my_assignment3

September 4, 2021

[2]:
```
%run MyLibrary.ipynb
```

[3]:
```
# Function to swap row1 and row2

def swap_rows(Ab,row1,row2):
    temp = Ab[row1]
    Ab[row1] = Ab[row2]
    Ab[row2] = temp
    return Ab



# Function for partial pivoting

def partial_pivot(Ab,m,nrows):
    pivot = Ab[m][m]    # declare the pivot
    if (Ab[m][m] != 0):
        return Ab    # return if partial pivot is not required
    else:
        for r in range(m+1,nrows):
            # check for non-zero pivot and swap rows with it
            if Ab[r][m] != 0:
                pivot = Ab[r][m]
                Ab=swap_rows(Ab,m,r)
                return Ab
            else:
                r+=1
    if (pivot==0):    # if there is no unique solution
        return None



# Gauss Jordan Elimiination method

def gauss_jordan(Ab,nrows,ncols):
    det=1
    r=0
    # does partial pivoting
    Ab = partial_pivot(Ab,r,nrows)
```

```
    for r in range(0,nrows):
        # if there is no solution
        if Ab==None:
            return Ab
        else:
            # Changes the diagonal elements to unity
            fact=Ab[r][r]
            det=det*fact # calculates the determinant
            for c in range(r,ncols):
                Ab[r][c]*=1/fact
            # makes all elements other than diagonal elements to zero
            for r1 in range(0,nrows):
                # does not change if it is already done
                if (r1==r or Ab[r1][r]==0):
                    r1+=1
                else:
                    factor = Ab[r1][r]
                    for c in range(r,ncols):
                        Ab[r1][c]-= factor * Ab[r][c]
    return Ab, det


# Function to extract inverse from augmented matrix

def get_inv(A,n):
    r=len(A)
    c=len(A[0])
    M=[[0 for j in range(n)] for i in range(n)]
    for i in range(r):
        for j in range(n,c):
            M[i][j-n]=A[i][j]
    return M


# Function to round off all elements of a matrix

def round_matrix(M,r):
    for i in range(len(M)):
        for j in range(len(M[0])):
            M[i][j]=round(M[i][j],r)
    return M
```

[4]:
```
A = [[1,1,1,1, 13],[2,3,0,-1, -1], [-3,4,1,2, 10],[1,2,-1,1, 1]]
print("The augmented matrix is: ")
print_matrix(A,4,5)
GJ, d=gauss_jordan(A,4,5)
if GJ!=None:
```

```
    M=round_matrix(A,2)
    print("Solutions are : ")
    for i in range(4):
        print(M[i][4])
else:
    print("No unique solution exists.")
```

```
The augmented matrix is:

1    1    1    1    13

2    3    0    -1    -1

-3    4    1    2    10

1    2    -1    1    1


Solutions are :
2.0
-0.0
6.0
5.0
```

[5]:
```
B = [[0,2,-3,-1],[1,0,1,0],[1,-1,0,3]]
print("The augmented matrix is: ")
print_matrix(B,3,4)
GJ, d=gauss_jordan(B,3,4)
if GJ!=None:
    print("Solutions are : ")
    for i in range(3):
        print(B[i][3])
else:
    print("No unique solution exists.")
```

```
The augmented matrix is:

0    2    -3    -1

1    0    1    0

1    -1    0    3


Solutions are :
1.0
-2.0
-1.0
```

[6]:

```python
C2 = [[0,2,1],[4,0,1],[-1,2,0]]     # matrix
C = [[0,2,1, 1,0,0],[4,0,1, 0,1,0],[-1,2,0, 0,0,1]]     # augmented matrix
print("The augmented matrix is: ")
print_matrix(C,3,6)
GJ, d=gauss_jordan(C,3,6)
if GJ!=None:
    # Finding the inverse and printing in rounded form
    # Multiply the matrices to verify if it is identity and then rounding at the␣
 ↪end
    M=get_inv(C,3)
    MM,k,l=matrix_multiply(M,3,3,C2,3,3)
    M=round_matrix(M,2)
    print("The inverse matrix is: ")
    print_matrix(M,3,3)
    print("Verification: ")
    MM=round_matrix(MM,2)
    print_matrix(MM,3,3)
else:
    print("No unique solution exists.")
```

```
The augmented matrix is:

0    2    1    1    0    0

4    0    1    0    1    0

-1    2    0    0    0    1


The inverse matrix is:

-0.33    0.33    0.33

-0.17    0.17    0.67

1.33    -0.33    -1.33


Verification:

1.0    0.0    0.0

0.0    1.0    0.0

0.0    0.0    1.0
```

[7]:
```
D = [[1,4,2,3, 1,0,0,0],[0,1,4,4, 0,1,0,0],[-1,0,1,0, 0,0,1,0],[2,0,4,1,␣
↪0,0,0,1]]
print("The augmented matrix is: ")
print_matrix(D,4,8)
GJ, d=gauss_jordan(D,4,8)
if GJ!=None:
    print("Determinant of the matrix = ",end='')
    print(round(d,4))
else:
    print("No unique solution exists.")
```

```
The augmented matrix is:

1    4    2    3    1    0    0    0

0    1    4    4    0    1    0    0

-1    0    1    0    0    0    1    0

2    0    4    1    0    0    0    1


Determinant of the matrix = 65.0
```