EECS 16A Designing Information Devices and Systems I
Spring 2019 Homework 3

This homework is due February 15, 2019, at 23:59.
Self-grades are due February 19, 2019, at 23:59.

Submission Format
Your homework submission should consist of **one** file.

- `hw2.pdf`: A single PDF file that contains all of your answers (any handwritten answers should be scanned) as well as your IPython notebook saved as a PDF.

  If you do not attach a PDF "printout" of your IPython notebook, you will not receive credit for problems that involve coding. Make sure that your results and your plots are visible. Assign the IPython printout to the correct problem(s) on Gradescope.

Submit each file to its respective assignment on Gradescope.

1. Lab Grades and Discussion Attendance

Lab grades and discussion attendance will be posted on Gradescope. These will be updated periodically; check the Piazza post for the most recent updates.

For lab grades, please submit a file named `sid.txt` to the "Lab Grades" assignment. This text file should contain only your SID number and nothing else (e.g. no "" marks or extra whitespace). You can find a template in the iPython folder.

For discussion grades, submit the same file `sid.txt` to "Discussion Attendance."

If you use a strange text editor, the autograder will give a strange error message complaining about the unicode format. If this happens, please use a different editor and reupload.

Describe how to find your lab and discussion grades.

Solution: Get the template file from the iPython folder and replace the numbers with your student ID without changing anything else. Submit this file to the assignment "Lab Grades" on Gradescope.

For the discussion grade, submit the same file to "Discussion Attendance."

2. Elementary Matrices

In lecture, we learned about an important technique for solving systems of linear equations called Gaussian elimination. It turns out that each row operation in Gaussian elimination can be performed by multiplying the augmented matrix on the left by a specific matrix called an *elementary matrix*. For example, suppose we want to row reduce the following augmented matrix:

$$\mathbf{A} = \left[\begin{array}{cccc|c} 1 & -2 & 0 & -5 & 15 \\ 0 & 1 & 0 & 3 & -7 \\ -2 & -3 & 1 & -6 & 9 \\ 0 & 1 & 0 & 2 & -5 \end{array}\right] \tag{1}$$

What matrix do you get when you subtract the 4th row from the 2nd row of $\mathbf{A}$ (putting the result in row 2)? (You don't have to include this in your solution.) Now, try multiplying the original $\mathbf{A}$ on the left by

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

(You don't have to include this in your solutions either.) Notice that you get the same thing.

$$\mathbf{EA} = \left[\begin{array}{cccc|c} 1 & -2 & 0 & -5 & 15 \\ 0 & 0 & 0 & 1 & -2 \\ -2 & -3 & 1 & -6 & 9 \\ 0 & 1 & 0 & 2 & -5 \end{array}\right]$$

$\mathbf{E}$ is a special type of matrix called an *elementary matrix*. This means that we can obtain the matrix $\mathbf{E}$ from the identity matrix by applying an elementary row operation – in this case, subtracting the 4th row from the 2nd row.

In general, any elementary row operation can be performed by left multiplying by an appropriate elementary matrix.

In other words, you can perform a row operation on a matrix $\mathbf{A}$ by first performing that row operation on the identity matrix to get an elementary matrix (see below), and then left multiplying $\mathbf{A}$ by the elementary matrix (like we did above).

$$\mathbf{I} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \xrightarrow{R_2 - R_4 \mapsto R_2} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \mathbf{E}$$

(a) Write down the elementary matrices required to perform the following row operations on a 4×5 augmented matrix.

i. $R_1 \mapsto R_3$
$R_3 \mapsto R_1$

ii. $-5R_3 \mapsto R_3$

iii. $3R_2 + R_4 \mapsto R_4$
$R_1 - R_2 \mapsto R_1$

Hint: For the last one, note that if you want to perform two row operations on the matrix $\mathbf{A}$, you can perform them both on the identity matrix and then left multiply $\mathbf{A}$ by the resulting matrix.

Solution:

We obtain each of the desired elementary matrices by performing the row operations on a 4×4 identity matrix.

i. Switching rows 1 and 3:

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

ii. Multiplying row 3 by -5:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

iii. Adding 3×row 2 to row 4 and subtracting row 2 from row 1:

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

Note that we obtained this last matrix by applying two elementary row operations to the identity matrix. We could have performed each elementary row operation on individual identity matrices and then multiplied them together to achieve the same result. In this case, the order of the matrices did not matter; however, this is not true in general.

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

(b) In lecture we emphasized using Gaussian Elimination to reach an upper triangular form to determine the number of solutions for a given system of linear equations. When there is a unique solution, however, it is useful to determine exactly what that solution is by continuing Gaussian Elimination to reach a "fully reduced" form like that shown below. An example of this process can be found in Note 1, Example 1.7.

Compute a matrix $\mathbf{E}$ (by hand) that fully row reduces the augmented matrix $\mathbf{A}$ given in Equation (1)– that is, find $\mathbf{E}$ such that $\mathbf{EA}$ is a diagonal matrix with 1s along the diagonal. Show that this is true by multiplying out $\mathbf{EA}$. When an augmented matrix is in this final form, it will have the form

$$\mathbf{EA} = \left[\begin{array}{cccc|c} 1 & 0 & 0 & 0 & b_1 \\ 0 & 1 & 0 & 0 & b_2 \\ 0 & 0 & 1 & 0 & b_3 \\ 0 & 0 & 0 & 1 & b_4 \end{array}\right]$$

Once you have found the required elementary matrices,

i. use IPython to find the matrix $\mathbf{E}$

ii. verify by hand that multiplying $\mathbf{E}$ and $\mathbf{A}$ gives you the identity matrix augmented with constants (as $\mathbf{EA}$ shown above).

Hint: As before, note that you can either ***apply a set of row operations to the same identity matrix*** *or* ***apply them to separate identity matrices and then multiply the matrices together****. Make sure, though, that you apply the row operations and multiply the matrices in the correct order.*

Solution:

We first need to row reduce $\mathbf{A}$ by hand to find the required row operations. The following row operations will do the trick (though you could have used a different set that does the same thing.)

- Step 1: Add 2×**Row 1** to Row 3
- Step 2: Add 2×**Row 2** to Row 1, add 7×**Row 2** to Row 3, and subtract **Row 2** from Row 4

- Step 3: Add **Row 4** to Row 1, add $3\times$**Row 4** to Row 2, and add $5\times$**Row 4** to Row 3
- Step 4: Multiply **Row 4** by -1

Note that we have grouped the row operations together, so that each step involves adding a scalar multiple of a particular row (bolded above) to the other rows. This will make calculating the elementary matrices for each step easier.

