Staying up to date with lectures is an important part of the learning process in this course. Here are links to the notes that you need to read for this week: Note 11 Note 12 Note 14

(a) Why is it sufficient to check that the matrix $\mathcal{C} = \begin{bmatrix} B & AB & \cdots & A^{n-1}B \end{bmatrix}$ is full rank (i.e., why is it sufficient to check whether the system is controllable in $n$ timesteps)?

**Solution:** The range of the columns $\vec{b}, A\vec{b}, \cdots, A^{k}\vec{b}$ will no longer grow after it has stopped growing, and the largest space we can span with vectors in $\mathbb{R}^n$ is $\mathbb{R}^n$. By the point we have $n$ columns, we will either have stopped growing our range or have reached the point we can span $\mathbb{R}^n$.

(b) Give a brief outline for how you would compute the Schur decomposition (i.e. upper-triangularization) of some general matrix under the assumption that all eigenvalues are real.

**Solution:**
- Compute an eigenvalue and eigenvector of your matrix, $A$.
- Create an orthonormal matrix with your eigenvector as the first vector: $V$.
- Transform your matrix: $V^\top AV$, then take the $n-1 \times n-1$ sub-matrix ignoring the first row and column with the eigenvalue inside. Iterate from the beginning.
- Stitch all matrices $V$ generated along the way into one single matrix so that we can express $A = VTV^\top$ where $T$ is upper triangular.

(c) What happens when we upper-triangularize a symmetric matrix?

**Solution:** For a symmetric matrix, it turns out that the Schur decomposition is the same as a diagonalization or eigendecomposition. We just get a diagonal matrix as our upper-triangular matrix.
2. Discrete systems and their orbits

The concept of controllability exists to tell us whether or not a system can be eventually driven from any initial condition to any desired state given no disturbances, perfect knowledge of the system, and knowledge of the initial state.

For a discrete-time system with \( n \)-dimensional state \( \vec{x} \) driven by a scalar input \( u(t) \):

\[
\vec{x}(t + 1) = A\vec{x}(t) + \vec{b}u(t)
\]  

this can be checked by seeing whether the controllability matrix

\[
C = \begin{bmatrix}
\vec{b} & A\vec{b} & A^2\vec{b} & \cdots & A^{n-1}\vec{b}
\end{bmatrix}
\]

has a range (span of the columns) that encompasses all of \( \mathbb{R}^n \). In other words, \( \text{span}(C) = \mathbb{R}^n \). If it does span the whole space, then the system (1) is controllable. If it does not, then the system is not controllable. Everything seems to hinge on the “orbit” that the vector \( \vec{b} \) takes as it repeatedly encounters \( A \). Does this orbit confine itself to a subspace, or does it explore the whole space? (Formally, this orbit is the infinite sequence \( \vec{b}, A\vec{b}, A^2\vec{b}, \ldots \))

(a) Suppose that we choose \( \vec{b} = \alpha \vec{v}_i \) for some eigenvector \( \vec{v}_i \) of \( A \). That is, \( A\vec{v}_i = \lambda \vec{v}_i \). Show that the rank of \( C \) will be 1 (i.e., show that the subspace spanned by the orbit of \( \vec{b} \) through \( A \) will be one dimensional).

**Solution:** If \( \vec{b} = \alpha \vec{v}_i \), then the controllability matrix is,

\[
C = \begin{bmatrix}
\alpha \vec{v}_i & A\alpha \vec{v}_i & A^2\alpha \vec{v}_i & \cdots & A^{n-1}\alpha \vec{v}_i
\end{bmatrix}
\]

But since \( \vec{v}_i \) is an eigenvector of \( A \) with eigenvalue \( \lambda_i \), the controllability matrix is,

\[
C = \begin{bmatrix}
\alpha \vec{v}_i & \alpha \lambda_i \vec{v}_i & \alpha \lambda_i^2 \vec{v}_i & \cdots & \alpha \lambda_i^{n-1} \vec{v}_i
\end{bmatrix}
\]

Hence, the columns of the controllability are scalar multiples of each other and therefore the rank of \( C(\vec{b}) \) is 1.

(b) If \( \vec{b} = \vec{v}_i \) is an eigenvector as in the previous part, **would the system be controllable if \( n > 1 \)?**

**Solution:** If \( \vec{b} = \vec{v}_i \) then the rank of \( C(\vec{b}) \) is 1 according to the previous part. So if \( n > 1 \), then the columns of the controllability matrix won’t span all of \( \mathbb{R}^n \) and hence the system will not be controllable. We can only move along \( \vec{v}_i \), nowhere else in the space.

(c) Suppose the matrix \( A \) has at least two distinct eigenvalues \( \lambda_1 \neq \lambda_2 \) with corresponding eigenvectors \( \vec{v}_1 \) and \( \vec{v}_2 \). If \( \vec{b} = \alpha_1 \vec{v}_1 + \alpha_2 \vec{v}_2 \), then show that \( A^2\vec{b} = \beta_1 A\vec{b} + \beta_0 \vec{b} \) for some choice of \( \beta_1 \) and \( \beta_0 \).

**Solution:** If \( \vec{b} = \alpha_1 \vec{v}_1 + \alpha_2 \vec{v}_2 \), then

\[
A\vec{b} = A(\alpha_1 \vec{v}_1 + \alpha_2 \vec{v}_2) = \alpha_1 A\vec{v}_1 + \alpha_2 A\vec{v}_2 = \alpha_1 \lambda_1 \vec{v}_1 + \alpha_2 \lambda_2 \vec{v}_2
\]

Therefore,

\[
A^2\vec{b} = A(\alpha_1 \lambda_1 \vec{v}_1 + \alpha_2 \lambda_2 \vec{v}_2) = \alpha_1 \lambda_1 A\vec{v}_1 + \alpha_2 \lambda_2 A\vec{v}_2 = \alpha_1 \lambda_1^2 \vec{v}_1 + \alpha_2 \lambda_2^2 \vec{v}_2
\]
Now, setting $A^2\vec{b} = \beta_1 A\vec{b} + \beta_0 \vec{b}$, we get

$$\alpha_1 \lambda_1^2 \vec{v}_1 + \alpha_2 \lambda_2^2 \vec{v}_2 = \beta_1 (\alpha_1 \lambda_1 \vec{v}_1 + \alpha_2 \lambda_2 \vec{v}_2) + \beta_0 (\alpha_1 \vec{v}_1 + \alpha_2 \vec{v}_2) \quad (7)$$

Comparing the coefficients of $\vec{v}_1$ and $\vec{v}_2$, we get the following two equations,

$$\alpha_1 \lambda_1^2 = \beta_1 \alpha_1 \lambda_1 + \beta_0 \alpha_1 \quad (8)$$

$$\alpha_2 \lambda_2^2 = \beta_1 \alpha_2 \lambda_2 + \beta_0 \alpha_2 \quad (9)$$

Since these are two linear equations in two variables, $\beta_0$ and $\beta_1$, there will be a solution.

The above suffices for full credit for this part. But being more explicit:
There are two qualitatively distinct cases. First, if one of the $\alpha_j = 0$. If that happens, this is really just the previous case of having a $\vec{b}$ along a single eigenvector. In that case, we just have $\beta_1 = \lambda_i$ (for the one whose $\alpha_i \neq 0$ and $\beta_0 = 0$. After all $A^2\vec{b} = A^2 \alpha_i \vec{v}_i = \lambda_i A\vec{b}$.

The other case is the one dealt with in the next sub-part.

(d) In the previous part, if both the $\alpha_i \neq 0$, do the coefficients $\beta_i$ depend on the exact nonzero values of the $\alpha_i$?

Solution:
The set of equations whose solutions give $\beta_0$ and $\beta_1$ are

$$\alpha_1 \lambda_1^2 = \beta_1 \alpha_1 \lambda_1 + \beta_0 \alpha_1 \quad (10)$$

$$\alpha_2 \lambda_2^2 = \beta_1 \alpha_2 \lambda_2 + \beta_0 \alpha_2 \quad (11)$$

If both $\alpha_1$ and $\alpha_2$ are given to be non-zero, then we can divide the first equation by $\alpha_1$ and the second equation by $\alpha_2$. This would yield the following two equations,

$$\lambda_1^2 = \beta_1 \lambda_1 + \beta_0 \quad (12)$$

$$\lambda_2^2 = \beta_1 \lambda_2 + \beta_0 \quad (13)$$

Since these equations don’t have any of the $\alpha$ variables in there, the solutions don’t depend on the $\alpha$ values.

In particular, we have $\beta_1 (\lambda_1 - \lambda_2) = \lambda_1^2 - \lambda_2^2$ which implies

$$\beta_1 = \lambda_1 + \lambda_2 \quad (14)$$

by the factorization of $\lambda_1^2 - \lambda_2^2 = (\lambda_1 + \lambda_2)(\lambda_1 - \lambda_2)$. From this, we can see that $\beta_0 = \lambda_1^2 - \beta_1 \lambda_1 = \lambda_1 (\lambda_1 - \beta_1)$ which implies

$$\beta_0 = -\lambda_1 \lambda_2 \quad (15)$$

This is interesting. Where have we seen these before? These are just the negatives of the lower-degree coefficients of $(\lambda - \lambda_1)(\lambda - \lambda_2) = \lambda^2 - (\lambda_1 + \lambda_2) \lambda + \lambda_1 \lambda_2$.

Indeed, this is no coincidence at all.
(e) Consequently, in the previous part, if \( n > 2 \), would the system be controllable if the \( \vec{b} \) was a linear combination of only two eigenvectors?

**Solution:** No. It would not be controllable. The rank of the controllability matrix (i.e., the number of linearly independent columns) would just be 2.

To be controllable, it is important for the \( \vec{b} \) to include nontrivial representation from all the eigenvalues’ corresponding eigenvectors somehow. Otherwise, we can’t control that dimension.

It turns out that is actually a sufficient condition for controllability when all the eigenvalues are distinct and we have a full complement of eigenvectors. We will see this more clearly when we talk about polynomials later in 16B. Our understanding of polynomials will tell us instantly that the resulting controllability matrix must be full rank.

(f) Now consider a general square matrix \( A \) (not necessarily with distinct eigenvectors, etc.) with a specific \( \vec{b} \) such that the system defined by the pair \((A, \vec{b})\) is controllable. This means that the controllability matrix \([\vec{b}, A\vec{b}, \ldots, A^{n-1}\vec{b}]\) is invertible and hence gives us a basis for \( n \)-dimensional space. Consequently, for this specific vector \( \vec{b} \), we know \( A^n\vec{b} \) can be written in this basis. This means there exists \( \{\beta_i\}_{i=0}^{n-1} \) so that:

\[
A^n\vec{b} = \sum_{i=0}^{n-1} \beta_i A^i\vec{b} \quad (16)
\]

**Show that for all** \( j > 0 \), the vector \( \vec{w}_j = A^j\vec{b} \), also satisfies \( A^n\vec{w}_j = \sum_{i=0}^{n-1} \beta_i A^i\vec{w}_j \). (HINT: Multiply both sides of an equation by something and substitute.)

