EECS 16B Designing Information Devices and Systems II
Spring 2019 UC Berkeley **Homework 9**

This homework is due on Wednesday, April 10, 2019, at 11:59PM.
Self-grades are due on Saturday, April 13, 2019, at 11:59PM.

1. **Induced Matrix Norms** Often, the general effect of matrices on their inputs is really hard to predict. To overcome this, we usually try to "bound" the effect a matrix has on input vectors. We will work through a simple case of bounding the output of an $n \times n$ matrix T given that T is diagonalizable and symmetric. We will consider the system

$$\vec{y} = T\vec{x},$$

where $\vec{x}$ is the input vector and $\vec{y}$ is the output vector.

   (a) Let $U = [\vec{u}_1, \vec{u}_2, \ldots, \vec{u}_n]$ be a set of orthonormal eigenvectors of the matrix T. **Decompose the generic input vector $\vec{x}$ into a linear combination of these eigenvectors.**

   (b) Let the eigenvalues of T be $\{\lambda_1, \lambda_2, \ldots, \lambda_n\}$ with $|\lambda_1| \geq |\lambda_2| \geq \ldots |\lambda_n|$. If we represent $\vec{x}$ as a linear combination of the eigenvectors of T, **what is the Euclidean norm of the output vector $\vec{y}$, $\|\vec{y}\|$?**
   Hint: You can use the fact that Euclidean norms are preserved with orthonormal transforms.

   (c) Say you do not know all the eigenvalues of T, but you know the largest eigenvalue λ_1. If the norm $||\vec{x}|| = \alpha$, **how big could $\|\vec{y}\|$ be**?
   The maximum factor by which a square matrix can grow the norm of a vector is called the induced norm for that matrix. Although we had you do the derivation above for a symmetric matrix T, the fact that $\|A\vec{x}\| = \sqrt{\vec{x}^T A^T A \vec{x}}$ can be used to show how to generalize this concept of induced norm for general matrices.

2. **Frobenius Norm**

   In this problem we will investigate the basic properties of the Frobenius norm.

   Much like the norm of a vector $\vec{x} \in \mathbb{R}^N$ is $\|x\| = \sqrt{\sum_{i=1}^N x_i^2}$, the Frobenius norm of a matrix $A \in \mathbb{R}^{N \times N}$ is defined as

$$\|A\|_F = \sqrt{\sum_{i=1}^N \sum_{j=1}^N |A_{ij}|^2}.$$

   This is basically the norm that comes from treating a matrix like a big vector filled with numbers. Note that matrices have other types of natural norms as well, like the induced norm.

   (a) With the above definitions, **show that**

$$\|A\|_F = \sqrt{Tr\{A^T A\}}.$$

   Note: The trace of a matrix is the sum of its diagonal entries. For example, let $A \in \mathbb{R}^{N \times N}$, then,

$$Tr\{A\} = \sum_{i=1}^N A_{ii}$$

Homework 9 ©UCB EECS 16B, Spring 2019. All Rights Reserved. This may not be publicly shared without explicit permission.

(b) **Show that if U and V are orthonormal matrices, then**

$$\|UA\|_F = \|AV\|_F = \|A\|_F.$$

(HINT: You can proceed directly, or you can think about the relationship of the Frobenius norm of a matrix to the norms of each of its columns or rows. If you take the latter path, then the fact that orthonormal matrices don't change the norms of vectors might prove handy.)

(c) **Use the SVD decomposition to show that $\|A\|_F = \sqrt{\sum_{i=1}^{N} \sigma_i^2}$, where $\sigma_1, \cdots, \sigma_N$ are the singular values of A.**
(HINT: The previous part migh be quite useful.)

3. The Moore-Penrose pseudoinverse for "fat" matrices

Say we have a set of linear equations given by $A\vec{x} = \vec{y}$. If A is invertible, we know that the solution for $\vec{x}$ is $\vec{x} = A^{-1}\vec{y}$. However, what if A is not a square matrix? In 16A, you saw how this problem could be approached for tall "standing up" matrices A where it really wasn't possible to find a solution that exactly matches all the measurements, using linear least-squares. The linear least-squares solution gives us a reasonable answer that asks for the "best" match in terms of reducing the norm of the error vector.

