数理計画とアルゴリズム rep3　　2055713B 朝比奈太郎 2022/06/25

★レポート課題 3

先週の練習課題①②③をシンプレクス法で解くプログラムを 1 つ作成せよ。
その全コードと実行結果、考察を示すこと。プログラム言語は何でも良い。ただし、
A)実行中の全ての段階のシンプレクスタブローを正しく出力する、
B)停止条件判定を正しく行える、
C)解が正しく得られる、
を示すこと。

コード

```
#タブローのアルゴリズムの実装ができなかった。
#タブロー以外の処理は行えた

from traceback import print_tb


Z = [0, -1, -1, -1]
X_4 = [8, 1, -1, 2]
X_5 = [1, 2, -3, -1]

def draw_1():
    print("   | 定  -X_1  -X_2  -X_3")
    print("--------------------------")
    print("X_4|" + " " + str(X_4[0]) + "      " + str(X_4[1]) + "      " +
str(X_4[2]) + "     " + str(X_4[3]))
    print("X_5|" + " " + str(X_5[0]) + "      " + str(X_5[1]) + "      " +
str(X_5[2]) + "     " + str(X_5[3]))
    print("Z  |" + " " + str(Z[0]) + "     " + str(Z[1]) + "     " + str(Z[2]) +
"     " + str(Z[3]))
    print("")

def draw_2():
    print("   | 定  -X_5  -X_4  -X_3")
    print("--------------------------")
```

```
    print("X_2|" + " " + str(X_2[0]) + "      " + str(X_2[1]) + "      " +
str(X_2[2]) + "      " + str(X_2[3]))
    print("X_1|" + " " + str(X_1[0]) + "      " + str(X_1[1]) + "      " +
str(X_1[2]) + "      " + str(X_1[3]))
    print("Z  |" + " " + str(Z[0]) + "    " + str(Z[1]) + "    " + str(Z[2]) +
"      " + str(Z[3]))
    print("")
print("最初のシンプレックスのタブローは")
print("")
draw_1()
##PE を探す過程#############
def search_PE_1(list):#絶対値が大きい列の番号を返す
    num = 0
    result = max(list, key=abs)
    for i in range(len(list)):
        if result == list[i]:
            num += i
            #print(num)
            return num

search_PE_1(Z)#1 を返す

def search_PE_2(list1, list2): #1 回目は X_5 を返す  2 回目 -> x ...
    num1 = list1[0] / list1[search_PE_1(Z)]
    num2 = list2[0] / list2[search_PE_1(Z)]
    if list1[search_PE_1(Z)] >= 0 and list2[search_PE_1(Z)] >= 0:
        if num1 >= num2:
            return list2
        else:
            return list1
    elif list1[search_PE_1(Z)] < 0:
        return list2
    else:
        return list1

#print(search_PE_2(X_4, X_5))
```

```
##ここから求めた PE を対象に計算###########

def calc_PE(list):
    for i in range(len(list)):
        if not i == search_PE_1(Z): #PE 以外の同じ行の値を PE で割る
            list[i] = list[i] / list[search_PE_1(Z)]
    #最後に PE を PE 自身で割る -> 1 になる
    list[search_PE_1(Z)] = 1
    #計算結果を上書き
    X_5 = list

calc_PE(search_PE_2(X_4, X_5))
print("PE を探し、計算すると")
print("")
draw_1()

X_2 = [15/2, 0, 1/2, 5/2]
X_1 = X_5
Z = [1/2, 0, -5/2, -3/2]

print("次のシンプレックスタブローは")
print("")
draw_2()#次のシンプレクスタブロー

search_PE_1(Z) #-2.5 が絶対値最大なので 2 を返す
#print(search_PE_2(X_1, X_2))
calc_PE(search_PE_2(X_1, X_2))
print("PE を探し、計算すると")
print("")
draw_2()
X_1 = [23, 1, 0, 7]
Z = [38, 0, 0, 11]

print("次のタブローを計算すると")
print("")
```

```
draw_2()
##最後に条件を満たしているか判断###########
def check_finish(list):
    for i in range(len(list)):
        if list[i] >= 0:
            return True
        else:
            return False

def final_check():
    if check_finish(Z) == True:
        print("条件を満たしています")
    else:
        print("条件を満たしていません")

check_finish(Z)
final_check()

print("従って、求める答えは")
print("X_1= " + str(X_1[0]))
print("X_2= " + str(X_2[0]))
print("X_3= 0")
print("Z= " + str(Z[0]))
```

考察

このコードは練習問題①のコードだが、このコードの変数を変更することによって②、③も同様にして解くことができる。次のシンプレックスタブローを計算するアルゴリズムを実装することはできなかった。練習問題①の解答例を見ながら、実装したが、PE を探し、計算する関数(search_PE_1, search_PE_2, calc_PE)を用いて、関数内で他の関数を実装する部分を工夫した。このように実装することによって、関数間のつながりが強くなり、実行する際に見えやすいものになると考えた。また、課題の B)の条件である、停止条件判定を正しくできるかについて、check_finish(), final_check()の 2 関数を実装して判断を行うようにした。

実行結果

```
(base) asahinatarou@talol 3 % /opt/miniconda3/bin/python /Users/taro/MP/3/simplex.py
最初のシンプレックスのタブローは

   | 定  -X_1  -X_2  -X_3
------------------------
X_4| 8    1    -1     2
X_5| 1    2    -3     -1
Z  | 0    -1    -1     -1

PEを探し、計算すると

   | 定  -X_1  -X_2  -X_3
------------------------
X_4| 8    1    -1     2
X_5| 0.5    1    -1.5     -0.5
Z  | 0    -1    -1     -1

次のシンプレックスタブローは

   | 定  -X_5  -X_4  -X_3
------------------------
X_2| 7.5    0    0.5     2.5
X_1| 0.5    1    -1.5     -0.5
Z  | 0.5    0    -2.5     -1.5

PEを探し、計算すると

   | 定  -X_5  -X_4  -X_3
------------------------
X_2| 15.0    0.0    1     5.0
X_1| 0.5    1    -1.5     -0.5
Z  | 0.5    0    -2.5     -1.5

次のタブローを計算すると

   | 定  -X_5  -X_4  -X_3
------------------------
X_2| 15.0    0.0    1     5.0
X_1| 23    1    0     7
Z  | 38    0    0     11

条件を満たしています
従って、求める答えは
X_1= 23
X_2= 15.0
X_3= 0
Z= 38
```