VNUHCM – University of Science
Knowledge Engineering Department

Subject: Applied Mathematics and Statistics

REPORT: Project Gauss

Information Student:

1. Nguyễn Thái Hiệp ID: 20127496

Instructors:

Nguyễn Trọng Hiến

Nguyễn Văn Quang Huy

Nguyễn Đình Thúc

Võ Nam Thục Đoan

TP.HCM, May 27 2022

1. Mức độ hoàn thiện

=> 100%

- **Tất cả các vấn đề đã được giải quyết**

2. Ý tưởng thực hiện

Trong đại số tuyến tính, phép khử Gauss là một thuật toán có thể được sử dụng để **tìm nghiệm của một hệ phương trình tuyến tính** (tìm hạng của một ma trận, để tính ma trận nghịch đảo của một ma trận vuông khả nghịch). Phép khử Gauss được đặt theo tên của nhà toán học Đức là Carl Friedrich Gauss.

Các thao tác cơ bản về hàng được sử dụng trong suốt thuật toán. Thuật toán có 2 phần, phần thứ nhất là đưa ma trận bậc thang bằng hàm **Gauss_elimination** và phần thứ hai là trả về nghiệm của phương trình (hoặc vô số nghiệm, vô nghiệm) bằng hàm **back_substitution.**

3. Mô tả các hàm

3.1. Gauss Elimination

```python
def Gauss_elimination(matrix):
    """
    returns a Echelon matrix
    """
    # to do not change the parameter
    result = []
    for line in matrix:
        array = []
        for i in line:
            array.append(i)
        result.append(array)

    row = len(result)
    col = len(result[0])

    # fix col and row of matrix
    if col > row + 1:
        n_add = col - (row + 1)
        for i in range(n_add):
            result.append([0 for item in range(col)])
        row = len(result)
        col = len(result[0])

    fix_matrix = []
    for i in range(row):
        fix_matrix.append([0 for item in range(col)])

    for i in range(0, row, 1):
        # find the pivot index
        pivot = 0
        isDone = True
        for j in range(i, col, 1):
            isVector0 = True
            for k in range(i, row, 1):
                if result[k][j] != 0:
                    isDone = False
                    isVector0 = False
                    break
            if not isVector0:
                pivot = j
                break
        if isDone:
            for i in range(row):
                for j in range(row):
                    if result[i][j] == 1:
                        fix_matrix[j] = result[i].copy()
                        break
            if col == row + 1:
                return fix_matrix
            else:
                return fix_matrix[:row - 1]

        # if pivot = 0 then swap line
        if result[i][pivot] == 0:
            for j in range(i+1, row, 1):
                if result[j][pivot] != 0:
                    for k in range(pivot, col, 1):
                        result[i][k], result[j][k] = result[j][k], result[i][k]
                    break

        # if pivot is not leader
        if result[i][pivot] != 1:
            result[i] = [item / result[i][pivot] for item in result[i]]

        for j in range(i + 1, row, 1):
            temp = result[j][pivot]
            if temp == 0: continue
            for k in range(pivot, col, 1):
                result[j][k] -= temp * result[i][k]

    return result
```

Hàm Gauss_elimination

- Input: là một ma trận mở rộng của hệ phương trình
- Output: là một ma trận bậc thang

```
16      # fix col and row of matrix
17      if col > row + 1:
18          n_add = col - (row + 1)
19          for i in range(n_add):
20              result.append([0 for item in range(col)])
```

Ở hình ảnh bên trên, từ dòng 16 tới dòng 20 là em kiểm tra giá trị đầu vào đã chuẩn chưa và chỉnh lại.

VD:

$$\begin{pmatrix} 1 & -1 & 1 & -3 & 0 \\ 2 & -1 & 4 & -2 & 0 \end{pmatrix}$$

Thì em sẽ thêm 2 dòng nữa vào ma trận.

Sau đó, em duyệt và kiểm tra từng dòng.

(Hình ảnh chính 1)

```
29          # find the pivot index
30          pivot = 0
31          isDone = True
32          for j in range(i, col, 1):
33              isVector0 = True
34              for k in range(i, row, 1):
35                  if result[k][j] != 0:
36                      isDone = False
37                      isVector0 = False
38                      break
39              if not isVector0:
40                  pivot = j
41                  break
42          if isDone:
43              for i in range(row):
44                  for j in range(row):
45                      if result[i][j] == 1:
46                          fix_matrix[j] = result[i].copy()
47                          break
48              if col == row + 1:
49                  return fix_matrix
50              else:
51                  return fix_matrix[:row - 1]
```

Ở hình ảnh trên,

Từ dòng 30 tới dòng 41 là tìm vị trí pivot của dòng thứ i và nếu không tìm ra nghĩa là đã hoàn tất ma trận bậc thang. Sau đó từ dòng 42 tới dòng 51 em sẽ sắp xếp lại ma trận và trả về

VD 1:

$$\begin{pmatrix} 0 & 2 & -1 & -1 \\ 0 & 2 & 1 & 1 \\ 0 & 5 & -2 & -1 \end{pmatrix}$$

Ở ma trận như trên, em sẽ kiểm tra cột đầu tiên có là vector 0 hay không? Nếu đúng thì tiếp tục kiểm tra cột tiếp theo. Nếu sai, thì pivot ở vị trí cột đó.

