

Project 1

MSSV: 19127102

Họ tên: Võ Hoàng Gia Bảo

1. Gauss_elimination(A)

```
# Hoán đổi 2 dòng để tính toán
if A[0][0] == 0:
    for i in range(rows):
        if A[i][0] != 0:
            temp = A[0]
            A[0] = A[i]
            A[i] = temp
            break
```

Hoán đổi 2 dòng để thuận tiện tính toán

```
for i in range(rows):
    flag = False
    if A[i][i] == 0:
        # kiểm tra các dòng trên 1 cột trên đường chéo nếu != 0 thì hoán đổi 2 dòng
        for j in range(i + 1, rows):
            if A[j][i] != 0:
                temp = A[j]
                A[j] = A[i]
                A[i] = temp
                flag = True
                break
```

Bắt đầu xét các hệ số trên đường chéo, nếu hệ số là 0 sẽ xảy ra lỗi ZeroDivisionError nên sẽ tiếp tục xét các dòng trên 1 cột trên đường chéo chính, nếu có 1 hệ số khác 0 thì hoán đổi 2 dòng.

Sau khi hoán đổi thì sẽ chuyển flag để xét là đã hoán đổi rồi

```

        break

# Kiểm tra nếu không hoán đổi dc thì kiểm tra các hệ số trên 1 dòng != 0
if flag == False:
    for j in range(i + 1, cols):
        if A[i][j] != 0:
            for k in range(i + 1, rows):
                ratio = - (A[k][j] / A[i][j])
                for l in range(cols):
                    A[k][l] = A[k][l] + ratio * A[i][l]

```

Nếu gặp flag chưa hoán đổi thì tìm hệ số đầu tiên khác 0 của dòng và bắt đầu thực hiện biến đổi sơ cấp trên dòng

```

# Nếu hệ số trên đường chéo != 0 thì thực hiện bình thường
if A[i][i] != 0:
    for j in range(i + 1, rows):
        ratio = - (A[j][i] / A[i][i])
        for k in range(cols):
            A[j][k] = A[j][k] + ratio * A[i][k]

```

Ngược lại với trường hợp đầu tiên nếu hệ số trên đường chéo khác 0 thì thực hiện biến đổi sơ cấp trên dòng

2. back_substitution(A)

```

# Trường hợp vô nghiệm
for i in reversed(range(rows)):
    if A[i][0] == 0:
        for j in range(1, cols):
            if j != cols - 1 and A[i][j] != 0:
                break
            if j == cols - 1 and A[i][j] != 0:
                print("Hệ phương trình vô nghiệm\n")
                return None

```

Sau khi có được ma trận bậc thang từ hàm trên, bắt đầu kiểm tra nếu hệ phương trình vô nghiệm

```
# Kiểm tra hạng
B = [None] * (cols - 1)
rank = np.linalg.matrix_rank(A)
unknownSol = cols - 1 - rank
```

Bắt đầu kiểm tra hạng của ma trận, ở đây dùng hàm của numpy để lấy hạng của ma trận, sau đó kiểm tra số nghiệm ẩn

* Đối với trường hợp nghiệm duy nhất (nghiệm ẩn = 0)

```
# Trường hợp có nghiệm duy nhất
else:
    for i in reversed(range(len(B))):
        for j in reversed(range(cols)):
            if i == j:
                B[i] /= A[i][i]
                break
            if B[i] == None:
                B[i] = A[i][j]
            else:
                B[i] -= A[i][j] * B[j]
    np.round(B, 3)
    print(B)
    return B
```

Thực hiện tính lùi từ dòng cuối tính lên, từ cột phải qua trái, kết quả lưu lùi vào 1 array có size là cột - 1, mỗi lần như thế sẽ lấy lùi array kết quả để tính, và khi chạm được hệ số trên đường chéo chính thì chia cho hệ số đó.

