) + \begin{bmatrix} 1 \\ 1 \end{bmatrix} u(t) = \begin{bmatrix} -2 & 2 \\ -2 & 3 \end{bmatrix} \vec{x}(t) + \begin{bmatrix} 1 \\ 1 \end{bmatrix} \cdot ([ f_1 \quad f_2 ] \vec{x}(t)) = \{ \begin{bmatrix} -2 & 2 \\ -2 & 3 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} \cdot [ f_1 \quad f_2 ] \} \vec{x}(t)
\]

Thus, the closed loop system has the form
\[
\vec{x}(t + 1) = \begin{bmatrix} -2 + f_1 & 2 + f_2 \\ -2 + f_1 & 3 + f_2 \end{bmatrix} \lambda \vec{x}(t)
\]

Thus, finding the characteristic polynomial of the above system we have
\[
\det(\lambda I - \begin{bmatrix} -2 + f_1 & 2 + f_2 \\ -2 + f_1 & 3 + f_2 \end{bmatrix}) = (\lambda + 2 - f_1)(\lambda - 3 - f_2) - (2 - f_2)(2 - f_1) = \lambda^2 - (1 + f_1 + f_2)\lambda + f_1 - 2
\]

However, we want to place the eigenvalues at \( \lambda_1 = -\frac{1}{2}, \lambda_2 = \frac{1}{2} \). That means we want
\[
\lambda^2 - (1 + f_1 + f_2)\lambda + f_1 - 2 = (\lambda + \frac{1}{2})(\lambda - \frac{1}{2})
\]
or equivalently:
\[
\lambda^2 - (1 + f_1 + f_2)\lambda + f_1 - 2 = \lambda^2 - \frac{1}{4}
\]

Equating the coefficients of the different powers of \( \lambda \) on both sides of the equation, we get,
\[
1 + f_1 + f_2 = 0 \\
\lambda - \frac{1}{2} = -\frac{1}{4}
\]
The above system of equations gives us \( f_1 = \frac{7}{4}, f_2 = -\frac{11}{4} \).

(d) Now suppose we’ve got a seemingly different system described by the controlled scalar difference equation \( z(t + 1) = z(t) + 2z(t - 1) + u(t) \). **Write down the above system’s representation in the following matrix form:**

\[
\vec{z}(t + 1) = A_z \vec{z}(t) + B_z u(t).
\]

Please specify what the vector \( \vec{z}(t) \) consists of as well as the matrix \( A_z \) and the vector \( B_z \).

(HINT: Just as “state” in the case of continuous time refers to anything that has a derivative taken in the system of differential equations, for discrete time systems, the concept of state refers to memory.)
What, besides the current input, must you remember about the past/present to be able to figure out the future? In this case, you must know both \( z(t) \) and \( z(t-1) \). It may be helpful if you form your state with \( z(t-1) \) above \( z(t) \).

**Solution:** From the problem, we have \( z(t+1) = z(t) + 2z(t-1) + u(t) \). Defining our state variable as \( \vec{z}(t) = \begin{bmatrix} z(t-1) \\ z(t) \end{bmatrix} \), we can write the equation equivalently in the matrix form,

\[
\vec{z}(t+1) = \begin{bmatrix} 0 & 1 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} z(t-1) \\ z(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t)
\]

where \( A_z = \begin{bmatrix} 0 & 1 \\ 2 & 1 \end{bmatrix} \), \( B_z = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \).

(e) We will now show how the initial matrix representation for \( \vec{x}(t) \) can be converted to the canonical form for \( \vec{z}(t) \) using a change of basis. Suppose we do a transformation of the coordinates of the state \( \vec{x}(t) \) to \( \vec{z}(t) = P\vec{x}(t) \). Write down the state-transition matrices of \( \vec{z}(t) \) in terms of the state transition matrices of \( \vec{x}(t) \), i.e., express \( A_z \) and \( B_z \) in terms of \( A, B, \) and \( P \). For \( P = \begin{bmatrix} -1 & 1 \\ 0 & 1 \end{bmatrix} \), confirm that the resulting state space representation of the behavior of \( \vec{z}(t) \) is indeed the same as the previous part (i.e. we get the same \( A_z, B_z \)).

**Solution:** Given that the new vector is transformed by the following matrix, \( \vec{z}(t) = P\vec{x}(t) \).

As we know from before, \( \vec{x}(t+1) = A\vec{x}(t) + Bu(t) \).

Now,

\[
\vec{z}(t+1) = P\vec{x}(t+1)
\]

\[
\vec{z}(t+1) = P(A\vec{x}(t) + Bu(t))
\]

\[
\vec{z}(t+1) = PAP^{-1}\vec{z}(t) + PBu(t)
\]

Thus,

\[
A_z = PAP^{-1}
\]

\[
B_z = PB
\]

We confirm that,

\[
A_z = \begin{bmatrix} 0 & 1 \\ 2 & 1 \end{bmatrix} = PAP^{-1} = \begin{bmatrix} -1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} -2 & 2 \\ -2 & 3 \end{bmatrix} \begin{bmatrix} -1 & 1 \\ 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} -1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} -2 & 2 \\ -2 & 3 \end{bmatrix} \begin{bmatrix} -1 & 1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 2 & 1 \end{bmatrix}
\]

(Note: the above is not a typo. The inverse of this particular \( P \) matrix is really itself.)

