Homework 9

This homework is due on Saturday, October 28, 2023, at 11:59PM. Self-grades and HW Resubmissions are due on the following Saturday, November 4th, 2023, at 11:59PM.

1. Open-Loop and Closed-Loop Control

In last week's lab-related System ID problem, we 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 the System ID problem to control SIXT33N's trajectory to be a straight line.

More specifically, in this problem, we will explore how to use open-loop and closed-loop control to drive the trajectory of your car in a straight line.

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 $\theta_L, \theta_R, \beta_L, \beta_R$ values you learned from data. In the System ID problem and lab, we have modeled the velocity of the left and right wheels as

$$v_L[i] = d_L[i+1] - d_L[i] = \theta_L u_L[i] - \beta_L; \tag{1}$$

$$v_R[i] = d_R[i+1] - d_R[i] = \theta_R u_R[i] - \beta_R \tag{2}$$

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

(a) Find the open-loop control that would give us $v_L[i] = v_R[i] = v_t$. That is, **solve the model (Equations (1) and (2)) for the inputs $u_L[i]$ and $u_R[i]$ that make the velocities $v_L[i] = v_R[i] = v_t$.**

Solution: Starting from Equations (1) and (2) and substituting in the target velocity v_t, we get the following equations.

$$v_t = \theta_L u_L[i] - \beta_L \tag{3}$$

$$v_t = \theta_R u_R[i] - \beta_R \tag{4}$$

$$v_t + \beta_L = \theta_L u_L[i] \tag{5}$$

$$v_t + \beta_R = \theta_R u_R[i] \tag{6}$$

$$\frac{v_t + \beta_L}{\theta_L} = u_L[i] \tag{7}$$

$$\frac{v_t + \beta_R}{\theta_R} = u_R[i] \tag{8}$$

In practice, the $\theta_L, \theta_R, \beta_L, \beta_R$ parameters are learned from noisy data, and so can be wrong. This means that we will calculate the velocities for the two wheels incorrectly. When the velocities of the two wheels disagree, the car will go in a circle instead of a straight line. Thus, to make the car go in a straight line, we need the distances traveled by both wheels to be the same at each timestep.

This prompts us to simplify our model. Instead of having two state variables $\vec{v}_L$ and $\vec{v}_R$, we can just have a state variable determining how far we are from the desired behavior of going in a line – a state which we will want to drive to 0.

This prompts us to define our state variable δ to be the *difference* in the distance traveled by the left wheel and the right wheel at a given timestep:

$$\delta[i] := d_L[i] - d_R[i] \tag{9}$$

We want to find a scalar discrete-time model for $\delta[i]$ of the form

$$\delta[i+1] = \lambda_{\text{OL}}\delta[i] + f(u_L[i], u_R[i]). \tag{10}$$

Here λ_{OL} is a scalar and $f(u_L[i], u_R[i])$ is the control input to the system (as a function of $u_L[i]$ and $u_R[i]$).

(b) Suppose we apply the open-loop control inputs $u_L[i], u_R[i]$ to the original system. **Using Equations (1) and (2), write $\delta[i+1]$ in terms of $\delta[i]$, in the form of Equation (10). What is the eigenvalue λ_{OL} of the model in Equation (10)? Would the model in Equation (10) be stable with open-loop control if it also had a disturbance term?**

(HINT: For open-loop control, we set the velocities to $v_L[i] = v_R[i] = v_t$. What happens when we substitute that into Equations (1) and (2) and then apply the definition of $\delta[i]$ and $\delta[i+1]$?)

Solution: Proceeding by the hint,

$$\begin{aligned}
\delta[i+1] &= d_L[i+1] - d_R[i+1] && (11)\\
&= (v_L[i] + d_L[i]) - (v_R[i] + d_R[i]) && (12)\\
&= v_t + d_L[i] - (v_t + d_R[i]) && (13)\\
&= d_L[i] - d_R[i] && (14)\\
&= \delta[i] && (15)
\end{aligned}$$

From the derivation above, $\lambda_{\text{OL}} = 1$ and $f(u_L[i], u_R[i]) = 0$. To check stability, we already know our eigenvalue does not meet the stability criteria: $|\lambda_{\text{OL}}| = 1$, so we have an unstable system if we add disturbances (whereas if we don't then the system is *marginally stable*).

