Homework 7

This homework is due on Sunday, August 7th at 11:59 pm PT.

1. Administrivia

The Summer 2022 EECS 16B final is on Thursday, August 11, 2022, from 6-9 pm PDT. Please fill out this survey link, so we can understand your preferences and plan accordingly. **To get credit for this problem part, please attach a screenshot indicating that you have completed the survey.**

2. Inverse Kinematics

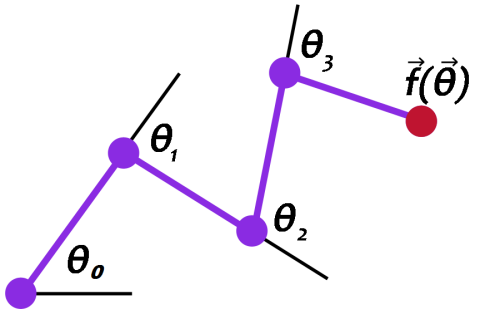

Figure 1: An example of an arm parameterized by θ_0, θ_1, θ_2, and θ_3 with the end effector at point $\vec{f}(\vec{\theta})$.

Suppose you have a robotic arm composed of several rotating joints. The lengths r_i of the arm are fixed, but you can control the arm by specifying the amount of rotation θ_i for each joint. If we have an arm with four joints, it can be parameterized by:

$$\vec{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}. \tag{1}$$

Suppose further that we have some target 2D point $\vec{t} \in \mathbb{R}^2$, and we would like for the end of the arm, called the end effector, to reach for the target. From physics and kinematics, we can find the function $\vec{f}(\vec{\theta})$ that given the angles of each joint can return the position of the end effector. Figure 1 shows a visualization of an arm rotated by $\vec{\theta}$. To make the arm reach for the target $\vec{t}$, we want to find where the function $\vec{g}$ defined as

$$\vec{g}(\vec{\theta}) = \vec{f}(\vec{\theta}) - \vec{t} \tag{2}$$

is equal to $\vec{0}$.

Note that this would be simple to do if $\vec{f}$ had an inverse. However due to physics and rotations, many sines and cosines appear in the forward kinematics and $\vec{f}$ becomes highly nonlinear. **Inverse kinematics is the problem of given this point in space that we want to reach, what should we set the joint angles of our arm to?**

To accomplish this, we use the spirit of Newton's method for solving potentially nonlinear equations.

1-D Case (for the purpose of intuition):

In the 1-D case, you have a real scalar function g of a single parameter θ and we want to find a $\widehat{\theta}$ so that $g(\widehat{\theta}) = 0$.

Step i of Newton's method does the following, where our current estimate of $\widehat{\theta}$ is $\theta^{(i)}$:

1. Linearize g around $\theta^{(i)}$ to get an approximation $\widetilde{g}$:

$$\widetilde{g}(\theta) = g(\theta^{(i)}) + g'(\theta^{(i)})(\theta - \theta^{(i)}) \tag{3}$$

2. We want to find the roots of g, and will get closer by finding the roots of this linear approximation $\widetilde{g}$. Thus we want to set $\widetilde{g}(\theta) = 0$ to get

$$0 = g(\theta^{(i)}) + g'(\theta^{(i)})(\theta - \theta^{(i)}) \tag{4}$$

$$\theta = \theta^{(i)} - \frac{g(\theta^{(i)})}{g'(\theta^{(i)})} \tag{5}$$

3. Therefore for the next iteration, we set $\theta^{(i+1)} = \theta^{(i)} - \frac{g(\theta^{(i)})}{g'(\theta^{(i)})}$, and repeat.

We will iterate this until $g(\theta)$ is close enough to 0 for our application. In practice, instead of solving exactly for $g(\theta) = 0$, in the second step of iteration i, we may chose to move $\theta^{(i)}$ by a fixed step-size η in the direction that the first-order approximation to the function suggests, but not all the way. This is done because the derivative $g'(\theta^{(i)})$ might be very different from $g'(\theta^{(i+1)})$. After all, linearization is only valid in a local neighborhood.

While you might have seen Newton's method as described above in your calculus courses, you might not have seen the vector-generalization of it. It follows exactly the same spirit.