Also, confirm that

\[
B_z = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = PB = \begin{bmatrix} -1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix}
\]

The last part of the hint is important for getting clean matrices, if you formatted otherwise you will get substantially different answers.
(f) For the previous part, Design a feedback $[\bar{f}_1 \bar{f}_2]$ to place the closed-loop eigenvalues at $\lambda_1 = -\frac{1}{2}, \lambda_2 = \frac{1}{2}$. Confirm that $[f_1 f_2] = [\bar{f}_1 \bar{f}_2]P$.

Solution: Solving for the new feedback matrix: The closed loop system using state feedback has the form

$$\ddot{z}[t + 1] = \begin{bmatrix} 0 & 1 \\ 2 & 1 \end{bmatrix} \dot{z}(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t) = \begin{bmatrix} 0 & 1 \\ 2 & 1 \end{bmatrix} \dot{z}(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \cdot ([\bar{f}_1 \bar{f}_2] \ddot{z}(t)) = \begin{bmatrix} 0 & 1 \\ 2 & 1 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \cdot ([\bar{f}_1 \bar{f}_2]) \ddot{z}(t)$$

Thus, the closed loop system has the form

$$\ddot{z}(t + 1) = \begin{bmatrix} 0 & 1 \\ 2 + \bar{f}_1 & 1 + \bar{f}_2 \end{bmatrix} \ddot{z}(t)$$

Thus, finding the eigenvalues of the above system we have

$$\det(\lambda I - \begin{bmatrix} 0 & 1 \\ 2 + \bar{f}_1 & 1 + \bar{f}_2 \end{bmatrix}) = 0 \Rightarrow \lambda^2 - (1 + \bar{f}_2)\lambda - (2 + \bar{f}_1) = 0$$

However, we want to place the eigenvalue at $\lambda_1 = -\frac{1}{2}, \lambda_2 = \frac{1}{2}$. Thus, this means that

$$\lambda^2 - (1 + \bar{f}_2)\lambda - \bar{f}_1 - 2 = \lambda + \frac{1}{2})\lambda - \frac{1}{2}$$

$$\lambda^2 - (1 + \bar{f}_2)\lambda - \bar{f}_1 - 2 = \lambda^2 - \frac{1}{4}$$

Equating the co-efficients of $\lambda$ on both sides, we get

$$1 + \bar{f}_1 = 0$$
$$-\bar{f}_1 - 2 = -\frac{1}{4}$$

The above system of equations gives us $\bar{f}_1 = -\frac{7}{4}, \bar{f}_2 = -1$

Matrix multiplication shows that

$$\begin{bmatrix} \frac{7}{4} & -\frac{11}{4} \end{bmatrix} = [f_1 f_2] = [\bar{f}_1 \bar{f}_2] P = \begin{bmatrix} -\frac{7}{4} & -1 \end{bmatrix} \begin{bmatrix} -1 & 1 \\ 0 & 1 \end{bmatrix}$$
(g) We are now ready to go through some numerical examples to see how state feedback works. Consider the first discrete-time linear system. Enter the matrix A and B from (a) for the system

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

into the Jupyter notebook and use the random input \( w(t) \) as the disturbance introduced into the state equation. Observe how the norm of \( x(t) \) evolves over time for the given A. **What do you see happening to the norm of the state?**

**Solution:** See Jupyter notebook for solution. The norm of \( x(t) \) increases with time for the given A. This is because the matrix A has eigenvalues with magnitude greater than one as we discussed in (b) and thus the state keeps growing at each time step.

(h) Add the feedback computed in part (c) to the system in the notebook and **explain how the norm of the state changes.**

**Solution:** The eigenvalues of the closed loop system are at \( \frac{1}{2} \) and \(-\frac{1}{2}\). Thus, the norm of the state variable is now bounded with time. Check the solution in the Jupyter notebook.

(i) Now we evaluate a system described by the following scalar system \( z(t + 1) = az(t) + u(t) + w(t) \) in the Jupyter notebook. Consider two values of \( a \), one case with \( a > 1 \) and one with \( a < 1 \), to observe the difference in the evolution of \( |z(t)| \) for the same error as part (g). **Describe the differences between the two.**

**Solution:** For \(|a| > 1\), the norm of \( z(t) \) grows with time and is not bounded, while it is bounded for \(|a| < 1\). This is because the eigenvalue of the evolution is given by ‘a’ itself and it determines if the state is stable or not depending on its magnitude. Check the solution in the Jupyter notebook.

(j) Suppose that the disturbance is actually coming from observation noise. We assume \( y(t) = z(t) + w(t) \) where \( w(t) \) is some random noise. Add a state feedback \( u(t) = ky(t) \) to the system so that the resulting closed loop system is described by \( z(t + 1) = (a + k)z(t) + kw(t) \). Say we know \( a = -1.25 \). **For what values of k’s will the result be bounded? Confirm with the norm of the closed loop system.**

**Solution:** The system will be stable for \(|a + k| < 1\). For these values of k, the state will not grow exponentially. Check the solution in the Jupyter notebook. Therefore, \( k > 0.25 \) and \( k < 2.25 \).

