EECS 16B Designing Information Devices and Systems II
Fall 2021 UC Berkeley

Homework 8

This homework is due on Friday, October 22, 2021, at 11:59PM. Self-grades and HW Resubmission are due on Tuesday, October 26, 2021, 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 the homework this week: Note 12. Note 13 on the website is optional and for those who enjoyed OMP in 16A. Note 14 is for Thu lecture.

   (a) Consider $A \in \mathbb{R}^{n\times n}$ where the columns of A (denoted by $\vec{a}_k, 1 \leq k \leq n$) are orthonormal. **What does the least squares solution $(A^\top A)^{-1}A^\top \vec{y}$ simplify to?**

   (b) Suppose we have two vectors $\vec{v}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ and $\vec{v}_2 = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \in \mathbb{R}^2$. Then, we use Gram-Schmidt Orthonormalization to construct $\vec{q}_1$ and $\vec{q}_2$. **Are $\vec{q}_1$ and $\vec{q}_2$ the only vectors that form an orthogonal basis for $\mathrm{Span}(\vec{v}_1, \vec{v}_2)$?**

2. Gram-Schmidt Basic

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

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

(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**

$$V(V^\top V)^{-1}V^\top \vec{w} = U(U^\top U)^{-1}U^\top \vec{w}. \tag{2}$$

Feel free to use numpy. No need to grind this out by hand.

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

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

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

As covered in EECS16A, this has a closed-form solution:

$$\vec{\hat{x}} = (A^\top A)^{-1} A^\top \vec{y}. \tag{3}$$

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

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

$$\vec{\epsilon}[i] = \vec{y} - A\vec{x}[i] \tag{4}$$

Show that if $\vec{x}[i] = \vec{\hat{x}}$, then $\vec{\epsilon}[i]$ is orthogonal to the columns of A, i.e. show $A^\top \vec{\epsilon}[i] = \vec{0}$.

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

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

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

We define the following update:

$$\vec{x}[i+1] = \vec{x}[i] + \alpha A^\top (\vec{y} - A\vec{x}[i]) \tag{5}$$

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

Notice also that taking one step of this iteration only requires $O(N\ell)$ operations if the height of $\vec{y}$ is ℓ and there are N different parameters in $\vec{x}$ that we are trying to fit. Basically, the complexity of one iteration is the size of the A matrix since every element participates exactly once in eq. (5).

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

$$\nabla\|A\vec{x}-\vec{y}\|^2 = 2A^\top(A\vec{x}-\vec{y}) \tag{6}$$

This can be derived directly by using vector derivatives (outside of 16B's class scope) or by carefully using partial derivatives as we will do for linearization, later in 16B. So, the heuristic update (5) is actually just taking a step along the negative gradient direction. This insight is what lets us adapt this heuristic for a kind of "linearization" applied to other optimization problems that aren't least-squares. (But all this is out-of-scope for 16B at this particular point in the course, and is something discussed further in 127 and 189. Here, at this point in 16B, (5) is just some discrete-time linear system that we have been given.)

To show that $\vec{x}[i] \to \vec{\hat{x}}$, we define a new state variable $\Delta\vec{x}[i] = \vec{x}[i] - \vec{\hat{x}}$. This parameter should be reminiscent of the variation $\vec{v}(t)$ you saw in the homework problem last week on trajectory tracking. In both cases, it represents the deviation from where we want to be.

Derive the discrete-time state evolution equation for $\Delta\vec{x}[i]$, and show that it takes the form:

$$\Delta\vec{x}[i+1] = (I-\alpha G)\Delta\vec{x}[i]. \tag{7}$$

(c) We would like to make the system such that $\Delta\vec{x}[i]$ converges to 0. As a first step, we just want to make sure that we have a stable system. To do this, we need to understand the eigenvalues of $I-\alpha G$. **Show that the eigenvalues of matrix $I-\alpha G$ are $1-\alpha\lambda_{j\{G\}}$, where $\lambda_{j\{G\}}$ $(j=1,2,...N)$ are the eigenvalues of G.**

(d) To be stable, we need all these eigenvalues to have magnitudes that are smaller than 1 (since this is a discrete-time system). Note that since the matrix G above has a special form, all of the eigenvalues of G are non-negative and real. (You'll see why next week in lecture.) **For what step-size/learning-rate α would the eigenvalue $1-\alpha\lambda_{\max\{G\}}=0$ where $\lambda_{\max\{G\}}$ is the largest eigenvalue of G. At this α, what would be the largest magnitude eigenvalue of $I-\alpha G$? Is the system stable?**

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

(e) **Above what value of α would the system (7) become unstable?** This is what happens if you try to set the α to be too high.