**Solution:** We are given that,

\[
A^n\vec{b} = \sum_{i=0}^{n-1} \beta_i A^i\vec{b} \quad (17)
\]

Multiplying both sides by \( A^j \), we get,

\[
A^j A^n\vec{b} = \sum_{i=0}^{n-1} A^j \beta_i A^i\vec{b} \quad (18)
\]

Rearranging, we get,

\[
A^n A^j\vec{b} = \sum_{i=0}^{n-1} \beta_i A^i A^j\vec{b} \quad (19)
\]

Now, simply substituting \( \vec{w}_j = A^j\vec{b} \), we get

\[
A^n\vec{w}_j = \sum_{i=0}^{n-1} \beta_i A^i\vec{w}_j. \quad (20)
\]

(g) Suppose your general square matrix \( A \) above has a specific \( \vec{b} \) such that the system defined by the pair
is controllable. This means that it satisfies (16) above. Use the previous part to show that in fact, \( A^n - \beta_n^{-1}A^{n-1} - \beta_{n-2}A^{n-2} - \ldots - \beta_1 A - \beta_0 I \) is the matrix of all zeros.

(HINT: The previous part might provide you with a very convenient basis to use. Then remember that the signature of the zero matrix is that no matter what it multiplies, it returns zero. It a matrix takes every basis vector to zero, then it must be the zero matrix.)

Solution: Since the pair \((A, \vec{b})\) is controllable, this means that any arbitrary vector \( \vec{d} \) can be represented as a linear combination of \( \{A^0\vec{b}, A^1\vec{b}, \ldots, A^{n-1}\vec{b}\} \), (where \( n \) is the size of \( A \)). As per the notation in the previous question this would be a linear combination of \( \{\vec{w}_0, \vec{w}_1, \ldots, \vec{w}_{n-1}\} \). Each of these \( \vec{w}_j \)s satisfy the equation

\[
A^n \vec{w}_j = \sum_{i=0}^{n-1} \beta_i A^i \vec{w}_j \tag{21}
\]

Hence, a linear combination of the \( \vec{w}_j \)’s would also satisfy this equation. Hence, every vector \( \vec{d} \) in \( R^n \) satisfies

\[
A^n \vec{d} = \sum_{i=0}^{n-1} \beta_i A^i \vec{d} \tag{22}
\]

Rearranging the terms, we get that for every vector \( \vec{d} \) in \( R^n \),

\[
(A^n - \beta_{n-1}A^{n-1} - \beta_{n-2}A^{n-2} - \ldots - \beta_1 A - \beta_0 I) \vec{d} = \vec{0} \tag{23}
\]

It is only possible that this expression is equal to the zero vector for any vector \( \vec{d} \) if \( A^n - \beta_{n-1}A^{n-1} - \beta_{n-2}A^{n-2} - \ldots - \beta_1 A - \beta_0 I \) is the matrix of all zeros, since we can just use the the \( n \) elementary basis vectors for \( \vec{d} \) to extract each column of the matrix and get zeros every time.

It turns out that this specific polynomial \( (\lambda^n - \beta_{n-1}\lambda^{n-1} - \ldots - \beta_1 \lambda - \beta_0) \) must in fact be the characteristic polynomial \( \det(\lambda I - A) \) of the matrix \( A \). So, this argument above shows that these kinds of matrices must satisfy their own characteristic polynomials. It is also easy to see this for diagonalizable matrices \( A \) (i.e., those with an eigenbasis), but this example shows that it holds more generally for controllable matrices.

This argument can actually be used to extend to all square matrices, even those that don’t have a full complement of linearly independent eigenvectors and aren’t controllable with a single scalar input. When extended all the way, it is called the Cayley-Hamilton theorem. It shows that a square matrix satisfies its own characteristic polynomial — no matter what. This is an important theorem that is proved in the upper-division linear-algebra course Math 110.

It turns out that the previous parts actually help us conceptually unlock a proof that every real or complex matrix \( A \) must have at least one (possibly complex) eigenvalue/eigenvector pair. In the next couple of parts, you will do this.

(h) The first thing we’re going to do is to eliminate the assumption that we have a \( \vec{b} \) so that the pair \((A, \vec{b})\) is controllable.
Show that for any nonzero vector $\vec{b}$ and any $n$-dimensional matrix $A$, it must be that there exists an $\ell > 0$ and $\{\gamma_i\}_{i=0}^{\ell-1}$ so that:

$$A^\ell \vec{b} = \sum_{i=0}^{\ell-1} \gamma_i A^i \vec{b}. \quad (24)$$

(HINT: If $\vec{b}$ were such that $(A, \vec{b})$ were controllable, then you’d be done since you could invoke eq. (16). But what do you know happens when $(A, \vec{b})$ is not controllable?)

Solution: We know from the previous part that if $\vec{b}$ were such that $(A, \vec{b})$ were controllable, then $A^n \vec{b} = \sum_{i=0}^{n-1} \gamma_i A^i \vec{b}$ (in other words $\ell = n$). If instead $\vec{b}$ is such that $(A, \vec{b})$ is not controllable, then the columns of the controllability matrix must be linearly dependent. Then, starting at $j = 1$, we can keep incrementing $i$ until $A^i \vec{b}$ is in the span of $\{\vec{b}, A\vec{b}, \ldots, A^{i-1} \vec{b}\}$. At this point, the span will not continue to grow, and we can set $\ell = i$. Note that at worst we will reach $\ell = n - 1$ for the case of $(A, \vec{b})$ not being controllable, since if we reach $n$ it implies that the first $n$ columns are linearly independent (and thus the pair is in fact controllable).

(i) Notice that we can define the polynomial $p(x) = x^\ell - \sum_{i=0}^{\ell-1} \gamma_i x^i$ using the $\{\gamma_i\}$ from eq. (24) and then we can equivalently write eq. (24) as:

$$p(A) \vec{b} = \vec{0}. \quad (25)$$

Here, we interpret $p(A)$ treating the definition of $p(x)$ like a macro and just substituting in the matrix $A$ wherever we had the variable $x$ in the definition. Here, we treat $A^0$ or the constant term as just the identity matrix.

The fundamental theorem of algebra further tells us that since $p(x)$ is a degree $\ell$ polynomial with complex coefficients, it can be completely factored. (This is true because the fundamental theorem of algebra guarantees us that any nonconstant polynomial over the complex numbers has at least one root — so we can factor out a term that corresponds to that root and get a polynomial of degree one lower.)

Solution: Many of you might not have seen any proof or even any argument as to why the fundamental theorem of algebra is even plausibly true. Indeed, fully rigorous arguments establishing this theorem are typically only given in Math 185 (complex analysis), and even rare in courses like Math 104 (real analysis), and even rarely in courses like Math 113 (abstract algebra). A 16B-style argument will unlock for us when we get to the final section of the course and introduce the ideas behind the Discrete Fourier Transform. It turns out that the elementary ideas behind one of the proofs of the Fundamental Theorem of Algebra involve the kinds of thinking that we engage with in 16B (especially the approximation thinking that you might associate with Bode plots), with one further element which is intuitively clear (that goes by the name of topology), but would require more mathematical sophistication than 16B to fully formalize. This allows us to take the natural proof that we all know for why odd-degree polynomials must have at least one real root (because you have to cross the x axis sometime to get from one side to the other) to the general case that includes even-degree polynomials.

This means that there exists a list $[\lambda_1, \lambda_2, \ldots, \lambda_\ell]$ (not necessarily all distinct) of possibly complex numbers so that

$$p(x) = \prod_{i=1}^{\ell} (x - \lambda_i). \quad (26)$$
This means that we also write

$$\vec{0} = p(A)\vec{b} = \left( \prod_{i=1}^{\ell} (A - \lambda_i I) \right) \vec{b}.$$  \hspace{1cm} (27)

**Show that this factorization implies that the matrix** \(A\) **must have an eigenvector** \(\vec{v}\) **so that** \(A\vec{v} = \lambda_1 \vec{v}\) **for at least one of the** \(\lambda_i\).

**(HINT: Write eq. (27) in the expanded form:**

\[
(A - \lambda_1 I)(A - \lambda_2 I) \cdots (A - \lambda_{\ell} I)\vec{b} = \vec{0}
\]

**(28)**

**and then reason about what can happen as you do that multiplication starting from the right and moving left. Somehow you have to end up at the zero vector. How can that happen? Can you turn this into a procedure that will return the eigenvalue, eigenvector pair?)**

**Solution:**

Using the hint, we can rewrite eq. (27) in the given expanded form:

\[
(A - \lambda_1 I)(A - \lambda_2 I) \cdots (A - \lambda_{\ell} I)\vec{b} = \vec{0}
\]

**(29)**

Then, since this expression on the left hand side must be equal to the zero vector, we know that either (i) \((A - \lambda_{\ell} I)\vec{b} = \vec{0}\), meaning that \(\vec{b}\) is in the nullspace of \((A - \lambda_{\ell} I)\), or (ii) \((A - \lambda_1 I)(A - \lambda_2 I) \cdots (A - \lambda_{\ell-1} I)\vec{b}_1 = \vec{0}\) for \(\vec{b}_1 = (A - \lambda_{\ell} I)\vec{b}\).

In case (i), \((A - \lambda_{\ell} I)\vec{b} = \vec{0}\) implies that \(A\vec{b} = \lambda_{\ell}\vec{b}\); thus, \(\vec{b}\) is an eigenvector of \(A\) with eigenvalue \(\lambda_{\ell}\).

In case (ii), we can recurse using the same two cases with the new vector \(\vec{b}_1\). If \((A - \lambda_{\ell-1} I)\vec{b}_1 = \vec{0}\), then \(\vec{b}_2\) is an eigenvector of \(A\) with eigenvalue \(\lambda_{\ell-1}\); otherwise, we continue to move to the right in the polynomial until we find a \(\vec{b}_k = (A - \lambda_{\ell-k} I) \cdots (A - \lambda_{\ell-1} I)(A - \lambda_{\ell} I)\vec{b}\) such that \((A - \lambda_{\ell-k} - 1 I)\vec{b}_k = \vec{0}\).

Essentially, at some point, we have to get a zero vector.