Part 2: Closed-Loop Control

Now, in order to make the car drive straight, we must implement closed-loop control – that is, control inputs that depend on the current state and are calculated dynamically – and use feedback in real time.

(c) **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: Let $i \geq i_{\text{start}}$. Then

$$\begin{aligned}
0 &= v_L[i] - v_R[i] && (16)\\
&= d_L[i+1] - d_L[i] - (d_R[i+1] - d_R[i]) && (17)\\
&= (d_L[i+1] - d_R[i+1]) - (d_L[i] - d_R[i]) && (18)
\end{aligned}$$

$$= \delta[i+1] - \delta[i]. \tag{19}$$

Thus

$$\delta[i+1] = \delta[i], \qquad i \geq i_{\text{start}} \tag{20}$$

which implies that

$$\delta[i] = \delta[i_{\text{start}}], \qquad i \geq i_{\text{start}}. \tag{21}$$

In other words, we have that for every timestep beyond i_{start}, the difference in distances the wheels have traveled does not change.

(d) **How is the condition you found in the previous part different from the condition:**

$$\delta[i] = 0, \qquad i \geq i_{\text{start}}? \tag{22}$$

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.

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_{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).

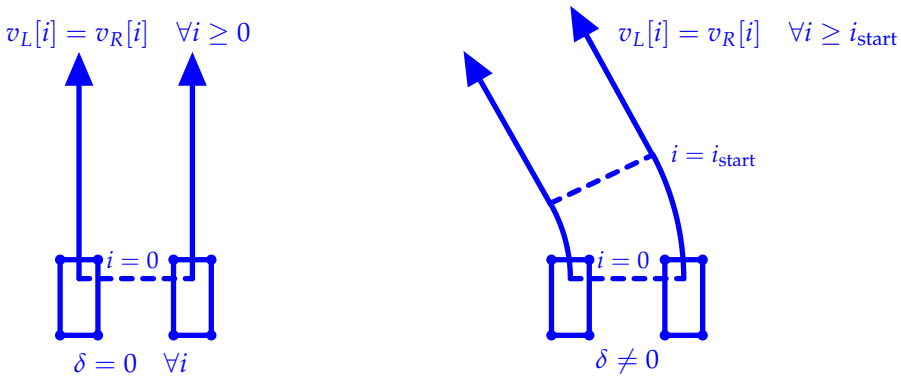

Figure 1

(e) 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 λ_{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 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.

$$v_L[i] = v_t - f_L\delta[i]; \tag{23}$$
$$v_R[i] = v_t + f_R\delta[i]. \tag{24}$$

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

$$v_t - f_L\delta[i] = \theta_L u_L[i] - \beta_L \tag{25}$$
$$v_t - f_L\delta[i] + \beta_L = \theta_L u_L[i] \tag{26}$$
$$\frac{v_t - f_L\delta[i] + \beta_L}{\theta_L} = u_L[i] \tag{27}$$

For the right wheel we have:

$$v_t + f_R\delta[i] = \theta_R u_R[i] - \beta_R \tag{28}$$
$$v_t + f_R\delta[i] + \beta_R = \theta_R u_R[i] \tag{29}$$
$$\frac{v_t + f_R\delta[i] + \beta_R}{\theta_R} = u_R[i] \tag{30}$$

(f) Using the control inputs $u_L[i]$ and $u_R[i]$ found in part (e), **write the closed-loop system equation for $\delta[i+1]$ as a function of $\delta[i]$. What is the closed-loop eigenvalue λ_{CL} for this system in terms of λ_{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. (10), and substitute into this equation our control expressions from the previous part.