This problem deals with the other case — when the matrix A is "lying down." In this case, there are generally going to be lots of possible solutions — so which should we choose? Why? We will walk you through the **Moore-Penrose pseudoinverse** that generalizes the idea of the matrix inverse and is derived from the singular value decomposition.

(a) Say you have the matrix

$$A = \begin{bmatrix} 1 & -1 & 1 \\ 1 & 1 & -1 \end{bmatrix}.$$

To find the Moore-Penrose pseudoinverse we start by calculating the SVD of A. That is to say, calculate U, Σ, V such that

$$A = U\Sigma V^T.$$

Here we will give you that the decomposition of A is:

$$A = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 2 & 0 & 0 \\ 0 & \sqrt{2} & 0 \end{bmatrix} \begin{bmatrix} 0 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 1 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}$$

where:

$$U = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}$$

$$\Sigma = \begin{bmatrix} 2 & 0 & 0 \\ 0 & \sqrt{2} & 0 \end{bmatrix}$$

$$V^T = \begin{bmatrix} 0 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 1 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}$$

It is a good idea to be able to calculate the SVD yourself as you may be asked to solve similar questions on your own in the exam.

Let us now think about what the SVD does. Let us look at matrix A acting on some vector $\vec{x}$ to give the result $\vec{y}$. We have

$$A\vec{x} = U\Sigma V^T\vec{x} = \vec{y}.$$

Observe that $V^T\vec{x}$ rotates the vector, Σ scales the result, and U rotates it again. We will try to "reverse" these operations one at a time and then put them together to construct the Moore-Penrose pseudoinverse.

If U "rotates" the vector $\left(\Sigma V^T\right)\vec{x}$, what operator can we derive that will undo the rotation?

(b) **Derive a matrix that will "unscale", or undo the effect of Σ where it is possible to undo.** Recall that Σ has the same dimensions as A. Ignore any division by zeros (that is to say, let it stay zero).

(c) **Derive an operator that would "unrotate" by V^T.**

(d) **Try to use this idea of "unrotating" and "unscaling" to derive an "inverse", denoted as $A^\dagger$.** That is to say,

$$\vec{x} = A^\dagger\vec{y}$$

The reason why the word inverse is in quotes (or why this is called a pseudo-inverse) is because we're ignoring the "divisions" by zero.

(e) **Use $A^\dagger$ to solve for a vector $\vec{x}$ in the following system of equations.**

$$\begin{bmatrix} 1 & -1 & 1 \\ 1 & 1 & -1 \end{bmatrix}\vec{x} = \begin{bmatrix} 2 \\ 4 \end{bmatrix}$$

(f) Now we will see why this matrix is a useful proxy for the matrix inverse in such circumstances. **Show that the solution given by the Moore-Penrose pseudoinverse satisfies the minimality property that if $\vec{\hat{x}}$ is the pseudo-inverse solution to $A\vec{x} = \vec{y}$, then $\|\vec{\hat{x}}\| \leq \|\vec{z}\|$ for all other vectors $\vec{z}$ satisfying $A\vec{z} = \vec{y}$.** *(Hint: look at the vectors involved in the V basis. Think about the relevant nullspace and how it is connected to all this.)*

This minimality property is useful in many applications. You saw a control application in lecture. You'll see a communications application in another problem. This is also used all the time in machine learning, where it is connected to the concept behind what is called ridge regression or weight shrinkage.

4. Rank 1 Decomposition

In this problem, we will decompose a few images into linear combinations of rank 1 matrices. Remember that outer product of two vectors gives a rank 1 matrix.

(a) Consider a standard 8×8 chessboard shown in Figure 1. Assume that black colors represent -1 and that white colors represent 1.

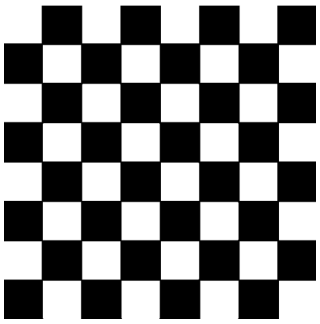


Figure 1: 8×8 chessboard.

