

LAB 1: Gauss Elimination and Back Substitution Algorithm with Python

Teacher: Trần Hà Sơn

Subject: Applied Mathematics and Statistics

I. Introduction

1. Student information

Full name	Student ID	Class
Nguyễn Hữu Thuận	21120566	21CTT5

2. Problem statement

Giải hệ phương trình tuyến tính bằng phương pháp Gauss Elimination và Back Substitution.

3. Problem description

Cho hệ phương trình tuyến tính $Ax = b$ với A là ma trận vuông, b là vector cột. Hãy giải hệ phương trình tuyến tính trên bằng phương pháp Gauss Elimination và Back Substitution.

II. Gauss Elimination implementation

1. Định nghĩa hàm `gauss_elimination`

```
def gauss_elimination(A):
```

Hàm tìm ma trận bậc thang của ma trận mở rộng của hệ phương trình tuyến tính, hàm nhận tham số đầu vào là ma trận A và trả về ma trận bậc thang của A .

2. Vòng lặp `for` sẽ duyệt qua các phần tử của ma trận A

```
for i in range(len(A)):
```

Duyệt theo chiều dọc, bắt đầu từ hàng đầu tiên (hàng 0) đến hàng cuối cùng.

3. Tìm hàng có giá trị pivot (hàng có giá trị đầu tiên khác 0 trong cột i).

```
pivot_row = i
while pivot_row < len(A) and A[pivot_row][i] == 0:
    pivot_row += 1
```

Sử dụng một vòng lặp `while` để duyệt qua các hàng kế tiếp của A, nếu phần tử của hàng đang xét tại cột i bằng 0 thì ta sẽ tiếp tục duyệt sang hàng kế tiếp (tăng biến `pivot_row` lên 1). Trong trường hợp không tìm thấy hàng nào có giá trị `pivot`, chương trình sẽ bỏ qua lệnh bên dưới và bắt đầu vòng lặp kế tiếp.

4. Kiểm tra xem có tìm thấy hàng có giá trị pivot hay không.

```
if pivot_row == len(A):
    continue
```

Nếu không tìm thấy, chương trình sẽ bỏ qua hàng đang xét và tiếp tục vòng lặp kế tiếp.

5. Hoán đổi vị trí giữa hàng có giá trị pivot với hàng đang xét.

```
A[i], A[pivot_row] = A[pivot_row], A[i]
```

Bằng cách này, ta sẽ đẩy các giá trị khác 0 trong cột i xuống dưới đáy ma trận A.

6. Thực hiện phép loại bỏ (elimination) các giá trị khác 0 trong cột i.

```
for j in range(i+1, len(A)):
    factor = A[j][i] / A[i][i]
    for k in range(i+1, len(A[0])):
        A[j][k] -= factor * A[i][k]
    A[j][i] = 0
```

a. Vòng lặp `for` thứ nhất duyệt qua các hàng A bắt đầu từ hàng sau của `pivot_row` đến hàng cuối cùng của ma trận.

Trong vòng lặp này, đầu tiên tính `factor` là thương của phần tử đang xét (`A[j][i]`) và phần tử pivot (`A[i][i]`).

b. Vòng lặp `for` thứ hai duyệt qua các phần tử trong hàng của A bắt đầu từ cột sau cột chứa pivot (cột i+1) đến cột cuối cùng của hàng (cột `len(A)`).

Trong vòng lặp này, mỗi phần tử `A[j][k]` được cập nhật giá trị bằng cách lấy nó trừ đi tích của `factor` và phần tử `A[i][k]`. Cuối cùng,

phần tử ở cột **pivot** của hàng đang xét ($A[j][i]$) được gán giá trị bằng 0 để loại bỏ nó.

III. Back Substitution implementation

1. Định nghĩa hàm `back_substitution(A)`:

```
def back_substitution(A):
```

Hàm giải hệ phương trình tuyến tính bằng phương pháp giải ngược.
Hàm nhận tham số đầu vào là ma trận A và trả về nghiệm của hệ phương trình tuyến tính.