Applying each of these sets of row operations to a 4×4 identity matrix gives us the following matrices:

- Step 1:
$$\mathbf{E}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 2 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
- Step 2:
$$\mathbf{E}_2 = \begin{bmatrix} 1 & 2 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 7 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix}$$
- Step 3:
$$\mathbf{E}_3 = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & 5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
- Step 4:
$$\mathbf{E}_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$$

We now multiply these matrices together as follows:

$$\mathbf{E} = \mathbf{E}_4\mathbf{E}_3\mathbf{E}_2\mathbf{E}_1 = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & 5 \\ 0 & 0 & 0 & 1 \end{bmatrix}}\underbrace{\begin{bmatrix} 1 & 2 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 7 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 2 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}$$

$$= \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & 5 \\ 0 & 0 & 0 & -1 \end{bmatrix}\begin{bmatrix} 1 & 2 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 2 & 7 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & -2 & 0 & 3 \\ 2 & 2 & 1 & 5 \\ 0 & 1 & 0 & -1 \end{bmatrix}$$

Note the order in which we multiplied the matrices. $\mathbf{E}_1$ gets applied first, so it is furthest to the right (i.e. it will act on the augmented matrix first), etc. Also, note that we could have applied the row operations to the identity matrices in different groups. For example, we could have written an elementary matrix

for each individual row operation and multiplied all of them together, making sure to maintain the correct order. We also could have applied all of the row operations, in the correct order, to a single identity matrix. **The important thing is that we maintain the correct order of row operations – either when we're applying them to an individual identity matrix or multiplying the elementary matrices together.** You should have done these multiplications in the IPython notebook.

To show that **E** does, in fact, row reduce **A**, we calculate **EA**.

$$\mathbf{EA} = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & -2 & 0 & 3 \\ 2 & 2 & 1 & 5 \\ 0 & 1 & 0 & -1 \end{bmatrix} \left[\begin{array}{cccc|c} 1 & -2 & 0 & -5 & 15 \\ 0 & 1 & 0 & 3 & -7 \\ -2 & -3 & 1 & -6 & 9 \\ 0 & 1 & 0 & 2 & -5 \end{array}\right]$$

$$= \left[\begin{array}{cccc|c} 1 & 0 & 0 & 0 & 3 \\ 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & -2 \end{array}\right]$$

3. Mechanical Inverses

For each of the following matrices, state whether the inverse exists. If so, find the inverse, $\mathbf{A}^{-1}$. If not, show why no inverse exists. **Solve the inverses by hand. You may use IPython for parts (a)-(d) to visualize how the matrix A changes a vector.**

For parts (a)-(d), in addition to finding the inverse (if it exists), describe how the original matrix, **A**, changes a vector it's applied to. For example, if $\mathbf{A}\vec{b} = \vec{c}$, then **A** could scale $\vec{b}$ by 2 to get $\vec{c}$, or **A** could reflect $\vec{b}$ across the x axis to get $\vec{c}$, etc. *Hint:* It may help to plot a few examples to recognize the pattern.

(a) $\mathbf{A} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$

Solution:

The inverse does exist.

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

The original matrix **A** flips the x and y components of the vector. Any correct equivalent sequence of operations warrants full credit. Notice how the inverse does the exact same thing–that is, it switches the x and y components of the vector it's applied to. This makes sense–switching x and y twice on a vector $\begin{bmatrix} x \\ y \end{bmatrix}$ gives us $\begin{bmatrix} x \\ y \end{bmatrix}$.

(b) **(PRACTICE)**

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

Solution:

The inverse does exist.

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

The original matrix **A** reflects the vector across the y-axis, i.e. it multiplies the vector's x-component by a factor of -1. Reflecting the vector across the y-axis again with $\mathbf{A}^{-1} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$ will give you the original vector.

(c) **(PRACTICE)**

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

Solution:

We see here that the inverse does not exist because the first row (and second column) are the zero vector so the columns are linearly dependent. Since the columns of the matrix are linearly dependent, the inverse does not exist.

The original matrix $\mathbf{A}$ removes the x component of the vector it's applied to and keeps the same y component. Graphically speaking, this matrix can be thought of as taking the "shadow" of the vector on the y-axis if you were to shine a light perpendicular to the y-axis.

(d) Hint: What does the result look like when you apply this matrix to $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$? Draw it out. In the iPython notebook for this homework, we have provided you with a rotation function that may help to visualize what is happening.

$\mathbf{A} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$

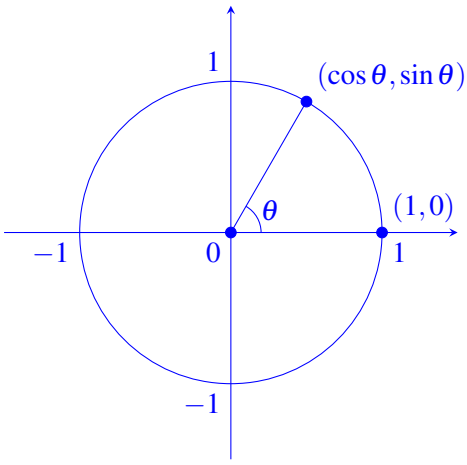
Solution:

The inverse does exist.

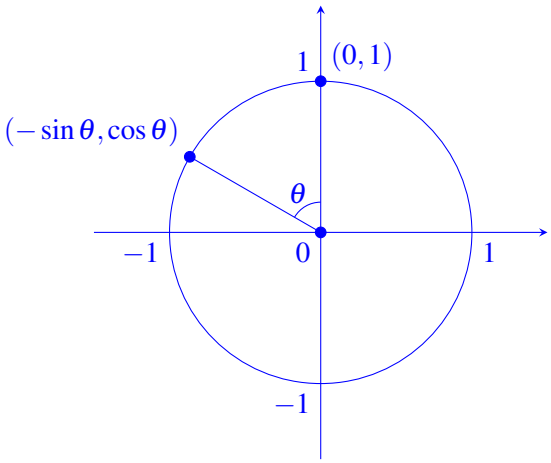
$\mathbf{A}^{-1} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$

The original matrix $\mathbf{A}$ is the two-dimensional rotation matrix. We'll look at the hint; namely, what the output looks like when we apply the matrix to $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$. In the diagrams below, we can see that applying the rotation matrix to a vector will rotate the vector by θ in the counter-clockwise direction.

$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix}$$



$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -\sin\theta \\ \cos\theta \end{bmatrix}$$



This is an important matrix to remember, but if you ever forget what the form of the rotation matrix is, you know the definition of matrix-vector multiplication in terms of the columns of the matrix

$$\begin{aligned}\mathbf{A}\vec{x} &= \begin{bmatrix} | & | & & | \\ \vec{a}_0 & \vec{a}_1 & \cdots & \vec{a}_n \\ | & | & & | \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} \\ &= \vec{a}_0 x_0 + \vec{a}_1 x_1 + \cdots + \vec{a}_n x_n \\ &= \sum_{i=0}^{n} \vec{a}_i x_i\end{aligned}$$

We can generalize this to matrix-matrix multiplication and go one column of $\mathbf{V}$ at a time (we changed from x to v to avoid confusion in notation):

$$\begin{aligned}\mathbf{AV} &= \mathbf{A} \begin{bmatrix} | & | & & | \\ \vec{v}_0 & \vec{v}_0 & \cdots & \vec{v}_m \\ | & | & & | \end{bmatrix} \\ &= \begin{bmatrix} | & | & & | \\ \mathbf{A}\vec{v}_0 & \mathbf{A}\vec{v}_1 & \cdots & \mathbf{A}\vec{v}_m \\ | & | & & | \end{bmatrix}\end{aligned}$$

