EECS 16B Designing Information Systems and Devices II UC Berkeley Spring 2022

Homework 12

This homework is due on Sunday, April 24, 2022 at 11:59PM. Self-grades and HW Resubmissions are due the following Sunday, May 1, 2022 at 11:59PM.

1. Reading Lecture Notes

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 18

(a) We know that a scalar function $f(x)$ can be linearly approximated around a particular point $x = x^\star$ using Taylor's series expansion as follows:

$$f(x) \approx f(x^\star) + \frac{\mathrm{d}f}{\mathrm{d}x}(x^\star) \cdot (x - x^\star)$$

What is the equivalent linear approximation of a multivariate scalar function $f(x, u)$ around a particular expansion point $(x^\star, u^\star)$?

Solution: The linear approximation of $f(x, u)$ around the expansion point $(x^\star, u^\star)$ is given by

$$f(x, u) \approx f(x^\star, u^\star) + \frac{\partial f}{\partial x}(x^\star, u^\star) \cdot (x - x^\star) + \frac{\partial f}{\partial u}(x^\star, u^\star) \cdot (u - u^\star)$$

(b) Now assume we have a vector valued function given by

$$\vec{f}(\vec{x}, \vec{u}) = \begin{bmatrix} f_1(\vec{x}, \vec{u}) \\ f_2(\vec{x}, \vec{u}) \\ \vdots \\ f_n(\vec{x}, \vec{u}) \end{bmatrix}$$

Let the state $\vec{x}$ be n dimensional, and control $\vec{u}$ be k dimensional. **What is the linear approximation of the function $\vec{f}(\vec{x}, \vec{u})$ around a particular expansion point $(\vec{x}^\star, \vec{u}^\star)$?**

Solution: The linear approximation of $\vec{f}(\vec{x}, \vec{u})$ around the expansion point $(\vec{x}^\star, \vec{u}^\star)$ is given by

$$\vec{f}(\vec{x}, \vec{u}) \approx \vec{f}(\vec{x}^\star, \vec{u}^\star) + \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(\vec{x}^\star, \vec{u}^\star) & \dots & \frac{\partial f_1}{\partial x_n}(\vec{x}^\star, \vec{u}^\star) \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1}(\vec{x}^\star, \vec{u}^\star) & \dots & \frac{\partial f_n}{\partial x_n}(\vec{x}^\star, \vec{u}^\star) \end{bmatrix} \cdot (\vec{x} - \vec{x}^\star)$$
$$+ \begin{bmatrix} \frac{\partial f_1}{\partial u_1}(\vec{x}^\star, \vec{u}^\star) & \dots & \frac{\partial f_1}{\partial u_k}(\vec{x}^\star, \vec{u}^\star) \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial u_1}(\vec{x}^\star, \vec{u}^\star) & \dots & \frac{\partial f_n}{\partial u_k}(\vec{x}^\star, \vec{u}^\star) \end{bmatrix} \cdot (\vec{u} - \vec{u}^\star).$$

2. Inverse Kinematics

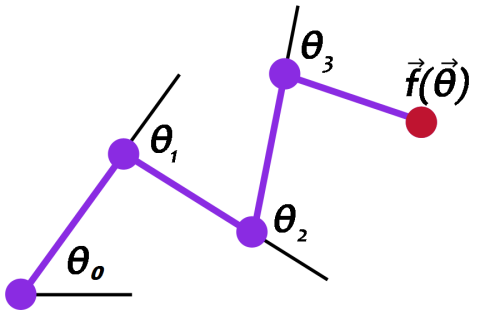

Figure 1: An example of an arm parameterized by θ_0, θ_1, θ_2, and θ_3 with the end effector at point $\vec{f}(\vec{\theta})$.

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. 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

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

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

© UCB EECS 16B, Spring 2022. All Rights Reserved. This may not be publicly shared without explicit permission.

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.

2. 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

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

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.

3. 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?)

Solution: See the IPython notebook for the solution.