VD 3:

$$\begin{pmatrix} 1 & -1 & 1 & -3 & 0 \\ 0 & 1 & 2 & 4 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Ở đoạn code trên có một biến isDone đánh dấu sự hoàn tất nhưng chưa duyệt hết dòng của ma trận.

Nghĩa là ở đây em kiểm tra được rằng dòng 3 và 4 là vector 0 nên isDone sẽ được gán lại cho True. Và những dòng code phía sau là để chỉnh sửa lại dòng cho hợp lí.

(Hình ảnh chính 2)

```
50          # if pivot = 0 then swap line
51          if result[i][pivot] == 0:
52              for j in range(i+1, row, 1):
53                  if result[j][pivot] != 0:
54                      for k in range(pivot, col, 1):
55                          result[i][k], result[j][k] = result[j][k], result[i][k]
56                      break
57
58          # if pivot is not leader
59          if result[i][pivot] != 1:
60              result[i] = [item / result[i][pivot] for item in result[i]]
61
62          for j in range(i + 1, row, 1):
63              temp = result[j][pivot]
64              if temp == 0: continue
65              for k in range(pivot, col, 1):
66                  result[j][k] -= temp * result[i][k]
```

Ở hình ảnh trên, từ dòng 51 tới dòng 56 là nếu pivot ở dòng i bằng với giá trị 0 thì em sẽ tìm từ dòng i + 1 trở đi nếu khác 0 thì sẽ hoán đổi vị trí dòng.

VD 1:

$$\begin{pmatrix} 0 & 0 & -2 & 0 & 7 \\ 6 & -3 & 0 & -5 & 3 \\ 8 & -4 & 28 & -44 & 11 \\ -8 & 4 & -4 & 12 & -5 \end{pmatrix}$$

Ở ma trận như trên, em sẽ hoán đổi dòng 1 với dòng 2. Từ dòng 58 tới dòng 60 là nếu pivot chưa là leader (nếu vị trí pivot = 1) thì em duyệt dòng đó và chia cho giá trị pivot. Từ dòng 62 tới dòng 66 là duyệt từ dòng ở dưới và chuyển các giá trị bên dưới cột pivot về 0.

3.2. Back substitution

```
1  def back_substitution(echelon_matrix):
2      """
3      returns a solution of the equation and a message
4      """
5      row = len(echelon_matrix)
6      col = len(echelon_matrix[0])
7
8      x = ["None" for i in range(col - 1)]
9
10     message = ""
11     for i in range(row - 1, -1, -1):
12         left_side_sum = 0
13         for k in range(i + 1, col - 1, 1):
14             left_side_sum += echelon_matrix[i][k] * float(x[k])
15         right_side = echelon_matrix[i][col - 1] - left_side_sum
16
17         if echelon_matrix[i][i] == 0:
18             if right_side == 0:
19                 message = "infinitely many roots"
20                 y = ["None" for i in range(col - 1)]
21                 for j in range(len(x)):
22                     if x[j] != 0:
23                         y[j] = str(x[j])
24                 cnt = 0
25                 y[i] = "a{}".format(cnt)
26                 cnt = cnt + 1
27                 for j in range(i - 1, -1, -1):
28                     check = False
29                     for k in range(j, col, 1):
30                         if echelon_matrix[j][k] != 0:
31                             check = True
32                     if check == False:
33                         y[j] = "a{}".format(cnt)
34                         cnt = cnt + 1
35                     else:
36                         left_sum = ""
37                         for k in range(j + 1, col - 1, 1):
38                             left_sum += "+ {} * {}".format(echelon_matrix[j][k], y[k])
39                         right_side = "({} - ({}))".format(echelon_matrix[j][col - 1], left_sum)
40                         y[j] = right_side
41                 return y, message
42             else:
43                 message = "the equation has no solution"
44                 return x, message
45         else:
46             x[i] = str(right_side)
47     message = "the equation has a solution"
48     return x, message
```

Hàm Back substitution

- Input: là một ma trận bậc thang của hệ phương trình
- Output: là nghiệm của phương trình (hoặc vô nghiệm, hoặc vô số nghiệm)

Ở hình ảnh trên,

Từ dòng 12 tới dòng 15, em sẽ tính giá trị **PHẦN BÊN PHẢI** của phương trình. Sau đó, tìm nghiệm theo như sau:

Ta có : $\mathbf{Ax = B}$

Nếu B khác 0 và A bằng 0 thì phương trình vô nghiệm.

Nếu B bằng 0 và A bằng 0 thì phương trình vô số nghiệm.

Ngược lại là phương trình có nghiệm duy nhất.

VD:

$$\begin{pmatrix} 1 & 2 & -1 & -1 \\ 0 & 1 & -1.5 & -1.5 \\ 0 & 0 & 1 & -1.0 \end{pmatrix}$$

Em sẽ duyệt từ dòng 3 lên trên. Giá trị vế bên phải của phương trình là -1.
Lúc này, A = 1 và B = -1 nên phương trình có nghiệm là x3 = -1.
Tương tự như vậy cho mọi trường hợp còn lại.

Hết.