So if we apply this to the vectors from the problem $\begin{bmatrix}1\\0\end{bmatrix}$ and $\begin{bmatrix}0\\1\end{bmatrix}$, we find that our vectors combine to make the identity matrix! Plugging this and the vectors we solved for above into the equation for $\mathbf{AV}$ above:

$$\begin{aligned}A\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} &= \begin{bmatrix}\cos\theta & \sin\theta\\ -\sin\theta & \cos\theta\end{bmatrix} \\ A &= \begin{bmatrix}\cos\theta & \sin\theta\\ -\sin\theta & \cos\theta\end{bmatrix}\end{aligned}$$

(e) $\mathbf{A} = \begin{bmatrix}1 & 1\\ 2 & 0\end{bmatrix}$

Solution:

We can use Gauss-Jordan method to find the inverse of the matrix.

$$\left[\begin{array}{cc|cc} 1 & 1 & 1 & 0 \\ 2 & 0 & 0 & 1 \end{array}\right] \sim \left[\begin{array}{cc|cc} 1 & 1 & 1 & 0 \\ 0 & 2 & 2 & -1 \end{array}\right] \sim \left[\begin{array}{cc|cc} 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & -\frac{1}{2} \end{array}\right] \sim \left[\begin{array}{cc|cc} 1 & 0 & 0 & \frac{1}{2} \\ 0 & 1 & 1 & -\frac{1}{2} \end{array}\right]$$

Inverse exists: $\mathbf{A}^{-1} = \begin{bmatrix} 0 & \frac{1}{2} \\ 1 & -\frac{1}{2} \end{bmatrix}$

(f) $\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 2 \\ 1 & 4 & 4 \end{bmatrix}$

Solution:

Inverse does not exist because the second and third column are equal, i.e., they are linearly dependent. Since the columns of the matrix are linearly dependent, the inverse does not exist.

(g) **PRACTICE:** $\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 1 \\ 0 & 1 & -1 \end{bmatrix}$

Solution:

Inverse does not exist because the third column is the negative of the second column, i.e., they are linearly dependent. Since the columns of the matrix are linearly dependent, the inverse does not exist.

(h) **PRACTICE:** $\mathbf{A} = \begin{bmatrix} -1 & 1 & -\frac{1}{2} \\ 1 & 1 & -\frac{1}{2} \\ 0 & 1 & 1 \end{bmatrix}$

Solution:

We can use Gauss-Jordan method to find the inverse of the matrix.

$$\left[\begin{array}{ccc|ccc} -1 & 1 & -\frac{1}{2} & 1 & 0 & 0 \\ 1 & 1 & -\frac{1}{2} & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{array}\right] \sim \left[\begin{array}{ccc|ccc} 0 & 2 & -1 & 1 & 1 & 0 \\ 1 & 1 & -\frac{1}{2} & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{array}\right] \sim \left[\begin{array}{ccc|ccc} 0 & 0 & -3 & 1 & 1 & -2 \\ 1 & 1 & -\frac{1}{2} & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{array}\right]$$

$$\sim \left[\begin{array}{ccc|ccc} 0 & 0 & -3 & 1 & 1 & -2 \\ 1 & 1 & -\frac{1}{2} & 0 & 1 & 0 \\ 0 & 1 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{array}\right] \sim \left[\begin{array}{ccc|ccc} 0 & 0 & -3 & 1 & 1 & -2 \\ 1 & 0 & -\frac{1}{2} & -\frac{1}{3} & \frac{2}{3} & -\frac{1}{3} \\ 0 & 1 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{array}\right] \sim \left[\begin{array}{ccc|ccc} 0 & 0 & 1 & -\frac{1}{3} & -\frac{1}{3} & \frac{2}{3} \\ 1 & 0 & -\frac{1}{2} & -\frac{1}{3} & \frac{2}{3} & -\frac{1}{3} \\ 0 & 1 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{array}\right]$$

$$\sim \left[\begin{array}{ccc|ccc} 0 & 0 & 1 & -\frac{1}{3} & -\frac{1}{3} & \frac{2}{3} \\ 1 & 0 & 0 & -\frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 1 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{array}\right] \sim \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & -\frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 1 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & 1 & -\frac{1}{3} & -\frac{1}{3} & \frac{2}{3} \end{array}\right]$$

Inverse exists: $\mathbf{A}^{-1} = \begin{bmatrix} -\frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ -\frac{1}{3} & -\frac{1}{3} & \frac{2}{3} \end{bmatrix}$

(i) **(PRACTICE)**

$\mathbf{A} = \begin{bmatrix} 3 & 0 & -2 & 1 \\ 0 & 2 & 1 & 3 \\ 3 & 1 & 0 & 4 \\ 1 & 0 & 0 & 1 \end{bmatrix}$

Hint 1: What do the linear (in)dependence of the rows and columns tell us about the invertibility of a matrix? Hint 2: We're reasonable people!

Solution:

Inverse does not exist because $\text{column}_1 + \text{column}_2 + \text{column}_3 = \text{column}_4$, which means that the columns are linearly dependent. Since the columns of the matrix are linearly dependent, the inverse does not exist.

4. Proofs about Independence, Span, and Subspaces

In this problem, we will explore the connections between independence, span, and subspaces. We will start with a list of definitions. Using proofs, you will explore how these definitions interact with each other.

Recall the definition of linear independence: a set of vectors $(\vec{v}_1, ..., \vec{v}_n)$ is **linearly dependent** if one of the vectors can be written as a linear combination of the other vectors. In mathematical notation, this means that for some i, $\vec{v}_i = \sum_{j \neq i} a_j \vec{v}_j$. We saw that this definition is equivalent to an alternate definition: that there exists some linear combination $\sum_{i=1}^n a_i \vec{v}_i = 0$ where at least one a_i is non-zero.

A set of vectors is **linearly independent** if it is not dependent. In other words, the only time that the linear combination $\sum_{i=1}^n a_i \vec{v}_i$ is equal to 0 is when all the a_i are 0.

Also, recall the definition of span: the **span** of a set of vectors $(\vec{v}_1, ..., \vec{v}_n)$ is the set of all linear combinations of these vectors, or any vector that can be written $\sum_{i=1}^n a_i \vec{v}_i$ for some scalars a_i.

(a) First, suppose we have some set of linearly independent vectors $(\vec{v}_1, ..., \vec{v}_n)$, and we have an extra vector $\vec{u}$. Show that if $\vec{u}$ is in the span of $(\vec{v}_1, ..., \vec{v}_n)$, then the set $(\vec{v}_1, ..., \vec{v}_n, \vec{u})$ is dependent. Hint: This should be a simple application of the definition.

Solution: If $\vec{u}$ is in the span of $(\vec{v}_1, ..., \vec{v}_n)$, then by definition of span, it can be written $\vec{u} = \sum_{i=1}^n a_i \vec{v}_i$. We can rearrange to find that $0 = -\vec{u} + \sum_{i=1}^n a_i \vec{v}_i$. This equation gives us a non-zero linear combination of the set $(\vec{v}_1, ..., \vec{v}_n, \vec{u})$ that reaches 0, so the set is dependent.