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

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

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

This is spiritually the machine-learning counterpart to the idea of critically damped systems, even though the math is different. The key to understanding this is that if you have something that is governed by many different eigenvalues that all have $|\lambda_i|<1$, it is the largest magnitude one that determines how fast we converge. So what this problem is asking you to do is to minimize the largest eigenvalue of $(I-\alpha G)$. If the currently largest magnitude eigenvalue of $(I-\alpha G)$ is negative, then making α slightly smaller would make that eigenvalue smaller in magnitude. If the currently largest magnitude eigenvalue of $(I-\alpha G)$ is positive, then making α slightly larger would make that eigenvalue smaller in magnitude. If the largest positive eigenvalue of $(I-\alpha G)$ wants to pull the α bigger and the largest negative eigenvalue of $(I-\alpha G)$ wants to pull the α smaller, what do you think that suggests has to happen to minimize the largest magnitude eigenvalue of $(I-\alpha G)$?

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

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

4. Motor Driver and System Identification

In the lab project, you will be designing SIXT33N, a mischievous little robot who *might* just do what you want — if you design it correctly. In phases 1 and 2, you will build the **legs** of SIXT33N: you will be designing SIXT33N's wheels and developing a linear model for the car system. The wheels will be driven by two 9-Volt DC motors whose driver control circuit is shown in Figure 1 .

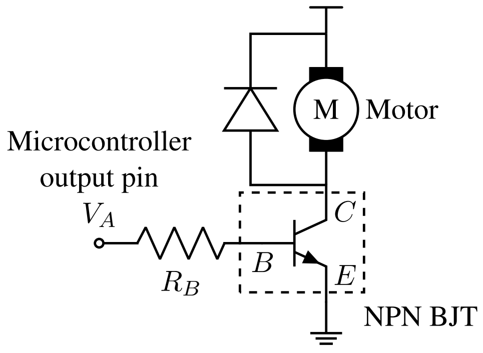


Figure 1: Motor Controller Circuit

There is some minimum voltage required to deliver enough power to the motors to overcome the static friction and start them, but after that point, we treat the motor speed as approximately **linear** with the applied voltage V_A (this will be the basis of the system model you will develop in this problem).

As it is difficult to use a microcontroller (MSP430 in hands-on lab; Arduino in lab sim) to generate a true adjustable DC signal, we will instead make use of its PWM function. A PWM, or pulse-width modulated, signal is a square wave with a variable duty cycle (the proportion of a cycle period for which the power source is turned on, or logic HIGH). PWM is used to digitally change the average voltage delivered to a load by varying the duty cycle. If the frequency is large enough, the on-off switching is imperceptible, but the average voltage delivered to the load changes proportionally with the duty cycle. Hence, changing the duty cycle corresponds to changing the DC voltage supplied to the motor. An example can be seen in figure 2.

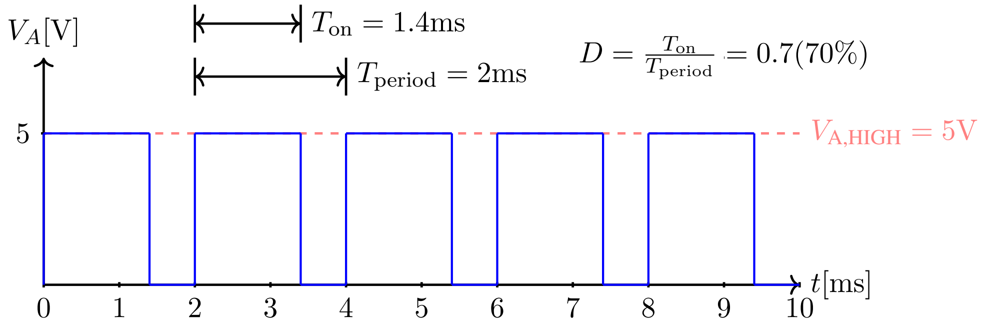
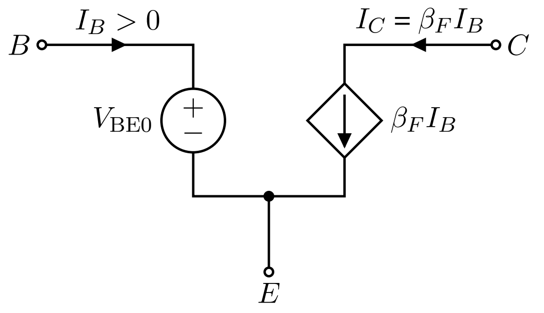