Hence, that the chessboard is given by the following 8×8 matrix C_1:

$$C_1 = \begin{bmatrix} 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \end{bmatrix}$$

Express C_1 as a linear combination of outer products. *Hint: In order to determine how many rank 1 matrices you need to combine to represent the matrix, find the rank of the matrix you are trying to represent.*

(b) For the same chessboard shown in Figure 1, now assume that black colors represent 0 and that white colors represent 1.

Hence, the chessboard is given by the following 8×8 matrix C_2:

$$C_2 = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

Express C_2 as a linear combination of outer products.

(c) Now consider the Swiss flag shown in Figure 2. Assume that red colors represent 0 and that white colors represent 1.

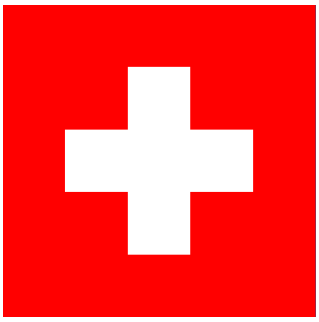


Figure 2: Swiss flag.

Assume that the Swiss flag is given by the following 5×5 matrix S:

$$S = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Furthermore, we know that the Swiss flag can be viewed as a superposition of the following pairs of images:

Figure 3: Pairs of images - Option 1

Figure 4: Pairs of images - Option 2

Express the S in two different ways: i) as a linear combination of the outer products inspired by the Option 1 images and ii) as a linear combination of outer products inspired by the Option 2 images.

5. Symmetric Matrices

We want to show that every real symmetric matrix is diagonalized by a matrix of its orthonormal eigenvectors. In other words, a symmetric matrix has full complement of eigenvectors that are all orthogonal to each other.

In discussion section, you have seen a recursive derivation of a related fact. Formally however, such recursive derivations are usually turned into proofs by using induction. This problem serves to both freshen your mind regarding induction as well as to give you a chance to prove for yourself this very important theorem. (This is the same essential proof as that of Schur upper-triangularization. So understanding this problem will help solidify your understanding of that proof as well.)

(a) You will start by proving a basic lemma about real symmetric matrices under an orthonormal change of basis. **Prove that if S is a symmetric matrix ($S = S^T$) and U is a matrix whose columns are orthonormal, then $U^T SU$ (that is, S represented in the basis U) is also symmetric**.

(b) Another useful lemma is that real symmetric matrices have real eigenvalues. **Prove that the eigenvalues λ of real, symmetric matrix A are real.**

(c) A third useful lemma is one about finding orthonormal bases. **Show that given a single nonzero vector $\vec{u}_0$ of dimension n, that it is possible to find an orthonormal set of n vectors, $\vec{v}_0, \ldots, \vec{v}_{n-1}$ such that $\vec{v}_0 = \alpha \vec{u}_0$ for some scalar α.**

(Hint: Use the Gram-Schmidt process on the list of $n+1$ vectors obtained by starting with the given vector and appending the standard basis.)

(d) For the main proof that every real symmetric matrix is diagonalized by a matrix of its orthonormal eigenvectors, we will proceed by formal induction. Recall that for a proof by induction, we have to start with a base case - this is also the base case in a recursive derivation.

Consider the trivial case of S having dimensions $[1 \times 1]$ ($n = 1$). **Does S have an eigenvector? Can this eigenbasis be made orthonormal? Is the matrix diagonal in this basis? Are the entries real?**

(e) After the base case, we do an inductive stage of the main proof. The first step in the inductive stage is to write down the induction hypothesis. Assume that the property that every real symmetric matrix is diagonalized by a matrix of its orthonormal eigenvectors holds for all symmetric matrices with size $[(n-1) \times (n-1)]$. **Write down the statement of the fact you want to assume in your own words using mathematical notation**. *(Hint: In general for proofs by induction, you want to start with the strongest version of what you want to prove. This gives you the most powerful inductive hypothesis.)*

(f) Now think about a symmetric matrix S with size $[n \times n]$. Consider a real eigenvalue λ_0 of S and the corresponding eigenvector $\vec{u}_0$ (a column vector with size n). **Use an appropriate orthonormal change of basis V to show that** $S = V \begin{bmatrix} \lambda_0 & \cdots & \cdot \\ \vec{0} & \ddots & \cdot \end{bmatrix} V^T$.