(b) Now, show that if $(\vec{v}_1, ..., \vec{v}_n)$ is independent but $(\vec{v}_1, ..., \vec{v}_n, \vec{u})$ is dependent, then $\vec{u}$ is in the span of $(\vec{v}_1, ..., \vec{v}_n)$.

Solution: If $(\vec{v}_1, ..., \vec{v}_n, \vec{u})$ is dependent, then there exists some linear combination $b\vec{u} + \sum_{i=1}^n a_i \vec{v}_i = 0$ where at least one coefficient is non-zero. b cannot be zero because otherwise, we would have $\sum_{i=1}^n a_i \vec{v}_i = 0$ for at least one non-zero a_i, but we know that $(\vec{v}_1, ..., \vec{v}_n)$ is independent. Therefore, we can rearrange and divide by b to find that $\vec{u} = -\sum_{i=1}^n \frac{a_i}{b} \vec{v}_i$.

(c) Recall that a **subspace** is a subset of vectors that contains the $\vec{0}$ vector and is closed under addition and scalar multiplication. In other words, if $\vec{u}$ and $\vec{v}$ are in the subspace, then $\vec{u} + \vec{v}$ is in the subspace. Also, if $\vec{u}$ is in the subspace, then $a\vec{u}$ is in the subspace for any scalar a.

Suppose we have a set of vectors $(\vec{v}_1, ..., \vec{v}_n)$. Show that the span of these vectors is a subspace.

Solution: **Showing closure under addition**: Suppose $\vec{u}$ and $\vec{v}$ are in the span of $(\vec{v}_1, ..., \vec{v}_n)$. Then, we can write them as $\vec{u} = \sum_{i=1}^n a_i \vec{v}_i$ and $\vec{v} = \sum_{i=1}^n b_i \vec{v}_i$, so their sum is

$$\sum_{i=1}^n a_i \vec{v}_i + \sum_{i=1}^n b_i \vec{v}_i = \sum_{i=1}^n (a_i + b_i) \vec{v}_i,$$

which is also in the span.

Showing closure under scalar multiplication: Suppose $\vec{u}$ is in the span of $(\vec{v}_1, ..., \vec{v}_n)$. Then, we can write it as $\vec{u} = \sum_{i=1}^n a_i \vec{v}_i$, so the product $a\vec{u}$ is

$$a \sum_{i=1}^n a_i \vec{v}_i = \sum_{i=1}^n (a * a_i) \vec{v}_i,$$

which is also in the span.

Showing inclusion of the $\vec{0}$ vector: The span of $(\vec{v}_1,...,\vec{v}_n)$ i.e. $\sum_{i=1}^{n} a_i\vec{v}_i$ will be $\vec{0}$ if a_1, a_2, ..., a_n are zero.

(d) Consider the span of the set $(\vec{v}_1,...,\vec{v}_n,\vec{u})$. Suppose $\vec{u}$ is in the span of $(\vec{v}_1,...,\vec{v}_n)$. Then, show that any vector in $span(\vec{v}_1,...,\vec{v}_n,\vec{u})$ is in $span(\vec{v}_1,...,\vec{v}_n)$.

Solution: By definition, we know that any vector in $span(\vec{v}_1,...,\vec{v}_n,\vec{u})$ can be written $b\vec{u}+\sum_{i=1}^{n} a_i\vec{v}_i$. Also, since $\vec{u}$ is in the span of $(\vec{v}_1,...,\vec{v}_n)$, we can write it as $\vec{u}=\sum_{i=1}^{n} b_i\vec{v}_i$. Therefore,

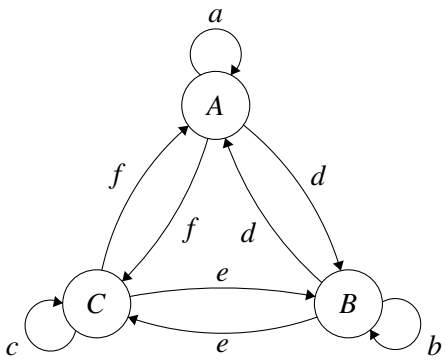
$$b\vec{u}+\sum_{i=1}^{n} a_i\vec{v}_i = b\sum_{i=1}^{n} b_i\vec{v}_i+\sum_{i=1}^{n} a_i\vec{v}_i = \sum_{i=1}^{n}(b*b_i+a_i)\vec{v}_i,$$

so this vector is also in $span(\vec{v}_1,...,\vec{v}_n)$.

Intuitively, $\vec{u}$ is redundant, so we can safely remove it without reducing our span.

5. Reservoirs That Give and Take

Consider a network of three water reservoirs A, B, and C. At the end of each day, water transfers among the reservoirs according to the directed graph shown below.



The parameters a, b, and c—which label the self-loops—denote the *fractions* of the water in reservoirs A, B, and C, respectively, that stay in the same reservoir at the end of each day n. The parameters d, e, and f denote the fractions of the reservoir contents that transfer to adjacent reservoirs at the end of each day, according to the directed graph above.

Assume that the reservoir system is conservative—no water enters or leaves the system, which means that the total water in the network is constant. Accordingly, *for each node*, the weights on its self-loop and its two outgoing edges sum to 1; for example, for node A, we have

$$a+d+f=1,$$

and similar equations hold for the other nodes. Moreover, assume that all the edge weights are positive numbers—that is,

$$0<a,b,c,d,e,f<1.$$

The state evolution equation governing the water flow dynamics in the reservoir system is given by $\vec{s}[n+1]=\mathbf{A}\vec{s}[n]$, where the 3×3 matrix $\mathbf{A}$ is the state transition matrix, and $\vec{s}[n]=\begin{bmatrix}s_A[n] & s_B[n] & s_C[n]\end{bmatrix}^T\in\mathbb{R}^3$ is the nonnegative state vector that shows the water distribution among the three reservoirs at the end of day n, as fractions of the total water in the network.

(a) Determine the state transition matrix $\mathbf{A}$.

Solution:

$$\mathbf{A} = \begin{bmatrix} a & d & f \\ d & b & e \\ f & e & c \end{bmatrix}$$

(b) For some systems, there exists an equilibrium state —that is, a state for which the following is true:

$$\vec{s}[n+1] = \vec{s}[n] = \vec{s}^*$$

Determine if for the systems described above, there exists a equilibrium state. If such a state exists, find it.

Hint: What's special about the rows of this matrix?

Solution:

Notice for the above matrix, $\mathbf{A}^T = \mathbf{A}$. Such a matrix is called symmetric. This implies both the rows and the columns of this matrix sum to one.

Consider what happens when you apply this matrix to a uniform vector, $\vec{s}[0] = [x\ x\ x]^T = x[1\ 1\ 1]^T$.

$$\vec{s}[1] = \mathbf{A}\vec{s}[0] = \begin{bmatrix} a & d & f \\ d & b & e \\ f & e & c \end{bmatrix} \begin{bmatrix} x \\ x \\ x \end{bmatrix} = \begin{bmatrix} ax+dx+fx \\ dx+bx+ex \\ fx+ex+cx \end{bmatrix} = \begin{bmatrix} x(a+d+f) \\ x(d+b+e) \\ x(f+e+c) \end{bmatrix} = \begin{bmatrix} x \\ x \\ x \end{bmatrix}$$

Since $\vec{s}[1] = \vec{s}[0]$, $\vec{s}[0]$ is the equilibrium state $\vec{s}^*$.

