EECS 16B Fall 2019 — Designing Information Devices and Systems II, UC Berkeley — HW 8

1. Orthonormalization

The idea of orthonormalization is that we take a list of vectors $\vec{a}_1, \vec{a}_2, \ldots, \vec{a}_n$ and get a new list of vectors $\vec{q}_1, \vec{q}_2, \ldots, \vec{q}_n$ such that the following properties are satisfied:

- Spans are preserved: For every $1 \leq \ell \leq n$, we know that $\text{span}(\vec{a}_1, \ldots, \vec{a}_\ell) = \text{span}(\vec{q}_1, \ldots, \vec{q}_\ell)$.
- The inner products of the $\vec{q}_i$ with each other are zero — they are orthogonal. That is: if $i \neq j$, we know $\vec{q}_i^T \vec{q}_j = 0$.
- The $\vec{q}_i$ have unit norm whenever they are nonzero. That is, if $\vec{q}_i \neq \vec{0}$, then $\vec{q}_i^T \vec{q}_i = 1$.

An algorithm for doing this was derived naturally in lecture building on what you learned in 16A about the nature of projections. This problem is about making sure that you understand it within the context of mathematical induction. Mathematical induction is a basic proof technique that is critical to understand as we build mathematical maturity. Our follow-on course CS70 assumes exposure to mathematical induction as a prerequisite and expects students to be grow to be able to craft reasonably intricate inductive proofs from scratch. Here in 16B, our goal is simply for you to be able to follow through with an induction that we set up for you, and to follow inductive arguments.

Anyway, first, let us explicitly state the iterative algorithm:

for $i = 1$ up to n **do** ▷ Iterate through the vectors
  $\vec{r}_i = \vec{a}_i - \sum_{j<i} \vec{q}_j (\vec{q}_j^T \vec{a}_i)$ ▷ Find the amount of $\vec{a}_i$ that remains after we project
  if $\vec{r}_i = \vec{0}$ **then**
    $\vec{q}_i = \vec{0}$
  else
    $\vec{q}_i = \frac{\vec{r}_i}{\|\vec{r}_i\|}$ ▷ Normalize the vector.
  end if
end for

(a) From the If/Then/Else statement in the algorithm above, it is almost completely clear that the third desired property holds by construction, at least for the case that $\vec{q}_i = \vec{0}$. **Show that** $\|\vec{q}_i\| = 1$ **if** $\vec{q}_i \neq \vec{0}$**.** i.e. Why does the "normalize the vector" line actually result in something whose norm is 1?

(b) To establish that the spans are the same, we need to proceed by induction over ℓ. This is a classic proof by induction. (You should always strongly suspect an inductive proof lurking when you see a for loop or a recursive construction in an algorithm.)

The statement is clearly true in the base case of $\ell = 1$ since the $\vec{q}_1$ is just a scaled version of $\vec{a}_1$. Now assume that it is true for $\ell = k-1$. What is true? We need to translate the spans being the same into math. Namely that whenever we have an $\vec{\alpha}$ so that $\vec{y} = \sum_{j=1}^{k-1} \vec{\alpha}[j]\vec{a}_j$, we know there exists $\vec{\beta}$ such that $\vec{y} = \sum_{j=1}^{k-1} \vec{\beta}[j]\vec{q}_j$. And vice-versa: from $\vec{\beta}$ to $\vec{\alpha}$.

Show that the spans are the same for $\ell = k$ as well.

(HINT: First write out what you need to show in one direction. Then just write $\vec{a}_k$ in terms of $\vec{q}_k$ and earlier $\vec{q}_j$ and then proceed. Don't forget the case that $\vec{q}_k = \vec{0}$. Then make sure you do the reverse direction as well.)

This establishes the induction step, and since we have the base case, we know that "all dominos must fall" and the statement is true for all ℓ. This follows the "dominos" picture for induction. Establishing the inductive step shows that each domino will knock over the next domino. The base case establishes that the first domino falls. And thus, they all must fall.

(c) To establish orthogonality, we also need to do another little proof by induction, where we again do induction over ℓ. The statement we want to prove is that for all $j < \ell$, it must be that $\vec{q}_j^T \vec{q}_\ell = 0$. The base case here of $\ell = 1$ is trivially true since there are no $j < 1$. So, we can focus on the induction part of the proof.

