МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное
учреждение высшего образования
«СЕВАСТОПОЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

МЕТОДЫ ОПТИМИЗАЦИИ И НЕЛИНЕЙНОЕ ПРОГРАММИРОВАНИЕ

Методические указания
к выполнению лабораторного практикума по дисциплине
для магистрантов направления подготовки
09.04.02 – «Информационные системы и технологии»
профиля «Интеллектуальные информационные системы»
всех форм обучения

Севастополь
СевГУ
2024

УДК 519.853(076.5)
ББК 22.18я73
М545

Р е ц е н з е н т ы :
В. С. Чернега – канд. техн. наук, доцент кафедры информационных систем
Т. В. Волкова – канд. техн. наук, доцент кафедры информационных технологий и компьютерных систем

Ответственный за выпуск:
И. П. Шумейко – канд. физ.-мат. наук, доцент заведующий кафедрой информационных систем

Составитель: В. Ю. Карлусов

М545 Методы оптимизации и нелинейное программирование : методические указания выполнению лабораторного практикума по дисциплине для магистрантов направления подготовки 09.04.02 – «Информационные системы и технологии» профиля «Интеллектуальные информационные системы» всех форм обучения / Севастопольский государственный университет ; сост. В. Ю. Карлусов. – Севастополь : СевГУ, 2024. – 85 с. – Текст : электронный.

Цель методических указаний: обеспечить качественное усвоение теоретических положений курса и наиболее полное овладение математическим и вычислительным аппаратами, применяемыми в многомерной условной и безусловной оптимизации.

УДК 519.853(076.5)
ББК 22.18я73

Рассмотрены и утверждены на заседании кафедры информационных систем, протокол № 8 от 16 апреля 2024 г.

Рекомендованы к изданию на заседании Учёного совета Института информационных технологий, протокол № 5 от 18 апреля 2024 г. года.

Содержание

ВВЕДЕНИЕ

Почему у меня такие
хорошие студенты?
Да потому, что я сам
не очень умный.
П. Эренфест

Большинство практических задач, встречающихся в научной и производственной деятельности, имеют множество (часто бесконечное) решений и сложное математическое описание. Выбор наилучшего по принятому критерию решения представляется сложной оптимизационной задачей. Немаловажным инструментом научных исследований при решении оптимизационных задач являются методы нелинейного программирования.

В самом общем случае, задача нелинейного программирования (НП-задача) задаётся математической моделью [[1, 5 – 7, 12 – 14]]

$$f(x_1,x_2,...,x_n) \to \min,$$
$$\begin{cases} g_1(x_1,x_2,...,x_n) \ge 0, \\ g_2(x_1,x_2,...,x_n) \ge 0, \\ ... \\ g_m(x_1,x_2,...,x_n) \ge 0. \end{cases}$$

где функции $f(X)$ и $g_i(X)$, $i = 1, m$, в общем случае, нелинейные.
Следует отметить, что универсального общеупотребимого метода решения указанных задач не существует.

В методических целях, чтобы обеспечить наглядное восприятие хода решения и контроль вычислительного процесса, сделаем следующие допущения:

- число переменных в модели положим равным двум ($n = 2$);
- в качестве целевой функции $f(X)$ будем использовать полином второй степени;
- функции $g_i(X)$, $i = 1, m$, выберем линейные.

Оные допущения,

- во-первых, нисколько не помешают изучению методов оптимизации; во-вторых, сведут общую постановку НП-задачи к задачам квадратичного программирования, для решения которых разработаны программы;
- в-третьих, результаты, полученные с использованием “фирменных” программ, могут быть сопоставлены с ответами, являющихся решениями НП-задач, рассчитанных посредством исследуемых в ходе выполнения лабораторных работ алгоритмами.

ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ И СОДЕРЖАНИЮ ОТЧЁТА

Автор считают целесообразным оформление отчета в отдельной тетради, которая по окончании изучения дисциплины (после сдачи экзамена или зачёта) сдается преподавателю. В этом случае,

- во-первых, результаты выполнения предыдущих работ доступны для анализа;
- во-вторых, наработанный материал может быть с пользою для дела употреблён на этапе подготовки к экзамену либо зачёту.

В общем случае, отчет по лабораторной работе должен содержать:

1. Титульный лист с указанием названия работы и исполнителя её.
2. Цель работы.
3. Математическую модель задачи согласно варианту задания.
4. Результаты предварительных расчетов и аналитических преобразований.
5. Отражение результатов, полученных в ходе вычислений с использованием PC.
6. Выводы, основанные на анализе полученных результатов и результатов предыдущих лабораторных работ.

Вариант задаётся в виде двух чисел:

- первое число, в диапазоне 00 – 99, определяет систему ограничений задачи;
- второе (00 – 49) – задаёт целевую функцию.

Оформленный отчёт защищается в устной форме, в ходе ответов на контрольные вопросы.

В качестве лабораторной установки для проведения исследований, используется персональный компьютер.

Вариант задания выдаётся преподавателем индивидуально на вводном занятии очной формы обучения, либо на установочной сессии заочной формы обучени.

ЛАБОРАТОРНАЯ РАБОТА № 1
ИССЛЕДОВАНИЕ ЗАДАЧ НЕЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ СРЕДСТВАМИ ПАКЕТА MATLAB

… цветная капуста – эта обычная капуста,
получившая позднее высшее образование…
Марк Твен

1. ЦЕЛЬ РАБОТЫ

1. Восстановить практические навыки в использовании среды `MatLab`, начать освоение пакета `Optimization toolbox`.

