EECS 16B Designing Information Devices and Systems II
Spring 2019 A. Sahai, J. Roychowdhury, K. Pister **Homework 8**

This homework is due on Wednesday, April 3, 2019, at 11:59PM.
Self-grades are due on Saturday, April 6, 2019, at 11:59PM.

1. Sampling a Continuous-Time Control System to Get a Discrete-Time Control System

The goal of this problem is to help us better understand how given a linear continuous-time system:

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

we can sample it every Δ seconds and get a discrete-time form of the control system. The discretization of the state equations is a *sampled* discrete-time system given by

$$\vec{x}_d(k+1) = A_d\vec{x}_d(k) + B_d\vec{u}_d(k) \quad (1)$$
$$\vec{y}_d(k) = C_d\vec{x}_d(k) \quad (2)$$

Here, the $\vec{x}_d(k)$ denotes $\vec{x}(k\Delta)$. This is a snapshot of the state. Similarly, the output $\vec{y}_d(k)$ is a snapshot of $\vec{y}(k\Delta)$.

The relationship between the discrete-time input $\vec{u}_d(k)$ and the actual input applied to the physical continuous-time system is that $\vec{u}(t) = \vec{u}_d(k)$ for all $t \in [k\Delta, (k+1)\Delta)$.

While it is clear from the above that the discrete-time state and the continuous-time state have the same dimensions and similarly for the control inputs, what is not clear is what the relationship should be between the matrices A, B and the matrices A_d, B_d. By contrast it is immediately clear that $C_d = C$.

(a) **Argue intuitively why if the continuous-time system is stable, the corresponding discrete-time system should be stable too.** Here, stability means that for bounded inputs, the state stays bounded.

(b) Consider the scalar case where A and B are just scalar constants. **What are the new constants A_d and B_d?**
(HINT: Think about solving this one step at a time. Every time a new control is applied, this is a simple differential equation with a new constant input. How does $\frac{d}{dt}x(t) = \lambda x(t) + u$ evolve with time if it starts at $x(0)$? Notice that $x(0)e^{\lambda t} + \frac{u}{\lambda}(e^{\lambda t} - 1)$ seems to solve this differential equation.)

(c) Consider now the case where A is a diagonal matrix and B is some general matrix. **What is the new matrix A_d and B_d?**

(d) Consider the case where A is a diagonalizable matrix — i.e. it has a full complement of eigenvectors. **Use a change of coordinates to figure out the new matrix A_d and B_d.**

2. Op-Amp Stability

In this question we will revisit the basic op-amp model that was introduced in EE16A and we will add a capacitance C_{out} to make the model more realistic (refer to figure 1). Now that we have the tools to do so, we will study the behavior of the op-amp in positive and negative feedback (refer to figure 2). Furthermore, we will look at the integrator circuit (refer to figure 3) to see how a capacitor in the negative feedback can behave, and why it ends up being close to an integrator.

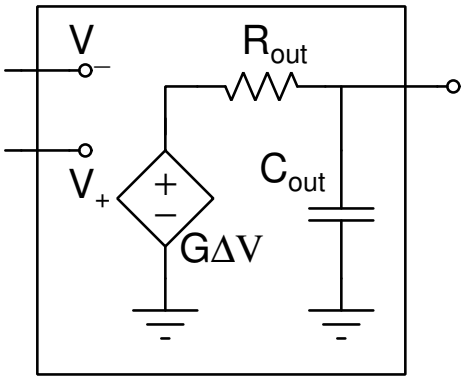
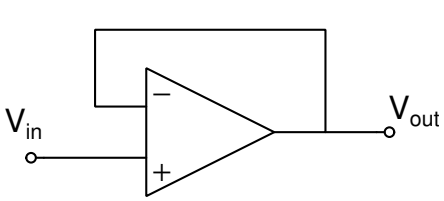
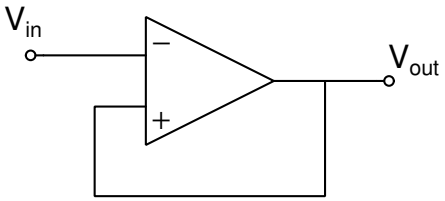


Figure 1: Op-amp model: $\Delta V = V_+ - V_-$



a Buffer in negative feedback



b "Buffer" in positive feedback that doesn't actually work as a buffer.