At this point, we know that \(\vec{b}_k\) is an eigenvector of \(A\) with eigenvalue \(\lambda_{\ell-k-1}\). Since this process must terminate (i.e., at worst, we will continue until \(k = \ell - 2\) and find eigenvector \(\vec{b}_{\ell-2}\) with eigenvalue \(\lambda_1\)), we have shown that every matrix \(A\) must have at least one eigenvalue, eigenvector pair.

Notice that this proof never needed determinants.
3. 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 $\vec{q}_i^\top \vec{q}_j = 0$.
- The $\vec{q}_i$ have unit norm whenever they are nonzero. That is, if $\vec{q}_i \neq \vec{0}$, then $\vec{q}_i^\top \vec{q}_i = \|\vec{q}_i\|^2 = 1$.

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
  2:    $\vec{r}_i = \vec{a}_i - \sum_{j < i} \vec{q}_j \left(\vec{q}_j^\top \vec{a}_i\right)$
  3:      if $\vec{r}_i = \vec{0}$ then
  4:        $\vec{q}_i = \vec{0}$
  5:      else
  6:        $\vec{q}_i = \frac{\vec{r}_i}{\|\vec{r}_i\|}$
  7:      end if
  8:  end for

(a) From the If/Then/Else statement in the algorithm above, 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 $\|\vec{q}_i\| = \vec{q}_i^\top \vec{q}_i$. Expanding this, we see

$$\|\vec{q}_i\|^2 = \frac{\vec{r}_i^\top \vec{q}_i}{\|\vec{r}_i\|^2} = \left(\frac{\vec{r}_i}{\|\vec{r}_i\|}\right)^\top \left(\frac{\vec{r}_i}{\|\vec{r}_i\|}\right).$$

(30)

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

$$\|\vec{q}_i\|^2 = \frac{1}{\|\vec{r}_i\|^2} \frac{\vec{r}_i^\top \vec{r}_i}{\|\vec{r}_i\|^2} = \frac{1}{\|\vec{r}_i\|^2} \|\vec{r}_i\|^2 = 1.$$  (31)

(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 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 $\vec{\alpha} = [\alpha_1 \cdots \alpha_{k-1}]$ so that $\vec{y} = \sum_{j=1}^{k-1} \alpha_j \vec{a}_j$, we know there exists $\vec{\beta} = [\beta_1 \cdots \beta_{k-1}]$ such that $\vec{y} = \sum_{j=1}^{k-1} \beta_j \vec{q}_j$. And vice-versa: from $\vec{\beta}$ 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:** Proof Outline: we first show that we can express $\vec{\beta}$ in terms of $\vec{\alpha}$, then vice versa. To do so, we express $\vec{a}_k$ or $\vec{q}_k$ using the algorithm above and then group $\vec{q}_j$ terms.

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

$$\sum_{j=1}^{k-1} \beta_j \vec{q}_j = \sum_{j=1}^{k-1} \alpha_j \vec{a}_j$$  \hspace{1cm} (32)

Then for $\ell = k$, consider a generic set of $\alpha$’s, $\alpha_1, \ldots, \alpha_{k-1}, \alpha_k$.

$$\sum_{j=1}^{k} \alpha_j \vec{a}_j = \alpha_k \vec{a}_k + \sum_{j=1}^{k-1} \alpha_j \vec{a}_j$$  \hspace{1cm} (33)

$$= \alpha_k \left( \|\vec{r}_k\| \vec{q}_k + \sum_{i<k} \vec{q}_i \left( \vec{q}_i^T \vec{a}_i \right) \right) + \sum_{j=1}^{k-1} \beta_j \vec{q}_j$$  \hspace{1cm} (34)

$$= \alpha_k \|\vec{r}_k\| \vec{q}_k + \sum_{j=1}^{k-1} \beta_j + \alpha_k \vec{q}_k \left( \vec{q}_k^T \vec{a}_k \right) \vec{q}_j$$  \hspace{1cm} (35)

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

$$\vec{a}_k = \vec{r}_k + \sum_{i<k} \vec{q}_i \left( \vec{q}_i^T \vec{a}_k \right)$$  \hspace{1cm} (36)

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

$$\vec{y} = \sum_{j=1}^{k} \beta_j \vec{q}_j.$$  \hspace{1cm} (37)

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

\[
\sum_{j=1}^{k-1} \beta_j \vec{q}_j = \sum_{j=1}^{k-1} \alpha_j \vec{a}_j \tag{38}
\]

Then for \( \ell = k \), consider a generic set of \( \beta \)'s, \( \beta_1, \ldots, \beta_{k-1} \).

\[
\vec{y} = \sum_{j=1}^{k} \beta_j \vec{q}_j = \beta_k \vec{q}_k + \sum_{j=1}^{k-1} \beta_j \vec{q}_j
\]

\[
= \beta_k \frac{\vec{r}_k}{\|\vec{r}_k\|} + \sum_{j=1}^{k-1} \beta_j \vec{q}_j
\]

\[
= \frac{\beta_k}{\|\vec{r}_k\|} \left( \vec{a}_k - \sum_{j<k} \vec{q}_j (\vec{q}_j^\top \vec{a}_k) \right) + \sum_{j=1}^{k-1} \beta_j \vec{q}_j
\]

\[
= \frac{\beta_k}{\|\vec{r}_k\|} \vec{a}_k + \sum_{j=1}^{k-1} \vec{q}_j \left( \beta_j - \frac{\beta_k}{\|\vec{r}_k\|} (\vec{q}_j^\top \vec{a}_k) \right). \tag{41}
\]

Notice that the second sum is only up through \( k - 1 \) and involves a linear combination of the \( \vec{q}_j \). 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.

\[
\vec{y} = \frac{\beta_k}{\|\vec{r}_k\|} \vec{a}_k + \sum_{j=1}^{k-1} \alpha_j \vec{a}_j \tag{43}
\]

Here, \( \alpha_k \) can just be defined to be \( \frac{\beta_k}{\|\vec{r}_k\|} \) and we have proved the result for \( \ell = k \). Thus, we have the new form of

\[
\vec{y} = \sum_{j=1}^{k} \alpha_j \vec{a}_j. \tag{44}
\]

(c) To establish orthogonality, we also need to do another little proof by induction over \( \ell \). The statement we want to prove is that for all \( j < \ell \), it must be that \( \vec{q}_j^\top \vec{q}_\ell = 0 \). The base case here of \( \ell = 1 \) is 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 \( \vec{q}_j^\top \vec{q}_i = 0 \) (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 \( \vec{q}_j^\top \vec{q}_i = 0 \).

(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 \( \bar{q}_k = \bar{0} \) is also easily true. So, focus on the case \( \bar{q}_k \neq \bar{0} \) and just expand what you know about \( \bar{q}_k \). 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 \( j < i \), \( \bar{q}_j^\top \bar{q}_j = 0 \). Remember that \( \bar{q}_j^\top \bar{q}_j = 1 \). All that remains is to deal with the case of \( \bar{q}_k \) itself. We need to verify that it is orthogonal to all the previous \( \bar{q}_j \). If \( ||\bar{r}_k|| = 0 \), then \( \bar{q}_k = \bar{0} \) and so it definitely holds since the zero vector is orthogonal to every vector. So now we can assume \( ||\bar{r}_k|| > 0 \) and check for \( i = k \).

For any \( j < k \),

\[
\bar{q}_j^\top \bar{q}_k = \bar{q}_j^\top \frac{1}{||\bar{r}_k||} \left( \bar{a}_k - \sum_{n<k} \bar{q}_n (\bar{q}_n^\top \bar{a}_k) \right) \\
= \frac{1}{||\bar{r}_k||} \left( \bar{q}_j^\top \bar{a}_k - \sum_{n<k} \bar{q}_n (\bar{q}_n^\top \bar{a}_k) \right) \\
= \frac{1}{||\bar{r}_k||} \left( \bar{q}_j^\top \bar{a}_k - \sum_{n<k} \bar{q}_j^\top \bar{q}_n (\bar{q}_n^\top \bar{a}_k) \right) \\
= \frac{1}{||\bar{r}_k||} (\bar{q}_j^\top \bar{a}_k - \bar{q}_j^\top \bar{a}_k) = 0
\]

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 \( \bar{q}_j^\top \bar{q}_j = 1 \) and \( \bar{q}_j^\top \bar{a}_k \) 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 \( A = [\bar{a}_1, \ldots, \bar{a}_n] \) and \( Q = [\bar{q}_1, \ldots, \bar{q}_n] \). If \( A \) and \( Q \) have the same column span then it must be the case that \( A = QU \) where \( U = [\bar{u}_1, \ldots, \bar{u}_n] \) is a square matrix. After all, this \( U \) tells us how we can find the \( \bar{\beta} \) that correspond to a particular \( \bar{\alpha} \) — namely \( \bar{\beta} = U \bar{\alpha} \).

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

*(HINT: Matrix multiplication tells you that \( \bar{a}_i = \sum_{j=1}^n \bar{u}_{i,j} \bar{q}_j \). What does the algorithm tell you about this relationship? Can you figure out what \( \bar{u}_{i,j} \) 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 lecture 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 \( \bar{q}_1 = \frac{1}{||\bar{a}_1||} \bar{a}_1 \), which in turn implies \( \bar{a}_1 = \bar{q}_1 ||\bar{a}_1|| = \bar{q}_1 \left( \bar{q}_1^\top \bar{a}_1 \right) \). From there, \( \bar{q}_2 = \frac{1}{||\bar{a}_2||} \bar{a}_2 \) and so it definitely holds since the zero vector is orthogonal to every vector. So now we can assume \( ||\bar{r}_k|| > 0 \) and check for \( i = k \).

For any \( j < k \),

\[
\bar{q}_j^\top \bar{q}_k = \bar{q}_j^\top \frac{1}{||\bar{r}_k||} \left( \bar{a}_k - \sum_{n<k} \bar{q}_n (\bar{q}_n^\top \bar{a}_k) \right) \\
= \frac{1}{||\bar{r}_k||} \left( \bar{q}_j^\top \bar{a}_k - \sum_{n<k} \bar{q}_n (\bar{q}_n^\top \bar{a}_k) \right) \\
= \frac{1}{||\bar{r}_k||} \left( \bar{q}_j^\top \bar{a}_k - \sum_{n<k} \bar{q}_j^\top \bar{q}_n (\bar{q}_n^\top \bar{a}_k) \right) \\
= \frac{1}{||\bar{r}_k||} (\bar{q}_j^\top \bar{a}_k - \bar{q}_j^\top \bar{a}_k) = 0
\]

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 \( \bar{q}_j^\top \bar{q}_j = 1 \) and \( \bar{q}_j^\top \bar{a}_k \) survives. So it is proven.
Reversing this equation, we see that again
\[ \vec{a}_2 = \vec{q}_2 \left( \vec{q}_2^\top \vec{a}_2 \right) + \vec{q}_1 \left( \vec{q}_1^\top \vec{a}_2 \right). \] (49)

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 \( \vec{a}_2 \).