The challenge here was basically in understanding the Numpy documentation about what the SVD function actually does. The fact that it has an optional argument that defaults to something other than what you want (i.e. it defaults to the full SVD as opposed to the compact form that

© UCB EECS 16B, Spring 2022. All Rights Reserved. This may not be publicly shared without explicit permission.

you prefer for this problem, and basically would prefer anytime the nullspace is not interesting to you) adds complexity, but this is something that you always have to watch out for when using libraries.

(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.

Solution: See the IPython notebook for the solution.

You might have wondered how you can systematically pick the value for the threshold ϵ and the answer here is that there is no such systematic approach. We had to guess something. The principled reason you need to pick a threshold is to prevent yourself from moving too far away after an update.

Now, our provided code was actually somewhat intelligent (i.e. adaptive) in picking this update's step-size η. It would shrink the step-size whenever it saw something odd. (This is a cheap variant of something called "line-search" — it is possible to do something more sophisticated, but it won't help in a big way for this problem.) And so, the real issue here was in avoiding a bad interaction with our adaptive η code.

(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.)*

Solution: See the IPython notebook for the solution.

In the real world, so-called "hyperparameters" like ϵ are set by more sophisticated variants of "trial and error." Basically, you try a value and see if it works for some of the cases that you want to make sure that it works for. If it doesn't, you adjust it. You keep trying. Once you've settled on something as being good enough, you just need to check to see whether you were deluded. To test that, you use another fresh and different set of test cases to make sure that things still work. In this problem, we thought of the animation as being like that final set of test cases.

(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.

Solution: You should observe the arm getting as close to the blue dot as possible in a smooth manner. You might also see it jump sometimes from one position to another. In these cases, the pseudoinverse version of Newton's Method fails to find a way to decrease the distance to the target. This shows one flaw in our method, and why much more advanced methods of inverse kinematics that take the nonlinear system of equations into account are used in practice. To learn more about this, take EECS 106A.

© UCB EECS 16B, Spring 2022. All Rights Reserved. This may not be publicly shared without explicit permission.

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.

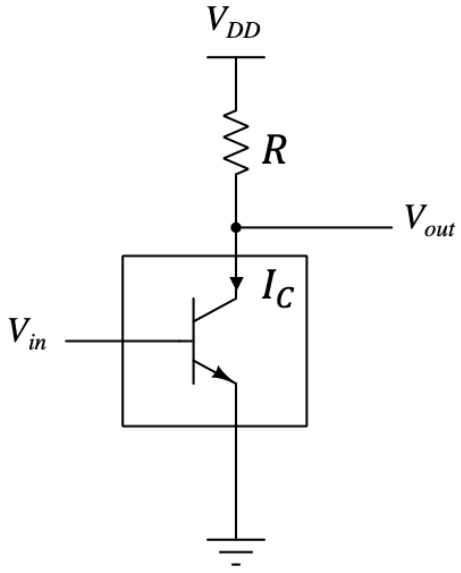

Figure 2: Voltage amplifier circuit using a BJT

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.**
Solution:

$$V_{\text{out}} = V_{\text{DD}} - RI_C \tag{9}$$

since we have a voltage drop of $I_C R$ across the resistor and the top voltage is V_{DD}.

(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:

© UCB EECS 16B, Spring 2022. All Rights Reserved. This may not be publicly shared without explicit permission.

$$I_C(V_{\text{in}}) = I_C(V_{\text{in}}^\star) + g_m(V_{\text{in}} - V_{\text{in}}^\star) = I_C^\star + g_m\,\delta V_{\text{in}} \tag{10}$$

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.)
Solution: We start out by writing out the linearization form that we are looking for.

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

Now, taking the first derivative of I_C around $V_{\text{in}}^\star$:

$$g_m = \frac{\mathrm{d}I_C(V_{\text{in}}^\star)}{\mathrm{d}V_{\text{in}}} \tag{11}$$

$$= \frac{1}{V_{\text{TH}}} I_S e^{\frac{V_{\text{in}}^\star}{V_{\text{TH}}}} \tag{12}$$

$$= \frac{I_C^\star}{V_{\text{TH}}} \tag{13}$$

where in the last line, we recognize that $I_C^\star = I_S e^{\frac{V_{\text{in}}^\star}{V_{\text{TH}}}}$, and therefore knowledge of I_S is not required to determine g_m if $I_C^\star$ and V_{TH} are known.