(k) **Is it advisable to have \( a + k \) close to zero if we want to minimize the magnitudes of the state \( x(t) \)?**

**How does the effect of the noise in the observation influence this?** Assign values of \( k \) close to \(-a\) to see the effect in the Jupyter notebook. Compare to values that are smaller, but still keep it stable.

**Solution:** No. It is not advisable to have \( a + k \) close to zero. This is because the effect of error \( w(t) \) which is weighted by \( k \) will be larger for large values of \( k \) and a spurious error will have a significant effect on the state even though the state is BIBO stable. The effect can be seen in the Jupyter notebook for \( k = 1.25 \) and \( k = 0.4 \), where the instantaneous value of \( z(t) \) can go to larger values.

4. **Tracking a Desired Trajectory in Continuous Time**

The treatment in 16B so far has treated closed-loop control as being about holding a system steady at some desired operating point, which was designated as zero in a linear model. This control used the actual current state to apply a control signal designed to bring the state to zero. Meanwhile, the idea of controllability itself was more general and allowed us to make an open-loop trajectory that went pretty much anywhere. This problem is about combining these two ideas together to make feedback control more practical — how can we get a system to more-or-less closely follow a desired trajectory, even though it might not start exactly where we wanted to start and in principle could be buffeted by small disturbances throughout.

The key conceptual idea is to realize that we can change coordinates in a time-varying way so that “zero” is the desired “open-loop” trajectory.
In this question, we will also see that everything that you have learned to do closed-loop control in discrete-time can also be used to do closed-loop control in continuous time.

Now, consider the specific 2-dimensional system

\[
\frac{d}{dt} \vec{x}(t) = A\vec{x}(t) + \vec{b}u(t) = \begin{bmatrix} 2 & 1 \\ 0 & 2 \end{bmatrix} \vec{x}(t) + \begin{bmatrix} 1 \\ 1 \end{bmatrix} u(t)
\]

(38)

where \( u(t) \) is a scalar valued continuous control input.

(a) **Would the given system be controllable if we viewed the \( A, \vec{b} \) as the parameters of a discrete-time system?**

**Solution:** By substituting the matrix \( A \) and \( \vec{b} \) into the controllability matrix, we have:

\[
\mathcal{R}_2 = \begin{bmatrix} \vec{b} & A\vec{b} \end{bmatrix} = \begin{bmatrix} 1 & 3 \\ 1 & 2 \end{bmatrix} \text{ (full rank)}
\]

Since \( \mathcal{R}_2 \) is full rank, the system would be controllable if it were a discrete-time system.

It turns out (although we will not prove so in this course), being controllable also means that we can navigate the system state to any desired state by choosing an input trajectory \( u(t) \) in continuous-time as well. Showing this is a bit more subtle than it is for the discrete-time case, which is why we don’t do it. But as this problem shows, we certainly can see that controllability allows us to place the closed-loop eigenvalues wherever we want. This is the same in discrete-time and continuous-time.

(b) Now, suppose we started at \( \vec{x}(0) = \vec{0} \) and had a nominal control signal \( u_n(t) \) that would make the system follow the desired trajectory \( \vec{x}_n(t) \) that satisfies (38) together with \( u_n(t) \).

**Change variables using** \( \vec{x}(t) = \vec{x}_n(t) + \vec{v}(t) \) **and** \( u(t) = u_n(t) + u_v(t) \) **and write out what (38) implies for the evolution of the trajectory deviation** \( \vec{v}(t) \) **as a function of the control deviation** \( u_v(t) \).

Now, add a bounded disturbance term \( \vec{w}(t) \) to the original state evolution in (38) and let’s see if we can absorb that entirely within an evolution equation for \( \vec{v}(t) \) you found above. Write out the resulting equation for the dynamics as:

\[
\frac{d}{dt} \vec{v}(t) = A_v\vec{v}(t) + \vec{b}_v u_v(t) + \vec{w}(t)
\]

(39)

**What are** \( A_v \) **and** \( \vec{b}_v \) **?**

**Solution:** As indicated by the problem, we are provided with control signal \( u_n(t) \) such that the system starting at \( \vec{x}(0) = \vec{0} \) is able to follow a desired trajectory:

\[
\frac{d}{dt} \vec{x}_n(t) = A\vec{x}_n(t) + \vec{b}_n u_n(t)
\]

(40)

To obtain the evolution of \( \vec{v}(t) \), we substitute the change of variables into the original state equation:

\[
\frac{d}{dt} \frac{d}{dt} \vec{x}(t) = A\vec{x}(t) + \vec{b}_n u(t)
\]

\[
\frac{d}{dt} \vec{x}_n(t) + \frac{d}{dt} \vec{v}(t) = A\vec{x}_n(t) + A\vec{v}(t) + \vec{b}_n u_n(t) + \vec{b}_v u_v(t)
\]

(42)

Now, move the terms over, and then substitute the definition of the differential equation

\[
\frac{d}{dt} \vec{v}(t) = A\vec{x}_n(t) - \frac{d}{dt} \vec{x}_n(t) + A\vec{v}(t) + \vec{b}_n u_n(t) + \vec{b}_v u_v(t)
\]

(43)

\[
\frac{d}{dt} \vec{v}(t) = A\vec{v}(t) + \vec{b}_v u_v(t)
\]

(44)
where $A = \begin{bmatrix} 2 & 1 \\ 0 & 2 \end{bmatrix}$, $\vec{b} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$, and equation (44) is obtained by substituting (40) into the expression and cancelling it from both sides.