2. Восстановить теоретические и практические положения математического анализа применительно к решению задач нелинейного программирования (НП-задач).

3. Подготовить тестовые примеры для сопоставления в ходе решения НП-задачи различными методами.

4. Исследовать путём сравнения, результаты теоретических вычислений и расчётов с использованием встроенных процедур `quadprog` и `linprog`.

2. ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

Задача квадратичного программирования, в своей канонической постановке записывается так [7, 11]

$$\begin{aligned} &f(X) = b^T X + \frac{1}{2} X^T C X \to \max, \\ &AX \le A_0,\ X \ge 0, \end{aligned} \tag{1}$$

где C — симметричная отрицательно определённая матрица размерностью [$n \times n$], b^T — вектор-строка размерностью [$1 \times n$], A — матрица системы ограничений размерностью [$m \times n$], A_0 — вектор свободных членов системы ограничений размерностью [$m \times 1$], n — число переменных, m – число ограничений.

Если задача (1) посвящена определению минимума целевой функции, то матрица C должна быть определена положительно.

Положительную или отрицательную определённость матрицы можно оценить по критерию Сильвестра [4], задающему порядок чередования знаков минорных определителей в зависимости от типа определённости.

Существуют случаи, когда матрица C не определена ни положительно, ни отрицательно. В этом случае функция $f(X)$ в отсутствие ограничений, не будет иметь глобального экстремума.

Переменная часть слагаемого (1) функции цели, в котором задействована матрица C, называется квадратичной формой, и, в невекторной записи, представима двойной суммой

$$X^T CX = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{i,j} \cdot x_i \cdot x_j \ . \qquad (2)$$

Из (2) следует, что диагональные элементы матрицы $c_{i,i}$ являются коэффициентами при квадратах неизвестных, а недиагональные $c_{i,j} = c_{j,i}$ – половине коэффициента в произведении $x_i \cdot x_j$. С учётом множителя в выражении (1), имеем, соответственно, $0{,}5 \cdot c_{i,i} \cdot x_i^2$ и $c_{i,j} \cdot x_i \cdot x_j$.

Решение задачи (1) можно производить двояко: воспользоваться конструктивной теоремой квадратичного программирования, либо каким-нибудь подходящим по условиям применения методом.

Констатирующая часть теоремы квадратичного программирования выглядит так [7]

$$\begin{aligned} & b + C \cdot X_0 - A^T \Lambda + V = 0, \text{ а)} \\ & A_0 - AX_0 - W = 0, \qquad \text{б)} \\ & \left.\begin{aligned} V^T X_0 = 0 \\ W^T \Lambda = 0 \end{aligned}\right\}. \qquad \text{в)} \end{aligned} \qquad (3)$$

Компоненты всех векторов Λ, W и V — неотрицательны, а вектора W и V могут быть нулевыми. Условия (3, а) и (3, б) образуют систему из $n + m$ уравнений для $2 \times (n + m)$ неизвестных компонентов X, Λ, V и W. Ограничения (3, в) есть условия дополняющей нежёсткости.

Выражения (3, а и б) являются основанием для построения эквивалентной задачи линейного программирования, в ходе решения которой представляется возможным определить значения элементов всех неизвестных векторов X_0, Λ, W и V и, после получения оптимальных значений ЗЛП, проверить условия дополняющей нежёсткости и неотрицательности.

1. Условия (3, а) и (3, б) необходимо представить в форме, обеспечивающей положительность элементов столбцов свободных членов

$$\begin{cases} A^T \Lambda - C \cdot X - V = b, \\ AX + W = A_0. \end{cases} \qquad (4)$$

2. Поскольку знак в ограничениях (4) – равенство, следует воспользоваться методом искусственного базиса. Необходимо добавить искусственные переменные $\{y_i\}$ и $\{z_i\}$, при этом система ограничений примет вид

$$\begin{cases} A^T \Lambda - C \cdot X - V + Z = b, \\ AX + W + Y = A_0. \end{cases} \quad (5)$$

3. Конструируется псевдоцелевая функция из искусственных переменных

$$\sum_{i=1}^{m} \mu \cdot y_i + \sum_{j=1}^{n} \mu \cdot z_j \to \min . \quad (6)$$

4. Решается эквивалентная ЗЛП с функцией цели (6) при ограничениях (5).

Если в ходе решения ЗЛП векторы Y и Z будут выведены из базиса (достигнут оптимум), а полученные значения X, Λ, V и W не отрицательны, а так же удовлетворяют (3, в), то компоненты вектора X представляют собой оптимальное решение исходной задачи квадратичного программирования (1).

5. Рассчитывается значение целевой функции.

В пакете Optimization toolbox имеются функции, с помощью которых можно получить решение (1):

- `quadprog` – непосредственное решение (1) методом, на который имеется ссылка в документации `MatLab`'а;
- `linprog` – решение эквивалентной ЗЛП, построенной с помощью (5) и (6).

и выполнить демонстрацию результатов:

- `plot` – построение одномерных графиков;
- `meshgrid` – разметка поля аргументов для построения объёмного изображения;
- `mesh` – построение поверхности (есть и другие процедуры)
- `contour` – построение линий равного уровня.

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Получить вариант задания.

2. Пользуясь матричной формой, построить развёрнутую функцию цели.

3. Исследовать поведение целевой функции:

- выпукла (вогнута, с перегибом),

- имеет ли экстремум,
- располагается ли экстремум внутри или вовне области ограничений,
- рассчитать значения независимых переменных, обеспечивающих оптимальное значение целевой функции и само значение.

