Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

Кафедра информатики

Отчет по лабораторной работе №4
Двойственный симплекс-метод.

Выполнил:

студент гр. 953505

Басенко К. А.

Руководитель:

доцент

Алёхина А. Э.

Минск 2022

Постановка задачи

Двойственный симплекс-метод — задача линейного программирования с n переменными и m ограничениями. При решении задачи ЛП обычным симплекс-методом свободные члены ограничений предполагались неотрицательными, а при решении задачи ЛП двойственным симплекс методом, свободные члены могут быть любыми числами. Задачу можно сформулировать следующим образом.

Дано:

- вещественный n-мерный вектор c,
- m × m-мерная вещественная матрица A
- вещественный m-мерный вектор b,

Целью двойственного симплекс-метода является поиск n-мерного вектора x, который максимизирует $c'x$, при $Ax = b$, $x \geq 0$, где c' обозначает транспонированный вектор.

Краткие теоретические сведения

Опишем двойственный симплекс-метод, который является специальным алгоритмом построения оптимального плана задачи линейного программирования посредством преобразования планов двойственной задачи.

Для решения задачи двойственным симплекс-методом, кроме исходных данных *c, b, A*, на каждой итерации необходимо знать следующие параметры:

1) текущий базисный двойственный план *y*;

2) соответствующий двойственному плану *y* базис $J_Б = \{j_1, j_2, \dots, j_m\}$;

3) $m \times m$-матрицу $B = A_Б^{-1}$, обратную к базисной матрице $A_Б = (A_j, j \in J_Б)$.

Опишем общую итерацию двойственного симплекс-метода по шагам.

Шаг 1. Найдем базисные компоненты псевдоплана $\aleph$, соответствующего базису $J_Б$: $\aleph_Б = (\aleph_j, j \in J_Б) = Bb$.

Шаг 2. Если выполняются неравенства $\aleph_j \geq 0,\ j \in J_Б$, то STOP: вектор $\aleph = (\aleph_Б, \aleph_Н = 0)$ является оптимальным планом задачи, а вектор *y* – оптимальным планом задачи. В противном случае перейдем к шагу 3.

Шаг 3. Среди базисных индексов $J_Б = \{j_1, j_2, \dots, j_m\}$ выберем индекс j_s, для которого $\aleph_{j_s} < 0$. Подставим *m*-вектор Δy и числа $\mu_j,\ j \in J_Н = J \setminus J_Б$, по правилам

$$\Delta y' = e_s' B; \quad \mu_j = \Delta y' A_j,\ j \in J_Н.$$

Если $\mu_j \geq 0,\ j \in J_Н$, то STOP: ограничения исходной задачи несовместны, а целевая функция двойственной задачи не ограничена снизу на множестве ее планов. В противном случае перейдем к шагу 4.

Шаг 4. Найдем минимум

$$\sigma_0 = \min_{j \in J_H, \ \mu_j < 0} (c_j - A'_j y) / \mu_j$$

и выберем в качестве индекса j_0 любой элемент из множества $\{j \in J_H : \mu_j < 0, (c_j - A'_j y) / \mu_j = \sigma_0\}$.

Шаг 5. Построим новый базисный двойственный план $\bar{y}$ и соответствующий ему базис $\bar{J}_Б$ по правилам

$$\bar{y} = y + \sigma_0 \Delta y; \quad \bar{J}_Б = (J_Б \setminus j_s) \cup j_0 = \{j_1, \ldots, j_{s-1}, j_0, j_{s+1}, \ldots, j_m\}.$$

Шаг 6. Вычислим матрицу $\bar{B}$, обратную к новой базисной матрице $\bar{A}_Б = (A_j, j \in \bar{J}_Б)$, по правилам, описанным на шаге 6 итерации прямого симплекс-метода.

Переходим к следующей итерации, исходя из новых значений для базисного двойственного плана $\bar{y}$, базиса $\bar{J}_Б$ и матрицы $\bar{B}$.

