EECS 16A Designing Information Devices and Systems I
Fall 2015 Anant Sahai, Ali Niknejad

Homework 3

This homework is due September 21, 2015, at Noon.

Note: Don't be intimidated by the amount of text in this HW. Some problem statements are long because we lead you towards the solution, and give examples along the way. Some problems have many parts, but each part takes only a small step toward the final solution.

Submission Format
Your homework submission should consist of **two** files.

- `hw3.pdf`: A single pdf file that contains all your answers (any handwritten answers should be scanned) as well as your ipython notebook saved as a pdf. You can do this by printing the ipython notebook page in your browser and selecting the save to pdf option (in Chrome, this is under `Destination`). Make sure any plots and results are showing. Also make sure you combine any separate pdf's into one file. There are many websites online that can do this.
- `hw3.ipynb` A single ipython notebook with all your code in it.

1. **Finding Null Spaces**

   (a) Consider the column vectors of any 3×5 matrix. What is the maximum possible number of linearly independent vectors you can pick from these column vectors?

   (b) Suppose we have the following 3×5 matrix after row reduction:

$$A = \begin{bmatrix} 1 & 1 & 0 & -2 & 3 \\ 0 & 0 & 1 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

   What is the minimum number of vectors spanning the range of A. Find a set of such vectors.

   (c) Recall that for every vector $\vec{x}$ in the null space of A, $A\vec{x} = \vec{0}$. The dimension of a the null space is the minimum number of vectors needed to span it. Find vectors that span the nullspace of A (the matrix in the previous part). What is the dimension of the nullspace of A?

   (d) Find vector(s) that span the null space of the following matrix:

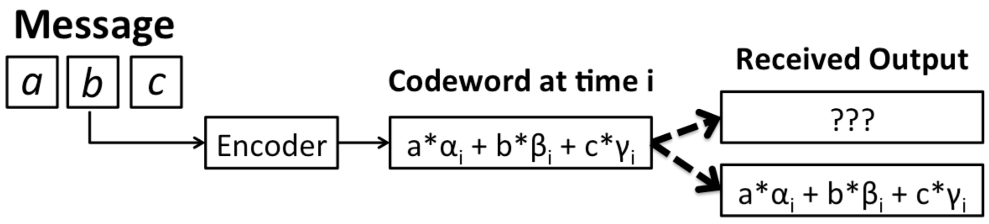
$$B = \begin{bmatrix} 1 & -2 & 2 & 4 \\ 1 & -2 & 3 & 5 \\ 2 & -4 & 5 & 9 \\ 3 & -6 & 7 & 13 \end{bmatrix}$$

2. **Fountain codes**
   Consider a sender, Alice, and a receiver, Bob. Alice wants to send a message to Bob, but the message is too big to send all at once. Instead, she breaks her message up into little chunks which she sends across a wireless channel one at a time (think radio transmitter to antenna). She knows some of the packets will be

corrupted or erased along the way (someone might turn a microwave on...), so she needs a way to protect her message from errors. This way, even if Bob gets a message missing parts of words he can still figure out what Alice is trying to say! One coding strategy is to use Fountain Codes. Fountain codes are a type of error-correcting codes based on principles of Linear Algebra. They were actually developed right here at Berkeley! The company that commercialized them, Digital Fountain, (started by a Berkeley grad, Mike Luby), was later acquired by Qualcomm. In this problem, we will explore some of the underlying principles that make Fountain codes work in a very simplified setting.

In this problem, we concentrate on the case with transmission erasures, i.e. where a bad transmission causes some parts the message to be erased. Let us say Alice wants to convey the set of her three favorite ideas covered in 16A lecture each day to Bob. For this, she maps each idea to a real number and wants to convey the 3-tuple $\begin{bmatrix} a & b & c \end{bmatrix}$ (Let us say there are an infinite number of ideas covered in 16A). At each time step, she can send one number, which we will call a "symbol" across. So one possible way for her to send the message is to first send a, then send b, and then send c. However, this method is particularly susceptible to losses. For instance, if the first symbol is lost, then Bob will receive $\begin{bmatrix} ? & b & c \end{bmatrix}$, and he will have no way of knowing what Alice's favorite idea is.

(a) The main idea in coding for erasures is to send redundant information so that we can recover from losses. So if we have three symbols of information, we might transmit six symbols for redundancy. One of the most naive codes is a called the repetition code. Here Alice would transmit $\begin{bmatrix} a & b & c & a & b & c \end{bmatrix}$. How much erasure can this transmission recover from? Are there specific patterns it cannot handle?