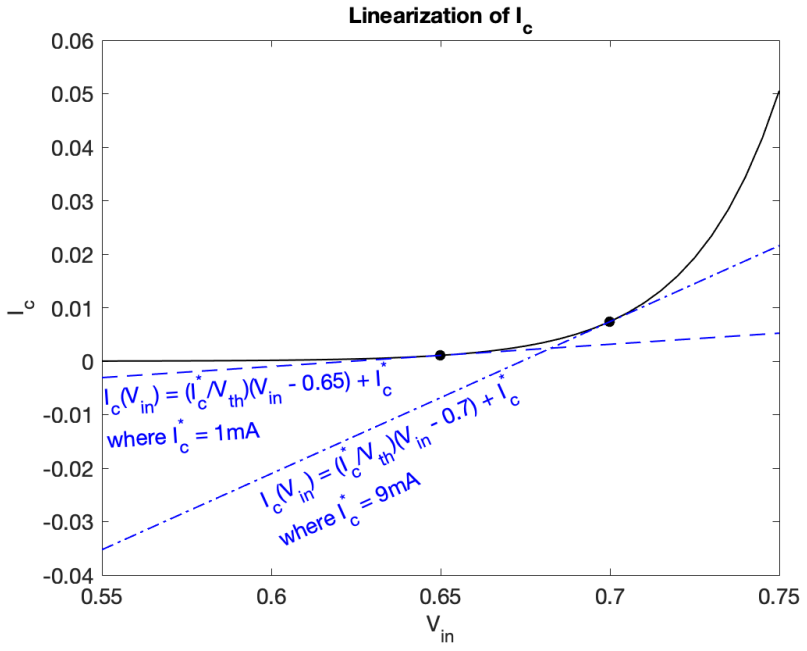

Figure 3: Linearization of the non linear I_C (black curve)

For understanding this linearization graphically, we can choose to plot I_C vs. V_{in} and look at how the slope (i.e. g_m) changes with different values of $(V_{\text{in}}^\star, I_C^\star)$. In Fig. 3, we see the linearizations around $V_{\text{in}}^\star = 0.65\,\text{V}$ ($I_C^\star = 1\,\text{mA}$) and $V_{\text{in}}^\star = 0.7\,\text{V}$ ($I_C^\star = 9\,\text{mA}$) given in parts (d) and (e) below.

© UCB EECS 16B, Spring 2022. All Rights Reserved. This may not be publicly shared without explicit permission.

(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.

Solution: We have two equations for V_{out}:

$$V_{\text{out}} = V_{\text{out}}^\star + \delta V_{\text{out}} \tag{14}$$

and

$$V_{\text{out}} = V_{\text{DD}} - RI_C \tag{15}$$

We know from equation (10) that $I_C = I_C^\star + g_m \delta V_{\text{in}}$, so we can re-write the above two equations as:

$$V_{\text{out}}^\star + \delta V_{\text{out}} = V_{\text{DD}} - R(I_C^\star + g_m \delta V_{\text{in}}) \tag{16}$$

We also know the output voltage operating point $V_{\text{out}}^\star$ is related to the current operating point $I_C^\star$ as $V_{\text{out}}^\star = V_{\text{DD}} - RI_C^\star$, hence:

$$V_{\text{DD}} - RI_C^\star + \delta V_{\text{out}} = V_{\text{DD}} - R(I_C^\star + g_m \delta V_{\text{in}}) \tag{17}$$

$$\Rightarrow \delta V_{\text{out}} = -R g_m \delta V_{\text{in}} \tag{18}$$

We re-arrange and solve for the small-signal voltage gain:

$$\frac{\delta V_{\text{out}}}{\delta V_{\text{in}}} = -R g_m = -\frac{RI_C^\star}{V_{\text{TH}}} \tag{19}$$