Vector-generalization of Newton's method

The first-order approximation to the vector valued function $\vec{g}(\vec{\theta})$ at $\vec{\theta}^{(i)}$ is now $\vec{g}(\vec{\theta}^{(i)}) + J_{\vec{g}}(\vec{\theta}^{(i)})(\vec{\theta} - \vec{\theta}^{(i)})$ where $J_{\vec{g}}(\vec{\theta})$ is the Jacobian matrix of the function $\vec{g}(\vec{\theta})$. For this problem, we will be using a robotic arm with 4 joints in a 2-dimensional space. Therefore, the Jacobian of $\vec{g}(\vec{\theta})$ will be a 2x4 matrix, and it is computed by calculating the partial derivatives of $\vec{g}(\vec{\theta})$:

$$J_{\vec{g}} = \begin{bmatrix} \frac{\partial g_x(\vec{\theta})}{\partial \theta_1} & \frac{\partial g_x(\vec{\theta})}{\partial \theta_2} & \frac{\partial g_x(\vec{\theta})}{\partial \theta_3} & \frac{\partial g_x(\vec{\theta})}{\partial \theta_4} \\ \frac{\partial g_y(\vec{\theta})}{\partial \theta_1} & \frac{\partial g_y(\vec{\theta})}{\partial \theta_2} & \frac{\partial g_y(\vec{\theta})}{\partial \theta_3} & \frac{\partial g_y(\vec{\theta})}{\partial \theta_4} \end{bmatrix}. \tag{6}$$

In this notation, we use $\vec{g}(\vec{\theta}) = [g_x(\vec{\theta}) \quad g_y(\vec{\theta})]^T$ where $g_x(\vec{\theta})$ is the x coordinate of the end effector and $g_y(\vec{\theta})$ is the y coordinate in our 2D space.

The Newton algorithm in this case is an iterative method that gives us successively better estimates for our vector $\vec{\theta}$. If we start with some guess $\vec{\theta}^{(i)}$, then the next guess is given by

$$\vec{\theta}^{(i+1)} = \vec{\theta}^{(i)} - \eta J^{\dagger}_{\vec{g}}(\vec{\theta}^{(i)})\vec{g}(\vec{\theta}^{(i)}) \tag{7}$$

where η is adjusted to determine how large of a step we make between $\vec{\theta}^{(i)}$ and $\vec{\theta}^{(i+1)}$. Notice that we need to invert the Jacobian matrix of first-partial-derivatives, and this matrix is not square. It is in fact a wide matrix. Fortunately, we now know how to "invert" any matrix, using the Moore-Penrose pseudoinverse that you saw in a previous homework.

The following problem will guide you step-by-step through the implementation of the pseudoinverse. The three steps of the pseudoinverse algorithm are:

1. First, compute the compact-form SVD of the input matrix.
2. Next, we compute Σ^{-1} by inverting each nonzero singular value σ_i. We assume that any singular value less than some ϵ is the same as 0, since inverting small values will cause numerical issues.
3. Finally, we compute the pseudoinverse by multiplying the matrices together in the right order.

The next parts guide you through the code to be written for the pseudoinverse.

(a) In the `pseudoinverse` function, **compute the SVD of the input matrix A by using the appropriate Numpy function.**

(HINT: It is useful to read the documentation for the Numpy functions involved so that you use them correctly. For example, for this problem you want the compact SVD so what argument should you call `svd` with? What exactly does the SVD function return in Numpy?)

(b) To save memory space, the NumPy algorithm returns the matrix Σ as a one-dimensional array of the singular values. Use this vector to **compute the diagonal entries of Σ^{-1}**. To be careful of numerical issues, first threshold the singular values, and only invert the singular values above a certain value ϵ, considering smaller ones as 0.

The reason is that you don't want to have very big entries in the pseudoinverse because that will defeat the point of you using a small step-size η to stay within the rough neighborhood that your linear approximation is valid. Considering small singular values as being 0 stops this from happening.