Then, for \( \vec{a}_3 \) we get the same pattern again and can express them all using matrix multiplications:

\[
\begin{bmatrix} \vec{a}_1 & \vec{a}_2 & \vec{a}_3 \end{bmatrix} = \begin{bmatrix} \vec{q}_1 & \vec{q}_2 & \vec{q}_3 \end{bmatrix} \begin{bmatrix} \vec{q}_1^\top \vec{a}_1 & \vec{q}_1^\top \vec{a}_2 & \vec{q}_1^\top \vec{a}_3 \\ \vec{q}_2^\top \vec{a}_2 & \vec{q}_2^\top \vec{a}_2 & \vec{q}_2^\top \vec{a}_3 \\ 0 & 0 & \vec{q}_3^\top \vec{a}_3 \end{bmatrix}
\] (51)

This pattern allows us to guess that \( \vec{u}_{ij} = \vec{q}_j^\top \vec{a}_i \).

And furthermore, we know why the terms below the diagonal are all zero — they are zero because we don’t need those \( \vec{q}_j \) to express the relevant \( \vec{a}_i \).

Having a clear understanding makes doing the proof much easier. Let us consider the \( i \)th column of \( A \), \( \vec{a}_i \). We want to understand the weights \( \vec{u}_i \) required to satisfy

\[ \vec{a}_i = \sum_{j=1}^n \vec{u}_{ij} \vec{q}_j. \] (52)

The relevant term in the algorithm is where \( \vec{r}_i \) is being computed. We know \( \vec{r}_i = \vec{a}_i - \sum_{j<i} \vec{q}_j \left( \vec{q}_j^\top \vec{a}_i \right) \) and hence \( \vec{a}_i = \vec{r}_i + \sum_{j<i} \vec{q}_j \left( \vec{q}_j^\top \vec{a}_i \right) \). If \( \vec{r}_i = \vec{0} \), we are already done since we have expressed \( \vec{a}_i \) in terms of \( \vec{q}_j \) with \( j < i \). Otherwise, we know that \( \vec{q}_i = \vec{r}_i \left( \vec{r}_i \right) \) and hence \( \vec{r}_i = \| \vec{r}_i \| \vec{q}_i \). So \( \vec{a}_i = \| \vec{r}_i \| \vec{q}_i + \sum_{j<i} \vec{q}_j \left( \vec{q}_j^\top \vec{a}_i \right) \) and in a sense, we are already done since we have expressed \( \vec{a}_i \) in terms of \( \vec{q}_j \) with \( j \leq i \).

However, it is nice to complete the story and notice that
\[ \begin{aligned}
\vec{q}_i^\top \vec{a}_i &= \| \vec{r}_i \| \vec{q}_i^\top \vec{q}_i + \sum_{j<i} \vec{q}_j^\top \vec{q}_j \left( \vec{q}_j^\top \vec{a}_i \right) = \| \vec{r}_i \| \\
&\quad \text{since all the cross terms cancel by the orthogonality proved in the previous part. So indeed} \vec{a}_i = \sum_{j=1}^i \vec{q}_j \left( \vec{q}_j^\top \vec{a}_i \right). \\
\end{aligned} \]

So the pattern we conjectured is actually proved.
4. Upper Triangularization

In this problem, you need to upper-triangularize the matrix

\[ A = \begin{bmatrix} 3 & -1 & 2 \\ 3 & -1 & 6 \\ -2 & 2 & -2 \end{bmatrix} \]

The eigenvalues of this matrix \( A \) are \( \lambda_1 = \lambda_2 = 2 \) and \( \lambda_3 = -4 \). We want to express \( A \) as

\[ A = \begin{bmatrix} \bar{x} & \bar{y} & \bar{z} \end{bmatrix} \begin{bmatrix} \lambda_1 & a & b \\ 0 & \lambda_2 & c \\ 0 & 0 & \lambda_3 \end{bmatrix} \begin{bmatrix} \bar{x}^\top \\ \bar{y}^\top \\ \bar{z}^\top \end{bmatrix} \]

where the \( \bar{x}, \bar{y}, \bar{z} \) are orthonormal. Your goal in this problem is to compute \( \bar{x}, \bar{y}, \bar{z} \) so that they satisfy the above relationship for some constants \( a, b, c \).

Here are some potentially useful facts that we have gathered to save you some computations, you’ll have to grind out the rest yourself.

\[
\begin{bmatrix} 3 & -1 & 2 \\ 3 & -1 & 6 \\ -2 & 2 & -2 \end{bmatrix} \begin{bmatrix} \sqrt{2} \\ \sqrt{2} \\ 0 \end{bmatrix} = \begin{bmatrix} \sqrt{2} \\ \sqrt{2} \\ 0 \end{bmatrix}.
\]

We also know that

\[
\begin{bmatrix} \sqrt{2} \\ \sqrt{2} \\ 0 \end{bmatrix}, \begin{bmatrix} \sqrt{2} \\ -\sqrt{2} \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}
\]

is an orthonormal basis, and

\[
\begin{bmatrix} \sqrt{2} & -\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 3 & -1 & 2 \\ 3 & -1 & 6 \\ -2 & 2 & -2 \end{bmatrix} \begin{bmatrix} \sqrt{2} \\ -\sqrt{2} \\ 0 \end{bmatrix} = \begin{bmatrix} 0 & -2\sqrt{2} & -2 \sqrt{2} \\ -2 \sqrt{2} & -2 \end{bmatrix}.
\]

We also know that

\[
\begin{bmatrix} 0 & -2\sqrt{2} \\ -2 \sqrt{2} & -2 \end{bmatrix}
\]

has eigenvalues 2 and -4. The normalized eigenvector corresponding to \( \lambda = 2 \) is \( \begin{bmatrix} -\frac{\sqrt{6}}{3} \\ \sqrt{\frac{2}{3}} \end{bmatrix} \) and \( \begin{bmatrix} \sqrt{\frac{3}{6}} \\ \sqrt{\frac{2}{3}} \end{bmatrix} \) is a vector that is orthogonal to that and also has norm 1.

**Based on the above information, compute \( \bar{x}, \bar{y}, \bar{z} \).** Show your work.

You don’t have to compute the constants \( a, b, c \) in the interests of time.

**Solution:** Let us denote \( U_3 = \begin{bmatrix} \bar{x} & \bar{y} & \bar{z} \end{bmatrix} \). Since matrix \( A \) is a \( 3 \times 3 \) matrix, \( U_3 \) is a \( 3 \times 3 \) matrix as well. In addition, since vectors \( \bar{x}, \bar{y}, \bar{z} \) are orthonormal, we know that \( U_3^\top U_3 = I \). Based on this information, we
have

\[
A = \begin{bmatrix} \bar{x} & \bar{y} & \bar{z} \end{bmatrix} \begin{bmatrix} \lambda_1 & a & b \\ 0 & \lambda_2 & c \\ 0 & 0 & \lambda_3 \end{bmatrix} \begin{bmatrix} \bar{x}^T \\ \bar{y}^T \\ \bar{z}^T \end{bmatrix} = U_3 \begin{bmatrix} \lambda_1 & a & b \\ 0 & \lambda_2 & c \\ 0 & 0 & \lambda_3 \end{bmatrix} U_3^T
\] (53)

Multiplying \( U_3 \) on both side, we get:

\[
AU_3 = U_3 \begin{bmatrix} \lambda_1 & a & b \\ 0 & \lambda_2 & c \\ 0 & 0 & \lambda_3 \end{bmatrix} U_3^T U_3 = U_3 \begin{bmatrix} \lambda_1 & a & b \\ 0 & \lambda_2 & c \\ 0 & 0 & \lambda_3 \end{bmatrix}
\] (54)

Focusing on the first column vector of \( AU_3 \), we have:

\[
A\bar{x} = \lambda_1 \bar{x}
\] (55)

From the given information from the problem:

\[
A \begin{bmatrix} \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} \\ 0 \end{bmatrix} = \lambda_1 \begin{bmatrix} \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} \\ 0 \end{bmatrix} = 2 \begin{bmatrix} \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} \\ 0 \end{bmatrix} = \begin{bmatrix} \sqrt{2} \\ \sqrt{2} \\ 0 \end{bmatrix}
\] (56)

We can conclude that

\[
\bar{x} = \begin{bmatrix} \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} \\ 0 \end{bmatrix}
\] (57)

Next, let us define \( T \) as

\[
T \triangleq \begin{bmatrix} \lambda_1 & a & b \\ 0 & \lambda_2 & c \\ 0 & 0 & \lambda_3 \end{bmatrix}
\] (58)

Since \( T \) is a \( 3 \times 3 \) upper-triangular matrix, we can find \( T \) by doing the following manipulation:

\[
A = U_3 T U_3^T
\] (59)

\[
AU_3 = U_3 T U_3^T U_3 = U_3 T
\] (60)

\[
U_3^T AU_3 = U_3^T U_3 T = T
\] (61)

Assuming the matrix \( U_3 \) has the following entries:

\[
U_3 = \begin{bmatrix} \bar{x} & R_2 U_2 \end{bmatrix}
\] (62)

Each column vector in \( R_2 \) and \( U_2 \) is orthonormal, and hence each column vector in \( U_3 \) is also orthonormal.
Then $T$ can be written as:

$$U_3^T A U_3 = T = \begin{bmatrix} \lambda_1 & \vec{y}^T \\ 0 & U_2^T R_2^T A R_2 U_2 \end{bmatrix}$$

(63)

where $\vec{y}^T = \begin{bmatrix} a & b \end{bmatrix}$ and $U_2^T Q U_2$ is a $2 \times 2$ upper-triangular matrix. Again, using the information given in the problem, let us denote $R_2$ as

$$R_2 = \begin{bmatrix} \frac{\sqrt{2}}{2} & 0 \\ -\frac{\sqrt{2}}{2} & 0 \\ 0 & 1 \end{bmatrix}$$

(64)

Note that each column vectors in $R_2$ is orthonormal basis. Since we used the first given orthonormal basis to find $\vec{x}$, we use the other two given orthonormal basis to form a matrix $R_2$. Using this information, we can compute $Q$ as follows:

$$Q = R_2^T A R_2 = \begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 3 & -1 & 2 \\ -1 & 3 & 6 \\ 2 & 6 & -2 \end{bmatrix} \begin{bmatrix} \frac{\sqrt{2}}{2} & 0 \\ -\frac{\sqrt{2}}{2} & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -2\sqrt{2} \\ -2\sqrt{2} & -2 \end{bmatrix}.$$  

