Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и кибербезопасности

Высшая школа компьютерных технологий и информационных систем

**Отчёт расчетной работе № 3**

Дисциплина: Системный анализ и принятие решений.

Выполнил студент гр. 5130901/10101 \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д.Л. Симоновский (подпись)

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.Г. Сиднев (подпись)

“18” октябрь 2023 г.

Санкт-Петербург

2023

Оглавление

[1. Условие 2](#_Toc147352035)

[2. Ход решения 2](#_Toc147352036)

[2.1. Геометрическая интерпретация задачи и её графическое решение 2](#_Toc147352037)

[2.2. Обозначение опорных точек и соответствующих им наборов базисных переменных 3](#_Toc147352038)

[2.3. Решение симплекс-методом в табличной форме. 3](#_Toc147352039)

[2.4. Решение симплекс-методом в табличной форме. 9](#_Toc147352040)

[2.5. Введение дополнительного ограничения, отсекающего оптимальную точку. Решение новой задачи двойственным симплексом-методом в табличной форме. 11](#_Toc147352041)

[2.6. Формулировка задачи, двойственной по отношению к исходной: 13](#_Toc147352042)

[2.7. Определение координат сопряженных опорных точек прямой и двойственной задач. Нахождение оптимального решения двойственной задачи по оптимальному решению прямой задачи: 14](#_Toc147352043)

[3. Вывод: 15](#_Toc147352044)

# Условие

Дана задача нелинейного программирования в следующей форме:

При следующих значениях:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Вариант |  |  |  |  |  |
| 17 | -7 | -7 | 2 | 34 | 50 |

Итоговая задача:

Градиент целевой функции:

Матрица Гессе:

# Ход решения

Создадим функции, которые будем использовать в ходе решения данной расчетной работы.

Функция для получения коэффициентов C:

**def** get\_C**():**

**return** np**.**array**([-**7**,** **-**7**,** 2**,** 34**,** 50**])**

Функция для получения матрицы Гессе, которая необходима в ходе использования многих методов:

**def** get\_H**(**C**):**

**return** np**.**array**([[**C**[**0**]** **\*** 2**,** C**[**2**]],** **[**C**[**2**],** C**[**1**]** **\*** 2**]])**

Функция для получения градиента целевой функции:

**def** calculate\_dfX**(**X**,** C**):**

**return** np**.**array**([**C**[**0**]** **\*** 2 **\*** X**[**0**]** **+** C**[**2**]** **\*** X**[**1**]** **+** C**[**3**],** C**[**1**]** **\*** 2 **\*** X**[**1**]** **+** C**[**2**]** **\*** X**[**0**]** **+ ㅤ** ㅤ + C**[**4**]])**

Функция для получения значения целевой функции:

**def** calculate\_fX**(**X**,** C**):**

**return** C**[**0**]** **\*** X**[**0**]** **\*\*** 2 **+** C**[**1**]** **\*** X**[**1**]** **\*\*** 2 **+** C**[**2**]** **\*** X**[**0**]** **\*** X**[**1**]** **+** C**[**3**]** **\*** X**[**0**]** **+** C**[**4**]** **\*** ㅤ + X**[**1**]**

Напишем функцию для отрисовки графика функции. Она будет получать значения x, y, чтоб нарисовать путь к итоговому решению:

**def** draw**(**x**,** y**,** C**):**

x\_1 **=** np**.**arange**(min(**x**)** **-** 1**,** **max(**x**)** **+** 1.01**,** 0.01**)**

x\_2 **=** np**.**arange**(min(**y**)** **-** 1**,** **max(**y**)** **+** 1.01**,** 0.01**)**

x\_1**,** x\_2 **=** np**.**meshgrid**(**x\_1**,** x\_2**)**

w **=** **(**C**[**0**]** **\*** x\_1 **\*\*** 2 **+** C**[**1**]** **\*** x\_2 **\*\*** 2 **+** C**[**2**]** **\*** x\_1 **\*** x\_2 **+** C**[**3**]** **\*** x\_1 **+** C**[**4**]** **\*** x\_2**)**

plt**.**plot**(**x**,** y**,** '.-'**)**

contours **=** plt**.**contour**(**x\_1**,** x\_2**,** w**,** 15**)**

contours**.**clabel**()**

Так же создадим ряд функций для вывода данных в виде таблицы:

table **=** **None**

**def** create\_table**(**header**):**

**global** table

table **=** PrettyTable**(**header**)**

**def** add\_row**(**row**):**

table**.**add\_row**(**row**)**

**def** print\_table**():**

**print(**table**)**

Естественно, это не самый правильный способ решения с точки зрения языка Python, однако цель этого расчетного задания – получить результат, поэтому автор решил оставить вывод в такой форме.