Figure 2: Op-amp in buffer configuration

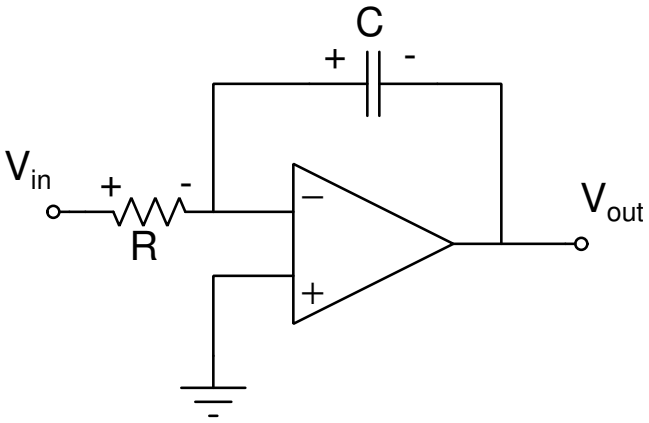


Figure 3: Integrator circuit

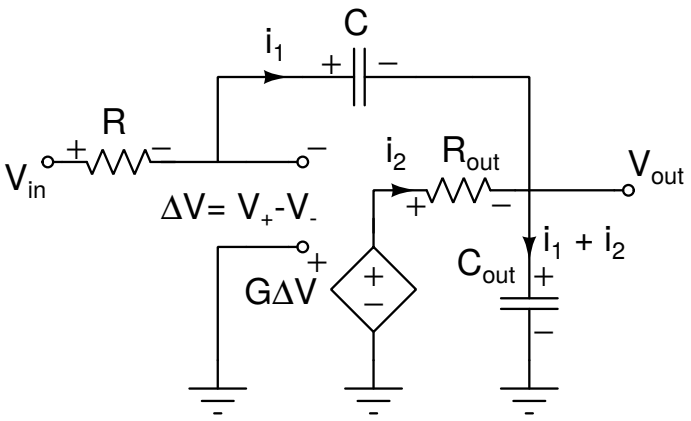


Figure 4: Integrator circuit with Op-amp model

(a) Using the op-amp model in figure 1 and the buffer in negative-feedback configuration in figure 2a, **draw a combined circuit**. Remember that $\Delta V = V_+ - V_-$, the voltage difference between the positive and negative labeled input terminals of the op-amp.

(HINT: Look at figure 4 to see how this was done for the integrator. That might help.)

Note: here, we have used the Thevenin-equivalent model for the op-amp gain to be compatible with what you have seen in 16A. In more advanced analog circuits courses, it is traditional to use a controlled current source with a resistor in parallel instead.

(b) Let's look at the op-amp in negtive feedback. From our discussions in EE16A, we know that the buffer in figure 2a should work with $V_{out} \approx V_{in}$ by the golden rules. **Write a differential-equation for V_{out} by replacing the op-amp with the given model and show what the solution will be as a function of time for a static V_{in}. What does it converge to as $t \to \infty$?** Note: We assume the gain $G > 1$ for all parts of the question.

(c) Next, let's look at the op-amp in positive feedback. We know that the configuration given in figure 2b is unstable and V_{out} will just rail. **Again, using the op-amp model in figure 1, show that V_{out} does not converge and hence the output will rail. For positive DC input $V_{in} > 0$, will V_{out} rail to the positive or negative side? Explain.**

(d) For an ideal op-amp, we can assume that it has an infinite gain, *i.e.*, $G \to \infty$. **Under these assumptions, show that the op-amp in negative feedback behaves as an ideal buffer, *i.e.*, $V_{out} = V_{in}$.**

(e) Let's extend our analysis to the integrator circuit shown in figure 3. Simplifying all the equations, we get a system of differential equations in two variables V_C and V_{out}, where V_C and V_{out} are the voltage drops across the capacitors C and C_{out}. **Fill in the missing term in the following matrix differential equation.**

$$\frac{d}{dt}\begin{bmatrix} V_{out}(t) \\ V_C(t) \end{bmatrix} = \begin{bmatrix} -\left(\frac{G+1}{R_{out}C_{out}} + \frac{1}{RC_{out}}\right) & -\left(\frac{1}{RC_{out}} + \frac{G}{R_{out}C_{out}}\right) \\ -\frac{1}{RC} & ? \end{bmatrix} \begin{bmatrix} V_{out}(t) \\ V_C(t) \end{bmatrix} + \begin{bmatrix} \frac{1}{RC_{out}} \\ \frac{1}{RC} \end{bmatrix} V_{in}(t) \tag{3}$$

(HINT: We picked an easier term to hide. You don't have to write out all the equations and do a lot of algebra to figure out what the missing term is.)