(65)

$Q$ has following eigenvalues and unit-length eigenvectors:

$$\lambda_1 = 2, \quad \vec{u}_{\lambda_1} = \begin{bmatrix} -\frac{\sqrt{6}}{3} \\ \frac{\sqrt{3}}{3} \end{bmatrix}, \quad \|\vec{u}_{\lambda_1}\|_2 = 1$$

(66)

$$\lambda_2 = -4, \quad \vec{u}_{\lambda_2} = \begin{bmatrix} \frac{\sqrt{3}}{3} \\ \frac{\sqrt{6}}{3} \end{bmatrix}, \quad \|\vec{u}_{\lambda_2}\|_2 = 1$$

(67)

$$\vec{u}_{\lambda_1} \perp \vec{u}_{\lambda_2}$$

(68)

Each column vector of $U_2$ is an eigenvector of $Q$ so that it forms orthogonal basis with length of 1.

$$U_2 = \begin{bmatrix} \vec{u}_{\lambda_1} & \vec{u}_{\lambda_2} \end{bmatrix} = \begin{bmatrix} -\frac{\sqrt{6}}{3} & \frac{\sqrt{3}}{3} \\ \frac{\sqrt{3}}{3} & \frac{\sqrt{6}}{3} \end{bmatrix}$$

(69)

$$U_3 = \begin{bmatrix} \vec{x} & \vec{y} & \vec{z} \end{bmatrix} = \begin{bmatrix} \vec{x} & R_2 U_2 \end{bmatrix} = \begin{bmatrix} \vec{x} & R_2 \vec{u}_{\lambda_1} & R_2 \vec{u}_{\lambda_2} \end{bmatrix}$$

(70)

Using the above information, we can finally compute $\vec{y}$ and $\vec{z}$.

$$\vec{y} = R_2 \vec{u}_{\lambda_1} = \begin{bmatrix} \frac{\sqrt{2}}{2} & 0 \\ -\frac{\sqrt{2}}{2} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} -\frac{\sqrt{6}}{3} \\ \frac{\sqrt{3}}{3} \end{bmatrix} = \begin{bmatrix} -\frac{\sqrt{3}}{3} \\ \frac{\sqrt{6}}{3} \end{bmatrix}$$

(71)

$$\vec{z} = R_2 \vec{u}_{\lambda_2} = \begin{bmatrix} \frac{\sqrt{2}}{2} & 0 \\ -\frac{\sqrt{2}}{2} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{\sqrt{3}}{3} \\ \frac{\sqrt{6}}{3} \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{6}}{3} \\ -\frac{\sqrt{3}}{3} \end{bmatrix}$$

(72)
\[
U_3 = \begin{bmatrix}
\sqrt{2}/2 & -\sqrt{3}/3 & \sqrt{6}/6 \\
\sqrt{2}/2 & \sqrt{3}/3 & -\sqrt{6}/6 \\
0 & \sqrt{2}/3 & \sqrt{3}/3 \\
\end{bmatrix}
\]  
(73)

Finally, we can find the values \(a, b, c\) by

\[
U_3^\top A U_3 = T = \begin{bmatrix}
\lambda_1 & a & b \\
0 & \lambda_2 & c \\
0 & 0 & \lambda_3 \\
\end{bmatrix}
\]  
(74)

\[
U_3^\top A U_3 = \begin{bmatrix}
\sqrt{2}/2 & \sqrt{2}/2 & 0 \\
\frac{\sqrt{3}}{3} & \frac{\sqrt{3}}{3} & \frac{\sqrt{3}}{3} \\
\frac{\sqrt{6}}{6} & -\frac{\sqrt{6}}{6} & \frac{\sqrt{6}}{6} \\
\end{bmatrix} \begin{bmatrix}
3 & -1 & 2 \\
3 & -1 & 6 \\
-2 & 2 & -2 \\
\end{bmatrix} \begin{bmatrix}
\frac{\sqrt{2}}{2} & -\frac{\sqrt{3}}{3} & \frac{\sqrt{6}}{6} \\
\frac{\sqrt{2}}{2} & \frac{\sqrt{3}}{3} & -\frac{\sqrt{6}}{6} \\
0 & \frac{\sqrt{3}}{3} & \frac{\sqrt{6}}{6} \\
\end{bmatrix} = \begin{bmatrix}
2 & 0 & 4\sqrt{3} \\
0 & 2 & 0 \\
0 & 0 & -4 \\
\end{bmatrix}
\]  
(75)

Hence, we have

\[
a = 0, \quad b = 4\sqrt{3}, \quad c = 0
\]  
(76)

which completes our upper triangularization.
5. Using upper-triangularization to solve differential equations

You know that for any square matrix $A$ with real eigenvalues, there exists a real matrix $V$ with orthonormal columns and a real upper triangular matrix $R$ so that $A = V R V^\top$. In particular, to set notation explicitly:

$$V = \begin{bmatrix} \vec{v}_1, \vec{v}_2, \ldots, \vec{v}_n \end{bmatrix}$$

$$R = \begin{bmatrix} \vec{r}_1^\top \\ \vec{r}_2^\top \\ \vdots \\ \vec{r}_n^\top \end{bmatrix}$$

where the rows of the upper-triangular $R$ look like

$$\vec{r}_1^\top = \begin{bmatrix} \lambda_1, r_{1,2}, r_{1,3}, \ldots, r_{1,n} \end{bmatrix}$$

$$\vec{r}_2^\top = \begin{bmatrix} 0, \lambda_2, r_{2,3}, r_{2,4}, \ldots, r_{2,n} \end{bmatrix}$$

$$\vec{r}_i^\top = \begin{bmatrix} 0, \ldots, 0, \lambda_i, r_{i,i+1}, r_{i,i+2}, \ldots, r_{i,n} \end{bmatrix}$$

$$\vec{r}_n^\top = \begin{bmatrix} 0, \ldots, 0, \lambda_n \end{bmatrix}$$

where the $\lambda_i$ are the eigenvalues of $A$.

Suppose our goal is to solve the $n$-dimensional system of differential equations written out in vector/matrix form as:

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

$$\vec{x}(0) = \vec{x}_0,$$  \hspace{1cm} (83) \hspace{1cm} (84)

where $\vec{x}_0$ is a specified initial condition and $\vec{u}(t)$ is a given vector of functions of time.

Assume that the $V$ and $R$ have already been computed and are accessible to you using the notation above. Assume that you have access to a function $\text{ScalarSolve}(\lambda, y_0, \vec{u})$ that takes a real number $\lambda$, a real number $y_0$, and a real-valued function of time $\vec{u}$ as inputs and returns a real-valued function of time that is the solution to the scalar differential equation

$$\frac{d}{dt} y(t) = \lambda y(t) + \vec{u}(t)$$

with initial condition $y(0) = y_0$.

Also assume that you can do regular arithmetic using real-valued functions and it will do the right thing. So if $u$ is a real-valued function of time, and $g$ is also a real-valued function of time, then $5u + 6g$ will be a real valued function of time that evaluates to $5u(t) + 6g(t)$ at time $t$.

**Use $V, R$ to construct a procedure for solving this differential equation**

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

$$\vec{x}(0) = \vec{x}_0,$$  \hspace{1cm} (83) \hspace{1cm} (84)

---

Homework 9, © UCB EECS 16B, Fall 2021. All Rights Reserved. This may not be publicly shared without explicit permission.
for $\vec{x}(t)$ by filling in the following template in the spots marked ♠, ♦, ♥, ♠.

(Note: It will be useful to upper triangularize $A$ by change of basis to get a differential equation in terms of $R$ instead of $A$.)

(HINT: The process here should be similar to diagonalization with some modifications. Start from the last row of the system and work your way up to understand the algorithm.)

1: $\vec{\tilde{x}}_0 = V^T \vec{x}_0$  \hspace{1cm} \triangleright Change the initial condition to be in $V$-coordinates
2: $\vec{\tilde{u}} = V^T \vec{u}$  \hspace{1cm} \triangleright Change the external input functions to be in $V$-coordinates
3: for $i = n$ down to 1 do  \hspace{1cm} \triangleright Iterate up from the bottom row
4: $\vec{\hat{u}}_i = \clubsuit + \sum_{j=i+1}^{n} \spadesuit$  \hspace{1cm} \triangleright Make the effective input for this level
5: $\vec{\tilde{x}}_i = ScalarSolve(♥, \vec{\tilde{x}}_0, i, \vec{\hat{u}}_i)$  \hspace{1cm} \triangleright Solve this level’s scalar differential equation
6: end for
7: $\vec{x}(t) = \heartsuit \begin{bmatrix} \vec{x}_1(t) \\ \vec{x}_2(t) \\ \vdots \\ \vec{x}_n(t) \end{bmatrix}$  \hspace{1cm} \triangleright Change back into original coordinates

(a) **Give the expression for ♥ on line 7 of the algorithm above.** (i.e., how do you get from $\vec{\tilde{x}}(t)$ to $\vec{x}(t)$?)

**Solution:** Since $\vec{\tilde{x}}_0 = V^T \vec{x}_0$ we know we are changing to $V$-basis. So, the implicit change of variable that we are doing is $\vec{x} = V^T \vec{\tilde{x}}$, this means that to come back, $\vec{x} = V \vec{\tilde{x}}$.

Thus, $♥ = V$.

(b) **Give the expression for ♦ on line 5 of the algorithm above.** (i.e., what are the $\lambda$ arguments to $ScalarSolve$, equation (2), for the $i^{th}$ iteration of the for-loop?)

(HINT: Convert the differential equation to be in terms of $R$ instead of $A$. It may be helpful to start with $i = n$ and develop a general form for the $i^{th}$ row.)