(b) A better strategy for transmission is to send linear combinations of symbols. Alice and Bob decide in advance on a collection of vectors $\vec{v}_i = \begin{bmatrix} \alpha_i \\ \beta_i \\ \gamma_i \end{bmatrix}$, $1 \le i \le 6$. These vectors define the code: at time i, Alice transmits the scalar

$$k_i = \begin{bmatrix} a & b & c \end{bmatrix} \vec{v}_i = a\alpha_i + b\beta_i + c\gamma_i.$$

How can you write the six transmitted symbols as a matrix multiplication in this case?

(c) What are the vectors $\begin{bmatrix} \alpha_i \\ \beta_i \\ \gamma_i \end{bmatrix}$, $1 \le i \le 6$ that generate the repetition code strategy in part (a)?

(d) Suppose now we choose a collection of seven vectors

$$\vec{v}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \vec{v}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \vec{v}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \vec{v}_4 = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}, \vec{v}_5 = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}, \vec{v}_6 = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}, \vec{v}_7 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

Again, at time i, Alice transmits the scalar $k_i = \begin{bmatrix} a & b & c \end{bmatrix} \vec{v}_i$. In this case, what patterns of losses let Bob recover the message?

(e) Suppose, using the collection of vectors in part (d), Bob receives $\begin{bmatrix} 6 & ? & ? & 2 & 3 & ? & ? \end{bmatrix}$. What was the transmitted message? Express the problem as a system of linear equations using matrix multiplication.

(f) Fountain codes build on these principles. The basic idea used by these codes is that Alice keeps sending linear combinations of symbols until Bob has received enough to decode. So at time 1, Alice sends the linear combination using $\vec{v}_1$, at time 2 she sends the linear combination using $\vec{v}_2$ and so on. After each new linear combination is sent, Bob sends back an acknowledgement to tell Alice whether or not he can decode her message (i.e. figure out the original $\begin{bmatrix} a & b & c \end{bmatrix}$ that she intended to communicate). So clearly, the minimum number of transmissions for Alice is 3. If Bob receives the first three linear combinations that are sent, Alice is done in three steps! But because of erasures, he might not. Suppose Alice used

$$\vec{v}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad \vec{v}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \quad \vec{v}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

as her first three vectors, but she has still not received an acknowledgement from Bob. Should she choose new vectors according to the strategy in part (a) or the strategy in part (d)? Why?

3. **Traffic Flows** Your goal is to measure the flow rates of vehicles along roads in a town. However, it is prohibitively expensive to place a traffic sensor along every road. You realize, however, that the number of cars flowing into an intersection must equal the number of cars flowing out. You can use this "flow conservation" to determine the traffic along all roads in a network by only measuring flow along only some roads. In this problem we will explore this concept.

(a) Let's begin with a network with three intersections, A, B and C. Define the flows t_1 as the rate of cars (cars/hour) on the road between B and A, t_2 as the rate on the road between C and B and t_3 as the rate on the road between C and A.

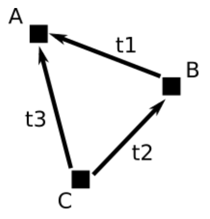


Figure 1: A simple road network.

(Note: The directions of the arrows in the figure are only the way that we define the flow by convention. If there were 100 cars per hour traveling from A to C, then $t_3 = -100$.)

We assume the "flow conservation" constraints: the total number of cars per hour flowing into each intersection is zero. For example at intersection B, we have the constraint $t_2 - t_1 = 0$. The full set of constraints (one per intersection) is:

$$\begin{cases} t_1 + t_3 & = 0 \\ t_2 - t_1 & = 0 \\ -t_3 - t_2 & = 0 \end{cases}$$

As mentioned earlier, we can place sensors a road to measure the flow through it. But, we have a limited budget, and we would like to determine all of the flows with the smallest possible number of sensors.

Suppose for the network above we have one sensor reading, $t_1 = 10$. Can we figure out the flows along the other roads? (That is, the values of t_2 and t_3).

(b) Now suppose we have a larger network, as shown in Figure 2.

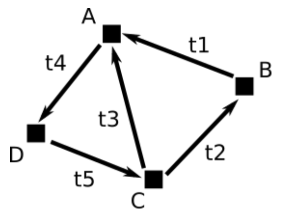


Figure 2: A larger road network.

We would again like to determine the traffic flows on all roads, using measurements from some sensors. A Berkeley student claims that we need two sensors placed on the roads AD and BA. A Stanford student claims that we need two sensors placed on the roads CB and BA. Is it possible to determine all traffic flows with the Berkeley student's suggestion? How about the Stanford student's suggestion?

(c) Suppose we write the traffic flow on all roads as a vector $\vec{t} = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \end{bmatrix}$. Show that the set of valid flows (which satisfy the conservation constraints) form a subspace. Then, determine the subspace of traffic flows for the network of Figure 2. Specifically, express this space as the span of two linearly independent vectors.

