Gentle Intro Workshop to Matrix Operations and Linear Algebra
ENGN/PHYS 225
March 10, 2021

1 Introduction

Vectors and matrices pop up everywhere in physics and engineering. Why? Two reasons. First, systems of equations naturally pop up when one thing is connected to another, and matrix math is a natural way to handle them. Second, all real world data sets are ***discrete*** samples in time. Say you have m sensors, each of which measured n samples. There's an $m \times n$ matrix of data to analyze!

The point of today's exercise is to get familiar with basic matrix math in concept and in practice. Hello, Matlab. Be sure to consult the reference page of basic matrix operations at the end of this guide, as needed (Sec 3). Good to know what's happening under the hood before cranking the matlab engine.

Lastly, for matlab: if and when you are feeling stuck or unsure how to use a function or even what function to use, just ask Dr. Google! For example want to take the inverse of a matrix in matlab. Just google "Matlab inverse" and the official Mathworks documentation page should come right up, function name and all!

2 Workshop Problems

1. Let's say we have two vectors that we want to add. (We'll do this often when solving coupled ODEs). Given

$$\vec{v}_1 = \begin{bmatrix} 2 \\ 1 \end{bmatrix} \quad \text{and} \quad \vec{v}_2 = \begin{bmatrix} -1 \\ 4 \end{bmatrix}$$

compute the sum:

$$\vec{v}_3 = \vec{v}_1 + \vec{v}_2$$

2. Also compute the following:

$$\vec{v}_4 = 5\,\vec{v}_1 + 2\,\vec{v}_2 = 5 \begin{bmatrix} 2 \\ 1 \end{bmatrix} + 2 \begin{bmatrix} -1 \\ 4 \end{bmatrix}$$

. Again, we'll see this variant very often when finding superpositions of ***mode shapes*** (the shape encoded in the vector).

3. Now let's multiply matrices: Cryptography application time! Write out the alphabet letters corresponding to the matrix M below. For example, a = 1, b = 2, c = 3, etc. What does it spell out? Let's say this is our secret message. We can ***encrypt*** the message in M by multiplying it by the secret ***encryption key*** K. Only the sender and receiver, Alice and Bob, should have access to K. Now we'll transmit the encrypted message message is $C = KM$. It should look like relatively gobbly-gook at first glance, at least compared to the readout of M. If Alice sends the encrypted message C, how can Bob recover (aka ***decrypt***) the original content of the message M? Let's find out! This problem explores ***matrix multiplication***, the ***inverse***, and the ***determinant***.

$$M = \begin{bmatrix} 9 & 8 & 5 & 1 & 18 \\ 20 & 13 & 1 & 20 & 8 \end{bmatrix}$$

$$K = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$

(a) Compute the encrypted message $C = KM$. Compute by-hand at least 3 elements of C, then you can ask matlab to do the rest.

(b) To recover the original message M, we need to "undo" the encryption (aka transformation) done by K. This invokes the concept of the inverse as follows. We'll left-multiply by the inverse of K, denoted K^{-1}, which assumes this matrix actually exists (more on that in a minute):

$$K^{-1}(C = KM) \Rightarrow K^{-1}C = K^{-1}KM = IM = M$$

In other words, just left-multiply the encrypted message C by the inverse of K to recover the original M! Use matlab to compute the inverse of matrix K. Then given you have message C in hand, show that you can actually recover the original message M per the above matrix mathemagic.

(c) Were we guaranteed that K^{-1} exists? Does an inverse exist for any matrix? Emphatically, no! But we can apply a quick check using the ***determinant***. Compute $det(K)$, by hand first, then verify with matlab. A deep theorem from linear algebra is that a matrix is ***invertible*** (aka ***non-singular***) if and only if $det(A) \neq 0$. Compute the determinant in matlab using the det(.) function like this >> $det(A)$.

(d) Based on your answer above, you can now see why the current K was an acceptable key. But not all possible matrices are. Invent a non-invertible key. Show explicitly it is non-invertible.

4. Let's say you have the following system of equations:

$$\begin{aligned} 5x_1 - 3x_2 &= 10 \\ -20x_1 + 12x_2 &= -40 \end{aligned}$$

Try to solve this system of equations by hand! (Hint: there are infinitely many solutions!) Next: write this system in $Ax = b$ form, where $x = [x_1, x_2]^T$. Then, compute the determinant of $det(A)$. Ask matlab to solve using $x = A^{-1}b$. The point of this exercise, is that the inverse only exists—i.e., there is a ***unique*** x if and only if $det(A) \neq 0$. This is an very important fact about linear systems you should commit to math memory.