(c) We now have all of the parts to compute the relevant pseudoinverse of A. **Finish the last line of the function to calculate the pseudoinverse.**

(HINT: `np.diag` *can be a very useful tool in converting a vector into a square diagonal matrix. Also remember to use* `.T` *to get transposes.)*

(d) There are three test cases that you can use to determine if your pseudoinverse function works correctly. In the first case, the arm is able to reach the target, and the end of the arm will be touching the target. In the second case, the arm should be pointing in a straight line towards the blue circle. The last case is the same as the second with the addition that a singular value will be very close to zero to test your pseudoinverse function's ability to handle small singular values.

There is also an animated test case that will move the target in and out of the reach of the arm. Verify that the arm follows the target correctly and points towards the target when it is out of reach.

Finally, there is a test case where you can drag the target position and the robot joints will update automatically as you track the target. You can also click anywhere on the plot and the robot will attempt to reach it.

Describe what you see happening in the animation as well as the plot where you drag the target position.

3. Linearizing for understanding amplification

Linearization isn't just something that is important for control, robotics, machine learning, and optimization — it is one of the standard tools used across different areas, including circuits.

The circuit below is a voltage amplifier, where the element inside the box is a bipolar junction transistor (BJT). You do not need to know what a BJT is to do this question.

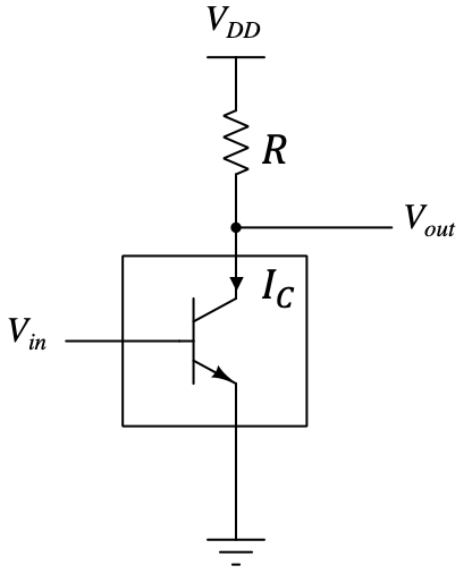

Figure 2: Voltage amplifier circuit using a BJT

The BJT in the circuit can be modeled quite accurately as a nonlinear, voltage-controlled current source, where the collector current I_C is given by:

$$I_C(V_{\text{in}}) = I_S \cdot e^{\frac{V_{\text{in}}}{V_{\text{TH}}}}, \tag{8}$$

where V_{TH} is the thermal voltage. We can assume $V_{\text{TH}} = 26\,\text{mV}$ at room temperature. I_S is a constant whose exact value we are not giving you because we want you to find ways of eliminating it in favor of other quantities whenever possible.

The goal of this circuit is to pick a particular point $(V_{\text{in}}^\star, V_{\text{out}}^\star)$ so that any small variation δV_{in} in the input voltage V_{in} can be amplified to a relatively larger variation δV_{out} in the output voltage V_{out}. In other words, if $V_{\text{in}} = V_{\text{in}}^\star + \delta V_{\text{in}}$ and $V_{\text{out}} = V_{\text{out}}^\star + \delta V_{\text{out}}$, then we want the magnitude of the 'amplification gain' given by $\left|\frac{\delta V_{\text{out}}}{\delta V_{\text{in}}}\right|$ to be large. We're going to investigate this amplification using linearization.

(NOTE: in this problem, δV is single variable indicating a small variation in V, not $\delta \times V$.)

(a) **Write a symbolic expression for V_{out} as a function of I_C, V_{DD} and R in Fig 2.**

(b) Now let's linearize I_C in the neighborhood of an input voltage $V_{\text{in}}^\star$ and a specific $I_C^\star$. Assume that you have a found a particular pair of input voltage $V_{\text{in}}^\star$ and current $I_C^\star$ that satisfy the current equation (8).