Так же стоит отметить, что во всех заданиях начальной точкой выбраны пары (1; 5), (5; 1), (-2; -3), а необходимая точность решения 0.01.

В дальнейшем код приводится не будет, его можно найти в приложении.

## Метод наискорейшего подъема:

Траектория поиска решения:

Где - длина шага, - вектор направления.

Вызовем программу и получим следующий результат:

Изображение выглядит как диаграмма, круг, линия, График

Автоматически созданное описание

Рис. 1. Метод наискорейшего подъема. (1,5)

Изображение выглядит как диаграмма, линия, круг, Параллельный

Автоматически созданное описание

Рис. 2. Метод наискорейшего подъема. (5,1)

Изображение выглядит как диаграмма, линия, круг, Параллельный

Автоматически созданное описание

Рис. 3. Метод наискорейшего подъема. (-2, -3)

Видим, что графики действительно соответствуют ожидаемомым для метода наискорейшего подъема.

Промежуточные данные выглядят следующим образом:

Для (1; 5):

+------+------------+------------+------------+

| i | x1 | x2 | fX |

+------+------------+------------+------------+

| 1 | 1.000 | 5.000 | 112.000 |

| 2 | 2.903 | 3.858 | 150.821 |

| 3 | 2.991 | 4.005 | 150.999 |

| 4 | 3.000 | 3.999 | 151.000 |

+------+------------+------------+------------+

Для (5; 1):

+------+------------+------------+------------+

| i | x1 | x2 | fX |

+------+------------+------------+------------+

| 1 | 5.000 | 1.000 | 48.000 |

| 2 | 2.863 | 3.891 | 150.816 |

| 3 | 3.004 | 3.995 | 151.000 |

| 4 | 3.000 | 4.000 | 151.000 |

+------+------------+------------+------------+

Для (-2, -3):

+------+------------+------------+------------+

| i | x1 | x2 | fX |

+------+------------+------------+------------+

| 1 | -2.000 | -3.000 | -297.000 |

| 2 | 2.595 | 4.220 | 149.332 |

| 3 | 2.981 | 3.974 | 150.994 |

| 4 | 2.998 | 4.001 | 151.000 |

| 5 | 3.000 | 4.000 | 151.000 |

+------+------------+------------+------------+

По таблице видно, что алгоритм нашел ответ для всех начальных точек, получил результат (3; 4), а целевая функция приняла значение 151.

## Метод Ньютона

Траектория поиска решения:

Где - длина шага, - вектор направления.

Вызовем программу и получим следующий результат:

Изображение выглядит как диаграмма, круг, линия, График

Автоматически созданное описание

Рис. 4. Метод Ньютона. (1,5)

Изображение выглядит как диаграмма, линия, круг, График

Автоматически созданное описание

Рис. 5. Метод Ньютона. (5,1)

Изображение выглядит как диаграмма, линия, круг, Параллельный

Автоматически созданное описание

Рис. 6. Метод Ньютона. (-2, -3)

Промежуточные данные выглядят следующим образом:

Для (1; 5):

+------+------------+------------+------------+

| i | x1 | x2 | fX |

+------+------------+------------+------------+

| 1 | 1.000 | 5.000 | 112.000 |

| 2 | 3.000 | 4.000 | 151.000 |

+------+------------+------------+------------+

Для (5; 1):

+------+------------+------------+------------+

| i | x1 | x2 | fX |

+------+------------+------------+------------+

| 1 | 5.000 | 1.000 | 48.000 |

| 2 | 3.000 | 4.000 | 151.000 |

+------+------------+------------+------------+

Для (-2, -3):

+------+------------+------------+------------+

| i | x1 | x2 | fX |

+------+------------+------------+------------+

| 1 | -2.000 | -3.000 | -297.000 |

| 2 | 3.000 | 4.000 | 151.000 |

+------+------------+------------+------------+

По таблице видно, что алгоритм нашел ответ для всех начальных точек, получил результат (3; 4), а целевая функция приняла значение 151.