(Hint: Use the claim of the correct student in the previous part)

(d) We would like a more general way of determining the possible traffic flows in a network. As a first step, let us try to write all the flow conservation constraints (one per intersection) as a matrix equation. Find a (4×5) matrix B such that the equation $B\vec{t} = \vec{0}$:

$$\begin{bmatrix} & & \\ & B & \\ & & \end{bmatrix} \begin{bmatrix} t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

represents the flow conservation constraints for the network of Figure 2.

(Hint: Each row is the constraint of an intersection. You can construct B using only $0, 1, -1$ *entries.)*

This matrix is called the *incidence matrix*. What does each row of this matrix represent? What does each column of this matrix represent?

(e) Notice that the set of all vectors $\vec{t}$ which satisfy $B\vec{t} = \vec{0}$ is exactly the nullspace of the matrix B. That is, we can find all valid traffic flows by computing the nullspace of B. Use Guassian Elimination to determine the dimension of the nullspace of B, and compute a basis for the nullspace. (You may use a computer to compute reduced-row-echelon-form). Does this match your answer to part (c)? Can you interpret the dimension of the nullspace of the incidence matrix, for the road networks of Figure 1 and Figure 2?

(f) Now let us analyze general road networks. Say there is a road network graph G, with incidence matrix B_G. If B_G has a k-dimensional nullspace, does this mean measuring the flows along *any* k roads is always sufficient to recover the exact flows? Prove or give a counterexample.

(Hint: Consider the Stanford student.)

(g) Let G be a network of n roads, with incidence matrix B_G that has a k-dimensional nullspace. We would like to characterize exactly when measuring the flows along a set of k roads is sufficient to recover the exact flow along all roads. To do this, it will help to generalize the problem, and consider measuring *linear combinations* of flows. If $\vec{t}$ is a traffic flow vector, assume we can measure linear combinations $\vec{m}_i^T\vec{t}$ for some vectors $\vec{m}_i$. Then making k measurements is equivalent to observing the vector $M\vec{t}$ for some $(k \times n)$ "measurement matrix" M (consisting of rows $\vec{m}_i^T$).

For example, for the network of Figure 2, the measurement matrix corresponding to measuring t_1 and t_4 (as the Berkeley student suggests) is:

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

Similarly, the measurement matrix corresponding to measuring t_1 and t_2 (as the Stanford student suggests) is:

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

For general networks G and measurements M, give a condition for when the exact traffic flows can be recovered, in terms of the nullspace of M and the nullspace of B_G.

(Hint: Recovery will fail iff there are two valid flows with the same measurements. Can you express this in terms of the nullspaces of M and B_G?)

(h) *(optional)* Express the condition of the previous part in a way that can be checked computationally. For example, suppose we are given a huge road network G of all roads in Berkeley, and we want to find if our measurements M are sufficient to recover the flows.

(Hint: Consider a matrix U whose columns form a basis of the nullspace of B_G. Then $\{U\vec{x} : \vec{x} \in R^k\}$ is exactly the set of all possible traffic flows. How can we represent measurements on these flows?)

(i) *(optional)* If the incidence matrix B_G has a k-dimensional nullspace, does this mean we can always pick a set of k roads such that measuring the flows along these roads is sufficient to recover the exact flows? Prove or give a counterexample.

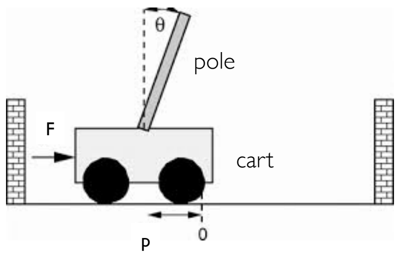
4. Bieber's Segway

After one too many infractions with the law J-Bieb's decides to find a new mode of transportation, and you suggest he get a segway.

He becomes intrigued by your idea and asks you how it works.

You let him know that a force (through the spinning wheel) is applied to the base of the segway, and this in turn controls both the position of the segway and angle of the stand. As you push on the stand the segway tries to bring itself back to the upright position, and it can only do this by moving the base.

J-Bieb's is impressed, to say the least, but he is a little concerned that only one input (force) is used to control two outputs (position and angle). He finally asks if it's possible for the segway to be brought upright and to a stop from any initial configuration. J-Bieb's calls up a friend who's majoring in mechanical engineering, who tells him that a segway can be modeled as a cart-pole system:



A cart-pole system can be fully described by its position p, velocity $\dot{p}$, angle θ, and angular velocity $\dot{\theta}$. We write this as a “state vector":

$$\vec{x} = \begin{bmatrix} p \\ \dot{p} \\ \theta \\ \dot{\theta} \end{bmatrix} \tag{1}$$

The input to this system u will just be the force applied to the cart (or base of the segway). At time step n, we can apply scalar input $u[n]$. The cart-pole system can be represented by a linear model:

$$\vec{x}[n+1] = A\vec{x}[n] + \vec{b}u[n], \tag{2}$$

where $A \in \mathbb{R}^{4\times 4}$ and $\vec{b} \in \mathbb{R}^{4\times 1}$. The model tells us how the the state vector will evolve over (discrete) time as a function of the current state vector and control inputs.