(f) **Solve for the eigenvalues for the matrix/vector differential equation in** (3). For simplicity, assume $C_{out} = C = 0.01F$ and $R = 1\Omega$ and looking at the datasheet for the TI LMC6482 (the op-amps used in lab), we have $G = 10^6$ and $R_{out} = 100\Omega$.

Feel free to assume $G+1 \approx G$ but do not make any other approximations. Feel free to use a scientific calculator or Jupyter to find the eigenvalues.

You should have seen that one eigenvalue corresponds to a slowly dying exponential and is close to 0. The other corresponds to a much faster dying exponential. The very slowly dying exponential is what corresponds to the desired integrator-like behavior. This is what lets it "remember."

(g) Again, assume we have an ideal op-amp, *i.e.*, $G \to \infty$. **Find the eigenvalues under this limit.** Feel free to make any reasonable approximations.

Here, you should see that the eigenvalue that used to be a slowly dying exponential stops dying out at all — corresponding to the ideal integrator's behavior of remembering forever.

3. Gram-Schmidt Basic

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

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

(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**

$$\mathbf{V}(\mathbf{V}^T\mathbf{V})^{-1}\mathbf{V}^T\vec{w} = \mathbf{U}(\mathbf{U}^T\mathbf{U})^{-1}\mathbf{U}^T\vec{w}.$$

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

4. Balance — linearizing a vector system

Justin is working on a small jumping robot named Salto. Salto can bounce around on the ground, but Justin would like Salto to balance on its toe and stand still. In this problem, we'll work on systems that could help Salto balance on its toe using its reaction wheel tail.

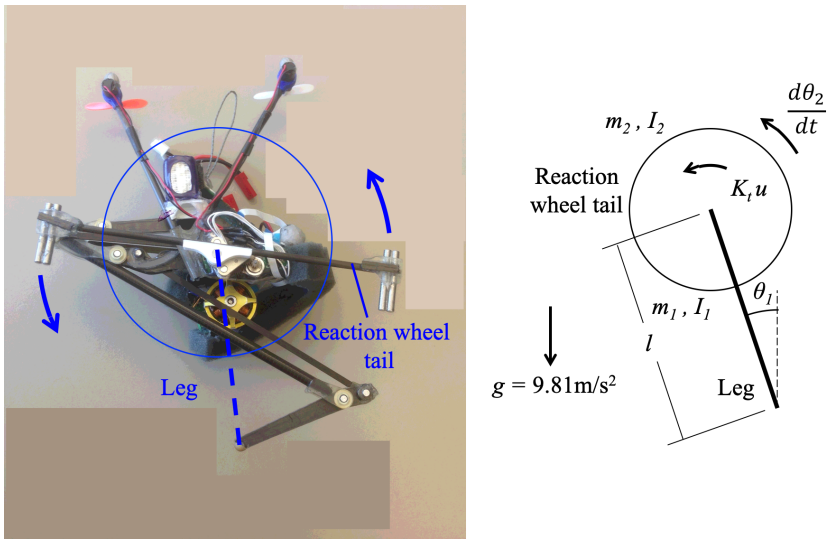


Figure 5: Picture of Salto and the x-z physics model. You can watch a video of Salto here: `https://youtu.be/ZFGxnF9SqDE`

Standing on the ground, Salto's dynamics in the x-z plane (called the sagittal plane in biology) look like an inverted pendulum with a flywheel on the end,

$$(I_1 + (m_1+m_2)l^2)\frac{d^2\theta_1(t)}{dt^2} = -K_t u(t) + (m_1+m_2)lg\sin\big(\theta_1(t)\big)$$
$$I_2\frac{d^2\theta_2(t)}{dt^2} = K_t u(t),$$

where $\theta_1(t)$ is the angle of the robot's body relative to the ground at time t ($\theta_1 = 0$ rad means the body is exactly vertical), $\frac{d\theta_1(t)}{dt}$ is the robot body's angular velocity, $\frac{d\theta_2(t)}{dt}$ is the angular velocity of the reaction wheel tail, and $u(t)$ is the current input to the tail motor. $m_1, m_2, I_1, I_2, l, K_t$ are positive constants representing system parameters (masses and angular momentums of the body and tail, leg length, and motor torque constant, respectively) and $g = 9.81\,\frac{\text{m}}{\text{s}^2}$ is the acceleration due to gravity.