Kết quả trong array sẽ được làm tròn 3 chữ số thập phân trước khi được in ra và trả về

* Trường hợp vô số nghiệm (nghiệm ẩn > 0)

```
# Trường hợp vô số nghiệm
if unknownSol > 0:
    alphabet_list = list(string.ascii_uppercase)
    alphabet_count = 0

    # Chuyển các nghiệm ẩn -> các chữ cái A, B, C, ...
    temp = unknownSol
    for i in range(rank):
        if A[i][i] == 0:
            B[i] = alphabet_list[alphabet_count]
            temp -= 1
            alphabet_count += 1
    if temp > 0:
        for i in reversed(range(len(B))):
            if temp == 0:
                break
            B[i] = alphabet_list[alphabet_count]
            temp -= 1
            alphabet_count += 1
```

Tạo 1 list với các chữ cái A,B,C... và dùng để lưu vào array kết quả vị trí của những biến ẩn đó

```

# Thực hiện phép tính
reverse_count = len(B) - 1
for i in reversed(range(rank)):
    flag = False
    for j in reversed(range(cols)):
        if flag == False:
            while B[reverse_count] != None:
                if reverse_count == 0:
                    print(B)
                    return B
                reverse_count -= 1
            flag = True

        if j == reverse_count and B[reverse_count] != None:
            break

        if B[reverse_count] == None:
            B[reverse_count] = str(round(A[i][j] / A[i][reverse_count], 3))

        else:
            B[reverse_count] += " + " + str(round(-A[i][j] / A[i][reverse_count], 3)) + " * (" + str(B[j]) + ")"

print(B)
return B

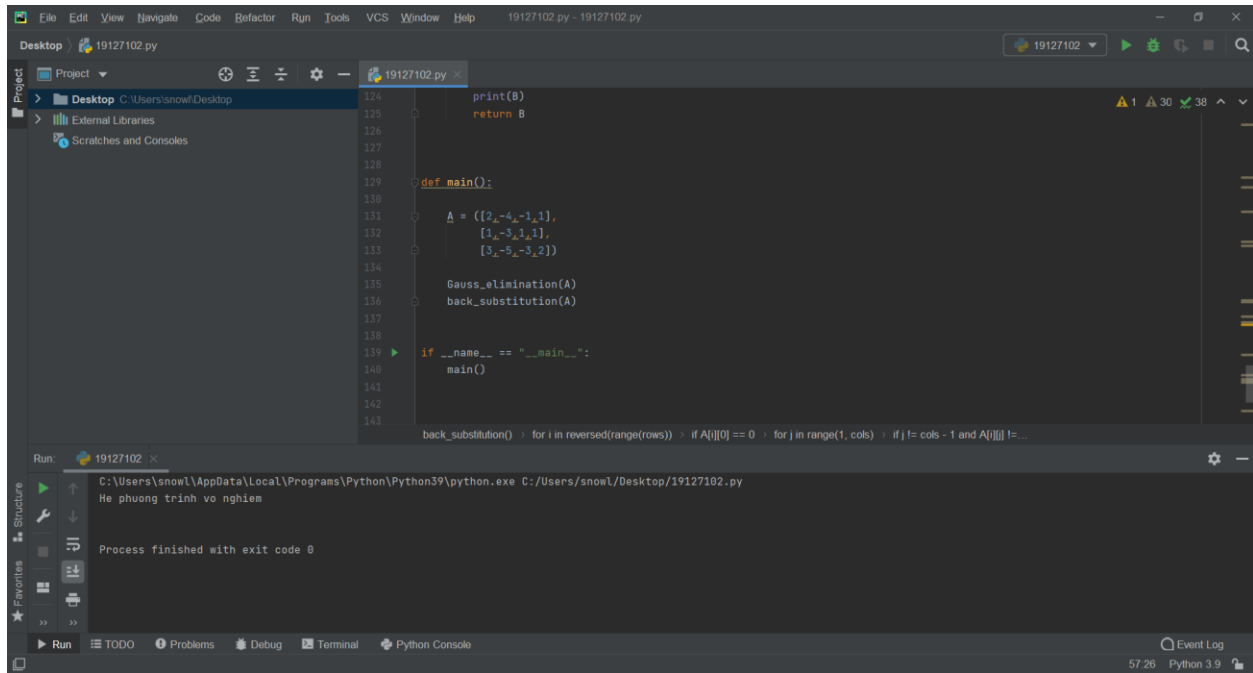
```

Thực hiện tính các nghiệm còn lại dựa trên nghiệm ẩn như trường hợp có nghiệm duy nhất.