You are not required to simplify it beyond this point. However, recognize that $I_C^\star R = V_{\text{DD}} - V_{\text{out}}^\star$, so we can relate the small-signal voltage gain directly to the output voltage operating point:

$$\frac{\delta V_{\text{out}}}{\delta V_{\text{in}}} = -\frac{V_{\text{DD}} - V_{\text{out}}^\star}{V_{\text{TH}}} \tag{20}$$

This suggests that we want the voltage "gap" between the supply voltage V_{DD} and the output voltage bias (i.e. DC) point $V_{\text{out}}^\star$ to be large if we want a large voltage gain. For a fixed V_{DD}, this means a lower $V_{\text{out}}^\star$. However, when you learn about BJT devices properly, you will see the output bias voltage can only go so low before our models fails. We also notice from equations (19) and (20) that to get a higher voltage gain, we need a larger bias (DC) current $I_C^\star$ (to get a lower $V_{\text{out}}^\star$ means we need a larger $I_C^\star$ through the resistor). In other words, to get higher voltage gain, we need to burn more power.

(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 $V_{\text{TH}} = 26\,\text{mV}$.)*

Solution: Just plugging in to equation (19):

$$\left|\frac{\delta V_{\text{out}}}{\delta V_{\text{in}}}\right| = \frac{I_C^\star R}{V_{\text{TH}}} = \left(\frac{1\,\text{mA} \times 1\,\text{k}\Omega}{26\,\text{mV}}\right) = \frac{1\,\text{V}}{26\,\text{mV}} \approx 38.$$

© UCB EECS 16B, Spring 2022. All Rights Reserved. This may not be publicly shared without explicit permission.

(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**.**

Solution:

$$\left|\frac{\delta V_{\text{out}}}{\delta V_{\text{in}}}\right| = \frac{I_C^\star R}{V_{\text{TH}}} = \left(\frac{9\,\text{mA} \times 1\,\text{k}\Omega}{26\,\text{mV}}\right) = \frac{9\,\text{V}}{26\,\text{mV}} \approx 346.$$

As an aside, notice here that $V_{\text{out}}^\star$ has already been pulled down to around 1 V ($= V_{\text{DD}} - RI_C^\star = 10\,\text{V} - 1\,\text{k}\Omega \times 9\,\text{mA}$). Realistically, this is close to as low as $V_{\text{out}}^\star$ can get for this device; the small-signal voltage gain is close to its upper limit. When you first saw the BJT circuit, it may not have been obvious that V_{DD} and V_{TH} provide the fundamental limit on the small-signal gain for such circuits - you may have been tempted to say the upper bound was related to the resistor value. But the simple linearization analysis in part (c) reveals V_{DD} and V_{TH} are setting the true limit. Courses like EE105 and EE140 further develop these insights in circuit design along with feedback control in interesting and very practical ways.

(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)?**

Solution: We would choose $V_{\text{in}}^\star = 0.7\,\text{V}$ since the magnitude of the small signal gain in this operating point is much higher than that at $V_{\text{in}}^\star = 0.65\,\text{V}$.

Note that since $I_C^\star$ is related to $V_{\text{in}}^\star$ by (8), and $V_{\text{out}}^\star$ is related to $I_C^\star$ by (9), just choosing $V_{\text{in}}^\star$ fixes the small signal gain of the circuit in Fig. 2.

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.

© UCB EECS 16B, Spring 2022. All Rights Reserved. This may not be publicly shared without explicit permission.

4. Linearization of a scalar system

In this question, we linearize the scalar differential equation

$$\frac{\mathrm{d}}{\mathrm{d}t}x(t) = \sin(x(t)) + u(t) \tag{21}$$

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 $0 = \sin(x^\star) + u^\star = \sin(x^\star)$.