Numerically substituting Salto's physical parameters, the differential equations become:

$$0.001\frac{d^2\theta_1(t)}{dt^2} = -0.025u(t) + 0.1\sin\big(\theta_1(t)\big)$$
$$5(10^{-5})\frac{d^2\theta_2(t)}{dt^2} = 0.025u(t)$$

For this problem, we will use one sensor on the robot: the output of the tail encoder, which measures the angular velocity of the tail relative to the angular velocity of the body.

$$y = \frac{d\theta_2(t)}{dt} - \frac{d\theta_1(t)}{dt}$$

(a) Using the state vector $\begin{bmatrix}\theta_1 \\ \frac{d\theta_1(t)}{dt} \\ \frac{d\theta_2(t)}{dt}\end{bmatrix}$, input u, and output y linearize the system about the point $\vec{x}^* = \begin{bmatrix}0\\0\\0\end{bmatrix}$ with nominal input $u^* = 0$. **Write the linearized equations as $\frac{d}{dt}\vec{x} = A\vec{x} + Bu$ and $y = C\vec{x}$. Write out the matrices with the physical numerical values, not symbolically.**

Note: Since the tail is like a wheel, we care only about the tail's angular velocity $\frac{d\theta_2(t)}{dt}$ and not its angle $\theta_2(t)$. This is why $\theta_2(t)$ is not a state.

Hint: The sin is the only nonlinearity that you have to deal with here.

(b) Your linearized system should have at least one eigenvalue that corresponds to a growing exponential. If we just do the formal test for controllability by checking the $(A, \vec{b})$ pair for the linearized system, **does it indicate that we could place the closed-loop eigenvalues wherever we want for the linearized system?**

(c) Using state feedback, Justin has selected the control gains $K = \begin{bmatrix} -20 & -5 & -0.01 \end{bmatrix}$ for his input $u = -K\vec{x}$. **What are the eigenvalues of the closed loop dynamics for the given K?**

Feel free to use numpy.

(d) (Optional — not in scope) Let's implement a controller for our system using an analog electrical circuit! You can use the following circuit components in Figure 2:

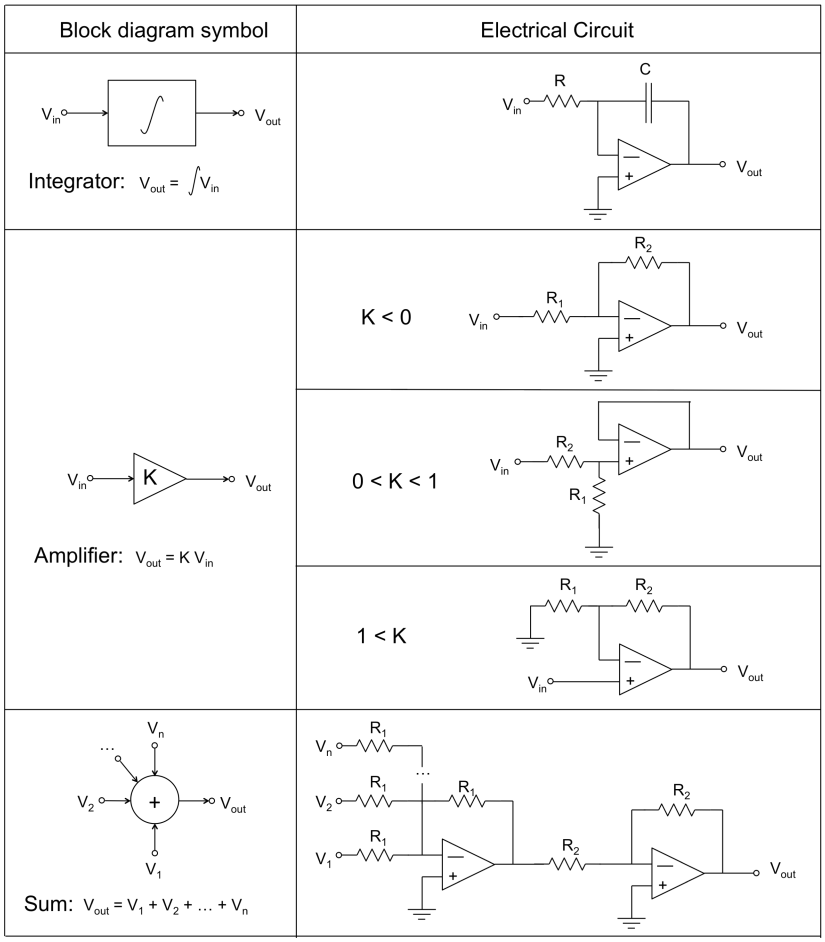


Figure 6: Circuit components and block diagram symbols.

