EECS 16A Designing Information Devices and Systems I
Spring 2016 Elad Alon, Babak Ayazifar Homework 3

This homework is due February 16, 2015, at Noon.

Optional Problems: We **do not** grade these problems. Nevertheless, you are responsible for learning the subject matter within their scope.
Bonus Problems: We **do** grade these problems. Doing them will provide an unspecified amount of extra credit; not doing them will not affect your homework grade negatively. We will specify if the problem is in or out of scope.

1. Homework process and study group

Who else did you work with on this homework? List names and student ID's. (In case of hw party, you can also just describe the group.) How did you work on this homework?
Working in groups of 3-5 will earn credit for your participation grade.

Solution: I worked on this homework with...

I first worked by myself for 2 hours, but got stuck on Problem 5 so I went to office hours on...

Then I went to homework party for a few hours, where I finished the homework.

2. Mechanical Inverses (Optional)

For each of the following matrices, state whether the inverse exists. If so, find the inverse. If not, prove that no inverse exists. **Solve these by hand, do NOT use numpy!**

(a) $\begin{bmatrix}1 & 1\\ 2 & 0\end{bmatrix}$ **Solution:** Exists. $\begin{bmatrix}0 & \frac{1}{2}\\ -1 & \frac{1}{2}\end{bmatrix}$

(b) $\begin{bmatrix}3 & 2\\ 1 & -1\end{bmatrix}$ **Solution:** Exists $\begin{bmatrix}\frac{1}{5} & \frac{2}{5}\\ \frac{1}{5} & \frac{-3}{5}\end{bmatrix}$

(e) $\begin{bmatrix}1 & 0 & 0\\ 0 & -1 & 1\\ 0 & 1 & 0\end{bmatrix}$ **Solution:** Exists $\begin{bmatrix}1 & 0 & 0\\ 0 & 0 & 1\\ 0 & 1 & 1\end{bmatrix}$

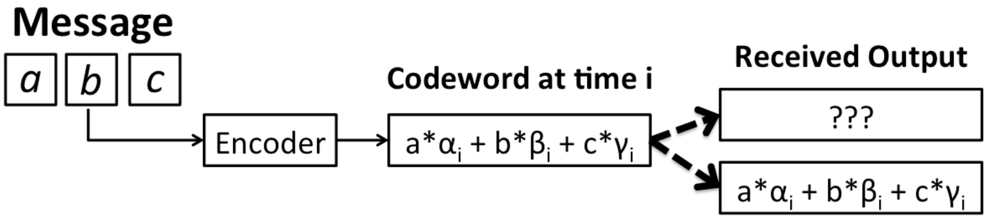
(f) $\begin{bmatrix}1 & 0 & 0\\ 0 & -1 & 1\\ 0 & 1 & -1\end{bmatrix}$ **Solution:** Does not exist. $v_3 = -v_2$. Since the c

3. 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?


Message
a
b
c
Encoder
Codeword at time i
$a*\alpha_i + b*\beta_i + c*\gamma_i$
Received Output
???
$a*\alpha_i + b*\beta_i + c*\gamma_i$

Solution: This pattern is robust as long as we have one of each point below:

- first or fourth entry
- second or fifth entry
- third or sixth entry

This means if we have a channel that erases every third element we cannot recover the message. If it erases every third element and even more, then we obviously also cannot recover the message since more erasures just make things worse.

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

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

Formulate the setup / the six transmitted symbols using matrix/vector notation.

Solution: Let T_X be the transmitted message (output of the encoding), we can write the encoding as the operation below:

$$T_X = \begin{bmatrix} a & b & c \end{bmatrix} \cdot \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 & \alpha_5 & \alpha_6 \\ \beta_1 & \beta_2 & \beta_3 & \beta_4 & \beta_5 & \beta_6 \\ \gamma_1 & \gamma_2 & \gamma_3 & \gamma_4 & \gamma_5 & \gamma_6 \end{bmatrix}$$

Notice here that we are using a row-vector convention for transmitted codewords and underlying messages. If you did the transpose of everything and followed a column-vector convention that would also be full credit.

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

Solution: The vectors are as below:

$$v_1 = v_4 = \begin{bmatrix}1 & 0 & 0\end{bmatrix}^T$$

$$v_2 = v_5 = \begin{bmatrix}0 & 1 & 0\end{bmatrix}^T$$

$$v_3 = v_6 = \begin{bmatrix}0 & 0 & 1\end{bmatrix}^T$$

(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 $k_i = \begin{bmatrix}a & b & c\end{bmatrix}\vec{v}_i$. Under what conditions, (i.e. what patterns of losses) can Bob still recover the message?