An alternative way of doing the problem would be to compute the nullspace of $\mathbf{A} - \mathbf{I}$ in the following way.

$$\begin{bmatrix} a & d & f \\ d & b & e \\ f & e & c \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} a & d & f \\ d & b & e \\ f & e & c \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} - \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \mathbf{0}$$

$$\Rightarrow (\mathbf{A} - \mathbf{I})[x_1 x_2 x_3]^T = \mathbf{0}$$

$$\begin{bmatrix} a-1 & d & f \\ d & b-1 & e \\ f & e & c-1 \end{bmatrix} \sim \begin{bmatrix} a-1 & d & f \\ d & b-1 & e \\ a+d+f-1 & b+d+e-1 & e+f+c-1 \end{bmatrix} \sim \begin{bmatrix} a-1 & d & f \\ d & b-1 & e \\ 0 & 0 & 0 \end{bmatrix}$$

Thus we can set x_3 as a free variable (say $x_3 = 1$), which gives us the following 2 equations for x_1, x_2

$$(a-1)x_1 + dx_2 + f = 0 = (-d-f)x_1 + dx_2 + f$$

$$dx_2 + (b-1)x_2 + e = 0 = dx_1 + (-d-e)x_2 + e$$

Adding the above two equations gives

$$f(1-x_1) + e(1-x_2) = 0$$

Thus

$$x_1 = x_2 = x_3 = 1$$

is a basis for the nullspace, and $[x\ x\ x]^T$ is a general solution.

(c) Suppose the state transition matrix for the network is given by

$$\mathbf{A} = \begin{bmatrix} \frac{1}{5} & \frac{2}{5} & \frac{2}{5} \\ \frac{2}{5} & \frac{1}{5} & \frac{2}{5} \\ \frac{2}{5} & \frac{1}{5} & \frac{1}{5} \end{bmatrix}.$$

Is it possible to determine the state $\vec{s}[n]$ from the subsequent state $\vec{s}[n+1]$? You do not have to find $\vec{s}[n]$ from $\vec{s}[n+1]$, just determine whether it is possible to do so.

Solution:

We row-reduce $\mathbf{A}$ and see what the pivots look like. If they're all nonzero, then $\mathbf{A}$ is invertible and reverse-time inference (obtaining $\vec{s}[n]$ from $\vec{s}[n+1]$) is possible. If any pivot is zero, then $\mathbf{A}$ is not invertible, in which case we cannot determine $\vec{s}[n]$ from $\vec{s}[n+1]$.

$$\begin{bmatrix} \frac{1}{5} & \frac{2}{5} & \frac{2}{5} \\ \frac{2}{5} & \frac{1}{5} & \frac{2}{5} \\ \frac{2}{5} & \frac{1}{5} & \frac{1}{5} \end{bmatrix} \sim \begin{bmatrix} \frac{1}{5} & \frac{2}{5} & \frac{2}{5} \\ 0 & -\frac{3}{5} & -\frac{2}{5} \\ 0 & -\frac{3}{5} & -\frac{3}{5} \end{bmatrix} \sim \begin{bmatrix} \frac{1}{5} & \frac{2}{5} & \frac{2}{5} \\ 0 & -\frac{3}{5} & -\frac{2}{5} \\ 0 & 0 & -\frac{1}{5} \end{bmatrix}$$

All three pivots are nonzero. Therefore $\mathbf{A}$ is invertible and $\vec{s}[n] = \mathbf{A}^{-1}\vec{s}[n+1]$. The problem does not ask us to compute $\mathbf{A}^{-1}$ but merely whether it's possible to determine $\vec{s}[n]$ from $\vec{s}[n+1]$. The answer is yes!

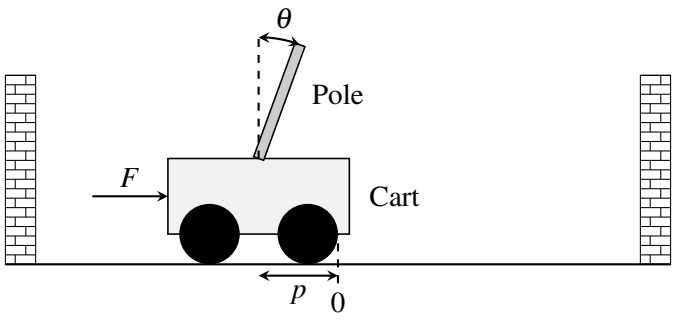
6. Segway Tours

Your friend has decided to start a new SF tour business, and you suggest he use segways.

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 the 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.

Your friend 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. He 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":

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

The input to this system u will just be the force applied to the cart (or base of the segway).[1]

At time step n, we can apply scalar input $u[n]$. The cart-pole system can be represented by a linear model:

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

where $\mathbf{A} \in \mathbb{R}^{4\times4}$ and $\vec{b} \in \mathbb{R}^{4\times1}$. 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 your friend'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 that 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 control?

You will solve this problem by walking through several steps.

(a) Express $\vec{x}[1]$ in terms of $\vec{x}[0]$ and the input $u[0]$. (*Hint:* This is easy.)

Solution:

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

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

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

Solution:

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

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

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

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

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

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

[1] You might note that velocity and angular velocity are derivatives of position and angle respectively. Differential equations are used to describe continuous time systems, which you will learn more about in EE 16B. But even without these techniques, we can still approximate the solution to be a continuous time system by modeling it as a discrete time system where we take very small steps in time. We think about applying a force constantly for a given finite duration and we see how the system responds after that finite duration.

By substituting $\vec{x}[2]$ from Equation (4), we get

$$\begin{aligned}\vec{x}[3] &= \mathbf{A}\vec{x}[2] + \vec{b}u[2] \\ &= \mathbf{A}\left(\mathbf{A}^2\vec{x}[0] + \mathbf{A}\vec{b}u[0] + \vec{b}u[1]\right) + \vec{b}u[2] \\ &= \mathbf{A}^3\vec{x}[0] + \mathbf{A}^2\vec{b}u[0] + \mathbf{A}\vec{b}u[1] + \vec{b}u[2] \end{aligned} \tag{5}$$

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

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

By substituting $\vec{x}[3]$ from Equation (5), we get

$$\begin{aligned}\vec{x}[4] &= \mathbf{A}\vec{x}[3] + \vec{b}u[3] \\ &= \mathbf{A}\left(\mathbf{A}^3\vec{x}[0] + \mathbf{A}^2\vec{b}u[0] + \mathbf{A}\vec{b}u[1] + \vec{b}u[2]\right) + \vec{b}u[3] \\ &= \mathbf{A}^4\vec{x}[0] + \mathbf{A}^3\vec{b}u[0] + \mathbf{A}^2\vec{b}u[1] + \mathbf{A}\vec{b}u[2] + \vec{b}u[3] \end{aligned} \tag{6}$$

(c) Now, derive an expression for $\vec{x}[N]$ in terms of $\vec{x}[0]$ and the inputs from $u[0], \ldots, u[N-1]$. (*Note: To obtain a compact expression, you can use a summation from* 0 *to* $N-1$.)