4. С использованием пакета Optimization toolbox рассчитать координаты решения задачи квадратичного программирования и выполнить необходимые графические построения. Коды представлены в приложении В, разделы В.1 и В.2.

5. Оформить отчет с приложением результатов вычислений, графической информации. Сделать ***содержательный*** вывод, оценивающий полученные результаты.

6. Защитить результаты выполнения лабораторной работы.

4. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какая задача является задачей нелинейного программирования?
2. Какая точка называется экстремальной?
3. Чем условный минимум или максимум отличаются от безусловного?
4. Что такое градиент функции?
5. Какая точка называется стационарной?
6. Когда функция называется выпуклой?
7. Когда функция называется выпуклой?
8. Как построить матрицы Гессе для функции?
9. Как оценить выпуклость или вогнутость функции с использованием матрицы Гессе?
10. Для чего применяется Якобиан?
11. Каковы необходимые условия существования экстремальных значений функций?
12. Как найти наибольшее и наименьшее значения функции нескольких переменных в замкнутой ограниченной области?

ЛАБОРАТОРНАЯ РАБОТА № 2
ИССЛЕДОВАНИЕ АЛГОРИТМА РЕШЕНИЯ ЗАДАЧ НЕЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ ПРЯМЫМИ МЕТОДАМИ

Как хорошо, когда благоденствие
человека основано на законах разума.
Пифагор

1. ЦЕЛЬ РАБОТЫ

1. Получить практические навыки в решении задачи нелинейного программирования.
2. Освоить аспекты применения функции `fminsearch` пакета `Optimization toolbox` среды `MatLab`.
3. Освоить эвристические алгоритмы решения НП-задач без ограничений.
4. Сравнить результат с аналитическими вычислениями, выполненными в ходе лабораторной работы № 1, функции `fminsearch` и собственноручно разработанного модуля.

2. ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

Решение НП-задач возможно разными методами. Исторически первыми возникли эвристические методы.

В данную группу входят прямые методы решения задачи, не использующие значения градиентов функций [10]. К таковым относятся метод траекторий Хука-Дживса и метод Розенброка. Эти методы применяются к оптимизации нелинейных функций без ограничений.

Идея метода Хука-Дживса состоит в следующем.

- Направления поиска ориентированы, так сказать, по сторонам света или вдоль осей координат в обе стороны.
- Вокруг текущей точки производится поиск направления, в котором функция убывает.
- Если такое направление найдено, то шаг поиска увеличивается, а поиск продолжается в выбранном направлении – устанавливается так называемый тренд поиска, и осуществляется до тех пор, пока функция в этом направлении убывает.
- Если факта убывания функции не обнаружено, то шаг поиска уменьшается.

Таким образом, делается попытка найти «овраг» (при минимизации) функции и двигаться вдоль него к точке минимума.

В задачи максимизации ищется направление возрастания функции и «хребет».

Алгоритм метода показан на рисунке 1.

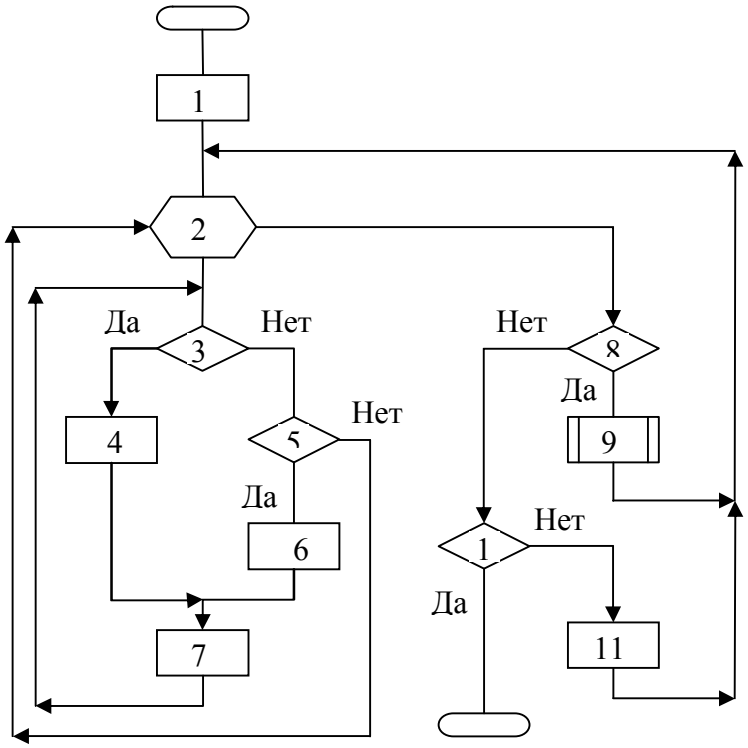

Рисунок 1 – Алгоритм метода Хука-Дживса

Входные данные. Начальное значение шага λ, координаты начальной точки X_0 ускоряющий множитель α, точность нахождения решения ε, список возможных направлений поиска $S = \{s_1, s_2, \ldots, s_n\}$. Для числа переменных равного двум, это орты: $s_1 = (0, 1)$ и $s_2 = (1,0)$.

Условие окончания: значение текущего шага укладывается в точность, $\varepsilon \geq \lambda$.

Блок № 1. Координаты начальной точки переприсваивается текущей переменной: $Y = X_0$.