Solution:

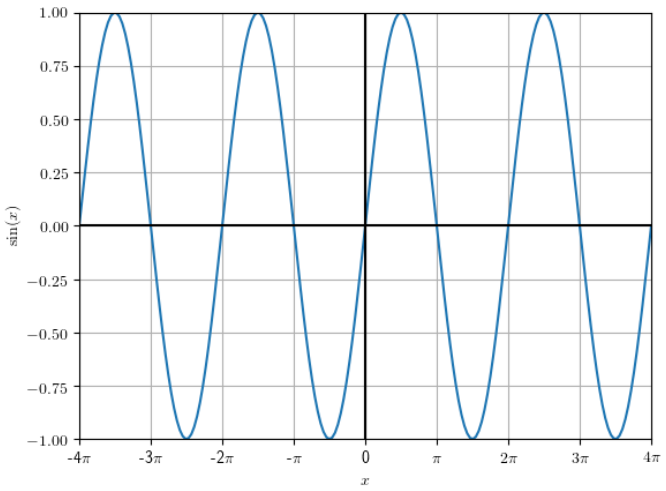

Figure 4: Plot of $\sin(x)$

We can see that all the multiples of π are where the line intersects the sine wave.

(b) From part (a), we can see, graphically, that the equilibria of system (21) 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** (21)**.**

Solution: At $(x_m^\star, u^\star)$ we have

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}x(t) &= \sin(x_m^\star) + u^\star && (22)\\ &= \sin(m\pi) + 0 && (23)\\ &= 0 && (24)\\ & && (25)\end{aligned}$$

so $(x_m^\star, u^\star)$ are equilibria of the system.

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 (21) 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** $\delta u(t) = u(t) - u^\star = u(t) - 0$**?**

Solution: We have

$$\frac{\mathrm{d}x}{\mathrm{d}t} = f(x(t), u(t)) = \sin(x(t)) + u(t) \tag{26}$$

© UCB EECS 16B, Spring 2022. All Rights Reserved. This may not be publicly shared without explicit permission.

$$\frac{\mathrm{d}}{\mathrm{d}t}\delta x(t) \approx \frac{\partial f}{\partial x}(x^\star, u^\star)\delta x(t) + \frac{\partial f}{\partial u}(x^\star, u^\star)\delta u(t) \tag{27}$$
$$= \cos(0)\delta x(t) + (1)\delta u(t) \tag{28}$$
$$= \delta x(t) + \delta u(t). \tag{29}$$

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

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = \lambda x(t) + bu(t) \tag{30}$$

into intervals of Δ gives the discrete-time system

$$x[i+1] = e^{\lambda\Delta}x[i] + \frac{b(e^{\lambda\Delta}-1)}{\lambda}u[i]. \tag{31}$$

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

$$\delta x[i+1] = e^{\Delta}\delta x[i] + \delta u[i](e^{\Delta}-1). \tag{32}$$

(d) **Is the (approximate) discrete-time system (32) stable?**
Solution: For a linear scalar discrete time recurrence relation, stability is determined by the coefficient of the system's variable, in this case δx. We know that if the magnitude of this coefficient is between -1 and 1, our system is stable. But $e^{\Delta} > 1$ for all positive Δ (and Δ by definition has to be positive). Hence, our system is not stable.

(e) Now linearize the system (21) around the equilibrium $(x^\star_{-1}, u^\star) = (-\pi, 0)$. **What is the resulting scalar differential equation for $\delta x(t) = x(t) - (-\pi)$ involving $\delta u(t) = u(t) - 0$?**
Solution: As before, we have

$$\frac{\mathrm{d}x}{\mathrm{d}t} = f(x(t), u(t)) = \sin(x(t)) + u(t) \tag{33}$$
$$\frac{\mathrm{d}}{\mathrm{d}t}\delta x(t) \approx \frac{\partial f}{\partial x}(x^\star_{-1}, u^\star)\delta x(t) + \frac{\partial f}{\partial u}(x^\star_{-1}, u^\star)\delta u(t) \tag{34}$$
$$= \cos(-\pi)\delta x(t) + (1)\delta u(t) \tag{35}$$
$$= -\delta x(t) + \delta u(t). \tag{36}$$

We again discretize the approximate linear system obtained in (e). Pattern-matching with (30) and using (31) (verify this!), we get

$$\delta x[i+1] = e^{-\Delta}\delta x[i] + \delta u[i](1 - e^{-\Delta}). \tag{37}$$

(f) **Is the (approximate) discrete-time system (37) stable?**
Solution: In this case, $0 < e^{-\Delta} < 1$ for all positive Δ, hence our system is stable.