Solution:

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

$$\vec{x}[N] = \mathbf{A}^N\vec{x}[0] + \mathbf{A}^{N-1}\vec{b}u[0] + \ldots + \mathbf{A}\vec{b}u[N-2] + \vec{b}u[N-1] \tag{7}$$

You might also use the compact expression:

$$\vec{x}[N] = \mathbf{A}^N\vec{x}[0] + \sum_{i=0}^{N-1} \mathbf{A}^i\vec{b}u[N-i-1] \tag{8}$$

Note that $\mathbf{A}^0$ is the identity matrix.

As a sanity check, plug the values $N = 1, 2, 3$, and 4 to obtain Equations (3), (4), (5), and (6), respectively.

For the next four parts of the problem, you are given the matrix $\mathbf{A}$ and the vector $\vec{b}$:

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

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

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 $u[0], u[1], \ldots$. The state vector $\vec{x}_f = \vec{0}$ corresponds to the cart-pole

(or segway) being upright and stopped at the origin. (Reaching $\vec{x}_f = \vec{0}$ in N steps means that, given that we start at $\vec{x}[0]$, we can find control inputs, such that we get $\vec{x}[N]$, the state vector at the Nth time step, equal to $\vec{x}_f$.)

Note: You may use IPython to solve the next three parts of the problem. You may use the function we provided (`gauss_elim(matrix)`) to help you find the upper triangular form of matrices. An example of Gaussian Elimination using this code is provide in the iPython notebook. You may also use the function (`np.linalg.solve`) to solve the equations.

(d) Can you reach $\vec{x}_f$ in *two* time steps? (*Hint: Express* $\vec{x}[2] - \boldsymbol{A}^2\vec{x}[0]$ *in terms of the inputs* $u[0]$ *and* $u[1]$. *Then determine if the system of equations can be solved to obtain* $u[0]$ *and* $u[1]$. *If we obtain valid solutions for* $u[0]$ *and* $u[1]$, *then we can say we will reach* $\vec{x}_f$ *in two time steps.*)

Solution:

No.

From Equation (4), we know that $\mathbf{A}^2\vec{x}[0] + \mathbf{A}\vec{b}u[0] + \vec{b}u[1] = \vec{x}[2]$ which is equivalent to $\mathbf{A}\vec{b}u[0] + \vec{b}u[1] = \vec{x}[2] - \mathbf{A}^2\vec{x}[0]$.

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:

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

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

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

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

In matrix form, this system of linear equations is

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

which yields the following augmented matrix:

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

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

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

Applying Gaussian elimination, we get the upper triangular form to be

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

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

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

Solution:

No.

Similar to the previous part, from Equation (5), we know that $\mathbf{A}^3\vec{x}[0]+\mathbf{A}^2\vec{b}u[0]+\mathbf{A}\vec{b}u[1]+\vec{b}u[2]=\vec{x}[3]$, which is equivalent to $\mathbf{A}^2\vec{b}u[0]+\mathbf{A}\vec{b}u[1]+\vec{b}u[2]=\vec{x}[3]-\mathbf{A}^3\vec{x}[0]$.

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:

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

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

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

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

In matrix form, this system of linear equations is

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

which yields the following augmented matrix:

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

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

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

Applying Gaussian elimination, we get the upper triangular form to be

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

which means that the system is inconsistent (due to the fourth row) and that 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 upper triangular form as long as you showed that the system of equations is inconsistent.

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

Solution:

Yes.

Similar to the previous part, from Equation (6), we know that $\mathbf{A}^4\vec{x}[0]+\mathbf{A}^3\vec{b}u[0]+\mathbf{A}^2\vec{b}u[1]+\mathbf{A}\vec{b}u[2]+\vec{b}u[3]=\vec{x}[4]$ which is equivalent to $\mathbf{A}^3\vec{b}u[0]+\mathbf{A}^2\vec{b}u[1]+\mathbf{A}\vec{b}u[2]+\vec{b}u[3]=\vec{x}[4]-\mathbf{A}^4\vec{x}[0]$.

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:

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

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

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

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

In matrix form, this system of linear equations is

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

By defining $\mathbf{Q}=\begin{bmatrix}| & | & | & |\\ \mathbf{A}^3\vec{b} & \mathbf{A}^2\vec{b} & \mathbf{A}\vec{b} & \vec{b}\\ | & | & | & |\end{bmatrix}$ 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

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

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

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

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

(g) If you have found that you can get to the final state in 4 time steps, find the required correct control inputs using IPython and verify the answer by entering these control inputs into the *Plug in your controller* section of 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.

(h) Let's return to a general matrix $\mathbf{A}$ and a general vector $\vec{b}$. What condition do we need on

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

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 ((c)) (Equation (8)) as

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

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

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

If we extend this sum, we get

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

This system of linear equations can be rewritten as

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

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

(i) What condition would we need on span $\left\{\vec{b}, \mathbf{A}\vec{b}, \mathbf{A}^2\vec{b}, \ldots, \mathbf{A}^{N-1}\vec{b}\right\}$ 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 ((c)) (Equation (8)) as

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

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

$$\sum_{i=0}^{N-1} \mathbf{A}^i\vec{b}u[N-i-1] = \vec{x}_f - \mathbf{A}^N x[0].$$

If we extend this sum, we get

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

This system of linear equations can be further rewritten as

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

For this system to be solvable, we need $\vec{x}_f-\mathbf{A}^N x[0]\in \text{span}\left\{\vec{b},\mathbf{A}\vec{b},\mathbf{A}^2\vec{b},\ldots,\mathbf{A}^{N-1}\vec{b}\right\}$. Since $\vec{x}_f$ can be any vector in $\mathbb{R}^4$, it also means that $\vec{x}_f-\mathbf{A}^N x[0]$ 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 space) of the matrix $\begin{bmatrix} | & | & \cdots & | & | \\ \mathbf{A}^{N-1}\vec{b} & \mathbf{A}^{N-2}\vec{b} & \cdots & \mathbf{A}\vec{b} & \vec{b} \\ | & | & \cdots & | & | \end{bmatrix}$ has to be all of $\mathbb{R}^4$.

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. Audio File Matching

Each day a wide variety of quantities we interact with can be expressed as vectors. For example, an audio clip or a sound wave (continuous function in time) can be sampled at regular intervals to make a discrete sequence of values and represented in vector form.

This problem explores using inner products for measuring similarity between two sound signals represented as vectors. The same ideas here will be further developed in the third module of EE16A where we will learn about Locationing and GPS.

Let us consider a very simplified model for an audio signal; one that is composed of just two tones. One tone is represented by the value -1 and the other by the value $+1$. A vector of length n makes up the audio file.