This describes the evolution of the trajectory deviation $\vec{v}(t)$ as a function of the control deviation $u_v(t)$. Next, to model the bounded disturbance, we have:

$$\frac{d}{dt} \vec{x}(t) = A \vec{x}(t) + \vec{b}u(t) + \vec{w}(t) = \begin{bmatrix} 2 & 1 \\ 0 & 2 \end{bmatrix} \vec{x}(t) + \begin{bmatrix} 1 \\ 1 \end{bmatrix} u(t) + \vec{w}(t)$$

(45)

By doing the changes of variables $\vec{x}(t) = \vec{x}_n(t) + \vec{v}(t)$ and $u(t) = u_n(t) + u_v(t)$ and using equation (40) into the equation, we have:

$$\frac{d}{dt} \vec{x}(t) = A \vec{x}(t) + \vec{b}u(t) + \vec{w}(t)$$

$$\frac{d}{dt} \vec{x}_n(t) + \frac{d}{dt} \vec{v}(t) = A \vec{x}_n(t) + A \vec{v}(t) + \vec{b}u_n(t) + \vec{b}u_v(t) + \vec{w}(t)$$

Therefore, the resulting equation is given by

$$\frac{d}{dt} \vec{v}(t) = A \vec{v}(t) + \vec{b}u_v(t) + \vec{w}(t)$$

By comparison, we can see that $A_v = A$, $\vec{b}_v = \vec{b}$ because nominally it is completed assuming no disturbance.

Notice that the disturbance $\vec{w}(t)$ is entirely something that must be dealt with in the $\vec{v}$ dynamics. It doesn’t effect the nominal trajectory at all.

(c) Based on what you have found above, how will the system behave over time? If there is some small disturbance, will we end up following the intended trajectory $\vec{x}_n(t)$ closely if we just apply the control $u_n(t)$ to the original system?

Solution:

The key is to study the system equation for $\vec{v}$, given by:

$$\frac{d}{dt} \vec{v}(t) = A_v \vec{v}(t) + \vec{w}(t)$$

where $u_v(t) = 0$ as indicated in the problem, i.e., just apply the control $u(t) = u_n(t)$.

If the state $\vec{v}(t)$ fluctuates around $\vec{0}$ under the small disturbance without any control, then we can end up following the intended trajectory $\vec{x}_n(t)$.

Our task, therefore, is to study the stability of the system.

So, what is the condition for stability in the continuous-time case? Recall it is that the real part of the eigenvalues must be less then zero

$$\Re(\lambda) < 0.$$ 

The form of the matrix $A_v$ was designed to make this easier to see. The second component of $\vec{v}$ experiences a scalar differential equation $\frac{d}{dt}v_2(t) = 2v_2(t) + u(t) + w_2(t)$. This differential equation, left to itself, would have exploding exponential trajectories $e^{2t}$ from what you have seen earlier. So this is clearly unstable. What makes it unstable? It is something about the eigenvalue 2.
Is it the size of 2? No. Because if it had been $-2$, we would have had a dying exponential $e^{-2t}$ and that would have been fine. The integral form of the solution to a scalar differential equation with inputs would have shown clear convergence.

So, it turns out that what matters is the real-part of the eigenvalues. If any of them have a real part that is not strictly negative, then over time the state $\vec{v}$ can grow without bound in response to a disturbance. However you derived it or reasoned about it, this means that the system is getting further and further away from the original desired trajectory.

Consequently, for stability reasons, we want the eigenvalues to have strictly negative real parts.

To summarize, since $A_\nu$ is a upper-triangular matrix, its eigenvalues lie on the diagonal, namely, 2 and 2. In this case, since they clearly have real parts greater than zero, we can see that the system is vulnerable to any disturbances $\vec{w}(t)$, and we will not end up following the intended trajectory $\vec{x}_n(t)$.

Now, we want to apply state feedback control to the system to get it to more or less follow the desired trajectory.

(d) Just looking at the $\vec{v}(t), u_\nu(t)$ system, how would you apply state-feedback to choose $u_\nu(t)$ as a function of $\vec{v}(t)$ that would place both the eigenvalues of the closed-loop $\vec{v}(t)$ system at $-10$.

**Solution:**

We can assume that the input $u_\nu = \begin{bmatrix} f_0 & f_1 \end{bmatrix} \vec{v}(t)$, which is a linear function of the current state $\vec{v}(t)$.

