Trường Đại học Khoa Học Tự Nhiên

Đại học Quốc gia Thành phố Hồ Chí Minh

ĐỒ ÁN 1

GIẢI HỆ PHƯƠNG TRÌNH BẰNG PHƯƠNG PHÁP GUASS

Giáo viên hướng dẫn :

- Đinh Ngọc Thanh
- Nguyễn Hữu Toàn
- Võ Nam Thục Đoan

Sinh viên thực hiện : Nguyễn Thuỵ Hoàng Yến

MSSV : 20127397 **Lớp** : 20CLC03

MỤC LỤC

1. Phương pháp Guass

- **Sơ lược về thuật toán**

  Phương pháp Guass , hay còn gọi là phương pháp tự sửa sai là một phương pháp lặp được sử dụng để giải một hệ phương trình tuyến tính. Ở trong đồ án này, em sẽ thực hiện giải một hệ phương trình bằng cách đưa ma trận về dạng ma trận bậc thang (như hình phía dưới) , và khi em đưa ma trận về dạng bậc thang tối giản em sẽ tiến hành tìm nghiệm của chúng.

  Giả sử ta có một ma trận sau:

$$\begin{bmatrix} 2 & 1 & -1 & 8 \\ -3 & -1 & 2 & -11 \\ -2 & 1 & 2 & -3 \end{bmatrix}$$

  Em sẽ tiến hành đưa ma trận trên về dạng ma trận bậc thang , bằng việc thực hiện các phép biến đổi sơ cấp trên dòng sao cho phù hợp, để có thể đưa về dạng bậc thang (như hình phía dưới)

$$\begin{bmatrix} 2 & 1 & -1 & 8 \\ 0 & \frac{1}{2} & \frac{1}{2} & 1 \\ 0 & 0 & -1 & 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & -1 \end{bmatrix}$$

  Khi đã đưa ma trận về dạng bậc thang rồi đưa về dạng bậc thang tối giản , em sẽ tiến hành tìm nghiệm của phương trình. Từ đó ta có thể suy ra được nghiệm của hệ phương trình trên.

- **Phân tích giải thuật**

  Phép thử Guass trên một ma trận NxM cần khoảng 2n^3/3 phép tính toán. Do đó độ phức tạp của thuật toán là O(n^3)

2. Ngôn ngữ sử dụng

Để thực hiện đồ án này, em sử dụng ngôn ngữ lập trình Python trên Pycharm IDE.

3. Sơ lược về ý tưởng thực hiện đồ án

Để thực hiện được giải thuật Guass , em sẽ tiến hành theo các bước sau:
Bước 1 : Tìm cột có giá trị bằng 0 với mục đích là đưa về theo dạng ma trận bậc thang. Nếu như cột đó có giá trị bằng 0 thì sẽ duyệt tới cột kế tiếp, tương tự cho đến khi thấy các giá trị đó khác 0.
Bước 2 : Tìm vị trí dòng khác 0 đầu tiên , với điều kiện nằm ở đầu cột. Nếu thoả mãn điều kiện trên thì ta tiến hành bước 3. Ngược lại, nếu giá trị khác 0 không là số hạng đầu tiên thì ta thêm một bước đổi dòng sau đó mới tiến hành bước kế tiếp.
Bước 3: Nhân dòng đó cho 1/a.
Bước 4: Ta thực hiện phép biến đổi sơ cấp trên dòng theo công thức: *di = di + kdj,* sau đó ta trừ hai dòng vừa tính và dòng được chọn là dòng cơ sở với nhau. Tuần tự cho đến khi ta đưa về được dạng ma trận bậc thang
Bước 5: Tiến hành kiểm tra nghiệm của hệ phương trình. Có ba loại :

- Phương trình có nghiệm : khi ma trận được đưa về dạng bậc thang , và không rơi vàp trường hợp vô số nghiệm hoặc vô nghiệm.
- Phương trình vô số nghiệm : ta kiểm định bằng việc kiểm tra xem có bao nhiêu dòng có giá trị toàn bộ bằng 0 , và nếu như tổng dòng – số dòng bằng 0 nhỏ hơn số cột của ma trận thì ta kết luận đó là phương trình vô số nghiệm.
- Phương trình vô nghiệm : khi ma trận đã đưa về dạng bậc thang tối giản thì ta xét giá trị dòng cuối cột cuối , nếu như giá trị đó bằng 0 mà phía bên phải ta có một số khác 0 thì ta kết luận đó là phương trình vô nghiệm.

4. Mô tả các hàm sử dụng

- **ShowMatrix()** : Hiện thị ma trận trên màn hình console

```
def showMatrix():
    for i in range(len(arr)):
        for j in range(len(arr[0])):
            print(arr[i][j], end='\t')
        print()
```

- **Check_Zero_Col(arr,col,row)** : Kiểm tra cột có giá trị bằng 0 hay không. Sử dụng một vòng lặp while với điều kiện dừng khi col truyền vào phải nhỏ hơn số cột của ma trận,nếu không thoả thì trả về FALSE, nếu thoả điều kiện trên, thì ta sẽ có một vòng lặp chạy từ row đến số dòng của ma trận, nếu giá trị tại dòng đó khác 0 thì ta trả về cột tại giá trị đó. Ngược lại, nếu giá trị tại dòng đó bằng 0 thì ta chuyển sang cột kết tiếp (col = col + 1).

```
def check_Zero_Col(arr, col, row):
    while (col < len(arr[0])):
        for i in range(row, len(arr)):
            if (arr[i][col] != 0):
                return col
        col = col + 1
    return -1
```