Figure 2: PWM Example with switching frequency 500Hz and 70% duty cycle

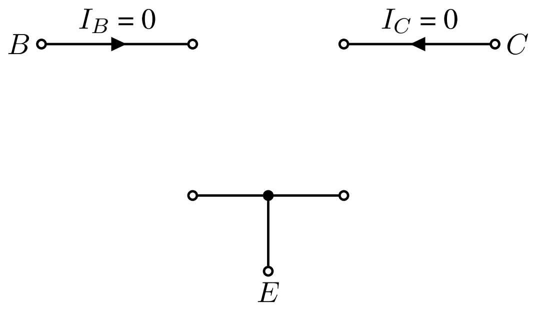
The PWM pin (V_A) is connected via a resistor (R_B) to the "Base (B)" of an NPN bipolar junction transistor (BJT). This transistor, in reality, behaves a bit differently from the NMOS with which you are familiar, but for this class, you may assume that it is functionally the same as an NMOS, behaving as a switch. On the BJT, the three terminals are analogous to those of an NMOS: the "Base (B)" is the gate, the "Collector (C)" is the drain, and the "Emitter (E)" is the source.

The BJT in Figure 1 is switching between **ON** and **OFF** modes when V_A is HIGH and LOW respectively. The model for both ON and OFF states are shown in Figure 3. When the BJT turns on, V_{BE} can be modeled as a fixed voltage source with voltage value V_{BE0}. In ON mode, there is a **Current Controlled Current Source** modeled between "Collector (C)" and "Emitter (E)", i.e., current at the "Collector (C)" is an amplified version of current at the "Base (B)" (notice that positive I_B has to flow into the "Base (B)" for the relation $I_C = \beta_F I_B$ to hold). β_F is called the **Common-Emitter Current Gain**.

The diode in parallel with the motor is needed because of the inductive characteristics of the motor. If the motor is on and V_A switches to LOW, the inductive behavior of the motor maintains the current and the diode provides the path to dissipate it as the BJT is turned off. When the BJT turns on, the diode is off so there is no current flow through the diode.



(a) Model of BJT in ON mode (when V_A is logic HIGH)



(b) Model of BJT in OFF mode (when V_A is logic LOW)

Figure 3: Model of NPN BJT in Different Modes

Please use $V_{\text{BE0}} = 0.8\,\text{V}$, $\beta_F = 100$, $V_{A,\text{HIGH}} = 5\,\text{V}$ and $V_{A,\text{LOW}} = 0\,\text{V}$ for all following calculations.

Part 1: Circuit analysis to construct the system model

(a) **Draw the equivalent motor controller circuit when the BJT is ON by substituting in the BJT model from Fig. 3a into Fig. 1. Express I_B and I_C for V_A = $V_{A,\text{HIGH}}$ as a function of R_B.**

(b) **Draw the equivalent motor controller circuit when the BJT is OFF by substituting in the BJT model from Fig. 3b into Fig. 1. Express I_B and I_C for V_A = $V_{A,\text{LOW}}$ as a function of R_B.**

(c) **Derive the average collector current, I_{AVG}, over one period, T_{period}, of the PWM signal, V_A, as a function of R_B and the duty cycle, D, of the PWM signal.** *Hint: The time average of some signal $f(t)$ from time t_0 to t_1 is given as $f_{\text{AVG}} = \frac{1}{t_1 - t_0}\int_{t_0}^{t_1} f(\tau)\,\mathrm{d}\tau$. Figure 2 may be useful.*

(d) **In the previous part, explain briefly why is it sufficient to take the average over only one period if we are actually interested in the average collector current over multiple periods?**

(e) **If $R_B = 2\,\text{k}\Omega$, what is the average collector current, I_{AVG}, that drives the motor when the duty cycle of the PWM signal is equal to 25%?**

Part 2: Learning a Car Model from data

To control the car, we need to build a model of the car first. Instead of designing a complex nonlinear model, we will approximate the system with a linear model to work for small perturbations around an equilibrium point. The following model applies separately to each wheel (and associated motor) of the car:

$$v_L[i] = \theta_L u_L[i] - \beta_L \tag{8}$$

$$v_R[i] = \theta_R u_R[i] - \beta_R \tag{9}$$

Notice that this particular model has no state variables since we are measuring velocity directly here.

To do system ID, we decide to use the exact same input $u_L[i] = u_R[i] = u[i]$ for both motors. We measure both velocities however.