Программная реализация

Весь двойственный симплекс-метод представлен в функции *Dual_Simplex(...)*, на вход в который идут матрица условий *_syst*, вектор ограничений *_b*, вектор стоимостей *_c*, множество базисных индексов *_Jb*:

```
def Dual_simplex(_syst, _b, _c, _Jb):
```

Создается начальный базисный план:

```
while True:
    _x = np.array([0] * n, dtype=float)
    _x[_Jb] = _b
```

По множеству базисных индексов составляются матрица A_B, вектор c_B:

```
    _Ab = _syst[:, _Jb]
    _cb = _c[_Jb]

    _Ab_inv = np.linalg.inv(_Ab)
```

Вычисляется двойственный базисный план и псевдоплан, соответствующий текущему базису:

```
    y = np.matmul(_cb, _Ab_inv)

    not_Jb = np.array([i for i in range(n) if i not in _Jb])

    pseudo_plan = np.matmul(_Ab_inv, _b)
```

Обновляем базисный план, проверяем его оптимальность:

```
    if all(_x[_Jb] >= 0):
        print("План оптимальный")
        print(f'F{tuple([round(xj, 3) for xj in _x])} = {np.array(_c).dot(_x)}')
        return
```

Среди базисных индексов $J_Б = \{j_1, j_2, \dots, j_m\}$ выберем индекс j_s, для которого $\aleph_{j_s} < 0$. Подставим *m*-вектор Δy и числа $\mu_j, j \in J_H = J \setminus J_Б$, по правилам $\Delta y' = e_s' B; \quad \mu_j = \Delta y' A_j, j \in J_H$.

```
minimum_pseudo_x_index = np.argmin(pseudo_plan)

delta_y = _Ab_inv[minimum_pseudo_x_index]

_mu = np.array([np.inf] * n)
_mu[not_Jb] = np.matmul(delta_y, _syst[:, not_Jb])
```

Проверяем $\mu_j \geq 0,\ j \in J_H$:

```
if not any(muj < 0 for muj in _mu):
    print("Задача несовместна")
    return
```

Найдем минимум $\sigma_0 = \min\limits_{j \in J_{H},\ \mu_j < 0} (c_j - A'_j y) / \mu_j$ и выберем в качестве индекса j_0 любой элемент из множества $\{j \in J_H : \mu_j < 0, (c_j - A'_j y) / \mu_j = \sigma_0\}$.

```
sigmas = []
for j in not_Jb:
    if _mu[j] >= 0:
        sigmas.append(np.inf)
    else:
        sigmas.append((_c[j] - _syst[:, j].dot(y)) / _mu[j])

sigma0 = min(sigmas)
if (sigma0 == np.inf):
    print('Целевая функция неограничена сверху')
    return

j0 = not_Jb[np.argmin(sigmas)]
```

Построим новый базисный двойственный план $\bar{y}$ и соответствующий ему базис $\bar{J}_Б$ по правилам $\bar{y} = y + \sigma_0 \Delta y;\quad \bar{J}_Б = (J_Б \setminus j_s) \cup j_0 = \{j_1, \ldots, j_{s-1}, j_0, j_{s+1}, \ldots, j_m\}$.

```
_Jb[minimum_pseudo_x_index] = j0

y += sigma0 * delta_y
```

Далее начинается новая итерация.

Тестовые примеры

Данные для тестирования:

```
test_cases = [
    [
        syst := [
            [-2, -1, -4, 1, 0],
            [-2, -2, -2, 0, 1],
        ],
        b := [-1, -3./2],
        c := [-4, -3, -7, 0, 0],
        Jb := [3, 4],
        res := [1/4., 1/2., 0, 0, 0]
    ],
```

```
    [
        syst := [
            [1, -5, 1, 0],
            [-3, 1, 0, 1],
        ],
        b := [-10, -12],
        c := [0, -6, 1, 0],
        Jb := [2, 3],
        res := [5, 3, 0, 0]
    ],
```

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

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

```
for *data, res in test_cases:
    Dual_simplex(*data)
    print(f'Правильный ответ:\nx = {tuple(res)}\n\n')
```

Вывод, соответствующий предоставленным входным данным:

```
План отпимальный
F(0.25, 0.5, 0.0, 0.0, 0.0) = -2.5
Правильный ответ:
x = (0.25, 0.5, 0, 0, 0)

План отпимальный
F(5.0, 3.0, 0.0, 0.0) = -18.0
Правильный ответ:
x = (5, 3, 0, 0)
```

```
План отпимальный
F(2.0, 3.0, 0.0, 0.0) = -15.0
Правильный ответ:
x = (2, 3, 0, 0)
```