PHY 422 week 5

Athira Sreejith
Roll no : ms18033

1 Problem 1

1. Write a simple code for multiplication of matrix A and B.

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

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

1.1 Algorithm

1. Two matrices a and b are defined.
2. A function matrix product is defined, which takes in two matrices x,y as input.

- If the number of columns of x is not equal to the number of rows of x, the function returns "The matrices cannot be multiplied"
- Else, a new zero matrix c is defined.
  Two for loops are initiated, with i and j iterating through the the rows of x, and the columns of y respectively.
  A new variable value = 0 is defined inside the 2nd for loop, and a new for loop is initiated with k iterating through all the columns of x, and adds x[i][k] * y[k][j] to value. The element c[i][j] of the matrix c is redefined as c[i][j] = value, when the innermost for loop terminates.
  After all the for loops terminate, the function returns the matrix c.
  3. The output of the function with inputs a and b,is printed as the matrix product of a,b.

1.2 Code

```
import numpy as np
a = [[1,2,3],[2,1,4],[1,4,3]]
b = [[2,1],[1,2],[2,1]]

def matrix_product(a,b):
        if len(a[0]) != len(b):
                return("The matrices cannot be multiplied")
        else:
                c = np.zeros([len(a), len(b[0])])
                for i in range(len(a)):
                        for j in range(len(b[0])):
                                value = 0
                                for k in range(len(b)):
                                        value += a[i][k] * b[k][j]
                                c[i][j] = value
                return(c)
print("The product of matrices, AB = \n", matrix_product(a,b))
```

Output

```
user1@user1-Inspiron-N5050:~/Desktop/Athi/MS18033_5$ python3 ms18033_5_code1.py
The product of matrices, AB =
 [[ 10.   8.]
 [ 13.   8.]
 [ 12.  12.]]
```

1.3 Summary

A function was defined and was used to multiply any two matrices, provided that the number of columns of the first matrix = number of rows of the second.

2 Problem 2

2. Use partial pivoting to solve using elimination $2x_2 + x_3 = 5$;
$4x_1 + x_2 - x_3 = -3$;

$-2x_1 + 3x_2 - 3x_3 = 5$

2.1 Algorithm

1. Two matrices are defined:

- a - the co-efficient matrix, $\begin{bmatrix} 0 & 2 & 1 \\ 4 & 1 & -1 \\ -2 & 3 & -3 \end{bmatrix}$
- b - [5, -3, 5]

2. The augmented matrix is created by using a for loop with i iterating through the indices of b and appending the corresponding elements to the ith list of a. 3. A function "solutions" is defined for back substitution, which takes in a matrix m as the input.

- In the function, a new variable sol is defined as a zero row matrix with the number of columns equal to the number of columns of m.
- The first element of sol is set as m[-1][-1]/m[-1][-2]
- Two for loops are initiated in which i is iterated through the indices of solution, and k takes values from 0 to i. For each k, the ith element of sol is redefined as : $sol[i] += -sol[k] * m[l-i-1][l-k-1]$, where l denotes the number of elements in m. This is again redefined as $sol[i] = (sol[i] + m[l-i-1][-1])/m[l-i-1][l-i-1]$ after the inner for loop terminates.
- Sol is then reversed and returned by the function.

4. A for loop is initiated, with k iterating through the columns of a (except the last two columns)
In the for loop:

- The row with maximum value for the kth column is obtained.
- The row with maximum value is swapped with the kth row with the help of a dummy variable.
- Two for loop are used to iterate through the rows below k (uisng a variable s), and then through the rows of each of those columns (using a variable t).
  The matrix elements are updated as : a[s][t] = a[s][t] - a[k][t]*a1/a2, where a1 = a[s][k] and a2 = a[k][k]. The updated matrix is printed.

5. The solutions are printed by using the function solutions(a).

2.2 Code

```
import numpy as np
a = [[0,2,1] , [4,1,-1] , [-2,3,-3]] #coefficient matrix
b = [5,-3,5]

#creating the augmented matrix
for i in range(len(b)):
        a[i].append(b[i])

#function for back substitution
def solutions(m):
        sol = np.zeros(len(m))
        sol[0] = m[-1][-1]/m[-1][-2]
        for i in range(1,len(sol)):
                for k in range(i):
                        sol[i] += - sol[k]*m[len(m)-i-1][len(m)-k-1]
                sol[i] = (sol[i] + m[len(m)-i-1][-1])/m[len(m)-i-1][len(m)-i-1]
        return(sol[::-1])



for k in range(len(a[0])-2): #iterating through columns of a

        #finding the row with maximum value for kth column
        maxi = a[k][k]
        row = k
        for r in range(k+1,len(a)):
                if maxi - a[r][k] < 0:
                        maxi = a[r][k]
                        row = r
```

```
        #swapping rows
        z = a[row]
        a[row] = a[k]
        a[k] = z


        #elimination
        for s in range(k+1,len(a)): #iterating through rows below k
                a1 = a[s][k]
                a2 = a[k][k]

                for t in range(len(a[0])): #iterating through colmuns of row s
                        a[s][t] = a[s][t] - a[k][t]*a1/a2

print("The solution of [x1,x2,x3] is ", solutions(a))
```