Стоит отметить, что метод Ньютона сделал это за минимальное число итераций, а именно одну.

## Метод сопряженных градиентов

Траектория поиска решения:

Где - длина шага, - вектор направления.

Вызовем программу и получим следующий результат:

Изображение выглядит как диаграмма, круг, линия, График

Автоматически созданное описание

Рис. 7. Метод сопряженных градиентов. (1,5)

Изображение выглядит как диаграмма, круг, линия, Параллельный

Автоматически созданное описание

Рис. 8. Метод сопряженных градиентов. (5,1)

Изображение выглядит как диаграмма, линия, круг, Параллельный

Автоматически созданное описание

Рис. 9. Метод сопряженных градиентов. (-2, -3)

Промежуточные данные выглядят следующим образом:

Для (1; 5):

+------+------------+------------+------------+

| i | x1 | x2 | fX |

+------+------------+------------+------------+

| 1 | 1.000 | 5.000 | 112.000 |

| 2 | 2.903 | 3.858 | 150.821 |

| 3 | 3.000 | 4.000 | 151.000 |

+------+------------+------------+------------+

Для (5; 1):

+------+------------+------------+------------+

| i | x1 | x2 | fX |

+------+------------+------------+------------+

| 1 | 5.000 | 1.000 | 48.000 |

| 2 | 2.863 | 3.891 | 150.816 |

| 3 | 3.000 | 4.000 | 151.000 |

+------+------------+------------+------------+

Для (-2, -3):

+------+------------+------------+------------+

| i | x1 | x2 | fX |

+------+------------+------------+------------+

| 1 | -2.000 | -3.000 | -297.000 |

| 2 | 2.595 | 4.220 | 149.332 |

| 3 | 3.000 | 4.000 | 151.000 |

+------+------------+------------+------------+

По таблице видно, что алгоритм нашел ответ для всех начальных точек, получил результат (3; 4), а целевая функция приняла значение 151.

## Метод релаксации

Траектория поиска решения:

Где - длина шага, - вектор направления.

Вызовем программу и получим следующий результат:

Изображение выглядит как диаграмма, круг, линия, текст

Автоматически созданное описание

Рис. 10. Метод релаксации. (1,5)

Изображение выглядит как диаграмма, круг, линия, текст

Автоматически созданное описание

Рис. 11. Метод релаксации. (5,1)

Изображение выглядит как диаграмма, линия, круг, Параллельный

Автоматически созданное описание

Рис. 12. Метод релаксации. (-2, -3)

Промежуточные данные выглядят следующим образом:

Для (1; 5):

+------+------------+------------+------------+

| i | x1 | x2 | fX |

+------+------------+------------+------------+

| 1 | 1.000 | 5.000 | 112.000 |

| 2 | 3.143 | 5.000 | 144.143 |

| 3 | 3.143 | 4.020 | 150.860 |

| 4 | 3.003 | 4.020 | 150.997 |

| 5 | 3.003 | 4.000 | 151.000 |

| 6 | 3.000 | 4.000 | 151.000 |

+------+------------+------------+------------+

Для (5; 1):

+------+------------+------------+------------+

| i | x1 | x2 | fX |

+------+------------+------------+------------+

| 1 | 5.000 | 1.000 | 48.000 |

| 2 | 2.571 | 1.000 | 89.286 |

| 3 | 2.571 | 3.939 | 149.741 |

| 4 | 2.991 | 3.939 | 150.974 |

| 5 | 2.991 | 3.999 | 150.999 |

| 6 | 3.000 | 3.999 | 151.000 |

| 7 | 3.000 | 4.000 | 151.000 |

+------+------------+------------+------------+

Для (-2, -3):

+------+------------+------------+------------+

| i | x1 | x2 | fX |

+------+------------+------------+------------+

| 1 | -2.000 | -3.000 | -297.000 |

| 2 | 2.000 | -3.000 | -185.000 |

| 3 | 2.000 | 3.857 | 144.143 |

| 4 | 2.980 | 3.857 | 150.860 |

| 5 | 2.980 | 3.997 | 150.997 |

| 6 | 3.000 | 3.997 | 151.000 |

| 7 | 3.000 | 4.000 | 151.000 |

+------+------------+------------+------------+

По таблице видно, что алгоритм нашел ответ для всех начальных точек, получил результат (3; 4), а целевая функция приняла значение 151.

