Учреждение образования
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»
Кафедра информатики

Отчет по лабораторной работе №2
Основная фаза симплекс метода.

Выполнил:
студент гр. 953505
Басенко К. А.

Руководитель:
доцент
Алёхина А. Э.

Минск 2022

Краткие теоретические сведения

Теория нам говорит, что базисных допустимых планов конечное число. Более того, если задача совместна, и целевая функция канонической задачи ограничена сверху, то существует оптимальный план, который является базисным.

Симплекс-метод перебирает базисные допустимые планы, но делает это систематически. А именно: на первой фазе симплекс-метода находит какой-нибудь базисный допустимый план, либо сообщает, что задача несовместна. Вторая фаза, или основная фаза симплекс-метода, итерационная. На первой итерации симплекс-метод берет найденный на первой фазе базисный допустимый план и по нему строит новый базисный допустимый план, а условием, что значение целевой функции на новом базисном допустимом плане не меньше, чем на старом. Новый базисный допустимый план объявляется текущим. По текущему базисному допустимому плану считается новый базисный допустимый план, для которого значение целевой функции, которое не меньше, чем текущее. И так далее. Процесс останавливается в тот момент, когда симплекс-метод обнаруживает, что достигнут оптимальный базисный план, либо, что целевая функция задачи не ограничена на множестве допустимых планов сверху.

Программная реализация основной фазы симплекс-метода

Симплекс-метод реализован в функции, на вход в которую идут: матрица условий *syst*, вектор ограничений *b*, вектор стоимостей *c*, начальный базисный план *x*, базисные индексы *Jb*:

```
def Simplex(syst, b, c, x, Jb):
```

По множеству базисных индексов составляются матрица A_B, вектор c_B:

```
cb = []
Ab = np.zeros([len(Jb)] * 2)
for i, j in enumerate(Jb):
    Ab[:, i] = syst[:, j]
    cb.append(c[j])

invAb = np.linalg.inv(Ab)
```

Высчитываются вектор потенциалов и оценок:

```
u = np.dot(cb, invAb)

delta = np.matmul(u, syst) - c
```

Проверяется оптимальность текущего плана:

```
optimal = True
for i in range(n):
    if i not in Jb and delta[i] < 0:
        optimal = False
        break

if optimal:
    F = np.dot(c, x)
    print("План отпимальный")
    print(f"F{tuple(x)} = {F}")

    return x, F, Jb
```

Если план неоптимальный, то среди небазисных отрицательных компонент выберем одну и запомним индекс выбранной компоненты как j_0:

```
j0 = 0
for i in range(len(delta)):
    if (delta[i] < 0 and i not in Jb)
        j0 = i
        break
```

Высчитывается вектор z = :

```
z = np.matmul(invAb, syst[:,j0])
```

Высчитываются числа *thetas*, количество которых равно количеству базисных индексов:

```
thetas = []
for i in range(len(Jb)):
    thetas.append(x[Jb[i]] / z[i] if z[i] > 0 else np.inf)
```

Среди них находим минимальное, запоминаем его индекс:

```
theta0 = min(thetas)
if (theta0 == np.inf):
    print('Целевая функция неограничена сверху')
    return
j0_new = thetas.index(theta0)
```

Далее меняем небазисный индекс *j0* и базисный индекс *j0_new* местами:

```
Jb_new = Jb.copy()
Jb_new[j0_new] = j0
```

Обновляем компоненты базисного множества:

```
for i in range(len(x)):
    if (i not in Jb):
        x[i] = 0

for i, j in enumerate(Jb):
    x[j] -= theta0 * z[i]
x[j0] = theta0
```

Задаем базис для следующей итерации:

```
Jb[:] = Jb_new
```

Для следующей итерации по множеству базисных индексов составляется вектор c_B:

```
cb = []
for j in Jb:
    cb.append(c[j])
```

Для следующей итерации находится матрица, обратная матрице A_B с заменой столбца:

```
invAb = InvMatrix(Ab, invAb, syst[:,j0], j0_new)
```

Далее заново высчитываются вектора потенциалов и оценок, проверяется оптимальность плана и т.д.

Программная реализация обращения матрицы с измененным столбцом

Это действо реализовано в функции, на вход в которую идут: матрица *A*, в которой будет меняться столбец, и обратная которой будет находиться; уже известная матрица, обратная матрице *A* -- *invA*; новый столбец *x*, индекс нового столбца *i*:

```
def InvMatrix(A, invA, x, i):
```

Меняем столбец в матрице *A*:

```
A[:,i] = x[:]
```

Находим $l = invA \cdot x$. Если $l = 0$, то матрица *A* необратима и метод завершает свою работу, в противном случае матрица *A* обратима:

```
l = invA.dot(x)
if (l[i] == 0):
    print("Матрица A необратима")
    return
```

Формируем вектор *l* заменой *i*-го элемента на -1. Находим *ls*:

```
ln = l.copy()
ln[i] = -1
ls = (-1/l[i]) * ln
```

Формируем матрицу *Q*, которая получается из единичной матрицы заменой *i*-го столбца на столбец *ln*, находим обратную матрицу перемножением *Q* и *invA*, возвращаем результат:

```
Q = np.eye(len(A))
Q[:,i] = ls
return np.matmul(Q, invA)
```

Тестовые примеры

Входные данные для тестирования:

```
test_cases = [
    [
        syst := [
            [-1, 1, 1, 0, 0],
            [1, 0, 0, 1, 0],
            [0, 1, 0, 0, 1],
        ],
        b := [1, 3, 2],
        c := [1, 1, 0, 0, 0],
        x := [0, 0, 1, 3, 2],
        Jb := [2, 3, 4],
        res := [3, 2, 2, 0, 0],
    ],
```

```
    [
        syst := [
            [1, 2, 1, 0, 0, 0],
            [2, 1, 0, 1, 0, 0],
            [1, 0, 0, 0, 1, 0],
            [0, 1, 0, 0, 0, 1]
        ],
        b := [10, 11, 5, 4],
        c := [20, 26, 0, 0, 0, 1],
        x := [0, 0, 10, 11, 5, 4],
        Jb := [2, 3, 4, 5],
        res := [4, 3, 0, 0, 1, 1],
    ],
```

```
    [
        syst := [
            [1, 1, 1, 0, 0, 0],
            [2, 5, 0, 1, 0, 0],
            [1, 0, 0, 0, 1, 0],
            [1, 0, 0, 0, 0, 1]
        ],
        b := [9, 30, 5, 7],
        c := [19, 29, -1, 0, 0, 0],
        x := [0, 0, 9, 30, 5, 7],
        Jb := [2, 3, 4, 5],
        res := [5, 4, 0, 0, 0, 2],
    ]
]
```

Выполняется метод с предоставленными данными, возвращается результат, выводится полученный результат, выводится ожидаемый результат:

```
for test_case in test_cases:
    syst, b, c, x, Jb, res = test_case
    x, F, Jb = Simplex(syst, b, c, x, Jb)
    print(f'правильный ответ:\nF{tuple(res)} = {np.array(c).dot(res)}\
```

Вывод, соответствующий предоставленным входным данным:

```
План отпимальный
F(3, 2, 2, 0, 0) = 5
правильный ответ:
F(3, 2, 2, 0, 0) = 5

План отпимальный
F(4, 3, 0, 0, 1, 1) = 159
правильный ответ:
F(4, 3, 0, 0, 1, 1) = 159

План отпимальный
F(5, 4, 0, 0, 0, 2) = 211
правильный ответ:
F(5, 4, 0, 0, 0, 2) = 211
```