With the new input, the system equation for $\vec{v}(t)$ without any disturbance is given by:

\[
\frac{d}{dt} \vec{v}(t) = A_\nu \vec{v}(t) + \vec{b} \begin{bmatrix} f_0 & f_1 \end{bmatrix} \vec{v}(t)
\]

\[
\frac{d}{dt} \vec{v}(t) = \begin{bmatrix} 2 + f_0 & 1 + f_1 \\ f_0 & 2 + f_1 \end{bmatrix} \vec{v}(t)
\]

where we denote $A_{cl} = \begin{bmatrix} 2 + f_0 & 1 + f_1 \\ f_0 & 2 + f_1 \end{bmatrix}$ as the state matrix for the closed loop system. The characteristic polynomial for finding the eigenvalues of $A_{cl}$ is given by:

\[
\text{det}(\lambda I - A_{cl}) = \begin{vmatrix} \lambda - 2 - f_0 & -1 - f_1 \\ -f_0 & \lambda - 2 - f_1 \end{vmatrix} = \lambda^2 - (4 + f_0 + f_1)\lambda + f_0 + 2f_1 + 4
\]

To set the eigenvalues to be where we want, we set this equal to $(\lambda + 10)(\lambda + 10) = \lambda^2 + 20\lambda + 100$.

By comparing the coefficients, we have:

\[-(4 + f_0 + f_1) = 20 \]
\[f_0 + 2f_1 + 4 = 100\]

Solving the above system of equations, we can find $f_0 = -144$, $f_1 = 120$. Therefore, we can design the state-feedback $u_\nu(t) = \begin{bmatrix} -144 & 120 \end{bmatrix} \vec{v}(t)$ which will place both the eigenvalues of the closed loop system at -10.

Why did we pick -10? So that it would be stable and aggressively reject disturbances.
(e) Based on what you did in the previous parts, and given access to the desired trajectory $\vec{x}_n(t)$, the nominal controls $u_n(t)$, and the actual measurement of the state $\vec{x}(t)$, come up with a way to do feedback control that will keep the trajectory staying close to the desired trajectory no matter what the small bounded disturbance $\vec{w}(t)$ does.

**Solution:**

From the previous parts, we have successfully found a feedback control law $u_v(t) = \begin{bmatrix} f_0 & f_1 \end{bmatrix} \vec{v}(t)$ such that the closed-loop system for $\vec{v}(t)$ can be kept around $\vec{0}$ as long as the disturbances are bounded. By changing variables $\vec{x}(t) = \vec{x}_n(t) + \vec{v}(t)$ and $u(t) = u_n(t) + u_v(t)$ that we performed in (c) and (d), the state $\vec{x}$, as a result, will stay close to the desired trajectory no matter what the small bounded disturbance $\vec{w}(t)$ does.

Explicitly $u(t) = u_n(t) + u_v(t) = u_n(t) + \begin{bmatrix} -144 & 120 \end{bmatrix} (\vec{x}(t) - \vec{x}_n(t))$ is the control law that we would invoke to achieve this.

This lets us figuratively have our cake and eat-it too! We can use the original system to plan and by using closed-loop feedback, we can make sure that we mostly follow our plan even in the face of disturbances.

5. Stability for information processing: solving least-squares via gradient descent with a constant step size

Although ideas of control were originally developed to understand how to control physical and electronic systems, they can be used to understand purely informational systems as well. Most of modern machine learning is built on top of fundamental ideas from control theory. This is a problem designed to give you some of this flavor.

In this problem, we will derive a dynamical system approach for solving a least-squares problem which finds the $\vec{x}$ that minimizes $||A\vec{x} - \vec{y}||^2$. We consider $A$ to be tall and full rank — i.e. it has linearly independent columns.

As covered in EE16A, this has a closed-form solution:

$$\vec{x} = (A^T A)^{-1} A^T \vec{y}.$$  

Direct computation requires the “inversion” of $A^T A$, which has a complexity of $O(N^3)$ where $(A^T A) \in \mathbb{R}^{N \times N}$. This may be okay for small problems with a few parameters, but can easily become unfeasible if there are lots of parameters that we want to fit. Instead, we will solve the problem iteratively using something called “gradient descent” which turns out to fit into our perspective of state-space dynamic equations. Again, this problem is just trying to give you a flavor for this and connect to stability, “gradient descent” itself is not yet in scope for 16B.

(a) Let $\vec{x}(t)$ be the estimate of $\vec{x}$ at time step $t$. We can define the least-squares error $\vec{e}(t)$ to be:

$$\vec{e}(t) = \vec{y} - A\vec{x}(t)$$

Show that if $\vec{x}(t) = \vec{x}$, then $\vec{e}(t)$ is orthogonal to the columns of $A$, i.e. show $A^T \vec{e}(t) = \vec{0}$.

This was shown to you in 16A, but it is important that you see this for yourself again.