Специфическая черта этого метода прослеживается как на графиках, так и на таблицах, он двигается вдоль осей координат. На графике это особенно видно, однако и на таблицах можно это заметить, что она из переменных изменяется в то время, как вторая – нет.

## Метод метрики Бройдена

Траектория поиска решения:

Где - длина шага, - вектор направления.

Вызовем программу и получим следующий результат:

Изображение выглядит как диаграмма, круг, линия, График

Автоматически созданное описание

Рис. 13. Метод метрики Бройдена. (1,5)

Изображение выглядит как диаграмма, линия, круг, Параллельный

Автоматически созданное описание

Рис. 11. Метод метрики Бройдена. (5,1)

Изображение выглядит как диаграмма, линия, круг, Параллельный

Автоматически созданное описание

Рис. 12. Метод метрики Бройдена. (-2, -3)

Промежуточные данные выглядят следующим образом:

Для (1; 5):

+------+------------+------------+------------+

| i | x1 | x2 | fX |

+------+------------+------------+------------+

| 1 | 1.000 | 5.000 | 112.000 |

| 2 | 2.903 | 3.858 | 150.821 |

| 3 | 2.895 | 3.978 | 150.924 |

| 4 | 2.896 | 3.968 | 150.924 |

| 5 | 2.936 | 4.033 | 150.959 |

| 6 | 3.005 | 4.032 | 150.993 |

| 7 | 3.018 | 4.018 | 150.996 |

| 8 | 3.001 | 3.999 | 151.000 |

| 9 | 3.000 | 4.000 | 151.000 |

+------+------------+------------+------------+

Для (5; 1):

+------+------------+------------+------------+

| i | x1 | x2 | fX |

+------+------------+------------+------------+

| 1 | 5.000 | 1.000 | 48.000 |

| 2 | 2.863 | 3.891 | 150.816 |

| 3 | 2.988 | 3.895 | 150.924 |

| 4 | 2.956 | 3.905 | 150.932 |

| 5 | 3.027 | 3.949 | 150.974 |

| 6 | 3.023 | 4.005 | 150.997 |

| 7 | 3.017 | 4.013 | 150.997 |

| 8 | 3.004 | 3.997 | 151.000 |

| 9 | 2.999 | 3.999 | 151.000 |

| 10 | 3.000 | 3.999 | 151.000 |

| 11 | 3.000 | 4.000 | 151.000 |

+------+------------+------------+------------+

Для (-2, -3):

+------+------------+------------+------------+

| i | x1 | x2 | fX |

+------+------------+------------+------------+

| 1 | -2.000 | -3.000 | -297.000 |

| 2 | 2.595 | 4.220 | 149.332 |

| 3 | 3.040 | 4.102 | 150.924 |

| 4 | 3.009 | 4.100 | 150.931 |

| 5 | 2.958 | 4.064 | 150.954 |

| 6 | 2.978 | 3.991 | 150.996 |

| 7 | 2.978 | 3.991 | 150.996 |

| 8 | 2.986 | 4.006 | 150.998 |

| 9 | 3.000 | 4.007 | 151.000 |

| 10 | 3.004 | 4.003 | 151.000 |

| 11 | 3.001 | 3.999 | 151.000 |

| 12 | 3.000 | 4.000 | 151.000 |

+------+------------+------------+------------+

По таблице видно, что алгоритм нашел ответ для всех начальных точек, получил результат (3; 4), а целевая функция приняла значение 151.

# Вывод:

В ходе расчетного задания были использованы некоторые методы для безусловной оптимизации заданной задачи нелинейного программирования.

По результатам работы видно, что наибыстрейшим методом является метод Ньютона, как и ожидалось, остальные методы тоже показали себя с хорошей стороны, успешно найдя точку максимума и её координаты во всех случаях.

# Ссылки:

С кодом программы можно ознакомиться на github: <https://github.com/DafterT/SADM>