**Solution:** We begin by taking our vector differential equation and substituting in our upper triangularization:

$$\frac{d}{dt} \vec{x}(t) = A\vec{x}(t) + \vec{u}(t) \quad (88)$$

$$\frac{d}{dt} \vec{x}(t) = RV^T \vec{x}(t) + \vec{u}(t) \quad (89)$$

Multiplying both sides by $V^T$ and using the fact that $V^TV = I$

$$V^T \frac{d}{dt} \vec{x}(t) = RV^T \vec{x}(t) + V^T \vec{u}(t) \quad (90)$$

Now, we perform change of variables, $\vec{\tilde{x}} = V^T \vec{x}$ and $\vec{\tilde{u}} = V^T \vec{u}$ so we get,

$$\frac{d}{dt} \vec{\tilde{x}}(t) = R\vec{\tilde{x}}(t) + \vec{\tilde{u}}(t) \quad (91)$$

Thus, the $i^{th}$ equation in this system is,

$$\frac{d}{dt} \vec{x}_i(t) = r_i \vec{\tilde{x}}(t) + \vec{\tilde{u}}_i(t) \quad (92)$$
Using, $\vec{r}_i^T = \begin{bmatrix} 0, \ldots, 0, \lambda_i, r_{i,i+1}, r_{i,i+2}, \ldots, r_{i,n} \end{bmatrix}$ we get,

$$\frac{d}{dt} \tilde{x}_i(t) = \lambda_i \tilde{x}_i(t) + r_{i,i+1} \tilde{x}_{i+1}(t) + r_{i,i+2} \tilde{x}_{i+2}(t) + \cdots + r_{i,n} \tilde{x}_n(t) + \tilde{u}_i(t)$$ (93)

Thus, $\frac{d}{dt} \tilde{x}_i(t) = \lambda_i \tilde{x}_i(t) + \tilde{u}_i(t) + \sum_{j=i+1}^{n} r_{i,j} \tilde{x}_j(t)$

Here, we can see that when solving the scalar differential equation for the $i$th row, the scaling term is $\lambda_i$: $\Diamond = \lambda_i$.

(c) Give the expression for $\clubsuit$ on line 4 of the algorithm above.

**Solution:**

Since, from above, $\frac{d}{dt} \tilde{x}_i(t) = \lambda_i \tilde{x}_i(t) + \tilde{u}_i(t) + \sum_{j=i+1}^{n} r_{i,j} \tilde{x}_j(t)$ we can see that the $\tilde{u}_i$ is the input term that does not depend on the inner sum. From this we conclude that $\clubsuit = \tilde{u}_i$.

(d) Give the expression for $\spadesuit$ on line 4 of the algorithm above.

**Solution:**

Since, from above, $\frac{d}{dt} \tilde{x}_i(t) = \lambda_i \tilde{x}_i(t) + \tilde{u}_i(t) + \sum_{j=i+1}^{n} r_{i,j} \tilde{x}_j(t)$ and so we know what is inside the inner sum:

$\spadesuit = r_{i,j} \tilde{x}_j(t)$.

Congratulations! You now know how to systematically solve any system of differential equations with constant coefficients, as long as you know how to solve the scalar case with inputs. The same argument style applies for recurrence relations. The only gap that remains is the assumption that all the eigenvalues are real, but once you understand orthogonality for complex vectors, you can also update your understanding of upper-triangularization to allow for complex matrices as well.

(e) (OPTIONAL) Let us complete the algorithm by investigating how $ScalarSolve(\lambda, y_0, \tilde{u})$ works.

Consider an input that is a weighted polynomial times and exponential.

$$\tilde{u}(t) = \alpha t^\beta e^{\gamma t}$$ (94)

Here, $\alpha$ is a real constant, $\beta$ is a non-negative integer, and $\gamma$ is a real exponent. In addition, we will assume that $\gamma = \lambda$ for simplicity. We encourage you to attempt solving this system if $\gamma \neq \lambda$ if you are curious.

**What function should $ScalarSolve(\lambda, y_0, \tilde{u})$ return for the above $\tilde{u}$? Remember, we are only considering the case where $\gamma = \lambda$.** Express the answer in terms of $\alpha, \beta, \gamma$.

**Solution:** $ScalarSolve(\lambda, y_0, \tilde{u})$ should return the solution to the differential equation

$$\frac{d}{dt} y(t) = \lambda y(t) + \tilde{u}(t)$$ (95)

with initial condition $y(0) = y_0$.

Recall from HW 2 the following general integral solution to such a differential equation

$$y(t) = y_0 e^{\lambda t} + \int_{0}^{t} \tilde{u}(\tau) e^{\lambda(t-\tau)} d\tau$$ (96)
Plugging $\tilde{u}(t)$ into (96) yields
\[ y(t) = y_0 e^{\lambda t} + \int_0^t \alpha \tau^\beta e^{\gamma \tau} e^{\lambda (t - \tau)} d\tau \]
\[ = y_0 e^{\lambda t} + \alpha e^{\lambda t} \int_0^t \tau^\beta e^{\gamma \tau - \lambda \tau} d\tau \]  
(97)  
(98)

There are now two cases: if $\gamma = \lambda$ and if $\gamma \neq \lambda$. Again, the problem only requires us to solve for the case where $\gamma = \lambda$, but we will show both.

**Case 1: $\gamma = \lambda$**

In this case, we can simplify (98) to
\[ y(t) = y_0 e^{\lambda t} + \alpha e^{\lambda t} \left[ \frac{1}{\beta + 1} \tau^{\beta + 1} \right]_{\tau=0}^{\tau=t} \]
\[ = y_0 e^{\lambda t} + \frac{\alpha}{\beta + 1} \tau^{\beta + 1} e^{\lambda t} \]  
(99)  
(100)  
(101)

Notice that in this case, we increment the power of $t$ — that is what happens when we encounter this exact $\lambda$ that matches the input exponential. **This is the solution to the question. Everything below is for your own curiosity**

**Case 2: $\gamma \neq \lambda$**

In this case, we must use integration by parts to solve for the integral in (98). It would have been fine if you had just looked up the formula. But for completeness, we show the procedure from integral calculus.

Using the tabular integration method, one can solve for $\int F(t)G(t)dt$ by creating a table where the function $F(t)$ is successively differentiated on the left column, and the function $G(t)$ is successively integrated in the right column. Every other entry is negated in the first column, and finally, take the sum of the entries from the first column times the entries in the second column that are one row below.

Considering the integral from (98): $I = \int_0^t \tau^\beta e^{\gamma \tau - \lambda \tau} d\tau$

<table>
<thead>
<tr>
<th>$F$</th>
<th>$G$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau^\beta$</td>
<td>$e^{\gamma \tau - \lambda \tau}$</td>
</tr>
<tr>
<td>$-\beta \tau^{\beta-1}$</td>
<td>$\frac{1}{(\gamma - \lambda)^2} e^{\gamma \tau - \lambda \tau}$</td>
</tr>
<tr>
<td>$\beta (\beta - 1) \tau^{\beta-2}$</td>
<td>$\frac{1}{(\gamma - \lambda)^3} e^{\gamma \tau - \lambda \tau}$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$(-1)^{\beta-1} \beta! \tau^1$</td>
<td>$\frac{1}{(\gamma - \lambda)^{\beta+1}} e^{\gamma \tau - \lambda \tau}$</td>
</tr>
<tr>
<td>$(-1)^{\beta} \beta!$</td>
<td>$\frac{1}{(\gamma - \lambda)^{\beta+2}} e^{\gamma \tau - \lambda \tau}$</td>
</tr>
<tr>
<td>0</td>
<td>$\frac{1}{(\gamma - \lambda)^{\beta+3}} e^{\gamma \tau - \lambda \tau}$</td>
</tr>
</tbody>
</table>

Writing the summation yields
\[ I = \sum_{m=0}^\beta (-1)^m \frac{\beta!}{(\beta - m)! (\gamma - \lambda)^{1+m}} \tau^{\beta-m} e^{\gamma \tau - \lambda \tau} \]  
(102)
We must evaluate $I$ at the integration bounds $\tau = 0$ and $\tau = t$, which yields

$$I_{\text{def}} = \sum_{m=0}^{\beta} (-1)^m \frac{\beta!}{(\beta - m)! (\gamma - \lambda)(1+m)} t^{\beta-m} e^t (\gamma - \lambda)$$  \hspace{1cm} (103)$$

Notice that this case doesn’t spawn any power of $t$ that is higher than the original $\beta$ that we started with, although lower powers of $t$ can be spawned in this process. The only way to get higher powers is to encounter the exact same $\lambda = \gamma$.

We use the integral we found $I_{\text{def}}$ to find the solution for (98) as:

$$y(t) = y_0 e^{\lambda t} + \alpha e^{\lambda t} \left( \sum_{m=0}^{\beta} (-1)^m \frac{\beta!}{(\beta - m)! (\gamma - \lambda)(1+m)} t^{\beta-m} e^t (\gamma - \lambda) \right)$$  \hspace{1cm} (104)$$

$$= y_0 e^{\lambda t} + \alpha \left( \sum_{m=0}^{\beta} (-1)^m \frac{\beta!}{(\beta - m)! (\gamma - \lambda)(1+m)} t^{\beta-m} e^t \right)$$  \hspace{1cm} (105)$$

**Conclusion of the entire problem:** The approach here is completely algorithmic and leans on the linear-algebra of upper-triangularization. In later courses (like 120), you will learn other techniques to get the same solutions that rely on complex analysis based approaches called Laplace Transforms. The overall work is the same in both cases. The advantage to the given approach is just that the proof/derivation is entirely elementary. Meanwhile, the Laplace Transform approach needs to rely on the uniqueness of Laplace Transforms which requires the techniques of Math 185 and beyond to establish.
6. Open-loop and closed-loop control

In the Front-End and System ID labs, we have built SIXT33N’s motor control circuitry and developed a linear model for the velocity of each wheel. We are one step away from our goal: to have SIXT33N drive in a straight line! We will see how to use the model we developed in System ID to control SIXT33N’s trajectory to be a straight line.

In the first part of this problem, we will see what happens if the parameters we got from system identification were not perfectly correct. In the second part, we’ll see how we can understand the dynamics so that we can do closed-loop control, and then see the consequence of bounded-input bounded-output stability.

Part 1: Open-Loop Control

An open-loop controller is one in which the input is predetermined using your system model and the goal, and not adjusted at all during operation. To design an open-loop controller for your car, you would set the PWM duty-cycle value of the left and right wheels (inputs \( u_L[i] \) and \( u_R[i] \)) such that the predicted velocity of both wheels is your target wheel velocity \( v_t \). You can calculate these inputs from the target velocity \( v_t \) and the \( \theta_{L,R}, \beta_{L,R} \) values you learned from data. In the previous homework problem and System ID lab, we have modeled the velocity of the left and right wheels as

\[
\begin{align*}
    v_L[i] &= d_L[i + 1] - d_L[i] = \theta_L u_L[i] - \beta_L; \\ \\
    v_R[i] &= d_R[i + 1] - d_R[i] = \theta_R u_R[i] - \beta_R
\end{align*}
\]  

where \( d_{L,R}[i] \) represent the distance traveled by each wheel.