Here, it is convenient to use what is sometimes called "strong induction" where we assume that we know for some $k-1$ that for all $i \leq k-1$, we have that for all $j < i$, that $\vec{q}_j^T \vec{q}_i = 0$. (i.e. We don't just assume that the statement is true for $\ell = k-1$, we assume it is true for all ℓ up to and including $k-1$.)

(In the dominos analogy for induction, strong induction is just the fancy name for assuming that all the dominos have fallen before this one. And then showing that this one also falls. This is spiritually not that different from assuming that the previous domino has fallen and then showing that this one also falls.)

Based on this strong induction hypothesis, **show by direct calculation that for all $i \leq k$ for all $j < i$, that $\vec{q}_j^T \vec{q}_i = 0$.**

(HINT: The cases $i \leq k-1$ are already covered by the induction hypothesis. So you can just focus on $i = k$. Next, notice that the case $\vec{q}_k = \vec{0}$ is also easily true. So, focus on the case $\vec{q}_k \neq \vec{0}$ and just expand what you know about $\vec{q}_k$. The strong induction hypothesis will then let you zero out a bunch of terms.)

This establishes the induction step, and since we have the base case, we know that "all dominos must fall" and the statement is true for all ℓ.

(d) It turns out that the fact that the spans are the same can be summarized in matrix form. Let $A = [\vec{a}_1, \cdots, \vec{a}_n]$ and $Q = [\vec{q}_1, \cdots, \vec{q}_n]$. If A and Q have the same column span then it must be the case that $A = QU$ where $U = [\vec{u}_1, \cdots, \vec{u}_n]$ is a square matrix. After all, this U tells us how we can find the $\vec{\beta}$ that correspond to a particular $\vec{\alpha}$ — namely $\vec{\beta} = U\vec{\alpha}$.

Show that a U can be found that is upper-triangular — that is that the i-th column $\vec{u}_i$ of U has zero entires in it for every row after the i-th position.

(HINT: Matrix multiplication tells you that $\vec{a}_i = \sum_{j=1}^{n} \vec{u}_i[j]\vec{q}_i$. What does the algorithm tell you about this relationship? Can you figure out what $\vec{u}_i[j]$ should be?)

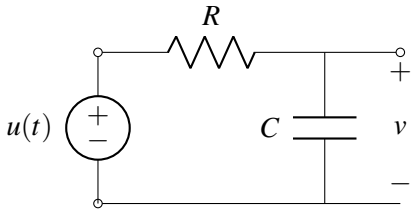
Notice that this explicit construction of U can serve as part of an alternative proof of the fact that the spans are the same. The fact that the span of Q is contained within the span of A is immediate from the fact that by construction, the columns of Q are linear combinations of the columns of A. The interesting part is the other direction — that the columns of A are also all linear combinations of the columns of Q.

2. BIBO Stability

(a) Consider the circuit below with $R = 1\Omega$, $C = 0.5$F, and $u(t) = \cos(t)$. Furthermore assume that $v(0) = 0$ (that the capacitor is initially discharged).

This circuit can be modeled by the differential equation

$$\frac{d}{dt}v(t) = -2v(t) + 2u(t) \tag{1}$$



Show that $v(t)$ remains bounded for all time.

Thinking about this helps you understand what bounded-input-bounded-output stability means in a physical circuit.

(b) Consider the discrete-time system

$$x(t+1) = -2x(t) + 2u(t) \tag{2}$$

with $x(0) = 0$.

Is this system stable or unstable? If stable, prove it. If unstable, find a bounded input sequence $u(t)$ that causes the system to 'blow up'.

(c) For the example in the previous part, **give an explicit sequence of inputs that are not zero but for which the state $x(t)$ will always stay bounded.**

(d) Consider a continuous-time scalar real differential equation with known solution

$$\frac{d}{dt}x(t) = ax(t) + bu(t) \qquad x(t) = e^{at}x(0) + \int_0^t e^{a(t-\tau)}bu(\tau)d\tau.$$

Show that if the system is unstable (has $a \geq 0$), then a bounded input can result in an unbounded output even if the initial condition was zero.

(e) **Repeat the previous part for the specific case of complex $a = r + j2\pi$ where $r > 0$ and zero initial condition $x(0) = 0$.**

All the other truly complex unstable cases are the same way for the same essential reason.

(f) **Repeat the previous part for the specific case of purely imaginary $a = j2\pi$ with zero initial condition $x(0) = 0$.**

All the other purely imaginary unstable cases are the same way for the same essential reason.