We can look at nearby input voltages and see how much the current changes. We can write the linearized expression for the collector current around this point as:

$$I_C(V_{\text{in}}) = I_C(V_{\text{in}}^\star) + g_m(V_{\text{in}} - V_{\text{in}}^\star) = I_C^\star + g_m\,\delta V_{\text{in}} \tag{9}$$

where $\delta V_{\text{in}} = V_{\text{in}} - V_{\text{in}}^\star$ is the change in input voltage, and g_m is the slope of the local linearization around $(V_{\text{in}}^\star, I_C^\star)$. **What is g_m here as a function of $I_C^\star$ and V_{TH}?**

If you take EE105, you will learn that this g_m is called "transconductance" and is an important parameter in analog circuit design.

(HINT: Find g_m by taking the appropriate derivative around the operating point. You should recognize a part of your equation is equal to the current operating point $I_C^\star = I_C(V_{\text{in}}^\star)$, so your final form should not depend on I_S. Also, note that in circuits terminology, "operating point" is defined to be the point around which we linearize input-output relationship.)

(c) We now have a linear relationship between small changes in current and voltage, $\delta I_C = g_m\,\delta V_{\text{in}}$ around a known solution $(V_{\text{in}}^\star, I_C^\star)$.

As a reminder, the goal of this problem is to pick a particular point $(V_{\text{in}}^\star, V_{\text{out}}^\star)$ so that any small variation δV_{in} in the input voltage V_{in} can be amplified to a relatively larger variation δV_{out} in the output voltage V_{out}. In other words, if $V_{\text{in}} = V_{\text{in}}^\star + \delta V_{\text{in}}$ and $V_{\text{out}} = V_{\text{out}}^\star + \delta V_{\text{out}}$, then we want the magnitude of the "amplification gain" given by $\left|\frac{\delta V_{\text{out}}}{\delta V_{\text{in}}}\right|$ to be large.

Plug in your linearized equation for I_C in the answer from part (a). It may help to define the output voltage operating point as $V_{\text{out}}^\star$, where

$$V_{\text{out}}^\star = V_{\text{DD}} - RI_C^\star$$

so that we can view $V_{\text{out}} = V_{\text{out}}^\star + \delta V_{\text{out}}$ when we have $V_{\text{in}} = V_{\text{in}}^\star + \delta V_{\text{in}}$.

Find the linearized relationship between δV_{out} and δV_{in}. The ratio $\frac{\delta V_{\text{out}}}{\delta V_{\text{in}}}$ is called the "small-signal voltage gain" of this amplifier around this operating point.

(d) Assuming that $V_{\text{DD}} = 10\,\text{V}$, $R = 1\,\text{k}\Omega$, and $I_C^\star = 1\,\text{mA}$ when $V_{\text{in}}^\star = 0.65\,\text{V}$, **verify that the magnitude of the small-signal voltage gain $\left|\frac{\delta V_{\text{out}}}{\delta V_{\text{in}}}\right|$ is approximately** 38**.** *(HINT: Remember $V_{\text{TH}} = 26\,\text{mV}$.)*

(e) If $I_C^\star = 9\,\text{mA}$ when $V_{\text{in}}^\star = 0.7\,\text{V}$ with all other parameters remaining fixed, **verify that the magnitude of the small-signal voltage gain $\left|\frac{\delta V_{\text{out}}}{\delta V_{\text{in}}}\right|$ between the input and the output around this operating point is approximately** 346**.**

(f) If you wished to make an amplifier with as large of a small signal gain as possible, **which operating (bias) point would you choose among $V_{\text{in}}^\star = 0.65\,\text{V}$ (part d) and $V_{\text{in}}^\star = 0.7\,\text{V}$ (part e)?**

This shows you that by appropriately biasing (choosing an operating point), we can adjust what our gain is for small signals. While we just wanted to show you a simple application of linearization here, these ideas are developed a lot further in EE105, EE140, and other courses to create things like op-amps and other analog information-processing systems. Simple voltage amplifier circuits like these

are used in everyday circuits like the sensors in your smartwatch, wireless transceivers in your phone, and communication circuits in CPUs and GPUs.

4. Tracking a Desired Trajectory in Continuous Time

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

In this question, we will also see that everything that you have learned to do closed-loop control in discrete-time can also be used to do closed-loop control in continuous time.