Блок № 2. Организуется цикл перебора направлений поиска $j = 1, n$, тело цикла составляют блоки №№ 3 – 7.

Сравнение в блоке № 3 проверяется факт убывания функции в выбранном направлении: $f(Y + \lambda \times s_j) < f(Y)$. Если функция убывает, выход «да» блока № 3, то текущая точка перемещается в выбранном направлении $Y = Y + \lambda \times s_j$ (блок № 4).

В противном случае, выход «нет» блока № 3, в блоке № 5 проверяется факт убывания функции в направлении, противоположном выбранному: $f(Y - \lambda \times s_j) < f(Y)$.

Если функция не убывает ни в выбранном, ни в противоположном направлениях (выход «нет», блок № 5), то выбирается очередное направление.

Если сравнение в блоке № 5 произошло удачно (выход «да»), то текущая точка решения перемещается в блоке № 6 по закону $Y = Y - \lambda \times s_j$.

Блок № 7 выполняет факультативное действие $\lambda = 2 \times \lambda$, которое увеличивает шаг, ускоряя тем самым поиск. В ряде реализаций алгоритма данное действие отсутствует.

Блок № 8 достигается в ходе работы алгоритма, после перебора всех направлений поиска и попадания в ситуацию, когда при заданном значении λ в окрестностях точки X_0 убывания функции не обнаружено.

Блок № 9 выполняет пересчёт координат текущей точки по формуле

$$Y = X_0 + \alpha \times (Y - X_0); \; X_0 = Y.$$

Блок № 10 осуществляет проверку окончания: $\varepsilon \geq \lambda$? Если «да» то алгоритм заканчивает работу, если «нет», то шаг поиска уменьшается

$$\lambda = \frac{\lambda}{2},$$

после чего начинается поиск вокруг текущей точки X_0 с уменьшенным шагом.

В качестве недостатка отмечается, что при попадании в область размытого локального минимума есть риск остановиться, не достигнув глобального минимума.

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Реализуйте расчёты по алгоритму Хука-Дживса в среде `MatLab` для Вашего варианта функции, предусмотрите подсчёт числа итераций. Коды представлены в приложении В.3.

2. Воспользовавшись функцией `fminsearch` пакета `Optimization toolbox` среды `MatLab` выполнить поиск решения.

3. Сравнить между собой по точности и по числу итераций результаты работы алгоритмов Хука-Дживса, `fminsearch` и выполненных в предыдущей лабораторной работе расчётов по нахождению глобального экстремума.

4. Оформить отчет с приложением результатов вычислений, необходимой графической информации. Сделать ***содержательный*** вывод, оценивающий полученные результаты.

5. Защитить результаты выполнения лабораторной работы.

4. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Как выглядит в общем виде модель для задачи нелинейного программирования?
2. Какие существуют типы задачи НП, чем они различаются?
3. Как формулируются необходимые условия оптимальности в задачах безусловной оптимизации?
4. Как следует понимать термины «выпуклая» и «вогнутая» функции применительно к задачам нелинейного программирования?
5. Что называется линией уровня функции двух переменных?
6. Почему метод получил ещё название метода траекторий?
7. Что является общей особенностью прямых методов?
8. Критичен ли метод к выбору начальной точки?
9. Почему прямые методы называют ещё и поисковыми?
10. В чём состоят достоинства метода Хука-Дживса?
11. В чём состоят недостатки метода Хука-Дживса?
12. Может ли данный алгоритм зацикливаться? Обоснуйте Ваш ответ.

ЛАБОРАТОРНАЯ РАБОТА № 3
ИССЛЕДОВАНИЕ АЛГОРИТМА РЕШЕНИЯ ЗАДАЧ НЕЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ МЕТОДОМ НАИСКОРЕЙШЕГО СПУСКА

Точное логическое определение понятий –
– главнейшее условие истинного знания.
Сократ

1. ЦЕЛЬ РАБОТЫ

1. Освоить алгоритм метода наискорейшего спуска.
2. Освоить аспекты применения функций `fminunc` и `fminbnd` пакета `Optimization toolbox` среды `MatLab`.
3. Сравнить результаты с данными, полученными аналитически, для функций среды `MatLab` и рассчитанными в предыдущих работах.
4. Получить практические навыки в решении задачи нелинейного программирования методом наискорейшего спуска.

2. ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

Метод *наискорейшего спуска* ещё называется методом *наискорейшего подъёма*.

При решении задачи на минимизацию [2] речь идёт о спуске, а в случае максимизации – о подъёме. В алгоритмах указанное отличие отражается в знаках при выборе направления, которое совпадает с направлением градиента в задачах максимизации и противоположно ему (знак «минус») в задачах минимизации. Граф-схема алгоритма минимизации показана на рисунке 1.