(a) Say we want to compare two audio files of the same length n to decide how similar they are. First consider two vectors that are exactly identical $\vec{X}_1=\begin{bmatrix}1 & 1 & \cdots & 1\end{bmatrix}^T$ and $\vec{X}_2=\begin{bmatrix}1 & 1 & \cdots & 1\end{bmatrix}^T$. What is the inner product/dot product of these two vectors, i.e. $\vec{X}_1^T\vec{X}_2$? What if $\vec{X}_1=\begin{bmatrix}1 & 1\cdots 1\end{bmatrix}^T$ and $\vec{X}_2=\begin{bmatrix}1 & -1 & 1 & -1\cdots 1 & -1\end{bmatrix}^T$ (where the length of the vector is an even number)? For pairs of vectors of length n made of ± 1's, does a larger dot product imply that the vectors are more similar or less similar?

Solution: From lecture, we define the dot product as the following:

$$\vec{u}^T\vec{v}=\sum_{i=1}^{n}u_i v_i$$

For the case where the two vectors $\vec{X}_1 = \vec{X}_2 = \begin{bmatrix}1 & 1 & \cdots & 1\end{bmatrix}^T$,

$$\vec{X_1}^T \vec{X}_2 = \sum_{i=1}^{n} 1(1) = n$$

When $\vec{X}_2 = \begin{bmatrix}1 & -1 & 1 & -1 & \cdots & 1 & -1\end{bmatrix}^T$ (with an even number of elements)

$$\vec{X_1}^T \vec{X}_2 = 1(1) + 1(-1) + \cdots + 1(1) + 1(-1) = 0$$

For pairs of vectors length n with values of ± 1, the larger the dot product, the more similar the vectors are. This notion of the dot product as a gauge of similarity between vectors will persist throughout the course.

In general, a dot product with a very negative value means the vectors are very different. However, it turns out humans are unable to perceive the "sign" of sound, so hearing a vector $\vec{X}$ sounds identical to hearing a vector $-\vec{X}$! That means that for the purposes of this problem, it's the *absolute value* of the dot product we care about. Don't take off points if you didn't mention absolute value.

(b) Next suppose we want to search for a short audio clip in a longer one. We might want to do this for an application like *Shazam* to be able to identify a song from a signature tune. Consider the vector of length 8, $\vec{X} = \begin{bmatrix}1 & -1 & -1 & 1 & 1 & 1 & -1 & 1\end{bmatrix}^T$. Let us label the elements of $\vec{X}$ so that $\vec{X} = \begin{bmatrix}x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8\end{bmatrix}^T$. Our goal is to find the short segment $\vec{Y} = \begin{bmatrix}1 & 1 & 1\end{bmatrix}^T$ in the longer vector (i.e. we want to find i, such that the sequence represented by $\begin{bmatrix}x_i & x_{i+1} & x_{i+2}\end{bmatrix}^T$ is the closest to $\vec{Y}$). Come up with an approach to do this. Can your approach be written as a matrix vector multiplication $\mathbf{A}\vec{x}$ where $\mathbf{A}$ is 6×8 and $\vec{x}$ is 8×1? Applying your technique, which i gives the best match for $\vec{Y} = \begin{bmatrix}1 & 1 & 1\end{bmatrix}^T$?

Solution: For each length-3 sub-sequence of $\vec{X}$ say $\vec{X}_i = \begin{bmatrix}x_i & x_{i+1} & x_{i+2}\end{bmatrix}^T$ (the subsequence which starts at position i), take the dot product of $\vec{X}_i$ and $\vec{Y}$. The length-3 sub-sequences $\vec{X}_i$ with the maximum-magnitude dot product with $\vec{Y}$ gives us the closest match with $\vec{Y}$. Computing the dot products:

$$i = 1 : \begin{bmatrix}1 & -1 & -1\end{bmatrix} \begin{bmatrix}1\\1\\1\end{bmatrix} = -1$$

$$i = 2 : \begin{bmatrix}-1 & -1 & 1\end{bmatrix} \begin{bmatrix}1\\1\\1\end{bmatrix} = -1$$

$$i = 3 : \begin{bmatrix}-1 & 1 & 1\end{bmatrix} \begin{bmatrix}1\\1\\1\end{bmatrix} = 1$$

$$i = 4 : \begin{bmatrix}1 & 1 & 1\end{bmatrix} \begin{bmatrix}1\\1\\1\end{bmatrix} = 3$$

$$i = 5 : \begin{bmatrix}1 & 1 & -1\end{bmatrix} \begin{bmatrix}1\\1\\1\end{bmatrix} = 1$$

$$i = 6 : \begin{bmatrix}1 & -1 & 1\end{bmatrix} \begin{bmatrix}1\\1\\1\end{bmatrix} = 1$$

We know that matrix-vector multiplication can be written as a series of dot products

$$A\vec{x} = \begin{bmatrix} - & \vec{a}_1^{\,T} & - \\ - & \vec{a}_2^{\,T} & - \\ & \vdots & \\ - & \vec{a}_n^{\,T} & - \end{bmatrix} \vec{x} = \begin{bmatrix} \vec{a}_1^{\,T}\vec{x} \\ \vec{a}_2^{\,T}\vec{x} \\ \vdots \\ \vec{a}_n^{\,T}\vec{x} \end{bmatrix}$$

so we can also implement the 6 dot products in one matrix multiplication. For instance, if $\vec{X} = \begin{bmatrix} x_1 & x_2 & \dots & x_8 \end{bmatrix}^T$ and $\vec{Y} = \begin{bmatrix} y_1 & y_2 & y_3 \end{bmatrix}^T$, the dot products (or correlation) $z_i = \vec{X}_i \cdot \vec{Y}$ can be represented as:

$$\begin{bmatrix} y_1 & y_2 & y_3 & 0 & 0 & 0 & 0 & 0 \\ 0 & y_1 & y_2 & y_3 & 0 & 0 & 0 & 0 \\ 0 & 0 & y_1 & y_2 & y_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & y_1 & y_2 & y_3 & 0 & 0 \\ 0 & 0 & 0 & 0 & y_1 & y_2 & y_3 & 0 \\ 0 & 0 & 0 & 0 & 0 & y_1 & y_2 & y_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix} = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \\ z_5 \\ z_6 \end{bmatrix}$$

We then pick the z_i with the largest magnitude and the corresponding $\vec{X}_i$ gives us the required substrings. This is connected to the ideas of cross-correlation which will be explored later on.

Note:

We sometimes extend beyond the length of $\vec{X}$—that is, we can include $i = 7, 8$ and $i = -1, 0$. Depending on the situation, we can either (1) assume all elements outside those in the original vector $\vec{X}$ are 0, (we call this "zero-padding"), or (2) take a cyclical approach to the elements of $\vec{X}$. If you did this, the equivalent matrix-vector multiplication will not have precisely the dimensions we specified. However, **if you correctly performed either the cyclical correlation or zero-padding and your matrix-vector multiplication matches your equations, you should give yourself full credit**.