Solution:

Clearly, at least three $\vec{v}_i$'s are needed to find a, b, c, so Bob cannot recover the message if more than four losses occur. Can he recover when there are exactly four losses? Sometimes yes and sometimes no. For example, $\{\vec{v}_1, \vec{v}_2, \vec{v}_3\}$ are clearly sufficient to recover the message, but $\{\vec{v}_1, \vec{v}_2, \vec{v}_4\}$ are insufficient because $\vec{v}_4 = \vec{v}_1 + \vec{v}_2$.

If fewer than four losses occur, the message can always be recovered. To see this, divide the $\vec{v}_i$'s into three groups:

- $G_1 = \{\vec{v}_1, \vec{v}_2, \vec{v}_3\}$
- $G_2 = \{\vec{v}_4, \vec{v}_5, \vec{v}_6\}$
- $G_3 = \{\vec{v}_7\}$

and consider the following cases when there are three losses (i.e. four successful transmissions):

- All three members from G_1 are transmitted: clearly $\vec{v}_1, \vec{v}_2, \vec{v}_3$ are sufficient to decode.
- Two members from G_1 are transmitted: only one of the vectors in G_2 is linearly dependent on two vectors from G_1 (that is the sum of the two vectors), and $\vec{v}_7$ is linearly independent of any two from G_1, so with four successful transmission you are guaranteed to have an extra linearly independent vector and will be able to decode.
- Only one member from G_1 is transmitted: then all except one of G_2 and G_3 were successful. If $\vec{v}_7$ was unsuccessful, then all of G_2 succeeded, and G_2 are all linearly independent and sufficient to decode. If $\vec{v}_7$ was successful, $\vec{v}_7$ is linearly independent of every set of two vectors from G_2, so we have three linearly independent vectors and can decode.
- Zero members of G_1 are transmitted: that means all of G_2 was successful, which has three linearly independent vectors, so Bob can decode.

We have provided iPython code in the accompanying notebook that reproduces the message in the next subproblem using every combination of 4 vectors. This is just for your benefit, we didn't expect you to necessarily do this yourself.

Full credit on this subproblem must identify that this scheme can always recover from three losses and sometimes four. If you did not fully justify why, do not take off credit. We didn't expect an explanation this detailed.

(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/vector notation.

Solution: Since we are using the 1st, 4th and 5th vectors, we can represent the transmission as the matrix multiplication below:

$$\begin{bmatrix} a & b & c \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 6 & 2 & 3 \end{bmatrix}$$

which can also be written as a system of linear equations

$$\begin{cases} a = 6 \\ a + b = 2 \\ a + c = 3 \end{cases}$$

The solution to this equation is $a = 6, b = -4, c = -3$.

(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 will send back an acknowledgement *if* 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, she might hear nothing (he hasn't decoded yet.) 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?

Solution: We can see the strategy in part (d) is more robust than that of part (a), since the strategy in part (d) can recover from more patterns of losses than the strategy in part (a) can. (d) can recover from any three losses, whereas (a) cannot.

Real-world fountain-codes are based on the idea of constantly generating new vectors $\vec{v}_i$ according to a known random seed. This way, they constantly provide fresh new mixtures of the underlying information. As soon as the receiver gets enough linearly independent measurements, it can decode. These sorts of ideas are also used in file-sharing and content-dissemination networks. You can learn more about those applications in EECS 121. More material on error-correcting codes will also appear in 16B as well as in the third course in this sequence, 70. In 16B, you will see a connection between polynomials and subspaces that can be used to build codes robust to erasures. And in 70, you will learn about finite fields that show how these ideas can work in the digital world without real numbers.

4. Cubic Polynomials (Optional)

Consider the set of real-valued canonical polynomials given below:

$$\varphi_0(t) = 1 \quad \varphi_1(t) = t \quad \varphi_2(t) = t^2 \quad \varphi_3(t) = t^3,$$

where $t \in [a, b] \subset \mathbb{R}$.

(a) Show that the set of all cubic polynomials

$$p(t) = p_0 + p_1 t + p_2 t^2 + p_3 t^3,$$

where $t \in [a,b]$ and the coefficients p_k are real scalars, forms a vector space. What is the dimension of this vector space? Explain.

Solution:

No escape (scaling) property:
$\alpha p(t) = \alpha p_0 + \alpha p_1 t + \alpha p_2 t^2 + \alpha p_3 t^3$
The function above is, itself, a cubic polynomial, so the no escape property holds.