The point of this problem: For the above system of equations, you may notice that the second equation is the same as the first one multiplied by -4. Thus, they are termed ***linearly dependent***. That is, both equations describe the same line. *Any* point on this line solves the system, thus there are infinitely many solutions, thus the solution is not unique.

5. Now Let's say you have the following system of equations:

$$\begin{aligned} 5x_1 - 3x_2 &= 10 \\ -10x_1 + 3x_2 &= -40 \end{aligned}$$

Per the problem above, solve this system of equations by hand. Then write in $Ax = b$ form, compute $det(A)$, and solve for the unknowns using matlab. Of course, you could easily extend this analysis to equations of N unknowns, where N might be 4 or 10 or 10^6. No sweat off your back, let matlab do the heavy lifting!

6. Let's check out the identify matrix—no identity crisis here. First consider the 2×2 ***identify matrix*** Given

$$I_{2\times 2} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

In general, an identity matrix is a ***square matrix*** that has all ones along the ***main diagonal*** and zeros everywhere else. Typically we leave off the $n \times n$ subscript the dimensions should be obvious based on the context of the problem.

Compute the result of multiplying the matrix A by I where

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

Does it matter if you right or left multiply by the identity matrix: $AI \stackrel{?}{=} IA$. This is equivalent to asking whether $1 \times 5 \stackrel{?}{=} 5 \times 1$ *Try this exercise first by hand, then with MATLAB.*

7. And now for a brief introduction to eigenvalues and eigenvectors. The name sounds terrifying but it really isn't so bad at all. Compute the following by hand: $B = A - \lambda I$, where $\lambda = 3$. Note: *lambda* is a scalar value and matlab is happy to multiply all elements of a matrix by this scalar value. Then show that the determinant of $B = A - \lambda I$ equals 0. You can and should double check your work in matlab.

8. Next, do the same as you did above, but for $\lambda = -1$. Does $det(B) = det(A - \lambda I) = 0$ again? (Hint: it should...but check!). This is no accident! It turns out that $\lambda = -1, 3$ are the ***eigenvalues*** of matrix A. You can verify this in matlab by simply using eig(Eigenvalues pop up everywhere: structural vibrations, AI/ML, disease modeling, and quantum mechanics, just to name a few. They are the backbone of applied linear algebra problems.

9. Continuing on our journey, we'll look at eigenvectors. Each eigenvalue has a best-buddy eigenvector. They *always* run around in pairs. The class eigenvalue problem can be written:

$$A\vec{x} = \lambda\vec{x}$$

This says that we operate on $\vec{x}$ per the "instructions" encoded in matrix A and the result just kicks back a scalar multiple of the original vector. These are very special vectors called eigenvectors. Find the eigenvectors corresponding to both $\lambda = -1$ and $\lambda = 3$.

It is often helpful to write out this equation long-hand:

$$\begin{bmatrix} 1 & -2 \\ -2 & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \lambda \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}$$

where we have defined $\vec{x} = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}$. Thus, we are on the hunt for values of a_1 and a_2 that solve the eigenvalue problem.

This is our launching point for solving some real world problems. Get ready, here comes diabetes/biophysical modeling 101!

10. Get ready for some computer graphics via matrix math! Let's define the 2D rotation matrix as follows:

$$R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ sin\theta & \cos\theta \end{bmatrix}$$

Imagine you have 3 vectors describing the location of three points in the Cartesian plane:

$$\vec{v}_1 = [x_1, y_1]^T = [1, 0]^T \qquad \vec{v}_2 = [x_2, y_2]^T = [1, 1]^T \qquad \vec{v}_3 = [x_3, y_3]^T = [0, 1]^T \qquad \vec{v}_4 = [x_4, y_4]^T = [0, 0]^T$$

Note that the $\vec{v}^T$ denotes the ***transpose*** of $\vec{v}$.

What happens if you transform these points by $R(\theta)$? That is, compute the new $\vec{v}_k^{'}$ ($k = 1, \ldots, 4$) as follows:

$$\vec{v}_k^{'} = R(\theta)\vec{v}_k$$

or in long-hand form:

$$\begin{bmatrix} x_k^{'} \\ y_k^{'} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_k \\ y_k \end{bmatrix}$$

Given $\theta = 45$ deg $= \pi/4$ radians, draw the initial constellation of points. Connect the dots to enclose a familiar shape. Then do the same for the transformed points. In the context of computer graphics, what is the net effect you see?