**Solution:** Plugging in $\vec{x} = (A^T A)^{-1} A^T \vec{y}$ for $\vec{x}$:
\[ \vec{e}(t) = \vec{y} - A\vec{x}(t) \]
\[ \vec{e}(t) = \vec{y} - A(A^T A)^{-1}A^T \vec{y} \]
\[ A^T \vec{e}(t) = A^T \left( \vec{y} - A(A^T A)^{-1}A^T \vec{y} \right) \]
\[ = A^T \vec{y} - (A^T A)(A^T A)^{-1}A^T \vec{y} \]
\[ = A^T \vec{y} - IA^T \vec{y} \]
\[ = A^T \vec{y} - A^T \vec{y} \]
\[ = \vec{0} \]

(b) We would like to develop a “fictional” state space equation for which the state \( \vec{x}(t) \) will converge to \( \hat{\vec{x}}(t) \), the true least squares solution. The evolution of these states reflects what is happening computationally.

Here \( A\vec{x}(t) \) represents our current reconstruction of the output \( \vec{y} \). The difference \( (\vec{y} - A\vec{x}(t)) \) represents the current residual.

We define the following update:

\[ \vec{x}(t + 1) = \vec{x}(t) + \alpha A^T (\vec{y} - A\vec{x}(t)) \] (46)

that gives us an updated estimate from the previous one. Here \( \alpha \) is the step-size that we get to choose. For us in 16B, it doesn’t matter where this iteration comes from. But if you want, this can be interpreted as a tentative sloppy projection. If \( A \) had orthonormal columns, then \( A^T (\vec{y} - A\vec{x}(t)) \) would take us exactly to where we need to be. It would update the parameters perfectly. But \( A \) doesn’t have orthonormal columns, so we just move our estimate a little bit in that direction where \( \alpha \) controls how much we move. You can see that if we ever reach \( \vec{x}(t) = \hat{\vec{x}} \), the system reaches equilibrium — it stops moving. At that point, the residual is perfectly orthogonal to the columns of \( A \). In a way, this is a dynamical system that was chosen based on where its equilibrium point is.

By the way, it is no coincidence that the gradient of \( ||A\vec{x} - \vec{y}||^2 \) with respect to \( \vec{x} \) is

\[ \nabla ||A\vec{x} - \vec{y}||^2 = 2A^T (A\vec{x} - \vec{y}) \]

This can be derived directly by using vector derivatives (outside of 16B’s class scope) or by carefully using partial derivatives as we will do for linearization, later in 16B. So, the heuristic update (46) is actually just taking a step along the negative gradient direction. This insight is what lets us adapt this heuristic for a kind of “linearization” applied to other optimization problems that aren’t least-squares. (But all this is currently out-of-scope for 16B at this point, and is something discussed further in 127 and 189. Here, at this point in 16B, (46) is just some discrete-time linear system that we have been given.)

To show that \( \vec{x}(t) \rightarrow \hat{\vec{x}} \), we define a new state variable \( \Delta \vec{x}(t) = \vec{x}(t) - \hat{\vec{x}} \).

Derive the discrete-time state evolution equation for \( \Delta \vec{x}(t) \), and show that it takes the form:

\[ \Delta \vec{x}(t + 1) = (I - \alpha G)\Delta \vec{x}(t) \] (47)
Solution:

\[ \Delta \vec{x}(t+1) = \vec{x}(t+1) - \vec{x} \]
\[ = \vec{x}(t) - \alpha A^T (A \vec{x}(t) - \vec{y}) - \vec{x} \]
\[ = (\vec{x}(t) - \vec{x}) - \alpha A^T (A \vec{x}(t) - \vec{y}) \]
\[ = \Delta \vec{x}(t) - \alpha A^T A \vec{x}(t) - \alpha A^T \vec{y} \]
\[ = \Delta \vec{x}(t) - \alpha A^T A (\vec{x}(t) - \vec{x}) \]
\[ = \Delta \vec{x}(t) - \alpha A^T A (\Delta \vec{x}(t)) \]
\[ = (I - \alpha A^T A) \Delta \vec{x}(t) \]

So \( G = A^T A \).

(c) We would like to make the system such that \( \Delta \vec{x}(t) \) converges to 0. As a first step, we just want to make sure that we have a stable system. To do this, we need to understand the eigenvalues of \( I - \alpha G \). **Show that the eigenvalues of matrix \( I - \alpha G \) are \( 1 - \alpha \lambda_{i(G)} \), where \( \lambda_{i(G)} \) are the eigenvalues of \( G \).**

**Solution:**

We actually know that \( G \) is symmetric and of the form \( A^T A \). This means that it has all non-negative eigenvalues and orthonormal eigenvectors. But this is not important for this part.

Here all we need to notice that if \( \lambda_{i(G)}, \vec{v} \) is an eigenvalue eigenvector pair for \( G \), then \( (I - \alpha G) \vec{v} = \vec{v} - \alpha \lambda_{i(G)} \vec{v} = (1 - \alpha \lambda_{i(G)}) \vec{v} \).

Hence, the eigenvalues of \( I - \alpha G \) are \( 1 - \alpha \lambda_{i(G)} \).

