Jacobi method

In [1]:

```
coefficient_matrix = []
```

In [6]:

```
n = int(input("Number of unknows : "))
```

In [2]:

```
# coefficient_matrix.append([1,2,3])

for i in range(n):
    coefficient_matrix.append(list(eval(input(f"Enter the coefficients of x,y,z for eq {i} : "))))
```

In [3]:

```
coefficient_matrix
```

Out[3]:

```
[[5, -2, 3], [-3, 9, 1], [2, -1, -7]]
```

In [4]:

```
constant_matrix = list(eval(input("Enter the constants : ")))
constant_matrix
```

Out[4]:

```
[-1, 2, 3]
```

In [99]:

```
solution_matrix = []

solution_matrix.append(list(eval(input("Enter the initial approximation for variables : "))))

solution_matrix
```

Out[99]:

```
[[0, 0, 0]]
```

In [100]:

```
iteration = 0

while True:
    sol = []
    print(f"Iteration {iteration+1} : ", end="")
    for i in range(n):
        sol.append(constant_matrix[i])
        # print(sol[i])
        for j in range(len(coefficient_matrix[i])):
            if i!=j:
                sol[i] -= coefficient_matrix[i][j]*solution_matrix[iteration][j]

        sol[i] = round(sol[i]/coefficient_matrix[i][i], 5)

        print(sol[i] , end=',')

    print()
    solution_matrix.append(sol)

    if solution_matrix[iteration+1] == solution_matrix[iteration]:
        break


    iteration += 1
```

```
Iteration 1 : -0.2,0.22222,-0.42857,
Iteration 2 : 0.14603,0.20317,-0.51746,
Iteration 3 : 0.19174,0.32839,-0.41587,
Iteration 4 : 0.18088,0.33234,-0.4207,
Iteration 5 : 0.18536,0.32926,-0.42437,
Iteration 6 : 0.18633,0.33116,-0.42265,
Iteration 7 : 0.18605,0.33129,-0.42264,
Iteration 8 : 0.1861,0.3312,-0.42274,
Iteration 9 : 0.18612,0.33123,-0.42271,
Iteration 10 : 0.18612,0.33123,-0.42271,
```

In [102]:

```
coefficient_matrix
```

Out[102]:

```
[[5, -2, 3], [-3, 9, 1], [2, -1, -7]]
```

In [103]:

```
constant_matrix
```

Out[103]:

```
[-1, 2, 3]
```

In [104]:

```
solution_matrix
```

Out[104]:

```
[[0, 0, 0],
 [-0.2, 0.22222, -0.42857],
 [0.14603, 0.20317, -0.51746],
 [0.19174, 0.32839, -0.41587],
 [0.18088, 0.33234, -0.4207],
 [0.18536, 0.32926, -0.42437],
 [0.18633, 0.33116, -0.42265],
 [0.18605, 0.33129, -0.42264],
 [0.1861, 0.3312, -0.42274],
 [0.18612, 0.33123, -0.42271],
 [0.18612, 0.33123, -0.42271]]
```

In [101]:

```
sol
```

Out[101]:

```
[0.18612, 0.33123, -0.42271]
```

In [105]:

```
round(5*solution_matrix[-1][0]-2*solution_matrix[-1][1]+3*solution_matrix[-1][2]+1)
```

Out[105]:

```
0
```

In [109]:

```
round(-3*solution_matrix[-1][0]+9*solution_matrix[-1][1]+solution_matrix[-1][2]-2)
```

Out[109]:

```
0
```

In [107]:

```
round(2*solution_matrix[-1][0]-solution_matrix[-1][1]-7*solution_matrix[-1][2]-3)
```

Out[107]:

```
0
```

In [ ]:

```

```