No escape (addition) property:
Let us define two cubic polynomials:
$p(t) = p_0 + p_1 t + p_2 t^2 + p_3 t^3$
$q(t) = q_0 + q_1 t + q_2 t^2 + q_3 t^3$
$p(t) + q(t) = (p_0 + q_0) + (p_1 + q_1)t + (p_2 + q_2)t^2 + (p_3 + q_3)t^3$
The function above is also a cubic polynomial, so this no escape property also holds.

Since both no escape properties hold, the set of cubic polynomials is a vector space. The dimension of this space is 4 because there are 4 degrees of freedom that can be varied.

(b) Show that every real-valued cubic polynomial

$$p(t) = p_0 + p_1 t + p_2 t^2 + p_3 t^3$$

defined over the interval $[a,b]$ can be written as a linear combination of the canonical polynomials $\varphi_0(t)$, $\varphi_1(t)$, $\varphi_2(t)$, and $\varphi_3(t)$. In particular, show that

$$p(t) = c^{\mathsf{T}} \varphi(t),$$

where

$$c^{\mathsf{T}} = [c_0\ c_1\ c_2\ c_3]$$

is a vector of appropriately chosen coefficients, and

$$\varphi(t) = \begin{bmatrix} \varphi_0(t) \\ \varphi_1(t) \\ \varphi_2(t) \\ \varphi_3(t) \end{bmatrix} = \begin{bmatrix} 1 \\ t \\ t^2 \\ t^3 \end{bmatrix}.$$

(c) Determine the coefficients c_k for $k = 0,1,2,3$. That is to say, for the setup above determine what values would be in the c vector.

(d) True or False? The canonical polynomials $\varphi_k(t) = t^k$, for $k = 0,1,2,3$, constitute a basis for the vector space of real-valued cubic polynomials defined over the interval $[a,b]$. Briefly justify your answer

Solution:

$$\begin{bmatrix} c_0 & c_1 & c_2 & c_3 \end{bmatrix} \begin{bmatrix} 1 \\ t \\ t^2 \\ t^3 \end{bmatrix} = c_0 + c_1 t + c_2 t^2 + c_3 t^3$$

$c_0 = p_0\ c_1 = p_1\ c_2 = p_2\ c_3 = p_3$
True, the canonical polynomials constitute a basis.

(e) A curve is a continuous mapping from the real line to $\mathbb{R}^N$. A cubic Bézier curve—used extensively in computer graphics—is a type of curve that uses as a basis the following special subset of what are more broadly known as *Bernstein polynomials*:

$$\beta_0(t) = (1-t)^3, \quad \beta_1(t) = 3t(1-t)^2, \quad \beta_2(t) = 3t^2(1-t), \quad \text{and } \beta_3(t) = t^3.$$

Prove that the Bernstein polynomials $\beta_k(t)$ defined above do indeed form a basis. To do this, show that any real-valued polynomial

$$p(t) = p_0 + p_1 t + p_2 t^2 + p_3 t^3$$

can be expressed as a linear combination of the polynomials $\beta_k(t)$, and determine the coefficients in that linear combination. In particular, determine the coefficients in the expansion

$$p(t) = \widehat{p}_0\beta_0(t) + \widehat{p}_1\beta_1(t) + \widehat{p}_2\beta_2(t) + \widehat{p}_3\beta_3(t)$$

.

Hint: Determine a matrix R such that

$$\beta(t) = R\,\varphi(t),$$

where

$$\beta(t) = \begin{bmatrix} \beta_0(t) \\ \beta_1(t) \\ \beta_2(t) \\ \beta_3(t) \end{bmatrix} = \begin{bmatrix} (1-t)^3 \\ 3t(1-t)^2 \\ 3t^2(1-t) \\ t^3 \end{bmatrix}$$

and show that R is invertible. Determine its inverse R^{-1}, from which you can determine the coefficients $\widehat{p}_k$.