(g) Consider the discrete-time real system with known solution:

$$x(t+1) = ax(t) + bu(t) \qquad x(t) = a^t x(0) + \sum_{\ell=0}^{t-1} a^{t-1-\ell}bu(\ell)$$

Show that if the system is quite unstable (has $|a| > 1$), then a bounded input can result in an unbounded output. Assume a zero initial condition here.

(h) **Repeat the previous part for the specific case of $a = -1$.**

(i) Now consider the discrete-time stable case where a is complex and has $|a| < 1$. **Show that as long as $|u(t)| < k$ for some k, that the solution $x(t)$ will be bounded for all time t.**

(HINT: There are a few helpful facts about absolute values and inequalities that, while obvious, are helpful in such proofs. First: $|\sum_j a_j| \leq \sum_j |a_j|$. Second $|ab| \leq |a| \cdot |b|$. Third: $|e^{j\theta}| = 1$ no matter what real number θ is. And fourth, if $a_i > 0$ and $b_i > 0$, and $b_i \leq B$, then $\sum_i a_i b_i \leq \sum_i a_i B = B\sum a_i$.)

3. Eigenvalue Placement through State Feedback

Consider the following discrete-time linear system:

$$\vec{x}(t+1) = \begin{bmatrix} -2 & 2 \\ -2 & 3 \end{bmatrix} \vec{x}(t) + \begin{bmatrix} 1 \\ 1 \end{bmatrix} u(t).$$

In standard language, we have $A = \begin{bmatrix} -2 & 2 \\ -2 & 3 \end{bmatrix}, B = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ in the form: $\vec{x}(t+1) = A\vec{x}(t) + Bu(t)$.

(a) **Is this system controllable?**

(b) **Is this discrete-time linear system stable on its own?**

(c) Suppose we use state feedback of the form $u(t) = \begin{bmatrix} f_1 & f_2 \end{bmatrix} \vec{x}(t)$

Find the appropriate state feedback constants, f_1, f_2 so that the state space representation of the resulting closed-loop system has eigenvalues at $\lambda_1 = -\frac{1}{2}, \lambda_2 = \frac{1}{2}$.

(d) Now suppose we've got a seemingly different system described by the controlled scalar difference equation $z(t+1) = z(t) + 2z(t-1) + u(t)$. **Write down the above system's representation in the following matrix form:**

$$\vec{z}(t+1) = A_z\vec{z}(t) + B_z u(t). \tag{3}$$

Please specify what the vector $\vec{z}(t)$ consists of as well as the matrix A_z and the vector B_z.
(HINT: Just as "state" in the case of continuous time refers to anything that has a derivative taken in the system of differential equations, for discrete time systems, the concept of state refers to memory. What, besides the current input, must you remember about the past/present to be able to figure out the future? In this case, you must know both $z(t)$ and $z(t-l)$.)

(e) We will now show how the initial matrix representation for $\vec{x}(t)$ can be converted to the canonical form for $\vec{z}(t)$ using a change of basis. Suppose we do a transformation of the coordinates of the state $\vec{x}(t)$ to $\vec{z}(t) = P\vec{x}(t)$. **Write down the state-transition matrices of $\vec{z}(t)$ in terms of the state transition matrices of $\vec{x}(t)$, i.e., express A_z and B_z in terms of A, B, and P.** For $P = \begin{bmatrix} -1 & 1 \\ 0 & 1 \end{bmatrix}$, confirm that the resulting state space representation of the behavior of $\vec{z}(t)$ is indeed the same as the previous part (i.e. we get the same A_z, B_z).

(f) For the previous part, **Design a feedback $\begin{bmatrix} \bar{f}_1 & \bar{f}_2 \end{bmatrix}$ to place the closed-loop eigenvalues at $\lambda_1 = -\frac{1}{2}, \lambda_2 = \frac{1}{2}$. Confirm that $\begin{bmatrix} f_1 & f_2 \end{bmatrix} = \begin{bmatrix} \bar{f}_1 & \bar{f}_2 \end{bmatrix} P$.**

(g) We are now ready to go through some numerical examples to see how state feedback works. Consider the first discrete-time linear system. Enter the matrix A and B from (a) for the system

$$\vec{x}(t+1) = A\vec{x}(t) + Bu(t) + w(t)$$

into the Jupyter notebook and use the random input $w(t)$ as the disturbance introduced into the state equation. Observe how the norm of $\vec{x}(t)$ evolves over time for the given A. **What do you see happening to the norm of the state?**

(h) Add the feedback computed in part (c) to the system in the notebook and **explain how the norm of the state changes.**