(d) To be stable, we need all these eigenvalues to have magnitudes that are smaller than 1 (since this is a discrete-time system). Since the matrix \( G \) above has a special form, all of the eigenvalues of \( G \) are non-negative and real. **For what \( \alpha \) would the eigenvalue \( 1 - \alpha \lambda_{\text{max}(G)} = 0 \) where \( \lambda_{\text{max}(G)} \) is the largest eigenvalue of \( G \). At this \( \alpha \), what would be the largest magnitude eigenvalue of \( I - \alpha G \)? Is the system stable?**

*(Hint: Think about the smallest eigenvalue of \( G \). What happens to it? Feel free to assume that this smallest eigenvalue \( \lambda_{\text{min}(G)} \) is strictly greater than 0.)*

**Solution:** Firstly, we have that \( \lambda_{i(G)} \geq 0 \). For the given condition, we want \( \alpha = \frac{1}{\lambda_{\text{max}(G)}} \).

To find the largest magnitude eigenvalue of \( (I - \alpha G) \), we need to maximize \( 1 - \alpha \lambda_{i(G)} \). We know that for the chosen \( \alpha \), the minimum value here is 0. To find the maximum value, the minus sign tells us that we must look at the minimum eigenvalue of \( G \). So the maximum eigenvalue of \( (I - \alpha G) \) is \( 1 - \alpha \lambda_{\text{min}(G)} = 1 - \frac{\lambda_{\text{min}(G)}}{\lambda_{\text{max}(G)}} \).

Since we are assuming \( \lambda_{\text{min}(G)} > 0 \), then the discrete-time system will be stable. Furthermore, all the eigenvalues will be in the range \([0, 1]\).

Only if \( \lambda_{\text{min}(G)} = 0 \) could we have a problem. That would happen if \( A^T A \) had a nullspace which also means that \( A \) would have to have a nullspace. This could happen if we had some redundant columns.

However, this seeming threat of instability is just an illusion. This is because we could just as well eliminate the redundant columns.

The relationship between the \( \lambda_{i(G)} \) and \( \lambda_{I - \alpha G} \) are visually shown on the number lines below.

(e) **Above what value of \( \alpha \) would the system \((47)\) become unstable?** This is what happens if you try to set the learning rate to be too high.
Figure 1: Original eigenvalues of the $G$ matrix. Note all eigenvalues are non-negative. The smallest and largest magnitude eigenvalues are specifically labeled.

Figure 2: The eigenvalues of $I - \alpha G$ for $\alpha = \frac{1}{\lambda_{\text{max}}(G)}$. Note that the largest $\lambda(G)$ is moved to 0, and the smallest $\lambda(G)$ is moved close to 1.

**Solution:** As we increase the $\alpha$, the eigenvalues of $(I - \alpha G)$ march to the left. They don’t change their order. So once the $\alpha > \frac{2}{\lambda_{\text{max}}}$, the set of eigenvalues will cross outside the unit circle because $1 - \alpha \lambda_{\text{max}} < -1$. It is the maximum eigenvalue of $G$ that matters here because it is the left-most eigenvalue for $I - \alpha G$.

Figure 3: Plot of $\lambda I - \alpha G$ for $\alpha = \frac{2}{\lambda_{\text{max}}}$. Note that there is an eigenvalue at $-1$, so the system is unstable.

(f) Looking back at the part before last (where you moved the largest eigenvalue of $G$ to zero), if you slightly increased the $\alpha$, would the convergence become faster or slower?

*(HINT: think about the dominant eigenvalue here. Which is the eigenvalue of $I - \alpha G$ with the largest magnitude?)*

**Solution:** It would converge faster because the dominant eigenvalue would get smaller. This is because the dominant eigenvalue (the one that makes the decay the slowest) is $0 > 1 - \alpha \lambda_{\text{min}}(G) < 1$. Increasing $\alpha$ slightly makes this closer to zero. Meanwhile, it only moves the maximum eigenvalue a little bit to the left from 0. So the system stays stable.

Figure 4: Plot of $\lambda I - \alpha G$ for $\alpha = \frac{1}{\lambda_{\text{max}}} + \delta_{\text{smart}}$. Note that all eigenvalues are still $|\lambda| < 1$, so the system remains stable. However, the one that corresponds to $\lambda_{\text{min}}$ has moved further away from 1 and so this is actually more stable.

(g) *(Optional — out of scope)* What is the $\alpha$ that would result in the system being stable, and converge fastest to $\Delta \vec{x} = 0$?

*(HINT: When would growing $\alpha$ stop helping shrink the biggest magnitude eigenvalue of $I - \alpha G$?)*

**Solution:**
This was only deemed out of scope because it involves reasoning about how to minimize the maximum here.

For a discrete system, the stability criteria is:

$$|\lambda_i| < 1$$

For $\Delta \vec{x}(t)$, the eigenvalues are $\lambda_i = 1 - \alpha \lambda_i(A^T A)$, where $\lambda_i(A^T A)$ are the eigenvalues of $A^T A$. Note that $A^T A$ is a symmetric matrix with all eigenvalues $\lambda_i(A^T A) \geq 0$:

$$-1 < 1 - \alpha \lambda_i(A^T A) < 1$$

If we meet the condition

$$-1 < 1 - \alpha \lambda_{\text{max}}(A^T A) < 1$$

where $\lambda_{\text{max}}(A^T A)$ is the largest eigenvalue of $A^T A$, then we will meet the stability criteria for all eigenvalues. With a little bit of algebraic manipulation, we can get the range of $\alpha$ that makes the system stable:

