TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN ĐHQG-HCM

KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN 2 – MA TRẬN NGHỊCH ĐẢO

Môn Học: Toán ứng dụng & thống kê

Giảng viên:

Nguyễn Trọng Hiến

Nguyễn Văn Quang Huy

Nguyễn Đình Thúc

Võ Nam Thục Đoan

Sinh viên thực hiện:

Nguyễn Tấn Phát 20127588

1. Nội dung đồ án:

Sinh viên viết hàm inverse(A), trong đó

- Input: A là ma trận vuông.
- Output: Ma trận nghịch đảo của ma trận A ban đầu nếu có, trường hợp không có ma trận nghịch đảo sẽ hiện thông báo " Ma trận không khả nghịch ". Lưu ý sinh viên phải sử dụng thuật toán đã được hướng dẫn trong phần bài tập để tìm nghịch đảo(dùng ma trận (A|I)). Sinh viên không được dùng các hàm có sẵn của các thư viện để tìm định thức hoặc ma trận nghịch đảo.

2. Môi trường làm việc:

- Ngôn ngữ lập trình: Python
- Text Editer: Visual Studio Code
- Thư viện hỗ trợ: numpy

3. Cơ sở và ý tưởng:

a) Xây dựng class và các đối tượng

- Tên class: **MyMatrix** (Là một lớp về ma trận)
- **self.root**: ma trận input và có giá trị không thay đổi (dạng numpy.array)
- **self.matrix**: ma trận input giúp đỡ trong quá trình biến đổi (dạng numpy.array)
- **self.inv**: ma trận đơn vị, biến đổi về ma trận nghịch đảo của input (dạng numpy.array)
- **self.numRow**: số hàng của ma trận (dạng int)
- **self.numCol**: số cột của ma trận (dạng int)

b) Ý tưởng giải quyết bài toán

Về lý thuyết:

$$[A|\, I_n] \rightarrow [I_n|\, B]$$

- Gộp ma trận A và ma trận đơn vị I thành ma trận [A|I]
- Biến đổi ma trận bằng các phép biến đổi sơ cấp trên dòng nhằm đưa về ma trận [I|B]
- Nếu đưa về [I|B] thành công thì B chính là ma trận nghịch đảo của A

Về thực hiện (code):

- Thay vì phải gộp 2 ma trận lại thì ta chỉ cần biến đổi ma trận A thành ma trận đơn vị một cách bình thường. Với điều khiển, bất kì sự thay đổi ở ma trận A thì cũng xảy ra ở ma trận đơn vị I.
- Nếu A không phải ma trận vuông => thông báo không có nghịch đảo và return None.
- Nếu A đã là ma trận đơn vị => Thông báo hoàn thành và return self.inv [I]
- Thực hiện Gauss_ Jordan để đưa A về ma trận đơn vị
- Nếu thành công thì trả về ma trận self.inv (đã được biến đổi cùng với A)

4. Các hàm hỗ trợ:

a. *swapRow(matrix, row1, row2)*

- *Chức năng: Đổi 2 dòng của ma trận cho nhau*
- VD:
  - Input: [[1, 2],[3, 4]], 2, 1
  - Output: [[3, 4], [1, 2]

b. *mulRow(matrix, k: float, row)*

- *Chức năng: Nhân một hàng của ma trận với hệ số khác 0.*
- VD:
  - Input: [[1, 2],[3, 4]], 2, 1
  - Output: [[1, 2], [6, 8]

c. *plusRow(matrix, rowI, k: float, rowJ)*

- Chức năng: hàng I = hàng I + k * hàng J
- VD:
  - Input: [[1, 2],[3, 4]], 0, 1, 1
  - Output: [[4, 6], [6, 8]

5. Giải thích thuật toán inverse():

- Nếu A không phải ma trận vuông => thông báo không có nghịch đảo và return None.
- Nếu A đã là ma trận đơn vị => thông báo hoàn thành và return ma trận đơn vị.
- Dùng vòng for để duyệt từng dòng của ma trận self.matrix với chỉ số i:
    - Nếu có dòng đang xét có giá trị matrix[i][i] == 0:
        - Tìm dòng khác thay thế để matrix[i][i] != 0
        - Nếu không có dòng nào cả thì thông báo không ma trận nghịch đảo và return None.
        - Thay vì kiểm tra matrix[i][i] == 0 ? Thì kiểm tra abs(matrix[i][i]) < 0,0001
        - Vì quá trình máy tính toán thì có thể làm tròn, nên đôi khi chỉ ra xấp xỉ 0 ~ 2e^–16 chứ không chính xác là 0.
    - Chia dòng I của ma trận matrix[i][i] để giá trị của matrix[i][i] = 1. Và làm tương tự với ma trận đơn vị self.inv
        - Tuy nhiên ta sẽ thực hiện với ma trận self.inv trước rồi mới đến self.matrix
        - Vì cả 2 đều chia hệ số của self.matrix[i][i] nên nếu thực hiện với self.matrix trước thì hệ số trên sẽ bị thay dổi.
    - Dòng vòng for để duyệt các dòng của ma trận self.matrix một lần nữa với chỉ số j:
        - Nếu j != i
            - Dòng j = dòng j – (matrix[i][i] * dòng i) đối với cả 2 ma trận self.inv và self.matrix.
            - Khi thực hiện phép biến đổi này, ta đang làm cho các giá trị nào không nằm trên đường chéo (self.matrix) đều về 0 để đưa về ma trận đơn vị.
- Return self.inv