Suppose we want control gains $K = \begin{bmatrix} -20 & -5 & -0.01 \end{bmatrix}$ for the continuous-time input $u(t) = -K\vec{x}(t)$. **Design the circuit that implements the controller. Use relatively reasonable component values.** Assume that you have the state available as a triple of voltages.

5. Speeding Up OMP

Recall the imaging lab from EE16A where we projected masks on an object to scan it to our computer using a single pixel measurement device, that is, a photoresistor. In that lab, we were scanning a 30×40 image having 1200 pixels. In order to recover the image, we took exactly 1200 measurements because we wanted our 'measurement matrix' to be invertible.

However, we saw in 16A lecture near the end of the semester that an iterative algorithm that does "matching and peeling" can enable reconstruction of a sparse vector (i.e. one that has mostly zeros in it) while reducing the number of samples that need to be taken from it. In the case of imaging, the idea of sparsity corresponds to most parts of the image being black with only a small number of light pixels. In these cases, we can reduce the overall number of samples necessary. This would reduce the time required for scanning the image. (This is a real-world concern for things like MRI where people have to stay still while being imaged. It is also a key principle that underlays a lot of modern machine learning.)

In this problem, we have a 2D image I of size 91×120 pixels for a total of 10920 pixels. The image is made up of mostly black pixels except for 476 of them that are white.

Although the imaging illumination masks we used in the lab consisted of zeros and ones, in this question, we are going to have masks with real values — i.e. the light intensity is going to vary in a more finely grained way. Say that we have an imaging mask M_0 of size 91×120. The measurements using the photoresistor using this imaging mask can be represented as follows.

First, let us vectorize our image to $\vec{i}$ which is a column vector of length 10920. Likewise, let us vectorize the mask M_0 to $\vec{m}_0$ which is a column vector of length 10920. Then the measurement using M_0 can be represented as

$$b_0 = \vec{m}_0^T \vec{i}.$$

Say we have a total of K measurements, each taken with a different illumination mask. Then, these measurements can collectively be represented as

$$\vec{b} = \mathbf{A}\vec{i},$$

where $\mathbf{A}$ is an $K \times 10920$ size matrix whose rows contain the vectorized forms of the illumination masks, that is

$$\mathbf{A} = \begin{bmatrix} \vec{m}_1^T \\ \vec{m}_2^T \\ \vdots \\ \vec{m}_K^T \end{bmatrix}.$$

To show that we can reduce the number of samples necessary to recover the sparse image I, we are going to only generate 6500 masks. The columns of $\mathbf{A}$ are going to be approximately uncorrelated with each other. The following question refers to the part of Jupyter notebook file accompanying this homework related to this question.

(a) In the jupyter notebook, we have completed a function `OMP` to run the naive OMP algorithm you learned in EE16A. Read through the code and understand the function `OMP`.

We have also supplied code that reads a PNG file containing a sparse image, takes measurements, and performs OMP to recover it. An example input image file is also supplied together with the code. Using `smiley.png`, generate an image of size 91×120 pixels of sparsity less than 400 and recover it using OMP with 6500 measurements.

Run the code `rec = OMP((height, width), sparsity, measurements, A)` and see the image being correctly reconstructed from a number of samples smaller than the number of pixels of your figure. What is the image? Report how many seconds this took to run.

Remark: Note that this remark is not important for solving this problem; it is about how such measurements could be implemented in our lab setting. When you look at the vector `measurements` you will see that it has zero average value. Likewise, the columns of the matrix containing the masks **A** also have zero average value. To satisfy these conditions, some entries need to have negative values. However, in an imaging system, we cannot project negative light. One way to get around this problem is to find the smallest value of the matrix **A** and subtract it from all entries of **A** to get the actual illumination masks. This will yield masks with positive values, hence we can project them using our real-world projector. After obtaining the readings using these masks, we can remove their average value from the readings to get measurements as if we had multiplied the image using the matrix **A**. This is being done silently for you in the code.