Flag để báo rằng khi lấy kết quả lùi từ array kết quả thì nếu tại index đó không rỗng thì bật flag để không lùi nữa vì nếu để vòng loop for sẽ dễ bị kẹt code.

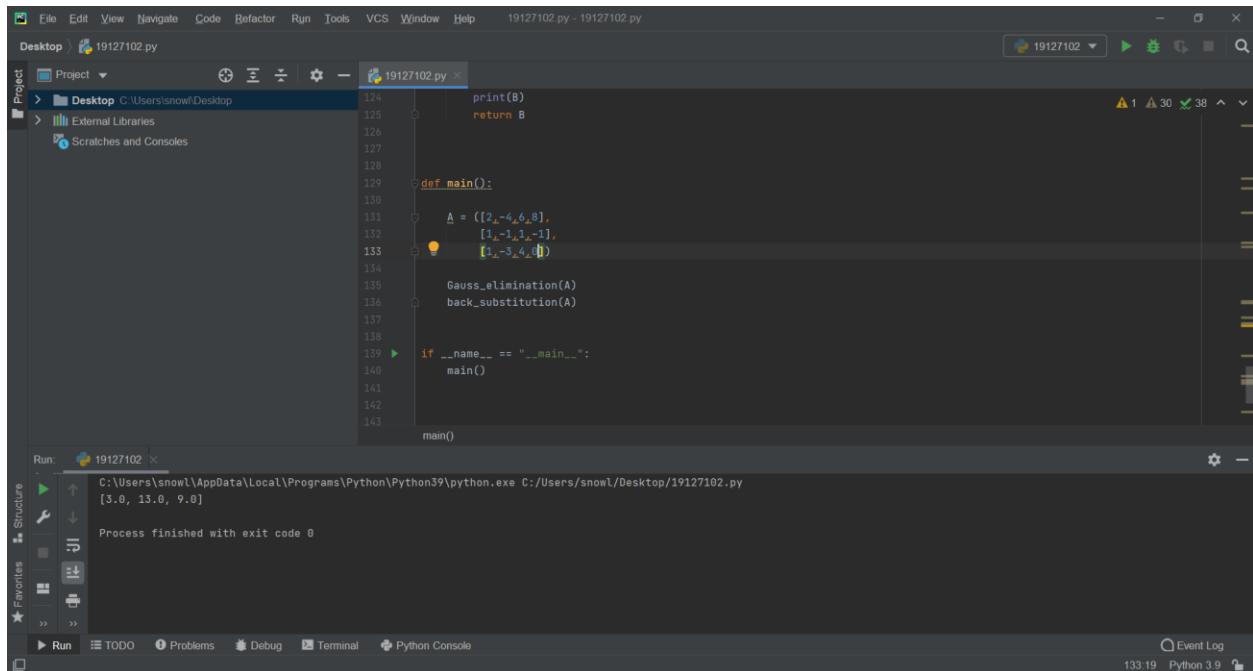
3. Demo

a. Trường hợp vô nghiệm



```
File Edit View Navigate Code Refactor Run Tools VCS Window Help 19127102.py - 19127102.py
Desktop 19127102.py
Project Desktop C:\Users\snowl\Desktop
External Libraries
Scratches and Consoles
19127102.py
124 print(B)
125 return B
126
127
128
129 def main():
130
131     A = ([2,-4,-1,1],
132          [1,-3,1,1],
133          [3,-5,-3,2])
134
135     Gauss_elimination(A)
136     back_substitution(A)
137
138
139 if __name__ == "__main__":
140     main()
141
142
143
back_substitution() for i in reversed(range(rows)): for j in range(1, cols): if j != cols - 1 and A[i][j] != ...
Run: 19127102
C:\Users\snowl\AppData\Local\Programs\Python\Python39\python.exe C:/Users/snowl/Desktop/19127102.py
He phương trình vô nghiệm
Process finished with exit code 0
Run TODO Problems Debug Terminal Python Console
Event Log
57:26 Python 3.9
```