Consider the specific 2-dimensional system

$$\frac{\mathrm{d}}{\mathrm{d}t}\vec{y}(t) = A\vec{y}(t) + \vec{b}u(t) + \vec{w}(t) = \begin{bmatrix} 2 & 1 \\ 0 & 2 \end{bmatrix} \vec{y}(t) + \begin{bmatrix} 1 \\ 1 \end{bmatrix} u(t) + \vec{w}(t) \tag{10}$$

where $u(t)$ is a scalar valued continuous control input and $\vec{w}(t)$ is a bounded disturbance (noise).

(a) In an ideal noiseless scenario, the desired control signal $u^*(t)$ makes the system follow the desired trajectory $\vec{y^*}(t)$ that satisfies the following dynamics:

$$\frac{\mathrm{d}}{\mathrm{d}t}\vec{y^*}(t) = A\vec{y^*}(t) + \vec{b}u^*(t). \tag{11}$$

The presence of the bounded noise term $\vec{w}(t)$ makes the actual state $\vec{y}(t)$ deviate from the desired $\vec{y^*}(t)$ and follow (10) instead. In the following subparts, we will analyze how we can adjust the desired control signal $u^*(t)$ in (11) to the control input $u(t)$ in (10) so that the deviation in the state caused by $\vec{w}(t)$ remains bounded.

Represent the state as $\vec{y}(t) = \vec{y^*}(t) + \Delta\vec{y}(t)$ and $u(t) = u^*(t) + \Delta u(t)$. Using (10) and (11), **show that we can represent the evolution of the trajectory deviation $\Delta\vec{y}(t)$ as a function of the control deviation $\Delta u(t)$ and the bounded disturbance $\vec{w}(t)$ as:**

$$\frac{\mathrm{d}}{\mathrm{d}t}\Delta\vec{y}(t) = A\Delta\vec{y}(t) + \vec{b}\Delta u(t) + \vec{w}(t). \tag{12}$$

(HINT: Write out equation (10) in terms of $\vec{y^*}(t)$, $\Delta\vec{y}(t)$, $u^*(t)$ and $\Delta u(t)$.))

(b) **Are the dynamics that you found for $\Delta\vec{y}(t)$ in part 4.a stable? Based on this, in the presence of bounded disturbance $\vec{w}(t)$, will $\vec{y}(t)$ in (10) follow the desired trajectory $\vec{y^*}(t)$ closely if we just apply the control $u(t) = u^*(t)$ to the original system in (10), i.e. $\Delta u(t) = 0$?**
(HINT: Use the numerical values of A and $\vec{b}$ from (10) in the solution from part (b) to determine stability of $\Delta\vec{y}(t)$.)

Now, we want to apply state feedback control to the system using $\Delta u(t)$ to get our system to follow the desired trajectory $\vec{y^*}(t)$.

(c) For the $\Delta\vec{y}(t), \Delta u(t)$ system, **apply feedback control by letting $\Delta u(t) = K\Delta\vec{y}(t) = \begin{bmatrix} k_0 & k_1 \end{bmatrix} \Delta\vec{y}(t)$ that would place both the eigenvalues of the closed-loop $\Delta\vec{y}(t)$ system at -10. Find k_0 and k_1.**

(d) Based on what you did in the previous parts, and given access to the desired trajectory $\vec{y}^*(t)$, the desired control $u^*(t)$, and the actual measurement of the state $\vec{y}(t)$, **come up with a way to do feedback control that will keep the trajectory staying close to the desired trajectory no matter what the small bounded disturbance $\vec{w}(t)$ does.** *(HINT: Express the control input $u(t)$ in terms of $u^*(t)$, $\vec{y}^*(t)$, and $\vec{y}(t)$.))*

5. Extending Orthonormality to Complex Vectors

So far in the course, we have only dealt with real vectors. However, it is often useful to also think about complex vectors, as it allows for useful signal processing tools like the Discrete Fourier Transform (DFT) which you will learn in later courses. In this problem, we will extend several important properties of orthonormal matrices to the complex case. You are already introduced to complex inner products in the lectures (refer to Note 2j.)