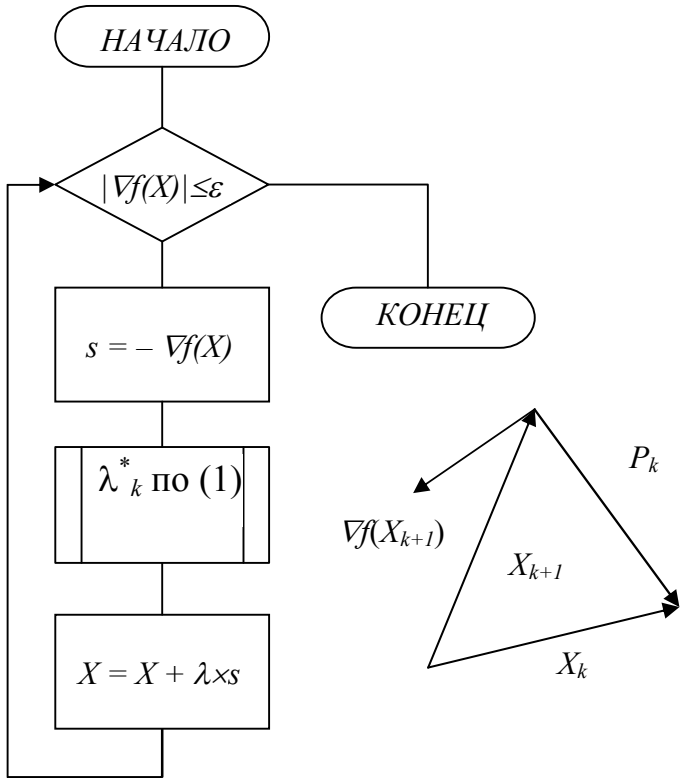

Рисунок 1 – Алгоритм метода наискорейшего спуска

Луи Огюст Коши предложил выбирать значение λ_k путём решения задачи одномерной минимизации из условия [3]

$$\lambda_k^* \Rightarrow \min_{\lambda>0} f(X_k + \lambda s_k), \tag{1}$$

где $s_k = -\nabla f(X_k)$ определяется значением градиента. При этом непосредственно значение λ_k может быть определено из уравнения

$$\frac{\partial f(X_k + \lambda s_k)}{\partial \lambda} = 0 \tag{2}$$

аналитически, либо путём численного решения задачи одномерной минимизации. Алгоритм метода, с точки зрения вычислительной сложности, тривиальный, изображён на рисунке 1.

Если ввести вектор, определяемый как $\vec{P}_k = \vec{X}_{k+1} - \vec{X}_k$, то вектор-градиент $\nabla\vec{f}(X_{k+1})$ перпендикулярен вектору приращения $\vec{P}_k$, то есть $\nabla\vec{f}(X_{k+1}) \perp \vec{P}_k$. Поэтому траектория спуска будет иметь, в общем случае, вид зигзага.

Тут уместно заметить, что метод наискорейшего спуска ***не обладает*** оптимальными свойствами, ***не обеспечивая минимум шагов*** при минимуме ***вычислений***, если только ***направление спуска не совпадает*** с направлением на минимум, локальный или глобальный.

Последнее справедливо только для ограниченного класса функций, поэтому метод считается ***полношаговым***.

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Для заданного варианта найти значение градиента функции аналитически.

2. Реализовать две версии алгоритма в части нахождения λ^*:

- с применением аналитического решения (2) и
- с применением численного решения (2) используя алгоритм одномерной минимизации `fminbnd`.

Коды представлены в приложении В.4.

3. Сравнить между собой по точности и по числу итераций результаты работы версии алгоритмов и результатов выполнения предыдущих лабораторных работ.

4. Оформить отчет с приложением результатов вычислений, необходимой графической информации. Сделать ***содержательный*** вывод, оценивающий полученные результаты.

5. Защитить результаты выполнения лабораторной работы.

4. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое градиент функции нескольких переменных?

2. Как определить составляющие вектора градиента?

3. Когда будет происходить наискорейший спуск, а когда – полношаговый?

4. Каковы основные свойства градиента функции в некой точке?

5. В чем заключается градиентные методы при поиске минимума функции?

6. Как определяется шаг поиска в градиентных методах?

7. Каковы будут значения градиента в точке глобального минимума?

8. Как адаптировать алгоритм метода наискорейшего спуска для поиска максимального значения функции многих переменных?

9. Как, не меняя программу, реализующую алгоритм наискорейшего спуска, найти точку максимума функции многих переменных?

10. Всегда ли метод наискорейшего спуска найдёт точку глобального минимума при наличии хотя бы одного локального?

11. Может ли данный алгоритм работать бесконечно?

ЛАБОРАТОРНАЯ РАБОТА № 4
ИССЛЕДОВАНИЕ АЛГОРИТМА РЕШЕНИЯ ЗАДАЧИ НЕЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ МЕТОДОМ СОПРЯЖЁННОГО ГРАДИЕНТА

Поистине, человеческий ум –
– большой мастер творить чудеса.
М. Монтень

1. ЦЕЛЬ РАБОТЫ

1. Освоить алгоритм метода сопряжённого градиента.

2. Закрепить способы применения функцию `fminunc` и `fminbnd` пакета `Optimization toolbox` среды `MatLab`.

3. Сравнить результаты с данными, полученными аналитически, для функций среды `MatLab` и рассчитанными в предыдущих работах.

4. Получить практические навыки в решении задачи нелинейного программирования методом сопряжённого градиента.

2. ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

В общем случае, система линейно-назависимых направлений поиска S_1, S_2, $...S_{n-1}$ называется *сопряжённой по отношению к некоторой положительно определённой матрице* Q если

$$S_i^T Q S_j = 0;\ \ i \neq j;\ \ i = 1,\overline{n-1},\ \ j = 1,\overline{n-1}.$$

Направление спуска S_{n+1} будет сопряжено направлению S_n, когда выполняется тождество:

$$S_n^T \nabla^2 f(x_n) S_{n+1} = 0.$$

Как известно из математики [4], для положительно определённой матрицы её собственные вектора образуют систему из $n - 1$ направлений и такая система всегда существует, по крайней мере, одна.

