AN INTRODUCTION TO LINEAR ALGEBRA USING PYTHON

Summer 2021
Zoom Lecture: Tu: 2:00-4:00 p.m.
National Science Foundation (NSF) Center for Integrated Quantum Materials (CIQM), DMR -1231319
Dr. Steven L. Richardson (srichards22@comcast.net)
Professor Emeritus of Electrical Engineering, Department of Electrical and Computer Engineering, Howard University, Washington, DC
and
Faculty Associate in Applied Physics, John A. Paulson School of Engineering and Applied Sciences, Harvard University, Cambridge, MA

PROBLEM SET II
(due Tuesday, June 1, 2021)

Problem 1

Use the method of **cofactors and minors** to find the determinant of the following matrix by expanding **both** about the second row and then expanding about the third column:

$$\mathrm{D} = \begin{pmatrix} 2 & -1 & 1 \\ 0 & 3 & -1 \\ 2 & -2 & 1 \end{pmatrix}$$

Problem 2

Use the method of **cofactors and minors** to find the determinant of the following matrix:

$$\mathrm{P} = \begin{pmatrix} 1 & sin\,x & cos\,x \\ 0 & cos\,x & -sin\,x \\ 0 & -sin\,x & -cos\,x \end{pmatrix}$$

Problem 3

Use the method of **cofactors and minors** to find the determinant of the following matrix in two different ways: (a) by expanding in cofactors about the second row of the matrix and (b) by expanding in cofactors about the third column of the matrix.

$$\mathrm{S} = \begin{pmatrix} 2 & -1 & 1 \\ 0 & 3 & -1 \\ 2 & -2 & 1 \end{pmatrix}$$

Problem 4

Find the values of x that satisfy the determinantal equation:

$$\begin{vmatrix} x & 1 & 1 & 1 \\ 1 & x & 0 & 0 \\ 1 & 0 & x & 0 \\ 1 & 0 & 0 & x \end{vmatrix} = 0$$

Problem 5

Find the values of x that satisfy the determinantal equation:

$$\begin{vmatrix} x & 1 & 0 & 1 \\ 1 & x & 1 & 0 \\ 0 & 1 & x & 1 \\ 1 & 0 & 1 & x \end{vmatrix} = 0$$

Problem 6

Use the method of **Gauss-Jordan elimination** to solve the following equations:

$$\begin{aligned} x_1 + 2x_2 - 3x_3 &= 4 \\ 2x_1 - x_2 + x_3 &= 1 \\ 3x_1 + 2x_2 - x_3 &= 5 \end{aligned}$$

Problem 7

Use the method of **Gauss-Jordan elimination** to solve the following equations:

$$\begin{aligned} 2x_1 - x_3 &= -1 \\ 3x_1 + 2x_2 &= 4 \\ 4x_2 + 3x_3 &= 6 \end{aligned}$$

Problem 8

Use the method of **Gauss-Jordan elimination** to solve the following equations:

$$\begin{aligned} x_1 + 2x_3 - x_4 &= 3 \\ x_2 + x_3 &= 5 \\ 3x_1 + 2x_2 - 2x_4 &= -1 \\ - x_3 + 4x_4 &= 13 \\ 2x_1 - x_3 + 3x_4 &= 11 \end{aligned}$$

Problem 9

Use the method of **Gauss-Jordan elimination** to solve the following equations:

$$\begin{array}{rcrcrcr} x_1 & + & x_2 & - & x_3 & = & 2 \\ 2x_1 & - & x_2 & + & 3x_3 & = & 5 \\ 3x_1 & + & 2x_2 & - & 2x_3 & = & 5 \end{array}$$

Problem 10

Use the method of **Gauss-Jordan elimination** to solve the following equations:

$$\begin{array}{rcrcrcr} x_1 & + & x_2 & + & x_3 & = & -2 \\ x_1 & - & x_2 & + & x_3 & = & 2 \\ -x_1 & + & x_2 & - & x_3 & = & -2 \end{array}$$

Python Exercise 2

NumPy is a Python library that is useful for numerical calculations. SymPy is a Python library that is useful for symbolic calculations or symbolic algebra. Why would you ever want to use SymPy?