(b) Now let us try using our naive implementation of OMP to recover a slightly less sparse image: An example input image file is supplied together with the code. Using `pika.png`, generate an image of size 100×100 pixels of sparsity less than 800 and recover it using OMP with 9000 measurements.

Run the corresponding code blocks in the accompanying jupyter notebook and report back the number of seconds it took to to reconstruct the image. (Take a well deserved break, this may take upwards of ten minutes to run!)

(c) As you saw, reconstructing the image with the staff's naive implementation took quite a while. Modify the code to run faster by using a Gram-Schmidt orthonormalization to speed it up. **Edit the code given to you in the jupyter notebook. Report back the number of seconds it took to run the reconstruct the image `pika.png`.** Note this should be less than previous part.

This is the only place in the problem where you should have to actually edit code.

(d) (Optional, not in scope) **Do any other modifications you want to further speed up the code.**

Hint: When possible, how would you safely extract multiple peaks corresponding to multiple pixels in one go and add them to the recovered list? Would this speed things up?

6. 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 thin and full rank.

As covered in EE16A, this has a closed-form solution:

$$\vec{\hat{x}} = (A^TA)^{-1}A^T\vec{y}.$$

Direct computation requires the "inversion" of A^TA, which has a complexity of $O(N^3)$ where $(A^TA) \in \mathbb{R}^{N\times N}$. 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 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 in scope for 16B.

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

$$\vec{\varepsilon}(t) = \vec{y} - A\vec{x}(t)$$

Show that if $\vec{x}(t) = \vec{\hat{x}}$, then $\vec{\varepsilon}(t)$ is orthogonal to the columns of A, i.e. show $A^T\vec{\varepsilon}(t) = 0$.

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 $\vec{x}(t)$ will converge to $\vec{x}(t) \to \vec{\hat{x}}$, the true least squares solution. The evolution of these states reflects what is happening computationally.

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

We define the following update:

$$\vec{x}(t+1) = \vec{x}(t) + \alpha A^T(\vec{y} - A\vec{x}(t)) \tag{4}$$

that gives us an updated estimate from the previous one. Here α is the step-size 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 $A^T(\vec{y} - A\vec{x}(t))$ 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 that if we ever reach $\vec{x}(t) = \vec{\hat{x}}$, the system reaches equilibrium — 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 fixed-point (or equilibrium point or DC operating point) is.

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

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

This can be derived directly by using vector derivatives (completely outside of class scope) or by carefully using partial derivatives as we did for linearization. So, the heuristic update (4) 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, and is something discussed further in 127 and 189. Here, (4) is just some discrete-time linear system that we have been given.)

To show that $\vec{x}(t) \to \vec{\hat{x}}$, we define a new state variable $\Delta\vec{x}(t) = \vec{x}(t) - \vec{\hat{x}}$.

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

$$\Delta\vec{x}(t+1) = (I - \alpha G)\Delta\vec{x}(t). \tag{5}$$

(c) We would like to make the system such that $\Delta\vec{x}(t)$ 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 $1 - \alpha\lambda_{i\{G\}}$, where $\lambda_{i\{G\}}$ 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.) Since the matrix G above has a special form, all of the eigenvalues of G are non-negative and real. **For what 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** (5) **become unstable?** This is what happens if you try to set the learning rate 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) (Optional — out of scope)**What is the α that would result in the system being stable, and converge fastest to $\Delta\vec{x} = 0$?**
(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.**

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

Writing your own problems is a very important way to really learn material. The famous "Bloom's Taxonomy" that lists the levels of learning 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.

8. Homework Process and Study Group

Citing sources and collaborators are an important part of life, including being a student! We also want to understand what resources you find helpful and how much time homework is taking, so we can change things in the future if possible.

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

(b) **Who did you work on this homework with?** List names and student ID's. (In case of homework party, you can also just describe the group.)

(c) **How did you work on this homework?** (For example, *I first worked by myself for 2 hours, but got stuck on problem 3, so I went to office hours. Then I went to homework party for a few hours, where I finished the homework.*)

(d) **Roughly how many total hours did you work on this homework?**

Contributors:

- Ioannis Konstantakopoulos.
- Anant Sahai.
- Aditya Arun.
- Justin Yim.
- Kyle Tanghe.
- Miki Lustig.