$$\delta[i+1] = \delta[i] + \theta_L u_L[i] - \theta_R u_R[i] - \beta_L + \beta_R \tag{31}$$
$$= \delta[i] + \theta_L\left(\frac{v_t - f_L\delta[i] + \beta_L}{\theta_L}\right) - \theta_R\left(\frac{v_t + f_R\delta[i] + \beta_R}{\theta_R}\right) - \beta_L + \beta_R \tag{32}$$
$$= \delta[i] + v_t - f_L\delta[i] - (v_t + f_R\delta[i]) \tag{33}$$
$$= \delta[i] - f_L\delta[i] - f_R\delta[i] \tag{34}$$
$$= (1 - f_L - f_R)\delta[i] \tag{35}$$

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

(g) **What is the condition on f_L and f_R for the closed-loop system in the previous part to be stable in the presence of disturbance?**

Solution:

$$|\lambda_{\text{CL}}| < 1 \tag{36}$$

$$\implies |1 - f_L - f_R| < 1 \tag{37}$$
$$\implies -1 < 1 - f_L - f_R < 1 \tag{38}$$
$$\implies 0 < f_L + f_R < 2 \tag{39}$$

Stability in this case means that δ is bounded and will not go arbitrarily high. In fact, if our calculated β and θ are perfectly accurate, then $\delta[i] \to 0$, so the car will (eventually) drive straight!

One question remains – what if our calculated β and θ are *not* perfectly accurate? The answer turns out to be that there is some small steady-state discrepancy that your δ will converge to. You will see how to quantify this in next week's homework.

2. Cayley-Hamilton and Controllability Matrix

(a) We can define the *characteristic polynomial* of a matrix $A \in \mathbb{R}^{n\times n}$ as

$$p_A(\lambda) = \lambda^n + c_{n-1}\lambda^{n-1} + \cdots + c_1\lambda + c_0\lambda^0 \tag{40}$$

where each $c_i \in \mathbb{R}$ is a constant. The characteristic polynomial has roots that are the eigenvalues of A. That is, we can equivalently define

$$p_A(\lambda) = \det\{\lambda I - A\} \tag{41}$$

We say that any of the eigenvalues of A "satisfy" the characteristic polynomial in that

$$p_A(\lambda_i) = 0 \tag{42}$$

where λ_i is the ith eigenvalue of A. Now, let A be a diagonalizable matrix, where we may write $A = V\Lambda V^{-1}$. **Prove that A satisfies its own characteristic polynomial.** In other words, prove that $p_A(A) = 0_{n\times n}$, where $0_{n\times n}$ is a $n \times n$ matrix of zeros.
(HINT: It is not correct to simply plug in $\lambda = A$ into $\det\{\lambda I - A\}$.)
Solution: Recall that $A^i = V\Lambda^i V^{-1}$. Hence,

$$\begin{aligned} p_A(A) &= A^n + c_{n-1}A^{n-1} + \cdots + c_1 A + c_0 A^0 && (43)\\ &= V\Lambda^n V^{-1} + c_{n-1}V\Lambda^{n-1}V^{-1} + \cdots + c_1 V\Lambda V^{-1} + c_0 I && (44)\\ &= V\left(\Lambda^n + c_{n-1}\Lambda^{n-1} + \cdots + c_1\Lambda + c_0 I\right)V^{-1} && (45)\end{aligned}$$

Notice that the ith diagonal entry of $\Lambda^n + c_{n-1}\Lambda^{n-1} + \cdots + c_1\Lambda + c_0 I$ is $\lambda_i^n + c_{n-1}\lambda_i^{n-1} + \cdots + c_1\lambda_i + c_0 = p_A(\lambda_i)$. Thus, we have that

$$p_A(A) = V\begin{bmatrix} p_A(\lambda_1) & & & \\ & p_A(\lambda_2) & & \\ & & \ddots & \\ & & & p_A(\lambda_n)\end{bmatrix}V^{-1} \tag{46}$$

Note that $p_A(\lambda_i) = 0$, so

$$\begin{aligned} p_A(A) &= V\begin{bmatrix} p_A(\lambda_1) & & & \\ & p_A(\lambda_2) & & \\ & & \ddots & \\ & & & p_A(\lambda_n)\end{bmatrix}V^{-1} && (47)\\ &= V\begin{bmatrix} 0 & & & \\ & 0 & & \\ & & \ddots & \\ & & & 0\end{bmatrix}V^{-1} && (48)\\ &= 0_{n\times n} && (49)\end{aligned}$$

where $0_{n\times n}$ is an $n \times n$ matrix of zeros.