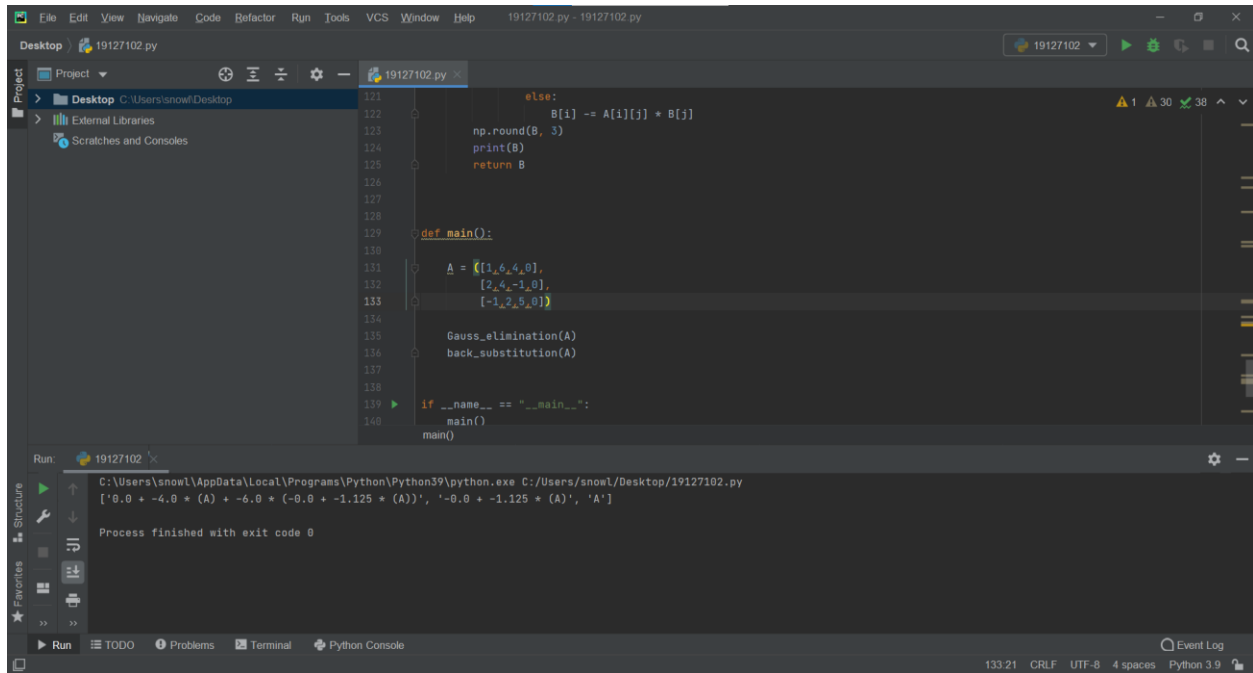
b. Trường hợp có nghiệm duy nhất



```
File Edit View Navigate Code Refactor Run Tools VCS Window Help 19127102.py - 19127102.py
Desktop 19127102.py
Project Desktop C:\Users\snowl\Desktop
External Libraries
Scratches and Consoles
19127102.py
124 print(B)
125 return B
126
127
128
129 def main():
130
131     A = ([2,-4,6,8],
132          [1,-1,1,-1],
133          [1,-3,4,6])
134
135     Gauss_elimination(A)
136     back_substitution(A)
137
138
139 if __name__ == "__main__":
140     main()
141
142
143
main()
Run: 19127102
C:\Users\snowl\AppData\Local\Programs\Python\Python39\python.exe C:/Users/snowl/Desktop/19127102.py
[3.0, 13.0, 9.0]
Process finished with exit code 0
Run TODO Problems Debug Terminal Python Console
Event Log
133:19 Python 3.9
```