Пусть исходное направление спуска (антиградиент) в некоторой точке X_n есть

$$S_n = - \nabla f(x_n),$$

а сопряжённое направление построено по алгоритму

$$S_{n+1} = - \nabla f(x_{n+1}) + \omega_{n+1} S_{n+1}, ,$$

где ω_{n+1} – весовой коэффициент сопряжения. Этот коэффициент должен обеспечивать условие сопряжённости, то есть выполняться (1).

Коэффициент сопряжения, как показано в [2, 9, 10], равен отношению квадратов норм (длин) векторов текущего и предшествующего градиентов:

$$\omega_{n+1} = \frac{\nabla^T f(x_{n+1}) \cdot \nabla f(x_{n+1})}{\nabla^T f(x_n) \cdot \nabla f(x_n)} = \frac{\left\| \nabla f(x_{n+1}) \right\|^2}{\left\| \nabla f(x_n) \right\|^2}.$$

Алгоритм расчётов, который назван по имени разработчиков алгоритмом Флетчера – Ривса, представлен ниже, на рисунке 1. Шаг λ^* отыскивается по процедуре, как и в методе наискорейшего спуска.

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Реализовать две версии алгоритма в части нахождения λ^*:

- с применением аналитического нахождения шага и
- с применением численного нахождения шага используя алгоритм одномерной минимизации `fminbnd`.

Коды представлены в приложении В.5.

2. Сравнить между собой по точности и по числу итераций результаты работы версии алгоритмов и результатов выполнения предыдущих лабораторных работ.

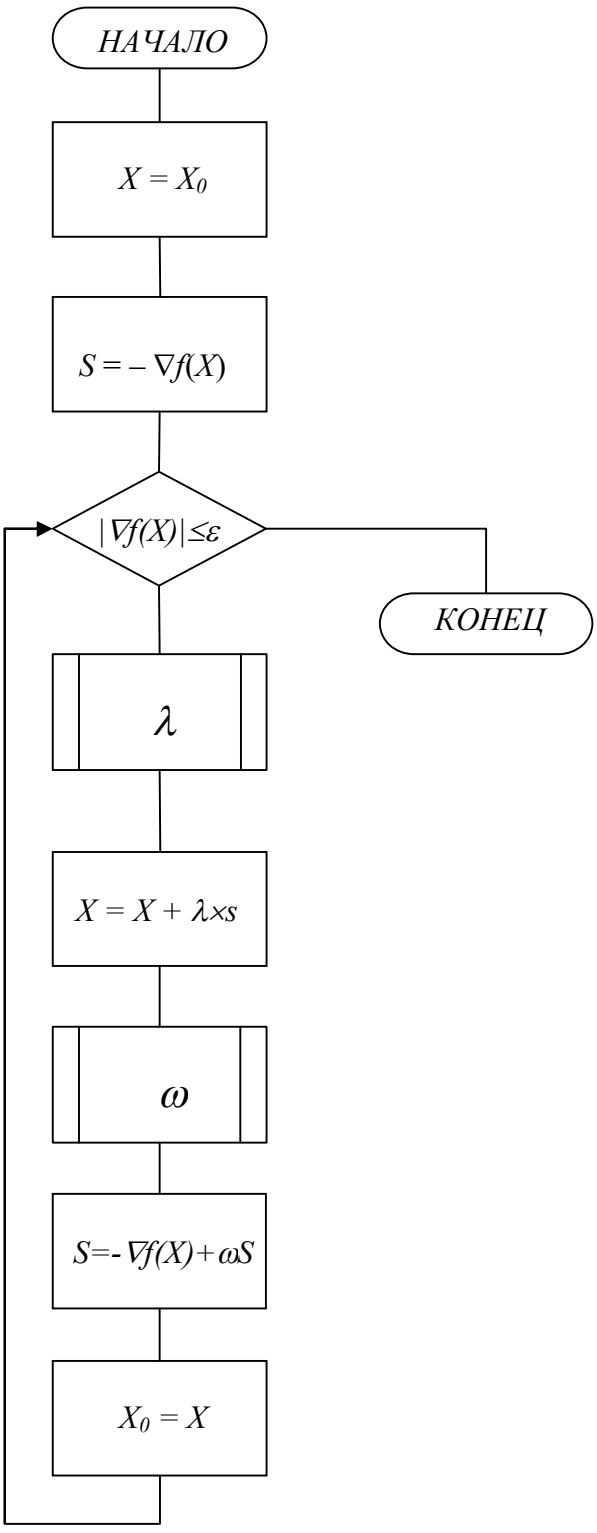

Рисунок 1 – Алгоритм метода Флетчера – Ривса

3. Оформить отчет с приложением результатов вычислений, необходимой графической информации. Сделать ***содержательный*** вывод, оценивающий полученные результаты.

4. Защитить результаты выполнения лабораторной работы.

4. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое сопряжённые направления?
2. Как называются векторы, если матрица, относительно которых они сопряжены нулевая с единицами по главной диагонали?
3. Каковы свойства градиента, используемые для отыскания минимальных и максимальных точек?
4. Всегда ли данный алгоритм способен отыскать точку оптимума?
5. Если у алгоритма склонность к зацикливанию?
6. Как Вы оцениваете ситуацию отсутствия градиента в некоторой текущей точке решения?
7. Что такое собственные векторы матрицы?
8. Для чего в задачах многомерной оптимизации применяются методы одномерной минимизации?
9. Может ли шаг λ оказаться равным нулю в ходе расчётов по мере спуска?
10. Что такое норма вектора?
11. Как рассчитать норму вектора?