(g) Suppose for the two linearized discrete-time systems (32) and (37), 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.

© UCB EECS 16B, Spring 2022. All Rights Reserved. This may not be publicly shared without explicit permission.

Solution: Let's begin with the first case, $x^\star = 0$. Based on our defintion of δx, we have, $\delta u[n] = -k\delta x[n]$. Substituting and grouping the terms, we get

$$\delta x[i+1] = \delta x[i]\left(e^{\Delta} - k(e^{\Delta} - 1)\right) \tag{38}$$

Hence, we want the above coefficient to between -1 and 1.

$$-1 < e^{\Delta} - k(e^{\Delta} - 1) < 1 \tag{39}$$

$$-(1 + e^{\Delta}) < -k(e^{\Delta} - 1) < 1 - e^{\Delta} \tag{40}$$

$$\implies 1 < k < \frac{e^{\Delta} + 1}{e^{\Delta} - 1} \tag{41}$$

Looking at the second case, with $x^\star = -\pi$, we get

$$\delta x[i+1] = e^{-\Delta}\delta x[i] + k\delta x[i](e^{-\Delta} - 1) + k\pi(e^{-\Delta} - 1). \tag{42}$$

Grouping terms, and further simplifying

$$\delta x[i+1] = \delta x[i]\left(e^{-\Delta} + k(e^{-\Delta} - 1)\right) + k\pi(e^{-\Delta} - 1). \tag{43}$$

As before, we want this coefficient to be between -1 and 1, hence

$$-1 < e^{-\Delta} + k(e^{-\Delta} - 1) < 1 \tag{44}$$

$$-(1 + e^{-\Delta}) < k(e^{-\Delta} - 1) < 1 - e^{-\Delta} \tag{45}$$

$$\implies -1 < k < \frac{1 + e^{-\Delta}}{1 - e^{-\Delta}} \tag{46}$$

© UCB EECS 16B, Spring 2022. All Rights Reserved. This may not be publicly shared without explicit permission.

5. 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

$$\frac{\mathrm{d}}{\mathrm{d}t}\vec{x}(t) = A\vec{x}(t) + \vec{b}u(t) + \vec{w}(t) = \begin{bmatrix} 2 & 1 \\ 0 & 2 \end{bmatrix} \vec{x}(t) + \begin{bmatrix} 1 \\ 1 \end{bmatrix} u(t) + \vec{w}(t) \tag{47}$$

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 $\vec{x^*}(t)$ that satisfies the following dynamics:

$$\frac{\mathrm{d}}{\mathrm{d}t}\vec{x^*}(t) = A\vec{x^*}(t) + \vec{b}u^*(t). \tag{48}$$

The presence of the bounded noise term $\vec{w}(t)$ makes the actual state $\vec{x}(t)$ deviate from the desired $\vec{x^*}(t)$ and follow (47) instead. In the following subparts, we will analyze how we can adjust the desired control signal $u^*(t)$ in (48) to the control input $u(t)$ in (47) so that the deviation in the state caused by $\vec{w}(t)$ remains bounded.

Represent the state as $\vec{x}(t) = \vec{x^*}(t) + \Delta\vec{x}(t)$ and $u(t) = u^*(t) + \Delta u(t)$. Using (47) and (48), **show that we can represent the evolution of the trajectory deviation $\Delta\vec{x}(t)$ as a function of the control deviation $\Delta u(t)$ and the bounded disturbance $\vec{w}(t)$ as:**

$$\frac{\mathrm{d}}{\mathrm{d}t}\Delta\vec{x}(t) = A\Delta\vec{x}(t) + \vec{b}\Delta u(t) + \vec{w}(t). \tag{49}$$

(HINT: Write out equation (47) in terms of $\vec{x^*}(t)$, $\Delta\vec{x}(t)$, $u^*(t)$ and $\Delta u(t)$.))