Solution: Let us define the vectors $p = \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix}$ and $\widehat{p} = \begin{bmatrix} \widehat{p}_0 \\ \widehat{p}_1 \\ \widehat{p}_2 \\ \widehat{p}_3 \end{bmatrix}$. We can find $\widehat{p}$ from p as follows:

$\widehat{p}^T\beta = p^T\varphi$
$\widehat{p}^T\beta = p^T R^{-1}\beta$
$\widehat{p}^T = p^T R^{-1}$

We can find the matrix R from the definitions of φ_k and β_k.

$$\begin{bmatrix} (1-t)^3 \\ 3t(1-t)^2 \\ 3t^2(1-t) \\ t^3 \end{bmatrix} = \begin{bmatrix} 1-3t+3t^2-t^3 \\ 3t-6t^2+3t^3 \\ 3t^2-3t^3 \\ t^3 \end{bmatrix} = \begin{bmatrix} 1 & -3 & 3 & -1 \\ 0 & 3 & -6 & 3 \\ 0 & 0 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ t \\ t^2 \\ t^3 \end{bmatrix}$$

The above matrix, R, is clearly invertible since there is a pivot in each column. We proceed to find the inverse:

$$\left[\begin{array}{cccc|cccc} 1 & -3 & 3 & -1 & 1 & 0 & 0 & 0 \\ 0 & 3 & -6 & 3 & 0 & 1 & 0 & 0 \\ 0 & 0 & 3 & -3 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{array}\right]$$

$$\tfrac{1}{3}\text{R3+R4} \rightarrow \text{R3} \quad \left[\begin{array}{cccc|cccc} 1 & -3 & 3 & -1 & 1 & 0 & 0 & 0 \\ 0 & 3 & -6 & 3 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & \frac{1}{3} & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{array}\right]$$

$$\tfrac{1}{3}\text{R2+2R3} \rightarrow \text{R2} \left[\begin{array}{cccc|cccc} 1 & -3 & 3 & -1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & \frac{1}{3} & \frac{2}{3} & 2 \\ 0 & 0 & 1 & 0 & 0 & 0 & \frac{1}{3} & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{array}\right]$$

$$\text{R2-R4} \rightarrow \text{R2} \left[\begin{array}{cccc|cccc} 1 & -3 & 3 & -1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & \frac{1}{3} & \frac{2}{3} & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & \frac{1}{3} & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{array}\right]$$

$$\text{R1+3R2} \rightarrow \text{R1} \left[\begin{array}{cccc|cccc} 1 & 0 & 3 & -1 & 1 & 1 & 2 & 3 \\ 0 & 1 & 0 & 0 & 0 & \frac{1}{3} & \frac{2}{3} & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & \frac{1}{3} & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{array}\right]$$

$$\text{R1+R4} \rightarrow \text{R1} \left[\begin{array}{cccc|cccc} 1 & 0 & 3 & 0 & 1 & 1 & 2 & 4 \\ 0 & 1 & 0 & 0 & 0 & \frac{1}{3} & \frac{2}{3} & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & \frac{1}{3} & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{array}\right]$$

$$\text{R1-3R3} \rightarrow \text{R1} \left[\begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & \frac{1}{3} & \frac{2}{3} & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & \frac{1}{3} & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{array}\right]$$

$$R^{-1} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & \frac{1}{3} & \frac{2}{3} & 1 \\ 0 & 0 & \frac{1}{3} & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$p^T R^{-1} = \begin{bmatrix} p_0 \\ p_0 + \frac{1}{3}p_1 \\ p_0 + \frac{2}{3}p_1 \\ \frac{1}{3}p_2 \end{bmatrix}$$

Thus,

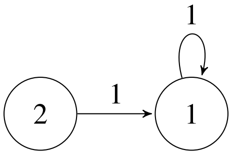
$\widehat{p_0} = p_0$

$\widehat{p_1} = p_0 + \frac{1}{3}p_1$

$\widehat{p_2} = p_0 + \frac{2}{3}p_1 + \frac{1}{3}p_2$

$\widehat{p_3} = p_0 + p_1 + p_2 + p_3$

5. Pumps Properties Proofs Part 2 (Optional)



Figure

(a) Suppose we have a pump setup as in Figure , with associated matrix A. Write out the state transition matrix A.

$$\begin{bmatrix} \quad A \quad \end{bmatrix} \begin{bmatrix} x[n] \end{bmatrix} = \begin{bmatrix} x[n+1] \end{bmatrix}$$

Solution:

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

(b) Suppose the reservoirs are initialized to have the following water levels: $x_1[0] = 0.5$, $x_2[0] = 0.5$. In a completely alternate universe the pools are initialized to have the following water levels: $x_1[0] = 0.3$, $x_2[0] = 0.7$. For both initial states, what are the water levels at timestep 1 ($x_1[1], x_2[1]$)?

Solution:

$$\begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1[0] \\ x_2[0] \end{bmatrix} = \begin{bmatrix} x_1[1] \\ x_2[1] \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0.3 \\ 0.7 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

(c) If you observe the pools at timestep 1, can you figure out what the initial ($x_1[0], x_2[0]$) water levels were? Why or why not?