Case 1: Zero Padding

In this case, we extend $\vec{X}$ beyond the signal of interest (the ± 1 values) to rewrite it

$$\vec{X}_{\text{zero-padded}} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ -1 \\ -1 \\ 1 \\ 1 \\ 1 \\ -1 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

For clarity, we maintain the same indexing as above.

$$i=-1:\begin{bmatrix}0 & 0 & 1\end{bmatrix}\begin{bmatrix}1\\1\\1\end{bmatrix}=1$$

$$i=0:\begin{bmatrix}0 & 1 & -1\end{bmatrix}\begin{bmatrix}1\\1\\1\end{bmatrix}=0$$

$$i=7:\begin{bmatrix}-1 & 1 & 0\end{bmatrix}\begin{bmatrix}1\\1\\1\end{bmatrix}=0$$

$$i=8:\begin{bmatrix}1 & 0 & 0\end{bmatrix}\begin{bmatrix}1\\1\\1\end{bmatrix}=1$$

This doesn't change the location of where the match occurs, but it will change your matrix-vector multiplication. Both of the following can be considered correct, though the resulting vector will be differently sized:

$$\begin{bmatrix}
y_1 & y_2 & y_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & y_1 & y_2 & y_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & y_1 & y_2 & y_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & y_1 & y_2 & y_3 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & y_1 & y_2 & y_3 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & y_1 & y_2 & y_3 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & y_1 & y_2 & y_3 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & y_1 & y_2 & y_3 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & y_1 & y_2 & y_3 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & y_1 & y_2 & y_3
\end{bmatrix}\vec{X}_{\text{zero-padded}}$$

$$\begin{bmatrix}
y_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
y_2 & y_3 & 0 & 0 & 0 & 0 & 0 & 0\\
y_1 & y_2 & y_3 & 0 & 0 & 0 & 0 & 0\\
0 & y_1 & y_2 & y_3 & 0 & 0 & 0 & 0\\
0 & 0 & y_1 & y_2 & y_3 & 0 & 0 & 0\\
0 & 0 & 0 & y_1 & y_2 & y_3 & 0 & 0\\
0 & 0 & 0 & 0 & y_1 & y_2 & y_3 & 0\\
0 & 0 & 0 & 0 & 0 & y_1 & y_2 & y_3\\
0 & 0 & 0 & 0 & 0 & 0 & y_1 & y_2\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & y_1
\end{bmatrix}\vec{X}$$

Case 2: Cyclical Correlation

Sometimes, instead of saying that $\vec{X}$ simply ends at its last element, we can act as though there's another copy of $\vec{X}$ attached before and after it. This situation applies when the sound defined by $\vec{X}$ is repeated several times in a row (see: the pop music industry), and we can describe this by extending $\vec{X}$ like so:

$$\vec{X}_{\text{cyclical}} = \begin{bmatrix} -1 \\ 1 \\ 1 \\ -1 \\ -1 \\ 1 \\ 1 \\ 1 \\ -1 \\ 1 \\ 1 \\ -1 \end{bmatrix}$$

All we've done here is take take the last two elements of $\vec{X}$ and placed them at the front, and taken the first two elements of $\vec{X}$ and placed them at the end. Maintaining the same indexing as the original vector,

$$i = -1 : \begin{bmatrix} -1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = 1$$

$$i = 0 : \begin{bmatrix} 1 & 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = 1$$

$$i = 7 : \begin{bmatrix} -1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = 1$$

$$i = 8 : \begin{bmatrix} 1 & 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = 1$$

Similar to the case with zero padding, there are several ways to write the matrix-vector multiplication which will provide the same information, albeit with differently sized output vectors.

$$\begin{bmatrix} y_1 & y_2 & y_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & y_1 & y_2 & y_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & y_1 & y_2 & y_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & y_1 & y_2 & y_3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & y_1 & y_2 & y_3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & y_1 & y_2 & y_3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & y_1 & y_2 & y_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & y_1 & y_2 & y_3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & y_1 & y_2 & y_3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & y_1 & y_2 & y_3 \end{bmatrix} \vec{X}_{\text{cyclical}}$$

$$\begin{bmatrix} y_3 & 0 & 0 & 0 & 0 & 0 & y_1 & y_2 \\ y_2 & y_3 & 0 & 0 & 0 & 0 & 0 & y_1 \\ y_1 & y_2 & y_3 & 0 & 0 & 0 & 0 & 0 \\ 0 & y_1 & y_2 & y_3 & 0 & 0 & 0 & 0 \\ 0 & 0 & y_1 & y_2 & y_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & y_1 & y_2 & y_3 & 0 & 0 \\ 0 & 0 & 0 & 0 & y_1 & y_2 & y_3 & 0 \\ 0 & 0 & 0 & 0 & 0 & y_1 & y_2 & y_3 \\ y_3 & 0 & 0 & 0 & 0 & 0 & y_1 & y_2 \\ y_2 & y_3 & 0 & 0 & 0 & 0 & 0 & y_1 \end{bmatrix} \vec{X}$$

(c) **(PRACTICE)** Now suppose our vector was represented using integers and not just by 1 and -1. We want to find the subsequence of the longer vector that is most similar to that of a sample vector. Say we wanted to locate the sequence closest in direction to $\vec{Y} = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}^T$ within $\vec{X} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{bmatrix}^T$. Can you explain why the approach in part (b) won't work? Consider using the norm/magnitude of the short segments (the norm/magnitude of a vector is defined as $\|\vec{z}\| = \sqrt{z_0^2 + z_1^2 + \ldots + z_n^2}$). How might you use this quantity to modify your approach for the new vectors to focus on the direction rather than the magnitude of the vectors?

Solution: Applying the approach in part (b), we get the best match to be $\begin{bmatrix} 6 & 7 & 8 \end{bmatrix}^T$ as this has the largest dot product with $\vec{Y} = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}^T$.

One way to modify the previous approach is by considering how close the "directions" of $\begin{bmatrix} 1 & 2 & 3 \end{bmatrix}^T$ and any length-3 substring of $\vec{X}$ is. The unit vector in the direction of $\begin{bmatrix} 1 & 2 & 3 \end{bmatrix}^T$ is $\vec{Y}_u = \frac{1}{\sqrt{14}} \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}^T$ and the unit vector in the direction of any length-3 substring $\vec{X}_i = \begin{bmatrix} x_i & x_{i+1} & x_{i+2} \end{bmatrix}^T$ is then given by $\vec{U}_i = \frac{1}{\|\vec{X}_i\|} \begin{bmatrix} x_i & x_{i+1} & x_{i+2} \end{bmatrix}^T$. The unit vector $\vec{U}_i$ which has the maximum-magnitude dot product with $\vec{Y}_u$ (i.e., $\vec{U}_i \cdot \vec{Y}_u$) is the one most aligned with the vector $\vec{Y}$. Thus in our example for $\vec{X} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{bmatrix}^T$ and $Y = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}^T$, the unit vector of the length-3 substring $\vec{X}_1 = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}^T$ has the maximum dot product ($\vec{U}_1 \cdot \vec{Y}_u = 1$).
What are other interesting ways to achieving this?

(d) In the IPython notebook, `prob3.ipynb`, complete part 1. **Solution:** See the solutions in the ipython notebook.

(e) In the IPython notebook, `prob3.ipynb`, complete part 2. **Solution:** See the solutions in the ipython notebook.

8. Homework Process and Study Group

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

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.