ЛАБОРАТОРНАЯ РАБОТА № 5
ИССЛЕДОВАНИЕ АЛГОРИТМА РЕШЕНИЯ ЗАДАЧИ НЕЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ КВАЗИНЬЮТОНОВСКИМИ МЕТОДАМИ

... как бы ни была совершенна теория,
она только приближается к истине.
А.М. Бутлеров

1. ЦЕЛЬ РАБОТЫ

1. Освоить алгоритмы квазиньютоновскимих методов решения задач нелинейного программирования.
2. Закрепить способы применения функциq `fminunc` и `fminbnd` пакета `Optimization toolbox` среды `MatLab`.
3. Сравнить результаты с данными, полученными аналитически, для функций среды `MatLab` и рассчитанными в предыдущих работах.
4. Получить практические навыки в решении задач нелинейного программирования квазиньютоновскими методами.

2. ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

Квазиньютоновские методы или методы переменной метрики производят построение аппроксимирующей матрицы, являющейся обратной по отношению к матрице Гессе, в процессе спуска к оптимальному значению [2, 7, 8, 10].

Если алгоритм движения точки подчиняется закону

$$X_{n+1} = X_n - \lambda_n D_n \nabla f(X_n),$$

где D_n – аппроксимирующая матрица, λ_n – стабилизирующий множитель, а аппроксимирующую матрицу D_n строят по правилу:

$$H^{-1}(X_n) \approx D_{n+1} = \omega(D_n + \Delta D_n),$$

ω – нормирующий множитель, то приращение ΔD_n, обеспечивающее корректное построение аппроксимации матрицы Гёссе $[H(X_n)]^{-1}$, выглядит следующим образом

$$\Delta D_n = \frac{1}{\omega} \cdot \frac{\Delta X_n}{\Delta g_n^T} \cdot \frac{Y^T}{Y} - \frac{D_n \Delta g_n}{\Delta g_n^T} \cdot \frac{Z^T}{Z},$$

где $\Delta g_n = \nabla f(X_{n+1}) - \nabla f(X_n)$, $\Delta X_n = \Delta X_{n+1} - \Delta f(X_n)$.

Фиктивные добавки – векторы Y и Z, размерности n, решают задачи:

- определимости матричных операций:
- сходимости D_n к $H^{-1}(X_n)$,
- её положительной определённости.

Определение фиктивных добавок – самостоятельная задача, решением которой занимались многие исследватели.

В алгоритме, предложенном Бройденом (Broyden C.G.), положено:
$\omega = 1$, $Y = Z = \Delta X_n - D_n \Delta g_n$.

В алгоритме Давидона-Флетчера-Пауэлла (Davidon W.C., Fletcher R, Powell M.J.D) аналогичные добавки определяются как:
$\omega = 1$, $Y = \Delta X_n$, $Z = D_n \Delta g_n$.

В обоих случаях, в качестве начального приближения используется единичная матрица I.

Для последнего алгоритма

$$D_{n+1} = D_n + \frac{\Delta X_n}{\Delta g_n^T} \cdot \frac{\Delta X_n^T}{\Delta X_n} - \frac{D_n \Delta g_n}{\Delta g_n^T} \cdot \frac{(D_n \Delta g_n)^T}{D_n \Delta g_n} = D_n + A_n - B_n$$

или

$$D_{n+1} = I + \sum_{i=1}^{n} A_i - \sum_{i=1}^{n} B_i ,$$

в котором A_n обеспечивает сходимость D_n к $H^{-1}(x_n)$, а B_n – положительную определённость.

Схема алгоритма является типовой для градиентных методов.

Параметр $\lambda_n^* \Rightarrow \min_{\lambda>0} f(X_n + \lambda_n s_n)$, где $s_n = -\nabla f(X_n)$, вычисляется по методу, предложенному Коши для алгоритма наискорейшего спуска.

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Реализовать две версии алгоритма в части нахождения λ^*:

- с применением аналитического нахождения шага и
- с применением численного нахождения шага используя алгоритм одномерной минимизации `fminbnd`.

Коды представлены в приложении В.6.

2. Сравнить между собой по точности и по числу итераций результаты работы версии алгоритмов и результатов выполнения предыдущих лабораторных работ.

3. Оформить отчет с приложением результатов вычислений, необходимой графической информации. Сделать ***содержательный*** вывод, оценивающий полученные результаты.

4. Защитить результаты выполнения лабораторной работы.

4. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. В чём состоят достоинства и недостатки метода Ньютона?
2. В чём предназначение квазиньютоновских методов?
3. Почёму методы получили такое название?
4. Для чего необходимо обеспечивать сходимость аппроксимирующей матрицы?
5. Для чего необходимо обеспечивать положительную определённость аппроксимирующей матрицы?
6. Какова цель ввода векторных компонент Y и Z в формулу для расчёта ΔD_{n+1}?
7. Будет ли алгоритм функционировать, если λ положить константному значению, сэкономив на времени его вычисления?
8. Как по Вашему, насколько критичен алгоритм к выбору начального значения?

9. Почему в реализациях алгоритма нормирующий множитель $\omega = 1$?
10. Что будет, если нормирующий множитель будет не единичен?