Hopefully this will help convince you that the matrix $R(\theta)$ is a ***rotation transformation***, which rotates a vector by some angle θ. Inside your computer graphics card, there are literally billions of these operations happening per second. Now you know!

Before saying ta-ta to this problem, develop an expression for the inverse rotation matrix $R^{-1}(\theta)$ which acts to rotate a vector in the opposite direction (CW). In other words:

$$R^{-1}(\theta)R(\theta) = R(\theta)R^{-1}(\theta) = I$$

11. One last fun bit with computer graphics: here's another fun and common ***linear transformation*** you'd find in many graphics rendering programs. Find an expression for a matrix H that reflects a point $[x_1, y_1]^T$ about the horizontal axis. In other words, you want to find a 2×2 matrix A such that

$$H \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} x_1 \\ -y_1 \end{bmatrix}$$

Show/draw a few test cases that your matrix H works as advertised.

12. Lastly, what happens if you rotate a point and then reflect it vs. reflect-then-rotate. Does the order of matrix operations matter (that is, are matrix operations ***commutative***?). Let's find out via a quick experiment. Starting with the same three 2-d position vectors as in problem 10, use matlab to compute the following (assume $\theta = \pi/4$ as before):

$$\vec{v}_k^{'} = H\, R(\theta)\, \vec{v}_k$$

$$\vec{v}_k^{'} = R(\theta)\, H\, \vec{v}_k$$

(a) Which of these first rotates then reflects? Briefly justify

(b) For both cases, plot the initial and final shapes (by connecting the dots between the constellation of points).

(c) Compare and contrast—is the final output the same? What does this result therefore imply about the commutative property of matrix operations? Hopefully, you will see that, in general, any two matrix operations A and B do NOT commute: $AB \neq BA$. Fun fact: this "simple" fact has profound results for quantum mechanics and the uncertainty principle!

3 Matrix Operations and Terminology

Below is a list of basic matrix operations and terms you should know—or at least recognize—by the end of this exercise. If you need a quick refresher, just follow the hyperlinks to some very helpful resources that are beautifully made and very readable! Note: you may need to download the document or view it in a pdf viewer for the links to show up. Chrome typically hides them (drats).

- Addition or subtraction. Element-wise operation: $C = A + B \qquad c_{ij} = a_{ij} + b_{ij}$
- Multiplication (but NOT division): "row by column" operation. Can only multiply matrices with same ***inner dimension***. If A is $m \times n$ and B is $n \times p$, then AB has dimensions $(m \times n) \times (n \times p) = m \times p$. The ijth element is the dot product of the ith row of A dotted with the jth row of B:

  $C = AB \qquad c_{ij} = \vec{a}_i \cdot \vec{b}_j \qquad c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$

  Note that *order of operation matters*, left-multiplying is NOT the same as right multiplying! In general, $AB \neq BA$.
- Transpose: Exchange the rows and columns of a matrix A. This is denoted as A^T.
- Determinant: Denoted as $\det(A)$. Familiar computation by hand for 2×2 and 3×3. Closed formula exists for higher dimensionality, but no on in their right mind does anything other than use a software package to compute it. One very special property is that the ***inverse exists if and only if the the determinant is non-zero***. $det(A) \neq 0 \Rightarrow A^{-1}$ exists.
- Identity matrix: $I_{n \times n}$ is a diagonal matrix with 1's on the main diagonal; 0 otherwise. $IA = A$ and $AI = A$.
- Inverse: Denoted as A^{-1}, has special property $AA^{-1} = I = A^{-1}A$. Only applies to ***square matrices*** (e.g. $n \times n$)
- Systems of linear equations: $Ax = b \Rightarrow A^{-1}Ax = Ix = x = A^{-1}b$. Here, A is the coefficient matrix, $x = [x_1, x_2, \ldots, x_n]^T$ are the unknowns, solved by using the inverse of A (if it exists).
- Linear transformations: $A\vec{x} = \vec{b}$. (Often, the vector symbol is dropped.) This says the matrix A operates on a vector $\vec{x}$, making a new vector $\vec{b}$. In other words, A *transforms* $\vec{x}$ into $\vec{b}$.
- Eigenvalues and eigenvectors: $Ax = \lambda x$ says that when a matrix A operates on vector x, it just spits out the same vector, scaled by a constant λ (the eigenvalue). The vectors x for which the above equation is true are special ones called eigenvectors. The constant λ is called the eigenvalue. They are solved using: $det(A - \lambda I) = 0$. Of fundamental importance to quantum physics, vibrations, analysis, and many other branches in physics and engineering!