Solution: Using the change of variables $\vec{x}(t) = \vec{x^*}(t) + \Delta\vec{x}(t)$ and $u(t) = u^*(t) + \Delta u(t)$ in (47), we get

$$\frac{\mathrm{d}}{\mathrm{d}t}\vec{x}(t) = A\vec{x}(t) + \vec{b}u(t) + w(t) \tag{50}$$

$$\implies \frac{\mathrm{d}}{\mathrm{d}t}\vec{x^*}(t) + \frac{\mathrm{d}}{\mathrm{d}t}\Delta\vec{x}(t) = A\vec{x^*}(t) + A\Delta\vec{x}(t) + \vec{b}u^*(t) + \vec{b}\Delta u(t) + \vec{w}(t) \tag{51}$$

$$\implies \frac{\mathrm{d}}{\mathrm{d}t}\Delta\vec{x}(t) = A\Delta\vec{x}(t) + \vec{b}\Delta u(t) + \vec{w}(t) + \left(A\vec{x^*}(t) + \vec{b}u^*(t) - \frac{\mathrm{d}}{\mathrm{d}t}\vec{x^*}(t)\right) \tag{52}$$

Using (48) we know that the last term in parenthesis in zero, so

$$\frac{\mathrm{d}}{\mathrm{d}t}\Delta\vec{x}(t) = A\Delta\vec{x}(t) + \vec{b}\Delta u(t) + \vec{w}(t) \tag{53}$$

Note that this implies the disturbance $\vec{w}(t)$ is entirely something that must be dealt with in the $\Delta\vec{x}$ dynamics. It doesn't affect the desired trajectory at all.

© UCB EECS 16B, Spring 2022. All Rights Reserved. This may not be publicly shared without explicit permission.

(b) **Are the dynamics that you found for $\Delta\vec{x}(t)$ in part 5.a stable? Based on this, in the presence of bounded disturbance $\vec{w}(t)$, will $\vec{x}(t)$ in (47) follow the desired trajectory $\vec{x^*}(t)$ closely if we just apply the control $u(t) = u^*(t)$ to the original system in (47), i.e. $\Delta u(t) = 0$?**

(HINT: Use the numerical values of A and $\vec{b}$ from (47) in the solution from part (b) to determine stability of $\Delta\vec{x}(t)$.))

Solution: If we just set $\Delta u(t) = 0$, then our $v(t)$ dynamics becomes open-loop so

$$\frac{\mathrm{d}}{\mathrm{d}t}\Delta\vec{x}(t) = A\Delta\vec{x}(t) + \vec{w}(t). \tag{54}$$

Recall that the condition for stability in the continuous-time case is that the real part of the eigenvalues of the state transition matrix A must be less then zero. Note that since A is an upper-triangular matrix, its eigenvalues lie on the diagonal, so they are 2 and 2. In this case, they have real parts greater than zero so the open-loop $\Delta\vec{x}(t)$ system is unstable.

$\Delta\vec{x}(t)$ will then follow a growing exponential trajectory in the form of e^{2t}, and will thus amplify any disturbance $\vec{w}(t)$ to the state. Therefore, $\Delta\vec{x}(t)$ will not go to $\vec{0}$ and we will not end up following the intended trajectory $\vec{x^*}(t)$.

