Lab #2a -- Introduction to Linear Algebra and Common Array and Matrix Operations in Matlab

Overview

This lab will contain work involving both analytical manipulation (i.e. by hand with pencil and paper) and running Matlab to perform a variety of array and matrix operations that are frequently needed in many engineering applications. In particular, 2-D (and higher order) arrays are commonly used for storage of numerical data (and other forms of information) and being able to properly extract the information of interest for a particular task is an essential operation that must be mastered -- and this involves having a good understanding of **array indexing** in Matlab. In addition, many practical engineering applications involve **formal linear algebra operations**, and the practicing engineer needs to be able to do many of these operations by hand and/or with the help of a computational tool (such as Matlab). Thus, this lab will also give you some hands-on practice with many of the linear algebra operations that are commonly used for many real-world applications.

By the end of this lab you should be quite comfortable with several basic linear algebra operations -- being able to perform these both via hand manipulation and within Matlab. Hopefully you will leave this lab with a much better understanding of the notation and manipulations that are involved, and have gained the knowledge and confidence needed to properly use Matlab, as needed, to simplify your life -- since doing these operations by hand can indeed be very tedious (or impractical as the size of the arrays increases beyond the illustrative 2×2 or 3×3 systems utilized here).

Good luck with the exercises below -- and do not hesitate to ask the lab instructor for personal help as needed, since the goal here is for everyone to leave the lab with sufficient knowledge to be functional with the introductory linear algebra and array manipulation concepts illustrated here…

Array Manipulations in Matlab -- Try the following tasks:

Note: For this exercise, you can work in interactive mode (i.e. directly in Matlab's Command Window) or you can put each of the commands in a Matlab script file and run the script file from the Editor, as desired. Either way, you should leave off the semi-colon at the end of the commands so that the desired results echo back to the screen -- the idea here is to immediately "see" the result of the various array indexing commands used. You might also use **format short** at the beginning to get a floating point format with 5 digits as the output -- to help streamline the numerical values output to the screen.

a. Enter the following command to create a 30×6 array, A, of random numbers using Matlab's **rand** command with the values scaled to vary between –2 and 2: **A = 4*rand(30,6) - 2**

b. Enter the following command to select only rows 4, 13, and 21 and echo these values to the screen: **B = A([4, 13, 21],:)**

c. Enter the following command to print columns 2, 4, 6 to the screen: **A(:,[2:2:6])**

d. Extract data from rows 2, 8, and 30 and columns 1 and 6 – this should give a 3×2 array: **???**

e. Use Matlab's **sum** command to add the values in column 3: **???**

f. What is the sum of all the values in the A array? **???**

g. Now divide the result from Part f by 30*6 = 180: **???** Is this approximately the result you expected? Does the command **mean(mean(A))** give the same result? What would the average result be if you had initially used a 300×6 array instead of a 30×6 array?

h. Find the **inner product** of rows 12 and 27 of the original A matrix -- the result here should be a 1×1 scalar (remember that the single quote is the transpose operator in Matlab): **???** Does the order of the vectors matter?

i. What will be the size of the **outer product** of rows 12 and 27 of the original A array: **???** Does the order of the vectors affect the resultant matrix?

j. Plot column 6 of A versus column 2 of A using only symbols to identify the discrete points (that is, do not use lines between the points). **Note:** You might try this task with lines just to see what you get -- do you understand what Matlab is doing here?

Some Matrix Multiplication Operations (by hand and with Matlab) -- Try the following tasks:

Given the following matrices

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

a. Perform the indicated operations via hand calculations (show intermediate results, as appropriate):

$\mathbf{B}^T\mathbf{B}$, $\mathbf{ABC}$, $\mathbf{x}^T\mathbf{Ax}$, $\mathbf{xx}^T$, and $\mathbf{A}^2$

b. Now, define the above arrays in a short Matlab script file and perform the indicated operations in Matlab. Do your results from Part a agree exactly with the Matlab results? If not, be sure to understand why and to fix any errors you may have (also don't hesitate to ask the lab instructor for clarification, as needed!!!)…

c. What is the difference between **A*A** and **A.*A** in Matlab? Make sure you understand what is going on here, since knowing the difference between "matrix multiplication" and "element by element multiplication" is essential for the proper use of Matlab in many applications…

The Matrix Inverse (by hand and with Matlab) -- Do the following:

Given the following data:

$$A = \begin{bmatrix} -1 & 1 & -3 \\ 3 & 0 & -1 \\ 5 & -2 & 5 \end{bmatrix} \qquad B = \begin{bmatrix} 2 & -1 & 0 \\ 1 & 0 & -2 \\ 0 & -3 & 2 \end{bmatrix} \qquad x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad y = \begin{bmatrix} 1 \\ 0 \\ -2 \end{bmatrix}$$

a. Calculate det **A**
   1. Using Laplace's Expansion
   2. By performing row operations to obtain an upper triangular matrix

b. Find the inverses of **A** and **B**

c. Solve the following matrix equation for the solution vector **x**: **Bx = y**

d. Parts a – c should be done via hand calculation. Now, for Part d, use Matlab to verify that you did these correctly, and address any differences that you get (note: just use Matlab's **det** command to do the Matlab implementation of Part a).

Note: If you were already familiar with some basic linear algebra concepts and the use of matrix algebra in Matlab, then the above tasks can be completed well within our 2-hr lab period -- if so, show the lab instructor your work and he/she will give you permission to leave class early, but only if everything is completed correctly!

On the other hand, if many of the above tasks are new to you, then you may struggle to finish all of them before the end of the lab. If so, please complete these tasks on your own outside of class to make sure that you are comfortable with array indexing and the notation and manipulations needed to perform a number of basic linear algebra operations -- both by hand and in Matlab (since there is a good chance that many of these same manipulations will be required in subsequent HWs and Quizzes/Exams)…