Solution:

No, at timestep 1 configuration 1 and 2 are indistinguishable. This implies the experiment inherently has linear dependence.

(d) Now generalize: if there exists a state transition matrix where two different initial state vectors lead to the same water levels/state vectors at a timestep in the future, can you recover the initial water levels? Prove your answer.

Solution: We are given that two different initial states, $x[0]$ and $y[0]$ lead to the same resulting state $x[1]$.

$Ax[0] = x[1]$

$Ay[0] = x[1]$

If we can recover the initial water levels, this would mean that there is some operation that can be performed on the vector representing the resulting state to yield the previous state. This operation would, by definition, be the inverse of the matrix A.

$x[0] = A^{-1}x[1]$

$y[0] = A^{-1}x[1]$

However, we can see that the above two equations contradict each other. Therefore, A is not invertible, and you cannot recover the previous state from the resulting state.

(e) Suppose there is a 2-pixel imaging experiment with a measurement matrix that happens to have the same numbers as the pump state transition matrix. In other words, the pump configuration also determines the measurements you take for a 2-pixel imaging experiment that is in all other respects completely unrelated. What do the answers to the previous subparts of this question imply about this experimental setup?

Solution: This implies that two different objects could yield the same image.

(f) Suppose there is a pump state transition matrix (and corresponding experiment matrix, similar to the last subpart) where every initial state is guaranteed to have a unique state vector in the next timestep.

Can you recover the initial state? Prove your answer, and explain what this implies about the experiment.

If you're not sure where to begin, consider the following: what does this statement imply about the system of linear equations represented by the state transition matrix A?

Solution: We are given that each input $x[i]$ leads to a unique output $x[i+1]$ when the matrix A is applied. This means that the set of all possible $x[i]$ vectors maps one-to-one to the set of all possible $x[i+1]$ through the linear transformation A. Given these sets, there must be some linear transformation, B, that maps each individual $x[i+1]$ to the corresponding $x[i]$

$x[i] = Bx[i+1]$

We only know B exists if each unique input to A leads to a unique output. This is because a linear transformation, such as B, cannot give different outputs for the same input. Plugging in our original matrix equation

$Ax[i] = x[i+1]$

Yields

$x[i] = BAx[i]$

Therefore, BA must be the identity matrix. Thus, $B = A^{-1}$

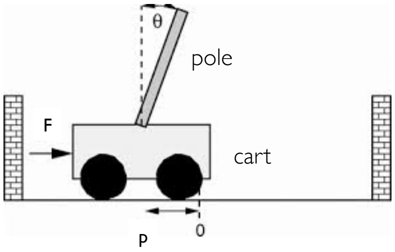
6. Bieber's Segway

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":

$$\vec{x} = \begin{bmatrix} p \\ \dot{p} \\ \theta \\ \dot{\theta} \end{bmatrix} \quad (1)$$

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 $u[0]$.

Solution: From Equation 2, we get (by substituting $n = 0$):

$$\vec{x}[1] = A\vec{x}[0] + \vec{b}u[0] \tag{3}$$

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

Solution: From Equation 2, we get (by substituting $n = 1$):

$$\vec{x}[2] = A\vec{x}[1] + \vec{b}u[1] \tag{4}$$

By substituting $\vec{x}[1]$ from Equation 3 we get

$$\begin{aligned} \vec{x}[2] &= A\vec{x}[1] + \vec{b}u[1] \\ &= A(A\vec{x}[0] + \vec{b}u[0]) + \vec{b}u[1] \\ &= A^2\vec{x}[0] + A\vec{b}u[0] + \vec{b}u[1] \end{aligned} \tag{5}$$

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

Solution: From Equation 2, we get (by substituting $n = 2$):

$$\vec{x}[3] = A\vec{x}[2] + \vec{b}u[2] \tag{6}$$

By substituting $\vec{x}[2]$ from Equation 5 we get

$$\begin{aligned} \vec{x}[3] &= A\vec{x}[2] + \vec{b}u[2] \\ &= A(A^2\vec{x}[0] + A\vec{b}u[0] + \vec{b}u[1]) + \vec{b}u[1] \\ &= A^3\vec{x}[0] + A^2\vec{b}u[0] + A\vec{b}u[1] + \vec{b}u[2] \end{aligned} \tag{7}$$