(b) Now, consider some vector $\vec{b} \in \mathbb{R}^n$. **Using the result from the previous part, show that $A^n\vec{b}$ is linearly dependent on $A^{n-1}\vec{b}, A^{n-2}\vec{b}, \ldots, A\vec{b}, \vec{b}$.**

Solution: From the previous part, we know that

$$A^n + c_{n-1}A^{n-1} + \cdots + c_1 A + c_0 I = 0_{n\times n} \tag{50}$$

Right multiplying both sides by $\vec{b}$, we get

$$A^n\vec{b} + c_{n-1}A^{n-1}\vec{b} + \cdots + c_1 A\vec{b} + c_0\vec{b} = \vec{0} \tag{51}$$

which we can rearrange to get

$$A^n\vec{b} = -\left(c_{n-1}A^{n-1}\vec{b} + \cdots + c_1 A\vec{b} + c_0\vec{b}\right) \tag{52}$$

Thus, $A^n\vec{b}$ is linearly dependent on $A^{n-1}\vec{b}, A^{n-2}\vec{b}, \ldots, A\vec{b}, \vec{b}$.

(c) Instead of setting $\vec{b}$ to be a vector, let it be a matrix $B \in \mathbb{R}^{n\times m}$. **Now, show that the columns of A^nB are linearly dependent on the columns of $A^{n-1}B, A^{n-2}B, \ldots, AB, B$.**

(HINT: If we were to write $B = \begin{bmatrix}\vec{b}_1 & \vec{b}_2 & \cdots & \vec{b}_m\end{bmatrix}$ where each column is n-dimensional, we can write $A^iB = \begin{bmatrix}A^i\vec{b}_1 & A^i\vec{b}_2 & \cdots & A^i\vec{b}_m\end{bmatrix}$. Make sure you convince yourself of this.)

Solution: We have that $A^nB = \begin{bmatrix}A^n\vec{b}_1 & A^n\vec{b}_2 & \cdots & A^n\vec{b}_m\end{bmatrix}$. From the previous part, we have that $A^n\vec{b}_i$ is linearly dependent on $A^{n-1}\vec{b}_i, A^{n-2}\vec{b}_i, \ldots, A\vec{b}_i, \vec{b}_i$. Since i is arbitrary here, we have that the columns of A^nB are linearly dependent on the columns of $A^{n-1}B, A^{n-2}B, \ldots, AB, B$.

(d) Consider a discrete time system of the form

$$\vec{x}[i+1] = A\vec{x}[i] + B\vec{u}[i] \tag{53}$$

where $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$. The controllability matrix for this discrete time system is given by

$$\mathcal{C} = \begin{bmatrix}A^{n-1}B & A^{n-2}B & \cdots & AB & B\end{bmatrix} \tag{54}$$

Conclude that the rank of your controllability matrix will not change if, instead, you made your controllability matrix $\begin{bmatrix}A^nB & A^{n-1}B & \cdots & AB & B\end{bmatrix}$ (i.e., you prepended A^nB to your original controllability matrix).

Solution: From the previous part, we have that the columns of A^nB are linearly dependent on the columns of $A^{n-1}B, A^{n-2}B, \ldots, AB, B$. Hence, if we were to prepend A^nB to our original controllability matrix, the rank would not change since each column of A^nB is linearly dependent on columns already in the controllability matrix.