(a) To get some practice computing complex inner products, **what are the complex inner products** $\left\langle \begin{bmatrix} 1+\mathrm{j} \\ 2 \end{bmatrix}, \begin{bmatrix} -3-\mathrm{j} \\ 2+\mathrm{j} \end{bmatrix} \right\rangle$ **and** $\left\langle \begin{bmatrix} -3-\mathrm{j} \\ 2+\mathrm{j} \end{bmatrix}, \begin{bmatrix} 1+\mathrm{j} \\ 2 \end{bmatrix} \right\rangle$**? Does the order of the vectors in the complex inner product matter i.e. is it commutative?**

(b) Let $Q = \begin{bmatrix} | & & | \\ \vec{q}_1 & \cdots & \vec{q}_n \\ | & & | \end{bmatrix}$ be an n by n *unitary* matrix, whose columns $\vec{q}_1, \vec{q}_2, \ldots, \vec{q}_n$ are orthonormal, i.e.

$$\langle \vec{q}_j, \vec{q}_i \rangle = \vec{q}_i^* \vec{q}_j = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \tag{13}$$

Show that $Q^{-1} = Q^*$, where Q^* is the conjugate transpose of Q.

(c) **Show that Q as defined in (b) preserves complex inner products, i.e. if $\vec{v}, \vec{w} \in \mathbb{C}^n$ are vectors of length n, then**

$$\langle \vec{v}, \vec{w} \rangle = \langle Q\vec{v}, Q\vec{w} \rangle . \tag{14}$$

(HINT: Note that $(AB)^* = B^* A^*$.)

(d) **Show that if $\vec{q}_1, \ldots, \vec{q}_n$ are the columns of a unitary matrix Q, they must be linearly independent.**

(HINT: Suppose $\vec{w} = \sum_{i=1}^n \alpha_i \vec{q}_i$, then first show that $\alpha_i = \langle \vec{w}, \vec{q}_i \rangle$. From here ask yourself whether a nonzero linear combination of the $\vec{q}_i$ could ever be identically zero.)

This fact shows how orthogonality is a very nice special case of linear independence.

(e) Now let V be another unitary matrix. **Show that QV is also an unitary matrix.**

6. Gram-Schmidt on Complex Vectors

Consider the three complex vectors

$$\vec{a}_1 = \begin{bmatrix} 1 \\ j \\ 0 \end{bmatrix}, \quad \vec{a}_2 = \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix}, \quad \vec{a}_3 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}. \tag{15}$$

Compute an orthonormal basis from this list of vectors with complex Gram-Schmidt.

(HINT: The complex version of Gram-Schmidt is Algorithm 45 of Note 2j.)

7. Adapting Proofs to the Complex Case

At many points in the course, we have made assumptions that various matrices or eigenvalues are real while discussing various theorems. If you have noticed, this has always happened in contexts where we have invoked orthonormality during the proof or statement of the result. Now that you understand the idea of orthonormality for complex vectors, and how to adapt Gram-Schmidt to complex vectors, you can go back and remove those restrictions. This problem asks you to do exactly that.

(a) The upper-triangularization theorem (Theorem 63 of Note 2j) says that every complex matrix $A \in \mathbb{C}^{n \times n}$ can be written as $A = UTU^*$, where $U \in \mathbb{C}^{n \times n}$ is unitary and $T \in \mathbb{C}^{n \times n}$ is upper triangular.

Adapt the proof from the real case with assumed real eigenvalues to prove this theorem.

Feel free to assume that any square matrix has an (potentially complex) eigenvalue/eigenvector pair. You don't need to prove this. But you can make no other assumptions.

(HINT: Use the exact same argument as before, just use conjugate-transposes instead of transposes.)

(b) The spectral theorem (Theorem 58 of Note 2j) for Hermitian matrices says that a Hermitian matrix $A \in \mathbb{C}^{n \times n}$ can be written as $A = U\Lambda U^*$ where U is a unitary matrix of eigenvectors of A, and Λ is a real and diagonal matrix of corresponding eigenvalues of A.

Adapt the proof from the real symmetric case to prove this theorem.

(HINT: As before, you should just leverage upper-triangularization and use the fact that $(ABC)^* = C^*B^*A^*$. There is a reason that this part comes after the first part.)