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

(c) For the $\Delta\vec{x}(t), \Delta u(t)$ system, **apply feedback control by letting $\Delta u(t) = F\Delta\vec{x}(t) = \begin{bmatrix} f_0 & f_1 \end{bmatrix}\Delta\vec{x}(t)$ that would place both the eigenvalues of the closed-loop $\Delta\vec{x}(t)$ system at -10. Find f_0 and f_1.**

Solution: With the new input, the system equation for $\Delta\vec{x}(t)$ is given by:

$$\frac{\mathrm{d}}{\mathrm{d}t}\Delta\vec{x}(t) = A_v\Delta\vec{x}(t) + \vec{b}\begin{bmatrix} f_0 & f_1 \end{bmatrix}\Delta\vec{x}(t) + \vec{w}(t) \tag{55}$$

$$\implies \frac{\mathrm{d}}{\mathrm{d}t}\Delta\vec{x}(t) = \begin{bmatrix} 2+f_0 & 1+f_1 \\ f_0 & 2+f_1 \end{bmatrix}\Delta\vec{x}(t) + \vec{w}(t) \tag{56}$$

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

$$\det(\lambda I - A_{cl}) = \begin{bmatrix} \lambda-2-f_0 & -1-f_1 \\ -f_0 & \lambda-2-f_1 \end{bmatrix} \tag{57}$$

$$= \lambda^2 - (4+f_0+f_1)\lambda + f_0 + 2f_1 + 4 \tag{58}$$

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

By comparing the coefficients, we have:

$$-(4+f_0+f_1) = 20 \tag{59}$$

$$f_0 + 2f_1 + 4 = 100 \tag{60}$$

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

Why did we pick -10? So that our closed-loop system would converge faster and aggressively reject disturbances.

(d) Based on what you did in the previous parts, and given access to the desired trajectory $\vec{x^*}(t)$, the desired control $u^*(t)$, and the actual measurement of the state $\vec{x}(t)$, **come up with a way to do feedback control that will keep the trajectory staying close to the desired trajectory no matter**

© UCB EECS 16B, Spring 2022. All Rights Reserved. This may not be publicly shared without explicit permission.

what the small bounded disturbance $\vec{w}(t)$ does. *(HINT: Express the control input $u(t)$ in terms of $u^*(t)$, $\vec{x^*}(t)$, and $\vec{x}(t)$.))*

Solution: From the previous parts, we have successfully found a feedback control law $\Delta u(t) = \begin{bmatrix} f_0 & f_1 \end{bmatrix} \Delta\vec{x}(t)$ such that the closed-loop system for $\Delta\vec{x}(t)$ is stable and convering to $\vec{0}$ as long as the disturbances are bounded. As a result, by changing variables $\vec{x}(t) = \vec{x^*}(t) + \Delta\vec{x}(t)$ and $u(t) = u^*(t) + \Delta u(t)$ that we performed in (b), we can infer that the state $\vec{x}(t)$ will stay close to the desired trajectory $\vec{x^*}(t)$ no matter what the bounded disturbance $\vec{w}(t)$ does.

From our initial change of variables, we want to set

$$u(t) = u^*(t) + \Delta u(t) = u^*(t) + \begin{bmatrix} -144 & 120 \end{bmatrix} \Delta\vec{x}(t) \tag{61}$$

$$= u^*(t) + \begin{bmatrix} -144 & 120 \end{bmatrix} (\vec{x}(t) - \vec{x^*}(t)) \tag{62}$$

as our overall system input to achieve this.

This lets us have our cake and eat it too! We can use the desired system dynamics from (48) to plan, and by using closed-loop feedback we can make sure that we mostly follow our plan even in the face of disturbances.

© UCB EECS 16B, Spring 2022. All Rights Reserved. This may not be publicly shared without explicit permission.

6. **(OPTIONAL) Make Your Own Problem.**

   Write your own problem about content covered in the course thus far, and provide a thorough solution to it.

   NOTE: This can be a totally new problem, a modification on an existing problem, or a Jupyter part for a problem that previously didn't have one. Please cite all sources for anything (including course material) that you used as inspiration.

   NOTE: High-quality problems may be used as inspiration for the problems we choose to put on future homeworks or exams.

7. **Homework Process and Study Group**

   Citing sources and collaborators are an important part of life, including being a student!

   We also want to understand what resources you find helpful and how much time homework is taking, so we can change things in the future if possible.

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

   (b) **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:

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

© UCB EECS 16B, Spring 2022. All Rights Reserved. This may not be publicly shared without explicit permission.