3. CCF Transformation and Controllability

(a) Consider the following discrete time system

$$\vec{x}[i+1] = A\vec{x}[i] + B\vec{u}[i] \tag{55}$$

Suppose we define a change of basis operation given by $M\vec{z}[i] = \vec{x}[i] \iff \vec{z}[i] = M^{-1}\vec{x}[i]$. This yields a new discrete time system of the form

$$\vec{z}[i+1] = \widetilde{A}\vec{z}[i] + \widetilde{B}\vec{u}[i] \tag{56}$$

for some $\widetilde{A}$ and $\widetilde{B}$ defined in terms of M, A, and B. **What is the controllability matrix for the system in eq. (56), in terms of M, A, and B?**

Solution: We have that

$$\vec{x}[i+1] = A\vec{x}[i] + B\vec{u}[i] \tag{57}$$
$$M\vec{z}[i+1] = AM\vec{z}[i] + B\vec{u}[i] \tag{58}$$
$$\vec{z}[i+1] = \underbrace{M^{-1}AM}_{\widetilde{A}}\vec{z}[i] + \underbrace{M^{-1}B}_{\widetilde{B}}\vec{u}[i] \tag{59}$$

We have that the controllability matrix for the z system is $\mathcal{C}_z = \begin{bmatrix}\widetilde{A}^{n-1}\widetilde{B} & \widetilde{A}^{n-2}\widetilde{B} & \cdots & \widetilde{A}\widetilde{B} & \widetilde{B}\end{bmatrix}$ where $\widetilde{A}^i\widetilde{B} = \left(M^{-1}A^iM\right)\left(M^{-1}B\right) = M^{-1}A^iB$. Hence, we can rewrite the controllability matrix as

$$\mathcal{C}_z = \begin{bmatrix}M^{-1}A^{n-1}B & M^{-1}A^{n-2}B & \cdots & M^{-1}AB & M^{-1}B\end{bmatrix} \tag{60}$$
$$= M^{-1}\begin{bmatrix}A^{n-1}B & A^{n-2}B & \cdots & AB & B\end{bmatrix} \tag{61}$$

(b) Consider the change of basis given by $\vec{z}[i] = T^{-1}\vec{x}[i]$ where, under this change of basis transformation, we have the following discrete time system

$$\vec{z}[i+1] = A_{\text{CCF}}\vec{z}[i] + B_{\text{CCF}}\vec{u}[i] \tag{62}$$

Using the result from the previous part, determine an expression for T in terms of $\mathcal{C}$, the controllability matrix of the original system in eq. (55), and $\mathcal{C}_{\text{CCF}}$, the controllability matrix of the system in eq. (62).

Solution: From the previous part, we can set $M = T$ and $\mathcal{C}_z = \mathcal{C}_{\text{CCF}}$ to obtain

$$\mathcal{C}_{\text{CCF}} = T^{-1}\mathcal{C} \tag{63}$$
$$T\mathcal{C}_{\text{CCF}} = \mathcal{C} \tag{64}$$
$$T = \mathcal{C}\mathcal{C}_{\text{CCF}}^{-1} \tag{65}$$

(c) We know that the controllability matrix for a system in CCF will always be full rank. **Using this, prove that you can find a transformation matrix T as in the previous part if and only if your original system is controllable.** *(HINT: To prove this, you can first show that, if such a T exists, then*

your original system is controllable. Then, you can show that, if your original system is controllable, there will exist such a transformation matrix T.) (HINT: Recall that T must be invertible (equivalently, full rank) in order for it to be a valid transformation matrix. You may use without proof the fact that $\text{rank}(AB) = \min(\text{rank}(A), \text{rank}(B))$.*)*

Solution: Following the hint, we have that, if T exists, then it must be full rank. Also, $\text{rank}(\mathcal{C}_{\text{CCF}}) = n$. From the previous part, we end up with

$$\text{rank}(\mathcal{C}_{\text{CCF}}) = \text{rank}(T^{-1}\mathcal{C}) \tag{66}$$
$$\text{rank}(\mathcal{C}_{\text{CCF}}) = \min(\text{rank}(T^{-1}), \text{rank}(\mathcal{C})) \tag{67}$$
$$n = \min(n, \text{rank}(\mathcal{C})) \tag{68}$$