2.3 Output

```
user1@user1-Inspiron-N5050:~/Desktop/Athi/MS18033_5$ python3 ms18033_5_code2.py
The solution of [x1,x2,x3] is  [-1.  2.  1.]
```

2.4 Summary

A system of linear equations in 3 variables were converted to matrix multiplication form, and the augmented matrix with both the co-efficient matrix and the b values is defined in the code.
Gauss elimination was carried out in the augmented matrix, and a function was defined and used to carry out back substitution. The solutions were thus obtained and printed.

3 Problem 3

3. Use Gauss Jordan to solve
$2x_1 + 4x_2 + x_3 = 3$;

$3x_1 + 2x_2 - 2x_3 = -2$;
$3x_1 - 3x_2 + 3x_3 = 18$

3.1 Algorithm

1. Two matrices are defined:

- a - the co-efficient matrix, $\begin{bmatrix} 2 & 4 & 1 \\ 3 & 2 & -2 \\ 3 & -3 & 3 \end{bmatrix}$
- b - [3, -2, 18]

2. The augmented matrix is created by using a for loop with i iterating through the indices of b and appending the corresponding elements to the ith list of a. 3. A function "solutions" is defined for back substitution, which takes in a matrix m as the input.

- In the function, a new variable sol is defined as a zero row matrix with the number of columns equal to the number of columns of m.
- A for loops are initiated in which i is iterated through the indices of solution, and k takes values from 0 to i. For each i, sol[i] is updated as $sol[i] = m[i][-1]/m[i][i]$
- The function returns sol

4. A for loop is initiated, with k iterating through the columns of a (except the last two columns)
In the for loop:

- The row with maximum value for the kth column is obtained (with the help of two dummy variables).
- The row with maximum value is swapped with the kth row with the help of a dummy variable.
- A for loop are used, with a variable a iterating through the indices of rows of a.
  If the row $s \neq k$, a for loop is used to iterate through columns of a,(with the variable t)
  The matrix elements are updated as : a[s][t] = a[s][t] - a[k][t]*a1/a2, where a1 = a[s][k], a2 = a[k][k] and t - column number(or index).

5. The solutions are obtained by using the function solutions(a) and printed.

3.2 Code

```
import numpy as np
a = [[2,4,1] , [3,2,-2] , [3,-3,3]] #coefficient matrix
b = [3,-2,18]

#creating the augmented matrix
for i in range(len(b)):
        a[i].append(b[i])

#function for substitution
def solutions(m):
        sol = np.zeros(len(m))

        for i in range(len(sol)):
                sol[i] = m[i][-1]/m[i][i]

        return(sol)


for k in range(len(a[0])-1): #iterating through columns of a

        #finding the row with maximum value for kth column
        maxi = a[k][k]
        row = k
        for r in range(k+1,len(a)):
                if maxi - a[r][k] < 0:
                        maxi = a[r][k]
                        row = r
```

```
    #swapping rows
    z = a[row]
    a[row] = a[k]
    a[k] = z

    #elimination
    for s in range(len(a)): #iterating through rows except k

            if s != k:

                    a1 = a[s][k]
                    a2 = a[k][k]

                    for t in range(len(a[0])): #iterating through colmuns of
                            a[s][t] = a[s][t] - a[k][t]*a1/a2


print("The solution of [x1,x2,x3] is ", solutions(a))
```

3.3 Output

```
user1@user1-Inspiron-N5050:~/Desktop/Athi/MS18033_5$ python3 ms18033_5_code3.py
The solution of [x1,x2,x3] is  [ 2. -1.  3.]
```

3.4 Summary

The given system of linear eqautions were converted to matrix multiplication form. Gauss Jordan elimination was carried out on the matrix and the solutions were obtained using a function.

4 Problem 5

5. Try to make the above codes generalize for any matrix

4.1 Summary

All the codes above are generalised for any matrix. The initial matrices that are defined in the code can instead be asked as input from the user.