$$\left[\begin{array}{ccc|ccc} 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 2 & 2 & 0 & 1 & 0 \\ 1 & 2 & 3 & 0 & 0 & 1 \end{array}\right] \xrightarrow{R_2 - R_1 \to R_2} \left[\begin{array}{ccc|ccc} 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & -1 & 1 & 0 \\ 1 & 2 & 3 & 0 & 0 & 1 \end{array}\right] \xrightarrow{R_3 - R_1 \to R_3} \left[\begin{array}{ccc|ccc} 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & -1 & 1 & 0 \\ 0 & 1 & 2 & -1 & 0 & 1 \end{array}\right]$$

$$\xrightarrow{R_3 - R_2 \to R_3} \left[\begin{array}{ccc|ccc} 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & -1 & 1 & 0 \\ 0 & 0 & 1 & 0 & -1 & 1 \end{array}\right] \xrightarrow{R_1 - R_2 \to R_1} \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 2 & -1 & 0 \\ 0 & 1 & 1 & -1 & 1 & 0 \\ 0 & 0 & 1 & 0 & -1 & 1 \end{array}\right]$$

$$\xrightarrow{R_2 - R_3 \to R_2} \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 2 & -1 & 0 \\ 0 & 1 & 0 & -1 & 2 & -1 \\ 0 & 0 & 1 & 0 & -1 & 1 \end{array}\right]$$

6. Kết quả chạy thử:

- Input:

```
A = [[1, 2, 1], [3, 7, 3], [2, 3, 4]]
B = [[1, -1, 2], [1, 1, -2], [1, 1, 4]]
C = [[1, 2, 3], [2, 5, 3], [1, 0, 8]]
D = [[-1, 3, -4], [2, 4, 1], [-4, 2, -9]]
I = [[1, 0, 0], [0, 1, 0], [0, 0, 1]]
```

- Main:

```
mtrix = MyMatrix(A)

mtrix.printMatrix()
print(mtrix.inverse())
print("\n A * A^-1 = I")
print(np.dot(A, mtrix.inverse()))
print("\n------------------------\n")

mtrix.clear()
mtrix = MyMatrix(B)

mtrix.printMatrix()
print(mtrix.inverse())
print("\n B * B^-1 = I")
print(np.dot(B, mtrix.inverse()))
print("\n------------------------\n")
```

```
mtrix.clear()
mtrix = MyMatrix(C)

mtrix.printMatrix()
print(mtrix.inverse())
print("\n C * C^-1 = I")
print(np.dot(C, mtrix.inverse()))
print("\n------------------------\n")

mtrix.clear()
mtrix = MyMatrix(D)
mtrix.printMatrix()
print(mtrix.inverse())
print("\n------------------------\n")
```

- Output:

```
[[1. 2. 1.]
 [3. 7. 3.]
 [2. 3. 4.]]
[[ 9.5 -2.5 -0.5]
 [-3.   1.   0. ]
 [-2.5  0.5  0.5]]

 A * A^-1 = I
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]

------------------------

[[ 1. -1.  2.]
 [ 1.  1. -2.]
 [ 1.  1.  4.]]
[[ 0.5         0.5         0.        ]
 [-0.5         0.16666667  0.33333333]
 [ 0.         -0.16666667  0.16666667]]

 B * B^-1 = I
[[1.00000000e+00 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 1.00000000e+00 0.00000000e+00]
 [0.00000000e+00 1.11022302e-16 1.00000000e+00]]
```

```
[[1. 2. 3.]
 [2. 5. 3.]
 [1. 0. 8.]]
[[-40.  16.   9.]
 [ 13.  -5.  -3.]
 [  5.  -2.  -1.]]

 C * C^-1 = I
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]

------------------------

[[-1.  3. -4.]
 [ 2.  4.  1.]
 [-4.  2. -9.]]
Khong co ma tran nghich dao
None

------------------------
```

7. Tài liệu tham khảo:

1. Slide bài giảng môn “Toán ứng dụng & thống kê”
2. Slide bài giảng môn “Đại số tuyến tính”

**********HẾT**********