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

Solution: From Equation 2, we get (by substituting $n = 3$):

$$\vec{x}[4] = A\vec{x}[3] + \vec{b}u[3] \tag{8}$$

By substituting $\vec{x}[3]$ from Equation 7 we get

$$\begin{aligned}\vec{x}[4] &= A\vec{x}[3] + \vec{b}u[3] \\ &= A(A^3\vec{x}[0] + A^2\vec{b}u[0] + A\vec{b}u[1] + \vec{b}u[2]) + \vec{b}u[1] \\ &= A^4\vec{x}[0] + A^3\vec{b}u[0] + A^2\vec{b}u[1] + A\vec{b}u[2] + \vec{b}u[3]\end{aligned} \tag{9}$$

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

Solution: Use the same procedure as above for N steps. You will obtain the following expression:

$$\vec{x}[N] = A^N x[0] + \sum_{i=0}^{N-1} A^i \vec{b}u[N-i-1] \tag{10}$$

Note that A^0 is the identity matrix.

As a sanity check, plug the values $N = 1, 2, 3$ and 4 to obtain Equations 3,5,7 and 9, respectively.

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

$$A = \begin{bmatrix} 1 & 0.05 & -0.01 & 0 \\ 0 & 0.22 & -0.17 & -0.01 \\ 0 & 0.10 & 1.14 & 0.10 \\ 0 & 1.66 & 2.85 & 1.14 \end{bmatrix} \tag{11}$$

$$\vec{b} = \begin{bmatrix} 0.01 \\ 0.21 \\ -0.03 \\ -0.44 \end{bmatrix} \tag{12}$$

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 $u[0], u[1], \dots$. (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.

[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.

(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* $u[1]$.)

Solution: No.

From Equation 5 we know that $A^2\vec{x}[0]+A\vec{b}u[0]+\vec{b}u[1]=\vec{x}[2]$ which is equivalent to $A\vec{b}u[0]+\vec{b}u[1]=\vec{x}[2]-A^2\vec{x}[0]$.

This means that in order to reach any state $\vec{x}_f$ in two time steps (that is, $\vec{x}[2]=\vec{x}_f$), we have to solve the following system of linear equations

$$A\vec{b}u[0]+\vec{b}u[1]=\vec{x}_f-A^2\vec{x}[0] \tag{13}$$

where $u[0]$ and $u[1]$ are the unknowns.

Since in our case we want to reach $\vec{x}_f=\begin{bmatrix}0\\0\\0\\0\end{bmatrix}$ then the system of linear equations simplifies to

$$A\vec{b}u[0]+\vec{b}u[1]=-A^2\vec{x}[0] \tag{14}$$

In matrix form, this system of linear equations is

$$\begin{bmatrix}| & | \\ A\vec{b} & \vec{b} \\ | & |\end{bmatrix}\cdot\begin{bmatrix}u[0]\\u[1]\end{bmatrix}=-A^2\vec{x}[0] \tag{15}$$

which yields the following augmented matrix

$$\left[\begin{array}{cc|c}| & | & | \\ A\vec{b} & \vec{b} & -A^2\vec{x}[0] \\ | & | & |\end{array}\right] \tag{16}$$

By plugging in the values of A, $\vec{b}$ and $\vec{x}[0]$ we get the following augmented matrix

$$\left[\begin{array}{cc|c}0.0208 & 0.01 & 0.02243475295 \\ 0.0557 & 0.21 & -0.30785116611 \\ -0.0572 & -0.03 & 0.0619347608 \\ -0.2385 & -0.44 & 1.38671325508\end{array}\right] \tag{17}$$

Applying Gaussian elimination (you could have done this with any software of your choice, we used the code from the gaussian elimination module that we sent out to the class) we get the reduced row echelon form to be

$$\left[\begin{array}{cc|c}1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0\end{array}\right] \tag{18}$$

Which means that the system is inconsistent (third row) and there are no solutions for $u[0]$ and $u[1]$. It is fine if you did not row reduce all the way to the reduced row echelon form, as long as you showed that the system of equations was inconsistent.

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

Solution: No.

Similar to the previous part, from Equation 7 we know that $A^3\vec{x}[0]+A^2\vec{b}u[0]+A\vec{b}u[1]+\vec{b}u[2]=\vec{x}[3]$ which is equivalent to $A^2\vec{b}u[0]+A\vec{b}u[1]+\vec{b}u[2]=\vec{x}[3]-A^3\vec{x}[0]$.

This means that in order to reach any state $\vec{x}_f$ in three time steps (that is, $\vec{x}[3]=\vec{x}_f$), we have to solve the following system of linear equations

$$A^2\vec{b}u[0]+A\vec{b}u[1]+\vec{b}u[2]=\vec{x}_f-A^3\vec{x}[0] \tag{19}$$

where $u[0],u[1]$ and $u[2]$ are the unknowns.

