TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

----------oOo----------

TOÁN ỨNG DỤNG THỐNG KÊ

Lab 1

Họ tên : Lê Nguyên Bình Nam
MSSV: 20127567

1. Mô tả các hàm:

a. Swap_row(A, i, j):

- Hàm nhận vào ma trận A và 2 giá trị i,j là 2 hàng cần hoán đổi vị trí, trả về ma trận đã được hoán đổi vị trí.

```
def swap_row(A,i, j):
    A[[i, j]] = A[[j, i]]
    return A
```

b. Mul_scalar(A,I,k):

- Hàm nhận vào ma trận A, giá trị I là vị trí dòng cần nhân và k là hệ số nhân, trả về ma trận A đã nhân hệ số k của dòng thứ I.

```
def mul_scalar(A, i,k):
    col = len(A[0])
    for j in range(col):
        A[i][j]= A[i][j] * k
    return A
```

c. Add_row(A, I, j, k):

- Hàm nhận vào ma trận A, dòng thứ I là dòng cần thay đổi, j là dòng thứ j và k là hệ số tương ứng. Hàm trả về ma trận đã thay đổi dòng thứ i

```
def add_row(A, i, j, k):
    col = len(A[0])
    for col12 in range(col):
        A[i][col12] = A[i][col12] + k*A[j][col12]
    return A
```

d. findNewRow(A, I, j):

- Hàm nhận vào ma trận A, I và j là dòng thứ I có hệ số nghiệm thứ j bằng 0. Sau đó dùng vòng lặp để tìm hàng mà tại đó hệ số của nghiệm thứ j khác 0, rồi thay đổi vị trí 2 hàng, trả về 1 nếu có và trả về 0 nếu không có.

```
def findNewRow(A , i, j):
    row = len(A)
    col = len(A[0])
    for x in range(i+ 1, row):
            if A[x, j] != 0:
                swap_row(A, i , x)
                return 1
    return 0
```

e. Gauss_elimination(A):

- Hàm nhận vào ma trận mở rộng.

- Đầu tiên kiểm tra xem hệ số đầu tiên của ma trận có bằng 0 hay không, nếu không thì sẽ dùng vòng lặp để kiểm tra và thay thế.
- Sau đó ta khởi tạo 1 biến để lưu trữ vị trí(posCol) nghiệm đã xét. Bắt đầu bằng 0, cho chạy vòng lặp với mỗi dòng. Nếu như posCol đã đi qua hết mọi nghiệm. thì sẽ break. Nếu không thì sẽ kiểm tra:
    - Nếu hệ số của nghiệm posCol bằng 0 thì ta sẽ gọi hàm findNewRow để tìm, nếu như tìm thấy thì sẽ thoát vòng lặp. Nếu không thì sẽ nhảy vị trí của posCol lên 1 đơn vị và tiếp tục.
- Sau khi xong thì ta sẽ chia hệ số của dòng I cho hệ số đầu tiên để hệ số đầu tiên của phương trình luôn bằng 1. Và từ những dòng tiếp theo thì ta lấy nhân với dòng đó với thương của hệ số dòng hiện tại chia cho hệ số dòng kế và lấy hiệu 2 dòng để để đưa về ma trận bậc thang
- Cuối cùng, ta sẽ xóa đi những dòng bằng 0 . Cho chạy vòng lặp từng dòng, với mỗi dòng sẽ chạy thêm 1 vòng nữa để kiếm ta, nếu như trong dòng đó có hệ số khác 0 thì thoát khỏi vòng lặp. Nếu không có hệ số khác 0 thì ta sẽ xóa dòng đó khỏi ma trận.
- Trả về ma trận bậc thang.

```
def gauss_elimination(A):
    row = len(A)
    col = len(A[0])
    if A[0][0] == 0:
        for i in range(1, row):
            if A[i , 0]!= 0:
                swap_row(A, 0 , i)
                break
    posCol = 0
    for i in range(1 ,row):
        if posCol == col -2:
            break
        if A[i-1][posCol] == 0:
            while(True):
                flag = findNewRow(A, i-1, posCol)
                if flag == 1:
                    break
                else:
                    posCol +=1
                    if A[i-1][posCol] != 0:
                        break
        A = mul_scalar(A, i-1, 1/A[i-1][posCol])
        temp = A[i-1][posCol]
        pos = i-1
        for j in range(i, row):
            devide = A[j][posCol]/temp
            A = add_row(A,j ,i-1, -devide)
        posCol+=1
```

```
# xoa di nhung phuong trinh ma tat ca he so bang 0
for i in range(row - 1, -1, -1):
    if any(A[i]):
        break
    A.pop(i)
    i -= 1
    row -= 1
return A
```

f. Back_substitution(A):

- Hàm nhận vào ma trận bậc thang A. Với mỗi ma trận đều có 3 trường hợp về nghiệm:
    - Trường hợp vô nghiệm: Ta sẽ kiểm tra rằng số phương trình có nhiều hơn số nghiệm hoặc tồn tại phương trình mà tất cả các hệ số của nghiệm bằng 0 còn hệ số tự do khác 0. Lúc đó ta kết luận phương trình vô nghiệm.

```
if(A[row-1][col -2]!= A[row-1][col-1] and A[row-1][col -2] == 0) or row > col -1:
    print("Phuong trinh vo nghiem")
    return None
```

    - Phương trình có nghiệm duy nhất: Ta sẽ khởi tạo một tập nghiệm rỗng toàn số 0. Sau đó mới mỗi phương trình, mắt đầu từ nghiệm xa nhất, ta lấy tập nghiệm nhân vô hướng với tập hệ số ứng với nghiệm của phương trình. Sau đó lấy hệ số tự do của phương trình trừ cho tích vô hướng vừa mới tình và chia cho hệ số của nghiệm đang tính(bằng 1). Cuối cùng ta cập nhật lại tập nghiệm.

```
# truong hop co nghiem duy nhat
i = row - 1
while i >= 0:
    sum = 0
    for j in range(col -1):
        sum = sum + res[j]*A[i][j]
    res[i] = A[i][-1]- sum
    i = i-1
return res
```

    - Trường hợp vô số nghiệm chưa hoàn thành.