Notice that $\text{rank}(\mathcal{C}) \leq n$, so $\min(n, \text{rank}(\mathcal{C})) = \text{rank}(\mathcal{C})$ and we conclude that $\text{rank}(\mathcal{C}) = n$. Next, we need to prove that if the original system is controllable (i.e., $\text{rank}(\mathcal{C}) = n$), then T exists. We already know how to compute T, so we need to show that $\text{rank}(T) = n$ (which would make it a valid basis transformation matrix). We have that

$$\text{rank}(T) = \text{rank}(\mathcal{C}\mathcal{C}_{\text{CCF}}^{-1}) \tag{69}$$
$$\text{rank}(T) = \min(\text{rank}(\mathcal{C}), \text{rank}(\mathcal{C}_{\text{CCF}}^{-1})) \tag{70}$$
$$\text{rank}(T) = \min(n, n) = n \tag{71}$$

so $\text{rank}(T) = n$ and it is thus a valid transformation matrix.

(d) Consider the following discrete-time dynamics model:

$$\vec{x}[i+1] = \underbrace{\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}}_{A} \vec{x}[i] + \underbrace{\begin{bmatrix} 0 \\ 1 \end{bmatrix}}_{\vec{b}} \vec{u}[i] \tag{72}$$

Find the transformation matrix T such that the dynamics model for $\vec{z}[i] = T^{-1}\vec{x}[i]$ is in CCF. You may use a calculator/computer to perform any computations, if you wish.

(HINT: First, find the characteristic polynomial of A. Use this to determine what A_{CCF} and $\vec{b}_{\text{CCF}}$ should be, and then use this to determined $\mathcal{C}_{\text{CCF}}$.)

Solution: Firstly, we can compute $\mathcal{C}$ as follows:

$$\vec{b} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{73}$$
$$A\vec{b} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{74}$$

so $\mathcal{C} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$ which is full rank. Hence, the transformation matrix T will exist. Following the hint, the characteristic polynomial of A is

$$p_A(\lambda) = \det\{A - \lambda I\} \tag{75}$$
$$= \det\left\{\begin{bmatrix} 1-\lambda & 1 \\ 0 & 1-\lambda \end{bmatrix}\right\} \tag{76}$$

$$= (\lambda - 1)^2 \tag{77}$$
$$= \lambda^2 - 2\lambda + 1 \tag{78}$$

Here, we pattern match $a_2 = 2$ and $a_1 = -1$. Recall that our A matrix in CCF will be

$$A_{\text{CCF}} = \begin{bmatrix} 0 & 1 \\ a_1 & a_2 \end{bmatrix} \tag{79}$$
$$= \begin{bmatrix} 0 & 1 \\ -1 & 2 \end{bmatrix} \tag{80}$$

and

$$\vec{b}_{\text{CCF}} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{81}$$

by the definition of CCF. Hence, $\mathcal{C}_{\text{CCF}}$ can be computed as follows:

$$\vec{b}_{\text{CCF}} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{82}$$
$$A_{\text{CCF}}\vec{b}_{\text{CCF}} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \tag{83}$$

so $\mathcal{C}_{\text{CCF}} = \begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix}$. Thus,

$$T = \mathcal{C}\mathcal{C}_{\text{CCF}}^{-1} = \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \tag{84}$$

4. Gram-Schmidt Basics

(a) **Use Gram-Schmidt to find a matrix U whose columns form an orthonormal basis for the column space of V.**

$$V = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \tag{85}$$

Solution: We start with the columns of V as our basis for the column space of V, and we want to find an orthonormal basis for this same space using Gram-Schmidt. For notational convenience, define

$$V = \begin{bmatrix} | & | & | \\ \vec{v}_1 & \vec{v}_2 & \vec{v}_3 \\ | & | & | \end{bmatrix}, U = \begin{bmatrix} | & | & | \\ \vec{u}_1 & \vec{u}_2 & \vec{u}_3 \\ | & | & | \end{bmatrix} \tag{86}$$

