Báo cáo

Họ và tên: Nguyễn Mạnh Hùng.

MSSV: 20127030

1/ Ý tưởng thực hiện

Chương trình của em sẽ cho người dùng nhập số ẩn có trong hệ phương tình tuyến tính cần giải quyết (n ẩn) và sau đó sẽ nhập tuần tự các hệ số $a_1,a_2,...,a_n$ và b để hình thành ra hệ phương trình với số phương trình bằng số ẩn cụ thể như sau:

$$\begin{cases} a11x1 + a12x2 + \cdots + a1nxn = b1 \\ a21x1 + a22x2 + \cdots + a2nxn = b2 \\ \qquad\qquad\qquad ... \\ an1x1 + an2x2 + \cdots + annxn = bn \end{cases}$$

Từ đó, ta biến đổi và có được ma trận mở rộng A gồm

$$A = \begin{bmatrix} a11\ a12\ ...\ a1n \mid b1 \\ a21\ a22\ ...\ a2n \mid b2 \\ \qquad\qquad ... \\ an1\ an2\ ...\ ann \mid bn \end{bmatrix}$$

và ma trận hệ số B gồm

$$B = \begin{bmatrix} a11\ a12\ ...\ a1n \\ a21\ a22\ ...\ a2n \\ \qquad ... \\ an1\ an2\ ...\ ann \end{bmatrix}$$

Sau đó ta tiến hành giải bài toán thông qua các hàm đã được lập trình.

2/ Mô tả chương trình.

Đầu tiên, cho người dùng nhập số ẩn có trong hệ phương trình và lưu vào biến n. Sau đó, ta tạo một ma trận mở rộng A với kích thước n*(n+1) và ma trận hệ số

Coef với kích thước n*n. Kế đến ta cho người dùng nhập lần lượt các hệ số và biến vào để hoành thành các phần tử bên trong ma trận. Sau khi nhập xong ta sẽ tính toán và kiểm tra các hạng của 2 ma trận A và Coef.

Nếu rank(A) = rank(Coef) = n thì hệ có nghiệm duy nhất.

Nếu rank(A) = rank(Coef) < n thì hệ có vô số nghiệm.

Nếu rank(A) > rank(Coef) thì hệ vô nghiệm.

Đối với trường hợp có nghiệm thì ta sẽ dùng thuật toán phép khử Gauss để đưa về ma trận bậc thang.

- ***Hàm Gauss_Elimination(a):***

Tham số truyền vào là ma trận mở rộng a.

Quy trình các bước thực hiện của hàm:

Bước 1: Kiểm tra ma trận có phải ma trận dòng hay cột? Nếu phải trả về Â

Bước 2: *Kiểm tra toàn bộ phần tử thuộc cột đầu tiên có bằng 0?* Nếu **có** thì tìm ma trận bậc thang cho cột 2 bằng câu lệnh đệ quy nhằm chia ma trận lớn thành nhiều ma trận con để xử lý và biến đổi (b = Gauss_Elimination(a[:,1:]). Sau đó chèn cột 1 vào cột cuối cùng (return np.hstack([a[:,:1], b]).

Bước 3: Trường hợp có các trường hợp các phần tử khác 0 trong quá trình đệ quy không nằm ở dòng đầu tiên (i > 0) -> đổi dòng đó với dòng đầu tiên.

Bước 4: Lấy toàn bộ dòng đầu tiên chia cho phần tử đầu tiên của dòng (a[0] = a[0] / a[0][0]), kế đến biến đổi các dòng còn lại (*từ dòng 2 và cột 2 trở đi*) theo dòng đầu tiên sao cho thành về dạng bậc thang thông qua câu lệnh đệ quy. (b = Gauss_Elimination(a[1:,1:])

Bước 5: Ta thêm lại dòng đầu tiên và toàn bộ cột đầu tiên = 0 (nếu trường hợp điều kiện ở bước 2 xảy ra cho toàn bộ cột đầu = 0) vào vị trí cũ và thực hiện lặp lại các bước theo tiến trình đệ quy trước đó (gồm các ma trận nhỏ trước đó ở *Bước 2* và *Bước 4* thành ma trận hoàn chỉnh) thông qua câu lệnh return np.vstack([a[:1], np.hstack([a[1:,:1], b])]

Sau khi biến đổi thành ma trận bậc thang thành công ta bắt đầu tìm nghiệm của phương trình thông qua thuật toán giải ngược.

- ***Hàm Back_Substitution(x,a,n)***
  Tham số truyền vào là ma trận mở rộng a đã về dạng bậc thang, tập x rỗng để lưu các nghiệm phương trình và số ẩn của hệ phương trình n.

  Quy trình các bước thực hiện của hàm:

  Bước 1: Kiểm tra hệ số cuối cùng của x_n trong ma trận bậc thang có rơi vô trường hợp vô số nghiệm? ($a_n = 0$?) Nếu xảy ra thì ta gán cho nghiệm cuối cùng trong tập nghiệm = 0 (thay vì chọn alpha hay beta và để alpha và beta thuộc tập số thực thì nay ta gán cho giá trị cụ thể). Nếu không thì ta tiến hành chia bình thường.

  Bước 2: Tìm các nghiệm còn lại thông qua 2 vòng lặp for chạy từ x_{n-1} về ngược lại x_1, sau đó ta gán $x_{n-1} = x_n$ để khởi tạo giá trị và ở vòng lặp thứ 2 ta sẽ thế ngược giá trị và tìm các giá trị của x_{n-1}. Trường hợp, x_{n-1} rơi vào vô số nghiệm (a[i][i] = 0 ?) thì ta cứ tiếp tục gán các giá trị đó bằng 0 và thêm vào tập nghiệm x.

Sau cùng ta sẽ thu được kết quả mong muốn và kết thúc chương trình.

Riêng với trường hợp vô nghiệm thì ta sẽ thông báo vô nghiệm và kết thúc chương trình.

Tài liệu tham khảo:

- Sách Lập trình Python căn bản, Chương Matrix mà thầy cô đã cung cấp trên moodle.
- Advanced Indexing Numpy: https://www.tutorialspoint.com/numpy/numpy_advanced_indexing.htm
- Gauss Elimination: https://www.pythonpool.com/gaussian-elimination-python/