To answer J-Bieb’s question, you look at this general linear system, and try to answer the following question: Starting from some initial state $\vec{x}_0$, can we reach a final desired state, $\vec{x}_f$ in N steps?

The challenge seems to be that the state is 4-dimensional and keeps evolving and we can only apply a one dimensional control at each time. Is it possible to control something 4-dimensional with only one degree of freedom that we can command?

You solve this problem by walking through several steps.

(a) Express $\vec{x}[1]$ in terms of $\vec{x}[0]$ and the input $u[0]$.

(b) Express $\vec{x}[2]$ in terms of *only* $\vec{x}[0]$ and the inputs, $u[0]$ and $u[1]$.

(c) Express $\vec{x}[3]$ in terms of *only* $\vec{x}[0]$ and inputs $u[0]$, $u[1]$ and $u[2]$.

(d) Express $\vec{x}[4]$ in terms of *only* $\vec{x}[0]$ and inputs $u[0]$, $u[1]$, $u[2]$ and $u[3]$.

(e) Now, derive an expression for $\vec{x}[N]$ in terms of $\vec{x}[0]$ and the inputs from $u[0]...u[N-1]$. (*Hint: You can use a sum from* 0 *to* $N-1$.)

For the next four parts of the problem, you are given [1] the matrix A and the vector b:

$$A = \begin{bmatrix} 1 & 0.05 & -0.01 & 0 \\ 0 & 0.22 & -0.17 & -0.01 \\ 0 & 0.10 & 1.14 & 0.10 \\ 0 & 1.66 & 2.85 & 1.14 \end{bmatrix} \tag{3}$$

$$\vec{b} = \begin{bmatrix} 0.01 \\ 0.21 \\ -0.03 \\ -0.44 \end{bmatrix} \tag{4}$$

The state vector $\vec{0}$ corresponds to the cart-pole (or segway) being upright and stopped at the origin. Assume the cart-pole starts in an initial state $\vec{x}[0] = \begin{bmatrix} -0.3853493 \\ 6.1032227 \\ 0.8120005 \\ -14 \end{bmatrix}$, and you want to reach the desired state $\vec{x_f} = \vec{0}$ using the control inputs $u[0], u[1], \ldots$. (Reaching $\vec{x_f} = \vec{0}$ in N steps means that, given we start at $\vec{x}[0]$, the state vector in the Nth time step, we can find control inputs such that we get $\vec{x}[N]$ equal to $\vec{x_f}$.)

Note: You can use IPython to solve the next four parts of the problem.

(f) Can you reach $\vec{x_f}$ in *two* time steps? (*Hint: Express* $\vec{x}[2] - A^2\vec{x}[0]$ *in terms of the inputs* $u[0]$ *and* $u[1]$.)

(g) Can you reach $\vec{x_f}$ in *three* time steps?

(h) Can you reach $\vec{x_f}$ in *four* time steps?

(i) If the answer to the previous part is yes, find the required correct control inputs using IPython, and verify the answer by entering these control inputs into the code in the IPython notebook. The code has been written to simulate this system, and you should see the system come to a halt in four time steps! **Suggestion: See what happens if you enter all four control inputs equal to 0. This gives you an idea of how the system naturally evolves!**

(j) Let's return to a general matrix A and a general vector b. What condition do we need on

$$\text{span}\{\vec{b}, A\vec{b}, A^2\vec{b}, \ldots, A^{N-1}\vec{b}\}$$

for $\vec{x_f} = \vec{0}$ to be "reachable" from $\vec{x_0}$ in N steps?

(k) What condition would we need on $span\{\vec{b}, A\vec{b}, A^2\vec{b}, \ldots, A^{N-1}\vec{b}\}$ for *any* valid state vector to be reachable from $\vec{x_0}$ in N steps?

Wouldn't this be cool?

5. **Your Own Problem** Write your own problem related to this week's material and solve it. You may still work in groups to brainstorm problems, but each student should submit a unique problem. What is the problem? How to formulate it? How to solve it? What is the solution?

[1] Some of you might be wondering why it is that applying a force in this model immediately causes a change in position. You might have been taught in high school physics that force creates acceleration, which changes velocity (both simple and angular), which in turn causes changes to position and angle. Indeed, when viewed in continuous time this is true instantaneously. However here in this engineering system, we have discretized time, i.e. we think about applying this force constantly for a given finite duration and we see how the system responds after that finite duration. In this finite time, indeed the application of a force will cause changes to all four components of the state. But notice that the biggest influence is indeed on the two velocities, as should comport with your intuition from high school physics.