$$0 < \alpha < \frac{2}{\lambda_{\text{max}}(A^T A)}$$

If we only cared about the largest eigenvalue of $A^T A$, then for this system to converge the fastest, we would want $1 - \alpha \lambda_{\text{max}}(A^T A) = 0$, which means we would choose:

$$\alpha = \frac{1}{\lambda_{\text{max}}(A^T A)}$$

However, we also need to think about the other eigenvalues of $A^T A$. With $\alpha = \frac{1}{\lambda_{\text{max}}(A^T A)}$, there will be other eigenvalues of the error system larger than 0 if not all eigenvalues of $A^T A$ are the same value. The largest eigenvalue of the error corresponds to the minimum eigenvalue of $A^T A$, which we’ll call $\lambda_{\text{min}}(A^T A)$. As we increase $\alpha$ past $\frac{1}{\lambda_{\text{max}}(A^T A)}$, the error eigenvalue corresponding to $\lambda_{\text{min}}(A^T A)$ will decrease in magnitude, but the eigenvalue corresponding to $\lambda_{\text{max}}(A^T A)$ will increase (since the eigenvalue starts going negative). To find the optimal $\alpha$, we set the minimum and maximum eigenvalues’ magnitudes equal to each other:

$$1 - \alpha \lambda_{\text{min}}(A^T A) = \alpha \lambda_{\text{max}}(A^T A) - 1$$

Note that the eigenvalue corresponding to $\lambda_{\text{max}}(A^T A)$ flipped signs since $\alpha$ was large enough to make the eigenvalue negative. This gives us an optimal step size of:

$$\alpha = \frac{2}{\lambda_{\text{min}}(A^T A) + \lambda_{\text{max}}(A^T A)}$$

It turns out that this is related to a concept in numerical linear algebra called the condition number for a matrix. When the matrix $A$ is well conditioned, then we can get faster convergence to the solution. Basically, this requires the ratio of the maximum eigenvalue of $A^T A$ to the minimum eigenvalue of $A^T A$ to be small. When the eigenvalues are closer to each other, the whole cluster can be made to be around zero in the diagrams we have plotted for you.

(h) **Play with the given jupyter notebook and comment on what you observe.** Consider how the different step sizes relate to recurrence relations, and how a sufficiently small step size can approach a continuous solution.

**Solution:** The most important thing to notice is that plotting on the semilog scale clearly shows the exponential speed of convergence. Having a learning rate that is too high indeed makes the estimates go unstable. While the optimal learning rate really does converge faster. Pretty much any observations count for full credit. You will learn much more about these things in 127.
Figure 5: Plot of $\lambda_{I-\alpha G}$ for $\alpha = \frac{2}{\lambda_{\text{max}} + \lambda_{\text{min}}}$. Note that the largest and smallest eigenvalues are centered around 0. They have the same magnitude, so the convergence will be the fastest. Any bigger $\alpha$ would make the one corresponding to $\lambda_{\text{max}}$ closer to $-1$, thereby slowing down convergence. Meanwhile, any smaller $\alpha$ would make the one corresponding to $\lambda_{\text{min}}$ closer to $+1$ which would also slow down convergence. This particular $\alpha$ is just right.

6. Write Your Own Question And Provide a Thorough Solution.

Writing your own problems is a very important way to really learn material. The famous “Bloom’s Taxonomy” that lists the levels of learning is: Remember, Understand, Apply, Analyze, Evaluate, and Create. Using what you know to create is the top level. We rarely ask you any homework questions about the lowest level of straight-up remembering, expecting you to be able to do that yourself (e.g. making flashcards). But we don’t want the same to be true about the highest level. As a practical matter, having some practice at trying to create problems helps you study for exams much better than simply counting on solving existing practice problems. This is because thinking about how to create an interesting problem forces you to really look at the material from the perspective of those who are going to create the exams. Besides, this is fun. If you want to make a boring problem, go ahead. That is your prerogative. But it is more fun to really engage with the material, discover something interesting, and then come up with a problem that walks others down a journey that lets them share your discovery. You don’t have to achieve this every week. But unless you try every week, it probably won’t ever happen.

7. Homework Process and Study Group

Citing sources and collaborators are an important part of life, including being a student! We also want to understand what resources you find helpful and how much time homework is taking, so we can change things in the future if possible.

(a) What sources (if any) did you use as you worked through the homework?

(b) Who did you work on this homework with? List names and student ID’s. (In case of homework party, you can also just describe the group.)

(c) How did you work on this homework? (For example, I first worked by myself for 2 hours, but got stuck on problem 3, so I went to office hours. Then I went to homework party for a few hours, where I finished the homework.)

(d) Roughly how many total hours did you work on this homework?

Contributors:

- Regina Eckert.
- Anant Sahai.
- Nathan Lambert.
- Sidney Buchbinder.
- Varun Mishra.
• Ming Jin.
• Kyle Tanghe.
• Miki Lustig.
• Aditya Arun.