Suppose you have the matrix

$$\mathbf{Z} = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$$

Try using the appropriate commands from NumPy to find its determinant. You see the problem! SymPy will solve our problem. Actually, SymPy is more powerful than NumPy but we will not go into the differences in detail in our course.

Here is a first look at some useful commands in SymPy:

import sympy as sp #import the SymPy library and rename it as sp for convenience

from sympy import * #import from the SymPy library all required functions

sp.init_printing() #This enables you to call the special printing function init_printing inside SymPy to produce all output in very high quality. The function sp.init_printing() is an example of a function which does the high quality printing as output. The parentheses () indicate that no arguments are needed for you to pass to this function to run it.

a,b,c,d = sp.symbols ('a,b,c,d') #defines the symbols (a,b,c,d) to be used henceforth

display((a,b,c,d)) #shows the symbols as a matrix in high quality format

A = sp.Matrix([[a,b],[c,d]]) #this is how you build a matrix in SymPy which is called A where sp.Matrix is a function in SymPy

A #all you need do is type A to see the matrix in high quality format

A.det() #This takes the function **det** which calculates the determinant of the matrix A and prints it out. Again no arguments are required for you to pass this function to. By simply appending det() to the end of the matrix after a period, SymPy is smart enough to simply find the determinant of **A**.

1. Now using SymPy and the above commands find the determinant of the following matrix

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$$

Do you get what you expect? What about for the general examples of 3 x 3 and 4 x 4 matrices. Now you see why our "high school" approach for determinants of matrices does not work for matrices of order 4 and higher. For example, the 4 x 4 example has 24 terms and not eight terms!

2. You can time a Python command using the time package. The timer is not very accurate for very small times, say, measured in microseconds (10^{-6} seconds). You should run the command more than once; it can be a lot faster on the second or subsequent runs. First let us learn how to generate random (actually pseudo-random) matrices in Python.

import numpy as np

A = np.random.random((2,2)) # create a 2 x 2 matrix A with random (actually pseudo-random) matrix elements between 0 and 1

B = np.random.random((2,2)) # create a 2 x 2 matrix B with random (actually pseudo-random) matrix elements between 0 and 1

import time#import time module

start = time.time()#use a command from time module to note start time where start is a variable

A@B#this is how you multiply matrices in NumPy where we will discuss matrix multiplication in Lecture w

end = time.time()#use a command from time module to note end time when calculation is finished where end is a variable

print(end - start)

3. What is the maximum number of zeroes that a 3 x 3 matrix can have without having a zero determinant? Use either NumPy or SymPy to solve this problem. Justify your results with some reasoning.

4. Choose random 3 x 3 matrices **A** and **B** and compute det **AB** = det **A** det **B**. Repeat this process as often as needed to be reasonably certain about the relationship between these quantities. What is the relationship?

5. Use Python to evaluate the following determinant and show that it can be simplified to yield the following result. Note here you will have to use the Python command **factor** to factor your final result, define it as a new matrix, and print it out **(e.g. F = factor(S))** where **F** and **S** are arbitrary matrices.

$$\begin{vmatrix} a & b & c & d \\ -b & a & d & -c \\ -c & -d & a & b \\ -d & c & -b & a \end{vmatrix} = (a^2 + b^2 + c^2 + d^2)^2$$

6. Use Python to evaluate the following determinant and show that it can be simplified. Note here you will have to use the Python command **factor** to factor your final result, define it as a new matrix, and print it out **(e.g. F = factor(S))** where **F** and **S** are arbitrary matrices.

$$\begin{vmatrix} (a+b)^2 & c^2 & c^2 \\ a^2 & (b+c)^2 & a^2 \\ b^2 & b^2 & (c+a)^2 \end{vmatrix}$$

7. It is now time to look at the Gauss-Jordan elimination method using SymPy. Use the following commands to review all of the problems you have previously solved analytically using quite a bit of arithmetic to see that you get the desired matrix in reduced row echelon form. Don't forget the examples we worked on in Lecture 2!

```
import sympy as sp

from sympy import *

sp.init_printing()

B = sp.Matrix([[1, 0, 1, 3], [2, 3, 4, 7], [-1, -3, -3, -4]])#for example

B.rref()[0] # you will need the [0] syntax here using the rref function to get your answer
in simplest form where the rref command is apparently only defined in the SymPy library
```