Since in our case we want to reach $\vec{x}_f=\begin{bmatrix}0\\0\\0\\0\end{bmatrix}$ then the system of linear equations simplifies to

$$A^2\vec{b}u[0]+A\vec{b}u[1]+\vec{b}u[2]=-A^3\vec{x}[0] \tag{20}$$

In matrix form, this system of linear equations is

$$\begin{bmatrix} | & | & | \\ A^2\vec{b} & A\vec{b} & \vec{b} \\ | & | & | \end{bmatrix}\cdot\begin{bmatrix}u[0]\\u[1]\\u[2]\end{bmatrix}=-A^3\vec{x}[0] \tag{21}$$

which yields the following augmented matrix

$$\left[\begin{array}{ccc|c} | & | & | & | \\ A^2\vec{b} & A\vec{b} & \vec{b} & -A^3\vec{x}[0] \\ | & | & | & | \end{array}\right] \tag{22}$$

By plugging in the values of A, $\vec{b}$ and $\vec{x}[0]$ we get the following augmented matrix

$$\left[\begin{array}{ccc|c} 0.024157 & 0.0208 & 0.01 & 0.0064228470365 \\ 0.024363 & 0.0557 & 0.21 & -0.092123298431 \\ -0.083488 & -0.0572 & -0.03 & 0.178491836209001 \\ -0.342448 & -0.2385 & -0.44 & 1.246334243328597 \end{array}\right] \tag{23}$$

Applying Gaussian elimination (you could have done this with any software of your choice, we used the code from the gaussian elimination module that we sent out to the class) we get the reduced row echelon form to be

$$\left[\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{array}\right] \tag{24}$$

Which means that the system is inconsistent (fourth row) and there are no solutions for $u[0],u[1]$ and $u[2]$. It is fine if you did not row reduce all the way to the reduced row echelon form, as long as you showed that the system of equations was inconsistent.

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

Solution: Yes.

Similar to the previous part, from Equation 9 we know that $A^4\vec{x}[0]+A^3\vec{b}u[0]+A^2\vec{b}u[1]+A\vec{b}u[2]+\vec{b}u[3]=\vec{x}[4]$ which is equivalent to $A^3\vec{b}u[0]+A^2\vec{b}u[1]+A\vec{b}u[2]+\vec{b}u[3]=\vec{x}[4]-A^4\vec{x}[0]$.

This means that in order to reach any state $\vec{x}_f$ in four time steps (that is, $\vec{x}[4]=\vec{x}_f$), we have to solve the following system of linear equations

$$A^3\vec{b}u[0]+A^2\vec{b}u[1]+A\vec{b}u[2]+\vec{b}u[3]=\vec{x}_f-A^4\vec{x}[0] \tag{25}$$

where $u[0],u[1],u[2]$ and $u[3]$ are the unknowns.

Since in our case we want to reach $\vec{x}_f=\begin{bmatrix}0\\0\\0\\0\end{bmatrix}$ then the system of linear equations simplifies to

$$A^3\vec{b}u[0]+A^2\vec{b}u[1]+A\vec{b}u[2]+\vec{b}u[3]=-A^4\vec{x}[0] \tag{26}$$

In matrix form, this system of linear equations is

$$\begin{bmatrix}| & | & | & |\\ A^3\vec{b} & A^2\vec{b} & A\vec{b} & \vec{b}\\ | & | & | & |\end{bmatrix}\cdot\begin{bmatrix}u[0]\\u[1]\\u[2]\\u[3]\end{bmatrix}=-A^4\vec{x}[0] \tag{27}$$

By defining $Q=\begin{bmatrix}| & | & | & |\\ A^3\vec{b} & A^2\vec{b} & A\vec{b} & \vec{b}\\ | & | & | & |\end{bmatrix}$ and $\vec{u}_4=\begin{bmatrix}u[0]\\u[1]\\u[2]\\u[3]\end{bmatrix}$. We can now rewrite our system of linear equations as

$$Q\vec{u}_4=-A^4\vec{x}[0] \tag{28}$$

Refer to the code in the solution IPython notebook for a solution of the system above. The solution of the system is

$$\vec{u}_4=\begin{bmatrix}-13.24875075\\23.73325125\\-11.57181872\\1.46515973\end{bmatrix}, \tag{29}$$

that is the control input sequence is: $u[0]=-13.24875075$, $u[1]=23.73325125$, $u[2]=-11.57181872$, $u[3]=1.46515973$.