We summarize the first few steps of the Gram-Schmidt algorithm as follows:

i. $\vec{u}_1' = \vec{v}_1; \qquad \vec{u}_1 = \frac{\vec{u}_1'}{\|\vec{u}_1'\|}.$

ii. $\vec{u}_2' = \vec{v}_2 - \langle \vec{v}_2, \vec{u}_1 \rangle \vec{u}_1; \qquad \vec{u}_2 = \frac{\vec{u}_2'}{\|\vec{u}_2'\|}.$

iii. $\vec{u}_3' = \vec{v}_3 - \langle \vec{v}_3, \vec{u}_1 \rangle \vec{u}_1 - \langle \vec{v}_3, \vec{u}_2 \rangle \vec{u}_2; \qquad \vec{u}_3 = \frac{\vec{u}_3'}{\|\vec{u}_3'\|}.$

For the column space of V, this is

i. $\vec{u}_1' = \vec{v}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \end{bmatrix}^\top$. Since $\vec{u}_1'$ is already normalized, we simply set $\vec{u}_1 = \vec{u}_1'$.

ii.

$$\vec{u}_2' = \vec{v}_2 - \langle \vec{v}_2, \vec{u}_1 \rangle \vec{u}_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} \implies \vec{u}_2 = \begin{bmatrix} 0 \\ \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \\ 0 \\ 0 \end{bmatrix} \tag{87}$$

iii.

$$\vec{u}_3' = \vec{v}_3 - \langle \vec{v}_3, \vec{u}_1 \rangle \vec{u}_1 - \langle \vec{v}_3, \vec{u}_2 \rangle \vec{u}_2 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} - \frac{2}{\sqrt{2}} \begin{bmatrix} 0 \\ \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} \implies \vec{u}_3 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} \tag{88}$$

Thus, the matrix U is given by

$$U = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} \\ 0 & 0 & \frac{1}{\sqrt{2}} \end{bmatrix}. \tag{89}$$

(b) Show that you get the same resulting vector when you project $\vec{w} = \begin{bmatrix} 1 \\ -1 \\ 0 \\ -1 \\ 0 \end{bmatrix}$ onto the columns of V as you do when you project onto the columns of U, i.e. **show that**

$$V(V^\top V)^{-1}V^\top \vec{w} = U(U^\top U)^{-1}U^\top \vec{w}. \tag{90}$$

Feel free to use `NumPy` for the projection onto the columns of V, but compute the projection onto the columns of U by hand. Comment on whether projecting upon the V or U basis is computationally more efficient. *(HINT: Which of these matrices allow us to circumvent the matrix inversion in the projection formula?)*

Solution: Note that whatever basis we use for a subspace, when we project a vector onto that subspace, we get the same vector. For example, when we project the vector $\vec{w} = \begin{bmatrix} 1 & -1 & 0 & -1 & 0 \end{bmatrix}^\top$ onto the subspace using the V basis, we get

$$V(V^\top V)^{-1}V^\top \vec{w} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 3 & 3 \\ 1 & 3 & 5 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \\ 0 \\ -1 \\ 0 \end{bmatrix} \tag{91}$$

$$= \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{3}{2} & -\frac{1}{2} & 0 \\ -\frac{1}{2} & 1 & -\frac{1}{2} \\ 0 & -\frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} \tag{92}$$

$$= \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{3}{2} \\ 0 \\ -\frac{1}{2} \end{bmatrix} = \begin{bmatrix} 1 \\ -\frac{1}{2} \\ -\frac{1}{2} \\ -\frac{1}{2} \\ -\frac{1}{2} \end{bmatrix} \tag{93}$$

$$U(U^\top U)^{-1}U^\top \vec{w} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} \\ 0 & 0 & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 \\ -1 \\ 0 \\ -1 \\ 0 \end{bmatrix} \tag{94}$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} \\ 0 & 0 & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 \\ -\frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} 1 \\ -\frac{1}{2} \\ -\frac{1}{2} \\ -\frac{1}{2} \\ -\frac{1}{2} \end{bmatrix} \tag{95}$$

Note however that the projection using the U basis was much simpler. Since $U^\top U$ is the identity, we didn't need to do a matrix inversion.

Contributors:

- Bozhi Yin.
- Kaitlyn Chan.
- Yi-Hsuan Shih.
- Vladimir Stojanovic.
- Moses Won.
- Druv Pai.
- Anish Muthali.