(a) Do the open-loop control design that would give us \( v_L[i] = v_R[i] = v_t \). Treat the model parameters \( \beta_L, \beta_R, \theta_L, \theta_R \) as symbolic quantities along with the target velocity \( v_t \).

i.e. Solve the model (eqs. (106) to (107)) for the inputs \( u_L[i] \) and \( u_R[i] \) that make the nominal velocities \( v_L[i] = v_R[i] = v_t \).

**Solution:** Starting from eqs. (106) to (107) and substituting in the target velocity \( v_t \), we get the following equations.

\[
\begin{align*}
    v_t &= \theta_L u_L[i] - \beta_L \\ \\
    v_t &= \theta_R u_R[i] - \beta_R \\ \\
    v_t + \beta_L &= \theta_L u_L[i] \\ \\
    v_t + \beta_R &= \theta_R u_R[i] \\ \\
    v_t + \beta_L &= \theta_L u_L[i] \\ \\
    v_t + \beta_R &= \theta_R u_R[i]
\end{align*}
\]

(b) The challenge with reality is that it is not exactly as we think it is. The \( \theta_L, \theta_R \) parameters for example are learned from a finite amount data and so can be wrong. Let’s use star superscripts to denote the hidden true values for the parameters, and plain (without star superscripts) to denote what we think the parameters are.
Suppose that there is 10% mismatch between $\theta_L$ in the model and $\theta^*_L$ in the physical system (i.e.,
\[
\frac{\theta^*_L - \theta_L}{\theta_L} = 0.1.
\]

If we used the control inputs chosen in the previous part (a), what would be the resulting velocity mismatch, $\frac{v_{L[i]} - v_t}{v_t}$, if $\theta_L = 2$, $\beta_L = -2.5$, and $v_t = 200$, but $\theta^*_L = 2.2$?

Solution:

\[
v_{L[i]} = \theta^*_L u_{L[i]} - \beta_L \]
\[
= \theta^*_L \times \frac{1}{\theta_L} (v_t + \beta_L) - \beta_L \quad \text{(115)}
\]
\[
= \frac{\theta^*_L}{\theta_L} v_t + \frac{\theta^*_L - \theta_L}{\theta_L} \beta_L \quad \text{(116)}
\]
\[
= (1 + \frac{\theta^*_L - \theta_L}{\theta_L}) v_t + \frac{\theta^*_L - \theta_L}{\theta_L} \beta_L \quad \text{(117)}
\]

Therefore,

\[
\frac{v_{L[i]} - v_t}{v_t} = \frac{1}{\theta_t} \theta^*_L - \frac{\theta^*_L - \theta_L}{\theta_L} (v_t + \beta_L) \quad \text{(119)}
\]
\[
= \frac{1}{200} (200 - 2.5) \quad \text{(120)}
\]
\[
= 0.09875 \quad \text{(121)}
\]

Above you calculated it for the left wheel, but there’s no reason to believe that this discrepancy would magically be the same for the right wheel. What happens when the velocities of the two wheels disagree with each other? The robot car goes in a circle instead of a straight line. To actually go in a straight line, we need the distance traveled by both wheels to be the same.

To do this, we are going to have to use feedback control to be able to reject the disturbances that come from the fact that our model is wrong. But before we do that, we are going to need to set up a model that has some state in it that we can monitor and use to adjust the control inputs.

Here, it is useful to apply the philosophy that we saw in the trajectory tracking problem on the earlier homework. We should define our state as something that captures how far off we are from the desired behavior. Something that we would ideally like to have be zero.

This is what prompts us to define our state variable $\delta$, to be the difference in the distance traveled between the left and right wheels at a given timestep:

\[
\delta[i] := d_{L[i]} - d_{R[i]} \quad \text{(122)}
\]

We have the following scalar discrete-time system equation. Here $w[i]$ is a (hopefully) bounded disturbance sequence and $f(u_{L[i]}, u_{R[i]})$ is the control input to the system, which is a function of $u_{L[i]}$ and $u_{R[i]}$.

\[
\delta[i+1] = \lambda_{OL} \delta[i] + f(u_{L[i]}, u_{R[i]}) + w[i]. \quad \text{(123)}
\]

(c) Consider the scalar discrete-time system in eq. (123). If we apply the open-loop control and the system is exactly as we have modeled it, what is $\delta[i+1]$ in terms of $\delta[i]$? What is the eigenvalue $\lambda_{OL}$ of the system?
Consider the case that for all \( i \) the system is unstable. Alternatively, we can examine the influence of the noise \( \epsilon \), and then plugging in our open-loop control inputs \( u_L[i] = \frac{v_t + \beta_L}{\theta_L} \) and \( u_R[i] = \frac{v_t + \beta_R}{\theta_R} \), or making use of our hint.

If proceeding by the hint:

\[
\delta[i + 1] = d_L[i + 1] - d_R[i + 1] = v_L[i] + d_L[i] - (v_R[i] + d_R[i])
\]

\[
\delta[i + 1] = v_L[i] + d_L[i] - (v_R[i] + d_R[i])
\]

\[
\delta[i + 1] = d_L[i] - d_R[i]
\]

\[
\delta[i + 1] = \delta[i]
\]

From the derivation above, \( \lambda_{OL} = 1 \).

If starting from the original equations, we have from subtracting eqs. (106) to (107):

\[
d_L[i + 1] - d_L[i] - (d_R[i + 1] - d_R[i]) = \theta_L u_L[i] - \beta - (\theta_R u_R[i] + \beta_R)
\]

\[
(d_L[i + 1] - d_L[i]) - (d_R[i + 1] - d_R[i]) = \theta_L u_L[i] - \theta_R u_R[i] - \beta_L + \beta_R
\]

\[
\delta[i + 1] - \delta[i] = \theta_L u_L[i] - \theta_R u_R[i] - \beta_L + \beta_R
\]

\[
\delta[i + 1] = \delta[i] + \theta_L u_L[i] - \theta_R u_R[i] - \beta_L + \beta_R
\]

From the above equation, we observe that \( f(u_L[i], u_R[i]) = \theta_L u_L[i] - \theta_R u_R[i] - \beta_L + \beta_R \), and that \( \lambda_{OL} = 1 \). Even if we substitute our controls in, since they do not depend on \( \delta[i] \) (the exact meaning of open-loop, no feedback) we should observe no change in the coefficient of \( \delta[i] \). With open-loop control we have the following equation just as before when using the hint:

\[
\delta[i + 1] = \delta[i] + \theta_L u_L[i] - \theta_R u_R[i] - \beta_L + \beta_R
\]

\[
\delta[i + 1] = \delta[i] + (v_t + \beta_L) - (v_t + \beta_R) - \beta_L + \beta_R
\]

\[
\delta[i + 1] = \delta[i] + (v_t - v_t) + (\beta_L - \beta_L) + (\beta_R - \beta_R)
\]

\[
\delta[i + 1] = \delta[i]
\]

To check stability, we already know our eigenvalue does not meet the criteria \( |\lambda_{OL}| < 1 \), so we have an unstable system. Alternatively, we can examine the influence of the noise \( w[i] \) on the system for a specific \( w[i] \) to see if it grows and exceeds any bound. Since \( w[i] \) is bounded, for some \( \epsilon > 0 \), we can state the boundedness of \( w[i] \) mathematically:

\[
|w[i]| \leq \epsilon \quad \forall i
\]

Consider the case that for all \( i \), \( w[i] \) has the same value \( \epsilon \), i.e.,

\[
|w[i]| = \epsilon \quad \forall i
\]
Then,

$$\lim_{i \to \infty} \delta[i] = \lim_{i \to \infty} \lambda_{DL} \delta[0] + \sum_{j=1}^{i} \lambda_{DL}^{j-1} \epsilon$$  \hspace{1cm} (139)$$

$$= \lim_{i \to \infty} \delta[0] + \sum_{j=1}^{i} \epsilon \to \infty$$  \hspace{1cm} (140)$$

Thus the open-loop system is unstable.

**Part 2: Closed-Loop Control**

As we have seen in the open-loop case, if there is any mismatch in the model parameters, the velocity of the right wheel and left wheel would not necessarily be the same. To solve this problem, we must implement closed-loop control and use feedback in real time.

(d) **If we want the car to drive straight starting from some timestep** $i_{\text{start}} > 0$, i.e., $v_L[i] = v_R[i]$ for $i \geq i_{\text{start}}$, what condition does this impose on $\delta[i]$ for $i \geq i_{\text{start}}$?

**Solution:**

$$v_L[i] - v_R[i] = d_L[i + 1] - d_L[i] - (d_R[i + 1] - d_R[i])$$  \hspace{1cm} (141)$$

$$= (d_L[i + 1] - d_R[i + 1]) - (d_L[i] - d_R[i])$$  \hspace{1cm} (142)$$

$$= \delta[i + 1] - \delta[i]$$  \hspace{1cm} (143)$$

$$= 0$$  \hspace{1cm} (144)$$

$$\delta[i + 1] = \delta[i], i \geq i_{\text{start}}$$  \hspace{1cm} (145)$$

So we have that for every timestep beyond $i_{\text{start}}$, the difference in distances the wheels have traveled does not change.

$$\delta[i] = \delta[i_{\text{start}}], i \geq i_{\text{start}}$$  \hspace{1cm} (146)$$

(e) **How is the condition you found in (d) different from** $\delta[i] = 0$ for $i \geq i_{\text{start}}$? Assume that $i_{\text{start}} > 0$, and that $d_L[0] = 0, d_R[0] = 0$.

This is a subtlety that is worth noting and often requires one to adjust things in real systems so that control systems do what we actually want them to do.

**Solution:** At time $i = 0$, the car has not moved yet, so $\delta[0] = d_L[0] - d_R[0] = 0$. If at some later time $i_{\text{start}}$ we have $\delta[i_{\text{start}}] = 0$ and $\delta[i] = 0$ for later times as well, we remain moving in the same direction we started with. When $\delta[i] \neq 0$, this means the wheels have moved different distances, and therefore has moved along a curved path and changed the direction the car is pointing.

While not required, Fig. 1 illustrates the two different cases where $\delta[i] = 0$ for all times $i \geq 0$ (left) and when $\delta \neq 0$ initially but we have $\delta[i_{\text{start}}] = 0$ for some $i = i_{\text{start}}$ and $\delta[i] = 0$ for $i \geq i_{\text{start}}$ (right).