c. Trường hợp vô số nghiệm

* 1 nghiệm ẩn



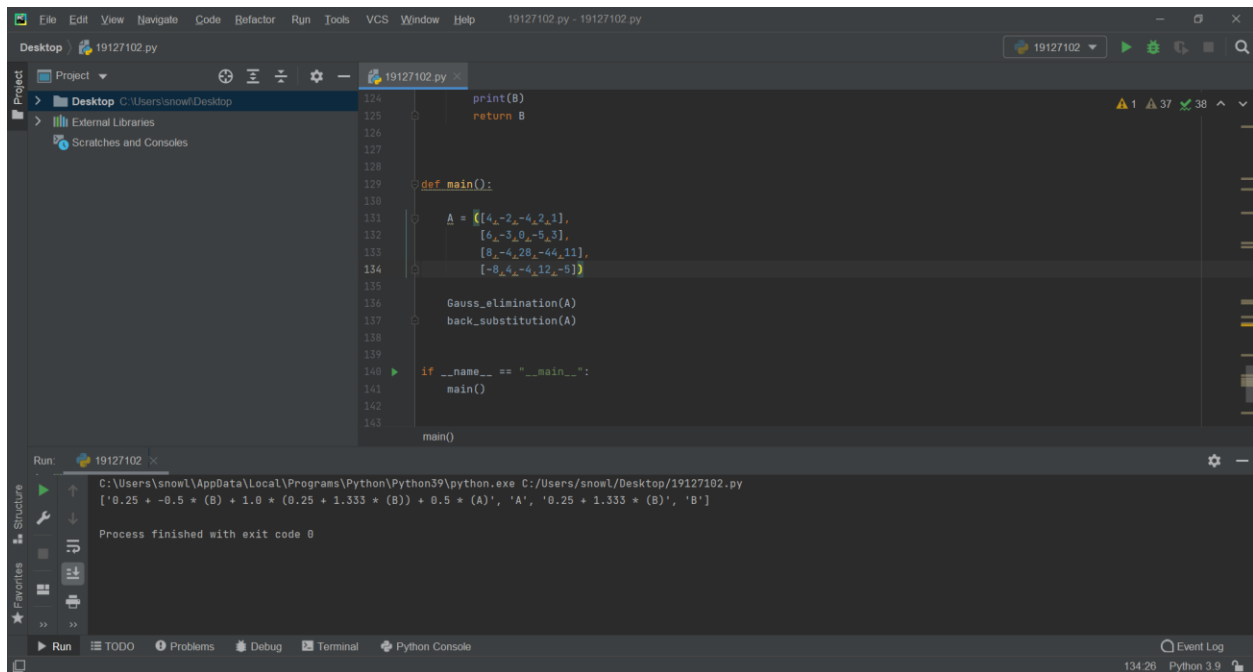
```
121         else:
122             B[1] -= A[1][j] * B[j]
123         np.round(B, 3)
124         print(B)
125         return B
126
127
128
129 def main():
130
131     A = [[1, 4, 4, 0],
132          [2, 4, -1, 0],
133          [-1, 2, 5, 0]]
134
135     Gauss_elimination(A)
136     back_substitution(A)
137
138
139 if __name__ == "__main__":
140     main()
```

Run: 19127102

C:\Users\snowl\AppData\Local\Programs\Python\Python39\python.exe C:/Users/snowl/Desktop/19127102.py
['0.0 + -4.0 * (A) + -6.0 * (-0.0 + -1.125 * (A))', '-0.0 + -1.125 * (A)', 'A']

Process finished with exit code 0

* 2 nghiệm ẩn



```
124         print(B)
125         return B
126
127
128
129 def main():
130
131     A = [[4, -2, -4, 2, 1],
132          [6, -3, 0, -5, 3],
133          [8, -4, 28, -44, 11],
134          [-8, 4, -4, 12, -5]]
135
136     Gauss_elimination(A)
137     back_substitution(A)
138
139
140 if __name__ == "__main__":
141     main()
142
143     main()
```

Run: 19127102

C:\Users\snowl\AppData\Local\Programs\Python\Python39\python.exe C:/Users/snowl/Desktop/19127102.py
['0.25 + -0.5 * (B) + 1.0 * (0.25 + 1.333 * (B)) + 0.5 * (A)', 'A', '0.25 + 1.333 * (B)', 'B']

Process finished with exit code 0

Chú thích: Code không áp dụng các phép toán để đưa kết quả về dạng ngắn gọn do các kết quả được lưu dưới dạng chuỗi kí tự.