(g) Continue the previous part **to show that in fact the matrix relation can be written as**

$$S = V \begin{bmatrix} \lambda_0 & \vec{0}^T \\ \vec{0} & Q \end{bmatrix} V^T$$

where Q is an $[n-1 \times n-1]$ symmetric matrix.

(h) According to our induction hypothesis, we can write Q as $U \Lambda U^T$ where U is an orthonormal $[n-1 \times n-1]$ square matrix and Λ is a diagonal matrix filled with real entries. **Use this fact to show that indeed there must exist an orthonormal $[n \times n]$ square matrix W such that**

$$S = W \begin{bmatrix} \lambda_0 & \vec{0}^T \\ \vec{0} & \Lambda \end{bmatrix} W^T.$$

(Hint: What is the product of orthonormal matrices?)

By induction, we are now done since we have proved that having the desired property for $n-1$ implies that we have the property for n and we also have a valid base case at $n=1$.

According to the base case and inductive steps we just proved, the statement, "every real symmetric matrix is diagonalized by a matrix of its real orthonormal eigenvectors" is proved by induction.

6. Rank in System Identification

Suppose that we have a system

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

$$y(t) = C\vec{x}(t) \tag{2}$$

where $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times 1}$, and $C \in \mathbb{R}^{1\times n}$ are unknown. **We may assume that the system is controllable and observable**, but other than that, A, B, and C are a total mystery. We would like to use input and output data collected from the system to learn about A, B, and C– that is, we'd like to do system identification. We've learned that, as long as you know how many states the system has, you can learn quite a bit: using only *input data* and *output data* with linear least-squares, you can identify matrices A_i, B_i, and C_i that are related to the true A, B and C by a coordinate change. Unfortunately, you can't use get the true A, B and C in the original state coordinates using only input and output data. But A_i, B_i, and C_i still give you a lot of useful information, such as the system eigenvalues.

However, this analysis required that we knew the the true number of states of the system. What if we *don't* know how many states the system has?

All we have are a scalar input trace $u(t)$ and a scalar output trace $y(t)$. We do not have direct information whatsoever about the state trace $x(t)$, excepting its initial condition. Can we still do system identification in this case?

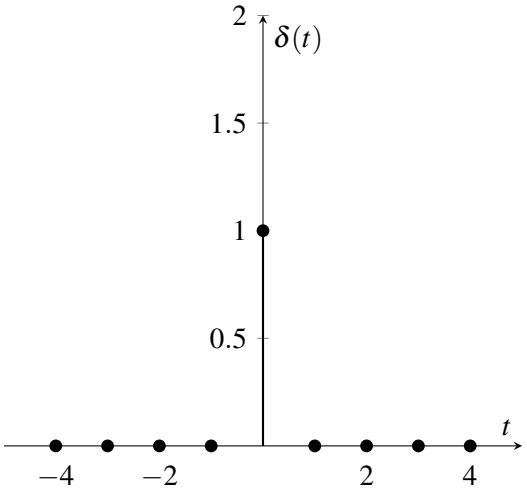
We could always proceed by trial and error – that is, assume random numbers of states and using least-squares until we get a good fit. However, it turns out that there is a more principled way forward: *we can use the data to learn the number of states using SVD*.

In this problem **you will learn how to estimate the state dimension n using SVD** and the response of the system to a special kind of input – an impulse response.

Consider the following discrete-time function:

$$\delta(t) = \begin{cases} 1 & \text{if } t = 0 \\ 0 & \text{if } t \neq 0 \end{cases} \tag{3}$$

This function, $\delta(t)$, is called the *impulse function*. When you plot $\delta(t)$, it looks like this:



It we give our system the initial condition $\vec{x}(0) = \vec{0}$ and the input $u(t) = \delta(t)$, then the *response* of the system to the impulse, that is the output $y(t)$ corresponding to the input $u(t) = \delta(t)$, is called the *impulse response*.

Our dataset will be the first $2T$ time steps of the impulse response, that is $y(0), y(1), \ldots, y(2T-1)$. We will assume that $T > n$ (where n is the dimension of the state $x(t)$), which is a reasonable assumption to make if we have a long data trace.