(f) From here, assume that we have reset the distance travelled counters at the beginning of this maneuver so that $\delta[0] = 0$. We will now implement a feedback controller by selecting two dimensionless positive coefficients, $f_L$ and $f_R$, such that the closed loop system is stable with eigenvalue $\lambda_{CL}$. To implement closed-loop feedback control, we want to adjust $v_L[i]$ and $v_R[i]$ at each timestep by an amount that’s...
proportional to $\delta[i]$. Not only do we want our wheel velocities to be close to some target velocity $v_t$, we also wish to drive $\delta[i]$ towards zero. This is in order to have the car drive straight along the initial direction it was pointed in when it started moving. If $\delta[i]$ is positive, the left wheel has traveled more distance than the right wheel, so relatively speaking, we can slow down the left wheel and speed up the right wheel to cancel this difference (i.e., drive it to zero) in the next few timesteps. The action of such a control is captured by the following velocities.

\[ v_L[i] = v_t - f_L \delta[i]; \]
\[ v_R[i] = v_t + f_R \delta[i]. \] (147) (148)

Give expressions for $u_L[i]$ and $u_R[i]$ as a function of $v_t$, $\delta[i]$, $f_L$, $f_R$, and our nominal system parameters $\theta_L$, $\theta_R$, $\beta_L$, $\beta_R$, to achieve the nominal velocities above.

Solution:
As in the open loop case, we substitute the velocity expressions above into the equations that relate $v[i]$ and $u[i]$.

For the left wheel we have:

\[ v_t - f_L \delta[i] = \theta_L u_L[i] - \beta_L \] (149)
\[ v_t - f_L \delta[i] + \beta_L = \theta_L u_L[i] \] (150)
\[ \frac{v_t - f_L \delta[i] + \beta_L}{\theta_L} = u_L[i] \] (151)

For the right wheel we have:

\[ v_t + f_R \delta[i] = \theta_R u_R[i] - \beta_R \] (152)
\[ v_t + f_R \delta[i] + \beta_R = \theta_R u_R[i] \] (153)
\[ \frac{v_t + f_R \delta[i] + \beta_R}{\theta_R} = u_R[i] \] (154)

(g) Using the control inputs $u_L[i]$ and $u_R[i]$ found in part (f) and assuming that our nominal parameters are actually true, write the closed-loop system equation for $\delta[i + 1]$ as a function of $\delta[i]$. What is the closed-loop eigenvalue $\lambda_{CL}$ for this system in terms of $\lambda_{OL}$, $f_L$, and $f_R$?

Solution:
We can take the system equation explicitly in terms of $u_L[i]$ and $u_R[i]$ from the solution of part (c) in eq. (132), and substitute into this equation our control expressions from the previous part.

\[
\delta[i + 1] = \delta[i] + \theta_L u_L[i] - \theta_R u_R[i] - \beta_L + \beta_R \\
= \delta[i] + \theta_L \left( \frac{v_t - f_L \delta[i] + \beta_L}{\theta_L} \right) - \theta_R \left( \frac{v_t + f_R \delta[i] + \beta_R}{\theta_R} \right) - \beta_L + \beta_R \\
= \delta[i] + v_t - f_L \delta[i] - (v_t + f_R \delta[i]) \\
= \delta[i] - f_L \delta[i] - f_R \delta[i] \\
= (1 - f_L - f_R) \delta[i]
\]

(155) \hspace{1cm} (156) \hspace{1cm} (157) \hspace{1cm} (158) \hspace{1cm} (159)

We see that our $\lambda_{CL}$ will end up being $1 - f_L - f_R$, which is equal to $\lambda_{OL} - f_L - f_R$.

(h) **What is the condition on $f_L$ and $f_R$ for the closed-loop system to be stable in the previous part?**

**Solution:**

\[
|\lambda_{CL}| < 1 \\
\Rightarrow |1 - f_L - f_R| < 1 \\
\Rightarrow -1 < 1 - f_L - f_R < 1 \\
\Rightarrow 0 < f_L + f_R < 2
\]

(160) \hspace{1cm} (161) \hspace{1cm} (162) \hspace{1cm} (163)

(i) Consider the following 10% mismatched mismatch between estimated model parameters $\theta_L, \theta_R$ and the real model parameters $\theta^*_L, \theta^*_R$.

\[
\frac{\theta^*_L - \theta_L}{\theta_L} = +0.1; \\
\frac{\theta^*_R - \theta_R}{\theta_R} = -0.1.
\]

(164) \hspace{1cm} (165)

With the mismatched mismatches above, what is the corresponding system equation? What is the closed-loop eigenvalue $\lambda_{CL}$ of the actual system?

**Solution:** To derive the system equation, we can substitute our closed-loop control inputs we found in part (f). However, the equation we substitute into is different as we have $\theta^*_L$ and $\theta^*_R$ instead.

\[
\delta[i + 1] = \delta[i] + \theta^*_L u_L[i] - \theta^*_R u_R[i] - \beta_L + \beta_R \\
= \delta[i] + \theta^*_L \left( \frac{v_t - f_L \delta[i] + \beta_L}{\theta_L} \right) - \theta^*_R \left( \frac{v_t + f_R \delta[i] + \beta_R}{\theta_R} \right) - \beta_L + \beta_R \\
= \delta[i] + \left( \frac{\theta^*_L}{\theta_L} - \frac{\theta^*_R}{\theta_R} \right) v_t - \left( \frac{\theta^*_L}{\theta_L} f_L + \frac{\theta^*_R}{\theta_R} f_R \right) \delta[i] + \left( \frac{\theta^*_L}{\theta_L} - 1 \right) \beta_L - \left( \frac{\theta^*_R}{\theta_R} - 1 \right) \beta_R
\]

(166) \hspace{1cm} (167) \hspace{1cm} (168)

We can evaluate the following quantities from our given mismatch values.

\[
\frac{\theta^*_L - \theta_L}{\theta_L} = \frac{\theta^*_L}{\theta_L} - 1 = +0.1 \\
\Rightarrow \frac{\theta^*_L}{\theta_L} = 1.1
\]

(169) \hspace{1cm} (170)

\[
\frac{\theta^*_R - \theta_R}{\theta_R} = \frac{\theta^*_R}{\theta_R} - 1 = -0.1
\]

(171)
(j) If there were no mismatch in the model parameters and there were no other source of disturbances

\[
\beta \quad \text{and} \quad \theta
\]

You should see that this is not zero, but instead depends on the target velocity

\[ f \quad \delta \quad \text{such that} \quad \lambda_{CL} = 1 - (1.1f_L + 0.9f_R). \]

Thus, we have the system equation \( \delta[i + 1] = (1 - (1.1f_L + 0.9f_R))\delta[i] + 0.1(\beta_L + \beta_R) + 0.2v_t \) and the closed-loop eigenvalue \( \lambda_{CL} = 1 - (1.1f_L + 0.9f_R). \)

If there were no mismatch in the model parameters and there were no other source of disturbances \( w[i], \) the state variable \( \delta[i] \) would eventually converge to 0 assuming the the system is stable, i.e., \(|\lambda_{CL}| < 1.\) However, with the mismatch introduced, \( \delta[i] \) may not converge to 0 but to some other constant, which is called the steady state error \( \delta_{SS} = \lim_{i \to \infty} \delta[i]. \)

Remember, BIBO stability just promises that a bounded disturbance gives rise to a bounded output — it doesn’t say that the result will be zero.

**What is the steady state error \( \delta_{SS} \) given 10\% mismatch in \( \theta_L \) and \( \theta_R \) as in eqs. (164) to (165)?**

Assume that even with the mismatch, you have chosen \( f_L \) and \( f_R \) such that \(|\lambda_{CL}| < 1.\)

You should see that this is not zero, but instead depends on the target velocity \( v_t \) as well as the \( \beta_R \) and \( \beta_L \) constants. Physically, this reflects the fact that the car will go straight, but it might turn a little before starting to go straight. (In later courses like 128, you’ll learn how to implicitly treat some of the steady mismatches as hidden states and cut down on such steady-state error.

**Solution:** Let \( \lambda_{CL} = 1 - (1.1f_L + 0.9f_R) \) and \( \epsilon = 0.2v_t + 0.1(\beta_L + \beta_R) \) such that

\[
\delta[i + 1] = \lambda_{CL}\delta[i] + \epsilon
\]

Then, we write out the limit using the closed-form expression for \( \delta[i] \) in terms of the initial condition \( \delta[0]: \)

\[
\delta_{SS} = \lim_{i \to \infty} \delta[i] = \lim_{i \to \infty} \lambda_{CL}^i \delta[0] + \sum_{j=0}^{i-1} \lambda_{CL}^j \epsilon
\]

\[
= \lim_{i \to \infty} \sum_{j=0}^{i-1} \lambda_{CL}^j \epsilon
\]

The term corresponding to the initial condition in the equation above goes to zero as \( i \to \infty \) since \(|\lambda_{CL}| < 1.\) Then we can rewrite the infinite geometric series above using the formula \( \sum_{k=0}^\infty ar^k = \frac{a}{1-r} \) for \(|r| < 1: \)

\[
\lim_{i \to \infty} \sum_{j=0}^{i-1} \lambda_{CL}^j \epsilon = \frac{\epsilon}{1 - \lambda_{CL}}
\]

\[
= \frac{0.2v_t + 0.1(\beta_L + \beta_R)}{1.1f_L + 0.9f_R}
\]

\[
= \frac{2v_t + \beta_L + \beta_R}{11f_L + 9f_R}
\]
So the steady-state error $\delta_{SS}$ converges to $\delta_{SS} = \frac{2v_t + \beta_L + \beta_R}{11f_L + 9f_R}$ as $i \rightarrow \infty$. 
7. 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 (from the bottom up) 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.

You need to write your own question and provide a thorough solution to it. The scope of your question should roughly overlap with the scope of this entire problem set. This is because we want you to exercise your understanding of this material, and not earlier material in the course. However, feel free to combine material here with earlier material, and clearly, you don’t have to engage with everything all at once. A problem that just hits one aspect is also fine.

Note: One of the easiest ways to make your own problem is to modify an existing one. Ordinarily, we do not ask you to cite official course materials themselves as you solve problems. This is an exception. Because the problem making process involves creative inputs, you should be citing those here. It is a part of professionalism to give appropriate attribution.

Just FYI: Another easy way to make your own question is to create a Jupyter part for a problem that had no Jupyter part given, or to add additional Jupyter parts to an existing problem with Jupyter parts. This often helps you learn, especially in case you have a programming bent.

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

(c) Roughly how many total hours did you work on this homework? Write it down here where you’ll need to remember it for the self-grade form.

Contributors:

• Anant Sahai.
• Michael Danielczuk.
• Regina Eckert.
• Nathan Lambert.
• Kuan-Yun Lee.
• Kunmo Kim.
• Sidney Buchbinder.
• Daniel Abraham.
• Bozhi Yin.
• Kaitlyn Chan.
• Yi-Hsuan Shih.
• Vladimir Stojanovic.
• Moses Won.