(i) Now we evaluate a system described by the following scalar system $z(t+1) = az(t) + u(t) + w(t)$ in the Jupyter notebook. Consider two values of a, one case with $a > 1$ and one with $a < 1$, to observe the difference in the evolution of $|z(t)|$ for the same error as part (g). **Describe the differences between the two.**

(j) Suppose that the disturbance is actually coming from observation noise. We assume $y(t) = z(t) + w(t)$ where $w(t)$ is some random noise. Add a state feedback $u(t) = ky(t)$ to the system so that the resulting closed loop system is described by $z(t+1) = (a+k)z(t) + kw(t)$. Say we know $a = -1.25$. **For what values of k's will the result be bounded. Confirm with the norm of the closed loop system.**

(k) **Is it advisable to have $a+k$ close to zero if we want to minimize the magnitudes of the state $x(t)$? How does the effect of the noise in the observation influence this?** Assign values of k close to $-a$ to see the effect in the Jupyter notebook. Compare to values that are smaller, but still keep it stable.

4. Tracking a Desired Trajectory in Continuous Time

The treatment in 16B so far has treated closed-loop control as being about holding a system steady at some desired operating point, which was designated as zero in a linear model. This control used the actual current state to apply a control signal designed to bring the state to zero. 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 can we 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 buffeted by small disturbances throughout.

The key conceptual idea is to realize that we can change coordinates in a time-varying way so that "zero" is the desired "open-loop" trajectory.

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.

Now, consider the specific 2-dimensional system

$$\frac{d}{dt}\vec{x}(t) = A\vec{x}(t) + \vec{b}u(t) = \begin{bmatrix} 2 & 1 \\ 0 & 2 \end{bmatrix} \vec{x}(t) + \begin{bmatrix} 1 \\ 1 \end{bmatrix} u(t) \tag{4}$$

where $u(t)$ is a scalar valued continuous control input.

(a) **Would the given system be controllable if we viewed the $A, \vec{b}$ as the parameters of a discrete-time system?**

(b) Now, suppose we started at $\vec{x}(0) = \vec{0}$ and had a nominal control signal $u_n(t)$ that would make the system follow the desired trajectory $\vec{x}_n(t)$ that satisfies (4) together with $u_n(t)$.

Change variables using $\vec{x}(t) = \vec{x}_n(t) + \vec{v}(t)$ and $u(t) = u_n(t) + u_v(t)$ and write out what (4) implies for the evolution of the trajectory deviation $\vec{v}(t)$ as a function of the control deviation $u_v(t)$.

Now, add a bounded disturbance term $\vec{w}(t)$ to the original state evolution in (4) and let's see if we can absorb that entirely within an evolution equation for $\vec{v}(t)$ you found above. Write out the resulting equation for the dynamics as:

$$\frac{d}{dt}\vec{v}(t) = A_v\vec{v}(t) + \vec{b}_v u_v(t) + \vec{w}(t) \tag{5}$$

What are A_v and $\vec{b}_v$?

(c) Based on what you have found above, how will the system behave over time? **If there is some small distubance, will we end up following the intended trajectory $\vec{x}_n(t)$ closely if we just apply the control $u_n(t)$ to the original system?**

Now, we want to apply state feedback control to the system to get it to more or less follow the desired trajectory.

(d) Just looking at the $\vec{v}(t), u_v(t)$ system, **how would you apply state-feedback to choose $u_v(t)$ as a function of $\vec{v}(t)$ that would place both the eigenvalues of the closed-loop $\vec{v}(t)$ system at -10.**

(e) Based on what you did in the previous parts, and given access to the desired trajectory $\vec{x}_n(t)$, the nominal controls $u_n(t)$, 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 what the small bounded disturbance $\vec{w}(t)$ does.**

5. 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 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 yet 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:

$$\vec{x}(t+1) = \vec{x}(t) + \alpha A^T(\vec{y} - A\vec{x}(t)) \tag{6}$$

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 see 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 equilibrium 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 (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 (6) 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 currently out-of-scope for 16B at this point, and is something discussed further in 127 and 189. Here, at this point in 16B, (6) 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:

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

(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** (7) **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.** Consider how the different step sizes relate to recurrence relations, and how a sufficiently small step size can approach a continuous solution.

6. 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.

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) **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:

- Regina Eckert.
- Anant Sahai.
- Sidney Buchbinder.
- Nathan Lambert.
- Varun Mishra.
- Ming Jin.
- Kyle Tanghe.
- Miki Lustig.
- Aditya Arun.