Meet the variables at play in this model: (Note: the β here have nothing to do with the previous part.)

- i - The current timestep of the model. Since we model the car as a discrete-time system, n will advance by 1 on every new sample in the system.
- $v_L[i]$ - The discrete-time velocity (in units of ticks/timestep) of the left wheel, reading from the motor.
- $v_R[i]$ - The discrete-time velocity (in units of ticks/timestep) of the right wheel, reading from the motor.
- $u[i]$ - The input to each wheel. The duty cycle of the PWM signal (V_A), which is the percentage of the square wave's period for which the square wave is HIGH, is mapped to the range $[0, 255]$. Thus, $u[i]$ takes a value in $[0, 255]$ representing the duty cycle. For example, when $u[i] = 255$, the duty cycle is 100 %, and the motor controller just delivers a constant signal at the system's HIGH voltage, delivering the maximum possible power to the motor. When $u[i] = 0$, the duty cycle is 0 %, and the motor controller delivers 0 V. The duty cycle (D) can be written as

$$\text{duty cycle (D)} = \frac{u[i]}{255} \tag{10}$$

- $\theta(\theta_L, \theta_R)$ - Relates change in input to change in velocity. **Its units are ticks/(timestep $\cdot$ duty cycle).** Since our model is linear, we assume that θ is the same for every unit increase in $u[i]$. This is empirically measured using the car. You will have a separate θ for your left and your right wheel(θ_L, θ_R).
- $\beta(\beta_L, \beta_R)$ - Similarly to θ, β is dependent upon many physical phenomena, so we will empirically determine it using the car. β represents a constant offset in the velocity of the wheel, and hence **its units are ticks/timestep**. Note that you will also typically have a different β for your left and your right wheel (i.e. $\beta_L \neq \beta_R$). These β_L and β_R are different from the β_F of the transistor.

(f) By measuring the car with a PWM signal at different duty cycles, we can collect the velocity data of the left and right wheel, as shown in the following table:

Table 1: The velocity of the left and the right wheel at different duty cycles of PWM signal

Duty Cycle $\times 255$ ($u[i]$)	Velocity of the left wheel ($v_L[i]$)	Velocity of the right wheel ($v_R[i]$)
80	147	127
120	218	187
160	294	253
200	370	317

Since the same input is applied to both the wheels, we can take advantage of the same "horizontal stacking" trick you've seen before to be able to reuse computation. To identify the system we need to setup matrix equations of left and right wheel in the form of:

$$D_{\text{data}} P \approx S \tag{11}$$

where $P = \begin{bmatrix} \theta_L & \theta_R \\ \beta_L & \beta_R \end{bmatrix}$. **Find the matrix D_{data} and matrix S needed to perform system identification to get the matrix of parameters of the left and right wheel, P.**

(g) **Solve the matrix equation $D_{\text{data}}P \approx S$ with least squares to find θ_L, θ_R, β_L, and β_R.** You may use a jupyter notebook for computation.

(h) In most advanced systems, we usually use a combination of a physics-based equation and a data-centric approach to build the model. In our case, the velocity of the motor can be written as

$$v[i] = kI_{\text{AVG}}(u[i]) - \beta \tag{12}$$

where $I_{\text{AVG}}(u[i])$ is the average collector current which is the function of the duty cycle that you have already derived in Part 1. k represents the response of your motor speed to the average current. In our simplified motor driver model in Part 1, you have already derived the expression for the I_{AVG} of the motor as a function of the circuit parameters and the duty cycle D. **If we assume that the model from Part 1 holds, determine the resistance ratio $\left(\frac{R_{\text{B,left}}}{R_{\text{B,right}}}\right)$ from the model parameters you identified in Part 2 item (f).** Assume that the left motor and the right motor respond the same, that is, $k_L = k_R$. The only difference is presumed to come from the resistors used.

(i) In order for the car to drive straight, the wheels must be moving at the same velocity. However, the data from Table 1 tell us that two motors cannot run at the same velocity if the duty cycles of driving PWM signals are the same. **Based on the model you extracted in Part 2 item (f), if we want the car to drive straight and $u_L = 100$, what should u_R be?**

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

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

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

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

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

6. 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:

- Kyle Tanghe.
- Anant Sahai.
- Miki Lustig.
- Aditya Arun.

- Nathan Lambert.
- Sidney Buchbinder.
- Bozhi Yin.
- Kaitlyn Chan.
- Yi-Hsuan Lin.
- Vladimir Stojanovic.
- Moses Won.