(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!**

Solution: See the solution to the previous part

(j) (Optional) 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?

Solution: Similar to the previous parts, the key step here is to rewrite the equation you derived in part (e) (Equation 10) as:

$$\sum_{i=0}^{N-1} A^i \vec{b} u[N-i-1] = \vec{x}[N] - A^N x[0] \tag{30}$$

We want $\vec{x}[N] = \vec{x_f} = \vec{0}$. Therefore the system of linear equations simplifies to

$$\sum_{i=0}^{N-1} A^i \vec{b} u[N-i-1] = -A^N x[0] \tag{31}$$

If we extend this sum we get

$$A^{N-1}\vec{b}u[0] + A^{N-2}\vec{b}u[1] + \cdots + A\vec{b}u[N-2] + \vec{b}u[N-1] = -A^N \vec{x}[0] \tag{32}$$

This system of linear equations can be rewritten as

$$\begin{bmatrix} | & | & \cdots & | & | \\ A^{N-1}\vec{b} & A^{N-2}\vec{b} & \cdots & A\vec{b} & \vec{b} \\ | & | & \cdots & | & | \end{bmatrix} \cdot \begin{bmatrix} u[0] \\ u[1] \\ \vdots \\ u[N-1] \end{bmatrix} = -A^N \vec{x}[0] \tag{33}$$

We need to find $\{u[0], u[1], \ldots, u[N-1]\}$ that satisfies this system of linear equations. For this system to be solvable, we need $-A^N x[0] \in span\{\vec{b}, A\vec{b}, A^2\vec{b}, \ldots, A^{N-1}\vec{b}\}$. That is, we need $-A^N x[0]$ to be in the range (column span) of the matrix $\begin{bmatrix} | & | & \cdots & | & | \\ A^{N-1}\vec{b} & A^{N-2}\vec{b} & \cdots & A\vec{b} & \vec{b} \\ | & | & \cdots & | & | \end{bmatrix}$.

(k) (Optional) 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?

Solution: Similar to the previous parts, the key step here is to rewrite the equation you derived in part (e) (Equation 10) as:

$$\sum_{i=0}^{N-1} A^i \vec{b} u[N-i-1] = \vec{x}[N] - A^N x[0] \tag{34}$$

The difference is that now $\vec{x}[N] = \vec{x_f}$ can be anything in $\mathbb{R}^4$. Therefore the system of linear equations can be written as

$$\sum_{i=0}^{N-1} A^i \vec{b} u[N-i-1] = \vec{x_f} - A^N x[0] \tag{35}$$

If we extend this sum we get

$$A^{N-1}\vec{b}u[0] + A^{N-2}\vec{b}u[1] + \cdots + A\vec{b}u[N-2] + \vec{b}u[N-1] = \vec{x_f} - A^N x[0] \tag{36}$$

This system of linear equations can be further rewritten as

$$\begin{bmatrix} | & | & \cdots & | & | \\ A^{N-1}\vec{b} & A^{N-2}\vec{b} & \cdots & A\vec{b} & \vec{b} \\ | & | & \cdots & | & | \end{bmatrix} \cdot \begin{bmatrix} u[0] \\ u[1] \\ \vdots \\ u[N-1] \end{bmatrix} = \vec{x_f} - A^N x[0] \tag{37}$$

For this system to be solvable, we need $\vec{x_f} - A^N x[0] \in span\{\vec{b}, A\vec{b}, A^2\vec{b}, \ldots, A^{N-1}\vec{b}\}$. Since $\vec{x_f}$ can be any vector in $\mathbb{R}^4$ it also means that $\vec{x_f} - A^N x[0]$ can be any vector in $\mathbb{R}^4$. This means that in order to be able to reach any state $\vec{x_f}$ in $\mathbb{R}^4$, the range (column span) of the matrix $\begin{bmatrix} | & | & \cdots & | & | \\ A^{N-1}\vec{b} & A^{N-2}\vec{b} & \cdots & A\vec{b} & \vec{b} \\ | & | & \cdots & | & | \end{bmatrix}$ has to be all $\mathbb{R}^4$.

P.S.: Congratulations! You have just derived the condition for “controllability” for systems with linear dynamics. When dealing with a system that evolves over time we can sometimes influence the behavior of the system through various control inputs (for example the steering wheel and gas pedal of a car, or the rudder of an airplane). It is of great importance to know what states (think positions and velocities of a car, or configurations of an aircraft) that our system can be controlled to. Controllability is the ability to control the system to any possible state or configuration.

7. **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?