2. Khởi tạo list x và biến n

```
n = len(A)
x = [0] * n
```

n là biến lưu số hàng của ma trận A . x là list chứa các nghiệm của A , được khởi tạo bằng 0 với độ dài bằng n .

3. Kiểm tra nếu A vô nghiệm hoặc có vô số nghiệm

```
if A[n-1][n-1] == 0 and A[n-1][n] != 0:
    return "The system has no solution"
elif A[n-1][n-1] == 0 and A[n-1][n] == 0:
    return "The system has infinite solutions"
```

Nếu $A[n-1][n-1] == 0$ và $A[n-1][n] != 0$, thì A không có nghiệm.
Nếu $A[n-1][n-1] == 0$ và $A[n-1][n] == 0$, thì A có vô số nghiệm.

4. Tính nghiệm sau cùng của A

```
x[n-1] = A[n-1][n] / A[n-1][n-1]
```

Tính giá trị của nghiệm sau cùng của $x[n-1]$ bằng cách chia giá trị của **phần tử cuối trong dòng cuối** cho giá trị của **phần tử cuối trong đường chéo chính**.

5. Tính các nghiệm còn lại của A

```
for i in range(n-2, -1, -1):
    sum = 0
    for j in range(i+1, n):
        sum += A[i][j] * x[j]
    x[i] = (A[i][n] - sum) / A[i][i]

return x
```

Sử dụng vòng lặp `for` để duyệt các phần tử trong cột cuối cùng của ma trận tam giác để tìm nghiệm của mỗi biến. `sum` được tính bằng cách duyệt từ phần tử **đứng ngay sau biến cần giải** đến phần tử **sau cùng của hàng đó**. Khi tính một biến, ta đã giải được các biến phía sau nó nên có thể sử dụng nó để tính nghiệm của biến hiện tại. Cuối cùng, trả về mảng `x` chứa các nghiệm của A.

IV. Technical details

- Sử dụng thư viện `numpy` để đọc file `input.txt`
- Hàm `print_matrix()` được sử dụng để in ma trận ra console theo định dạng đẹp

V. Usage

1. Nhập ma trận A vào file `input.txt` theo định dạng sau:

```
2 1 -1 8
-3 -1 2 -11
-2 1 2 -3
```

2. Chạy chương trình `21120566.py` để giải hệ phương trình tuyến tính.
3. Kết quả sẽ được xuất ra console như sau:

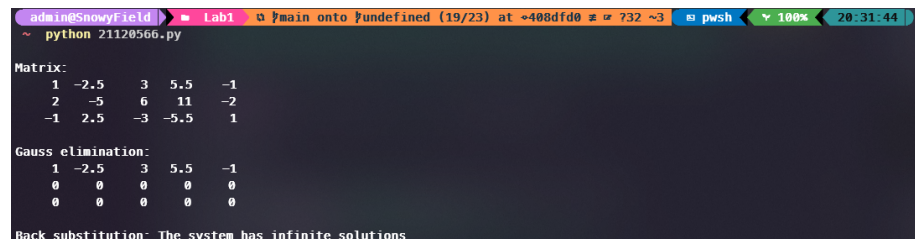
- Nếu A vô nghiệm: The system has no solution
- Nếu A có vô số nghiệm: The system has infinite solutions
- Nếu A có nghiệm: `[2, 3, -1]` (với $x_1 = 2$, $x_2 = 3$ và $x_3 = -1$)

VI. Example

`input.txt`

```
1 -2.5 3 5.5 -1
2 -5 6 11 -2
-1 2.5 -3 -5.5 1
```

`console`



```
admin@Snowfield ~$ python 21120566.py
Matrix:
1 -2.5 3 5.5 -1
2 -5 6 11 -2
-1 2.5 -3 -5.5 1
Gauss elimination:
1 -2.5 3 5.5 -1
0 0 0 0 0
0 0 0 0 0
Back substitution: The system has infinite solutions
```

VII. References

- Systems of Linear Equations