(a) Since we are given the initial condition $\vec{x}(0) = \vec{0}$, the impulse response at $t = 0$ is simply $y(0) = C\vec{0} = 0$. However, once the impulse function hits the system, we will begin to observe a nonzero output. Given the initial condition $\vec{x}(t) = \vec{0}$ and the input $u(t) = \delta(t)$, **show that the impulse response can be written as** $y(t) = CA^{t-1}B$ for $t \geq 1$.

(b) We will arrange our data into the following matrix,

$$H = \begin{bmatrix} y(1) & y(2) & y(3) & \ldots & y(T) \\ y(2) & y(3) & y(4) & \ldots & y(T+1) \\ y(3) & y(4) & y(5) & \ldots & y(T+2) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ y(T) & y(T+1) & k(T+2) & \ldots & y(2T-1) \end{bmatrix}. \tag{4}$$

Matrices in this form (where the values are constant along the anti-diagonals) are called *Hankel matrices*. However, the name isn't super important: what's more important is that information about A, B, and C is hidden inside H. To see this, **show that** H **can be factored into the form**

$$H = \mathscr{O}_T \mathscr{C}_T, \tag{5}$$

where $\mathscr{O}_T$ is the observability matrix

$$\mathscr{O}_T = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{T-1} \end{bmatrix} \tag{6}$$

and $\mathscr{C}_T$ is the controllability matrix

$$\mathscr{C}_T = \begin{bmatrix} B & AB & A^2B & \ldots & A^{T-1}B. \end{bmatrix} \tag{7}$$

(c) Given the assumptions we made about our system, **argue that $\mathscr{O}_T$ has linearly independent columns and that $\mathscr{C}_T$ has linearly independent rows.**

(d) Since we know a bit about the rank of $\mathscr{O}_T$ and $\mathscr{C}_T$, we can reason about the rank of their product H. Recall that our goal is to show that the rank of H is equal to the state dimension n. In this part, you will provide the first part of a proof of this fact, which we will complete for you below. For this part, **prove that since $\mathscr{C}_T$ has linearly independent rows, the matrix $\mathscr{C}_T\mathscr{C}_T^\top$ is invertible.**

Now for the second part of the proof. Since $\mathscr{C}_T\mathscr{C}_T^\top$ is invertible, we know that

$$\dim(\text{colspan}(\mathscr{O}_T\mathscr{C}_T)) \geq \dim(\text{colspan}(\mathscr{O}_T\mathscr{C}_T\mathscr{C}_T^\top)) = \dim(\text{colspan}(\mathscr{O}_T I)) = \dim(\text{colspan}(\mathscr{O}_T)) = n. \quad (8)$$

In other words, we know $\text{rank}(\mathscr{O}_T\mathscr{C}_T) \geq n$.

Furthermore, since H is the product of two matrices with rank n, and multiplying matrices can't cause the rank to grow, we also know that $\text{rank}(\mathscr{O}_T\mathscr{C}_T) \leq n$, which means it must be true that $\text{rank}(\mathscr{O}_T\mathscr{C}_T) = n$. This means that $\text{rank}(H) = n$, which is what we wanted to show.

(e) Since we know that the rank of H is equal to the state dimension, and H is made up of our data, we can now use our data to compute the state dimension! All we have to do is construct H, and use SVD to compute its rank. Why do we need the SVD? Because numerically or due to noise in the data, the rank would always come out to be full if we tried to find it using Gaussian Elimination. The SVD is a useful way to robustly find the "effective rank" of a matrix.

Take a look at the IPython notebook associated with this problem. In the last example in the notebook, you will complete some code that will allow you to numerically estimate the state dimension of an unknown system using only a noisy impulse response. There are a bunch of other system ID examples in the notebook as well which will help you understand system ID and SVD further.

For this part, **Complete the code in Example 6 in the notebook and comment on what you observe**. You need only comment on Example 6, but you can certainly comment on the other examples, too!

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:

- Siddharth Iyer.
- Aditya Arun.
- Justin Yim.
- Titan Yuan.
- Yu-Yun Dai.
- Sanjit Batra.
- Anant Sahai.
- Alex Devonport.