- **Find_Row_Zero(arr,row,col)** : Ta sẽ trả về vị trí của dòng đó với điều kiện giá trị dòng đó khác 0 và nằm đầu cột.

```
def find_Row_Zero(arr, row, col):
    for i in range(row, len(arr)):
        if (arr[i][col] != 0):
            return i
    return -1
```

- **Swap_Row(arr,row,temp)** : Ta có một vòng for chạy trong khoảng cột của ma trận và tiến hành đổi chỗ giữa hai dòng đó cho nhau.

```
def swapRow(arr, row, temp):
    for i in range(len(arr[0])):
        arr[row][i], arr[temp][i] = arr[temp][i], arr[row][i]
```

- **Add_Row(arr,col,row) :** Ta bắt đầu tại vị trí row + 1, và gọi k = arr[i]][col], duyệt các giá trị trong cột rồi ta tiến hành tính theo công thức : *di = di + kdj*

```
# di = di + k.dj
def add_Row(arr, col, row):
    for i in range(row + 1, len(arr)):
        k = arr[i][col]
        for j in range(col, len(arr[0])):
            arr[i][j] = arr[i][j] - k * arr[row][j]
```

- **Guass_Elimination(arr)** : Đặt n là số dòng, m là số cột , mặc định row , col là cột đầu tiên dòng đầu tiên. Sử dụng một vòng while với điều kiện dừng là row < n, nếu không thoả điều kiện trên thì trả về ma trận đó. Nếu như thoả, ta kiểm tra cột xem có giá trị nào bằng 0 hay không, khi không thoả điều kiện hàm check_Zero_Col thì ta trả về col = - 1(col = false) và trả về ma trận.
  Sau khi kiểm tra xong, ta chuyển qua bước 2, ta đặt temp = 0 , nếu như vị trí trả về sau khi kiểm tra khác vị trí 0 thì ta tiến hành đổi dòng cho nhau.
  Nếu như arr[row][col] khác 1 thì ta nhân theo công thức 1/a, sau đó ta cộng dòng như hàm add_Row.
- **Back_Substituation** : Ta sẽ có một biến đếm để đếm các dòng có tất cả giá trị đều bằng 0. Ở đây em dùng điều kiện if để kiểm tra xem trong các giá trị tại dòng đó có bằng 0 hay không, nếu như nó bằng 0 thì nó sẽ trả về 1 (TRUE) còn nếu một số nào đó khác 0 thì nó sẽ trả về 0(FALSE). Khi thấy dòng thoả điều kiện true, thì tăng biến count lên. Nếu như số dòng của ma trận trừ cho số dòng bằng 0 mà nhỏ hơn số cột và biến count đó khác 0 thì ta xuất ra hệ phương trình vô số nghiệm.
  Tương tự như trên, ta duyệt đến số cột – 1 để tìm các giá trị bằng 0, nếu như cột cuối có đáp án khác 0 mà bên trái ta đã đưa về dạng bậc thang thì ta xuất ra hệ phương trình vô nghiệm.
  Cấu trúc insert(vị trí chèn, giá trị chèn) để đưa kết quả arr[i][n] / arr[i][i] vào một mảng x khác.Sau đó lấy giá trị trong cột để nhân với kết quả chứa trong mảng mới.

  Khi chạy chương trình:

- Phương trình vô nghiệm

```
/Users/yennguyen/Desktop/Uni's_Life
--------------------
1   2   -1  -1
0   1   4   3
0   0   0   2
--------------------
1   2   -1  -1
0   1   4   3
0   0   0   2
--------------------
1   2   -1  -1
0   1   4   3
0   0   0   1.0
Hệ phương tình vô nghiệm

Process finished with exit code 0
```

- Phương trình vô số nghiệm

```
/Users/yennguyen/Desktop/Uni's_Life/20127397_Ga
---------------------
1   -2  3   -3
0   6   -6  6
0   -3  4   1
0   2   -2  2
---------------------
1   -2  3   -3
0   1.0 -1.0    1.0
0   0.0 0.16666666666666663 0.6666666666666666
0   0.0 0.0 0.0
---------------------
1   -2  3   -3
0   1.0 -1.0    1.0
0   0.0 1.0 4.000000000000001
0   0.0 0.0 0.0
Hệ phương tình vô số nghiệm

Process finished with exit code 0
```

- Phương trình có nghiệm

```
/Users/yennguyen/Desktop/Uni's_Life/201
--------------------
1   2   -1  -1
0   -2  3   3
0   -1  1   2
--------------------
1   2   -1  -1
0   1.0 -1.5    -1.5
0   0.0 0.25    -0.25
--------------------
1   2   -1  -1
0   1.0 -1.5    -1.5
0   0.0 1.0 -1.0
--------------------
Hệ phương trình có nghiệm
[4.0, -3.0, -1.0]

Process finished with exit code 0
```

5. Tài liệu tham khảo

- https://gist.github.com/j9ac9k/6b5cd12aa9d2e5aa861f942b786293b4
- - https://vi.wikipedia.org/wiki/Phép_khử_Gauss

-HẾT-