8. (PRACTICE) Linearization of a scalar system

In this question, we linearize the scalar differential equation

$$\frac{\mathrm{d}}{\mathrm{d}t}x(t) = \sin(x(t)) + u(t) \tag{16}$$

around equilibria, discretize it, and apply feedback control to stabilize the resulting system.

(a) The first step is to find the equilibria that we will linearize around. Recall that equilibria are the values of (x, u) such that $\frac{\mathrm{d}}{\mathrm{d}t}x(t) = 0$. Suppose we want to linearize around equilibria $(x^\star, u^\star)$ where $u^\star = 0$. **Sketch** $\sin(x)$ **for** $-4\pi \leq x \leq 4\pi$ **and intersect it with the horizontal line at** 0**.** This will show us the equilibrium points where $0 = \sin(x^\star) + u^\star = \sin(x^\star)$.

(b) From part (a), we can see, graphically, that the equilibria of system (16) where $u^\star = 0$ are $x_m^\star = m\pi$, for all $m \in \mathbb{Z}$. **Show that** $x_m^\star = m\pi$ **and** $u^\star = 0$ **are equilibria of system** (16)**.**

We will linearize around $x_{-1}^\star = -\pi$ and $x_0^\star = 0$. Looking at the sketch we made, these seem representative of the two types of equilibria where $u^\star = 0$.

(c) Linearize system (16) around the equilibrium $(x_0^\star, u^\star) = (0, 0)$. **What is the resulting linearized scalar differential equation for** $\delta x(t) = x(t) - x_0^\star = x(t) - 0$**, involving** $\delta u(t) = u(t) - u^\star = u(t) - 0$**?**

Now that we have an approximate linear system, we discretize time to intervals of Δ. We also approximate the input $u(t)$ as piecewise constant, i.e. $\delta u(t) = \delta u[i]$, in the time interval $t \in [i\Delta, (i+1)\Delta)$, where Δ is small relative to how fast the control input changes.
From Discussion 2A, we got that such a discretization of

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = \lambda x(t) + bu(t) \tag{17}$$

into intervals of Δ gives the discrete-time system

$$x[i+1] = e^{\lambda\Delta}x[i] + \frac{b(e^{\lambda\Delta} - 1)}{\lambda}u[i]. \tag{18}$$

Using this result on our approximate linear system via pattern-matching (verify this!) gives

$$\delta x[i+1] = e^{\Delta}\delta x[i] + \delta u[i](e^{\Delta} - 1). \tag{19}$$

(d) **Is the (approximate) discrete-time system** (19) **stable?**

(e) Now linearize the system (16) around the equilibrium $(x_{-1}^\star, u^\star) = (-\pi, 0)$. **What is the resulting scalar differential equation for** $\delta x(t) = x(t) - (-\pi)$ **involving** $\delta u(t) = u(t) - 0$**?**

We again discretize the approximate linear system obtained in (e). Pattern-matching with (17) and using (18) (verify this!), we get

$$\delta x[i+1] = e^{-\Delta}\delta x[i] + \delta u[i](1 - e^{-\Delta}). \tag{20}$$

(f) **Is the (approximate) discrete-time system** (20) **stable?**

(g) Suppose for the two linearized discrete-time systems (19) and (20), we apply the feedback law

$$\delta u[i] = -k(\delta x[i] - x^\star).$$

For what range of k values would the resulting linearized discrete-time systems be stable? Your answer will depend on Δ.

9. **Homework Process, Study Group, and Course Weekly Survey**

   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.

   At the same time, we want to check-in weekly regarding Discussions, Lectures, Lab, and Office Hours and see how effective they have all been for you as students.

   Please fill out this survey link. For your submission, please attach a screenshot indicating that you have completed the survey this week.

Contributors:

- Anirudh Rengarajan.
- Stephen Bailey.
- Ashwin Vangipuram.
- Anant Sahai.
- Kris Pister.
- Alex Devonport.
- Regina Eckert.
- Wahid Rahman.
- Ming Jin.
- Ayan Biswas.
- Tanmay Gautam.
- Nathan Lambert.
- Druv Pai.
- Sally Hui.