ЛАБОРАТОРНАЯ РАБОТА № 6
ИССЛЕДОВАНИЕ АЛГОРИТМА РЕШЕНИЯ ЗАДАЧ НЕЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ МЕТОДОМ ЗОЙТЕНДЕЙКА

Наука лишь постольку наука, поскольку
в неё входит математика
А.М. Бутлеров

1. ЦЕЛЬ РАБОТЫ

1. Освоить алгоритмы Зойтендейка методов решения задач нелинейного программирования.
2. Освоить и закрепить способы применения функции `nonlcon`, `linprog`, `fmincon` и `fminbnd` пакета `Optimization toolbox` среды `MatLab`.
3. Сравнить результаты с данными, полученными аналитически, для функций среды `MatLab` и рассчитанными в предыдущих работах.
4. Получить практические навыки в решении задач нелинейного программирования квазиньютоновскими методами.

2. ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

Метод Зойтендейка (G. Zoutendijk) [3, 9, 10] базируется на понятии независимого направления.

Вектор с ненулевыми компонентами S называется *возможным направлением* спуска в точке $X \in \mathfrak{R}$, если существует $\delta > 0$, такое, что $X + \lambda S \in \mathfrak{R}$ для всех $\lambda \in (0, \delta)$ и $f(X + \lambda S) < f(X)$.

По условию задачи с ограничением строится эквивалентная задача линейного программирования.

Существуют три разновидности указанного метода: при линейных ограничениях; при нелинейных ограничениях и улучшенной сходимости при нелинейных ограничениях.

Для квадратичного программирования характерно использование линейных ограничений. Постановка задачи, в этом случае, такова

$$\min f(X)$$
$$\begin{cases} AX \le b, \\ HX = h, \end{cases}$$

где $f(X)$ – квадратична, $A[m \times n]$, $H[l \times n]$ – матрицы, а $b[m]$ и $h[l]$ – вектора системы линейных ограничений.

Пусть в текущей точке все ограничения со знаком "≤" представимы в виде

$$\begin{cases} A_1 X = b_1, \\ A_2 X < b_2. \end{cases}$$

Матрицы исходной системы подлежат разделению или перестановке по знакам отношений

$$\begin{cases} A^T = \begin{bmatrix} A_1^T & A_2^T \end{bmatrix}, \\ b^T = \begin{bmatrix} b_1^T & b_2^T \end{bmatrix}. \end{cases}$$

Вектор S будет являться направлением спуска в точке X при выполнении условий:

$$\begin{cases} A_1 S \le 0, \\ HS = 0. \end{cases}$$

На компоненты вектора S налагаются дополнительные ограничения (требования) нормировки:

- на величину элементов

$$-1 \le s_j \le 1, \ j = \overline{1,n};$$

- на модуль вектора возможного направления S

$$S^T S \le 1, \ \|S\|^2 \le 1,$$

это ограничения объединяет условия $-1 \le s_j \le 1, \ j = \overline{1,n}$ и

$$\sum_{j=1}^{n} s_i \le 1, \ \sum_{j=1}^{n} s_i \ge -1;$$

- на величину целевой функции ЗЛП

$$\nabla^T f(X)S \geq -1.$$

Поэтому возможна одна из трёх постановок задачи минимизации, которая может быть решена соответствующим методом линейного программирования:

$$\min \nabla^T f(X)S \quad \begin{cases} A_1 S \leq 0, \\ HS = 0, \\ s_j \leq 1, \\ s_j \geq -1, \ j = 1, m. \end{cases} \qquad 1)$$

$$\min \nabla^T f(X)S \quad \begin{cases} A_1 S \leq 0, \\ HS = 0, \\ S^T S \leq 1. \end{cases} \qquad 2)$$

$$\min \nabla^T f(X)S \quad \begin{cases} A_1 S \leq 0, \\ HS = 0, \\ \nabla^T f(X)S \geq -1. \end{cases} \qquad 3)$$

Предварительный этап.

Нахождение точки X, удовлетворяющей системе ограничений задачи.

Итерация.

1. Найти множество активных ограничений в текущей точке и композицию матрицы системы ограничений $A^T = \begin{bmatrix} A_1^T & A_2^T \end{bmatrix}$ и решить ЗЛП вида 1, 2 или 3, по желанию.

2. *Проверка условия окончания*.

Если оптимальное значение целевой функции ЗЛП равно нулю, ***то*** текущие координаты X определяют точку Куна-Таккера.

3. ***Иначе*** решить задачу одномерной минимизации Коши: $\min\limits_{0 \leq \lambda \leq \lambda_{\max}} f(X + \lambda S)$, в которой граничное значение параметра λ определяется как

$$\lambda_{\max} = \begin{cases} \min\limits_k \left\{ \dfrac{\tilde{b}_k}{\tilde{s}_k} \middle| \tilde{s}_k > 0 \right\}, & \exists_k \tilde{s}_k > 0, \\ \infty, & \forall_k \tilde{s}_k \leq 0. \end{cases} \quad \text{где } \tilde{b} = b_2 - A_2 X, \qquad \tilde{S} = A_2 S.$$

4. Перейти в новую текущую точку $X = X + \lambda \times S$.

Алгоритм не сложен, он представлен на рисунке 1.

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Реализовать две версии алгоритма в части нахождения λ^*:

- с применением аналитического нахождения шага и

- с применением численного нахождения шага используя алгоритм одномерной минимизации `fminbnd`.

Коды представлены в приложении В.7.

2. Для решения задачи линейного программирования воспользоваться процедурой `linprog`.

3. Выполнить расчёт тестового примера с использованием `fmincon`.

4. Для разделения ограничений на ограничения равенства и строгие неравенства применить `nonlcon`.

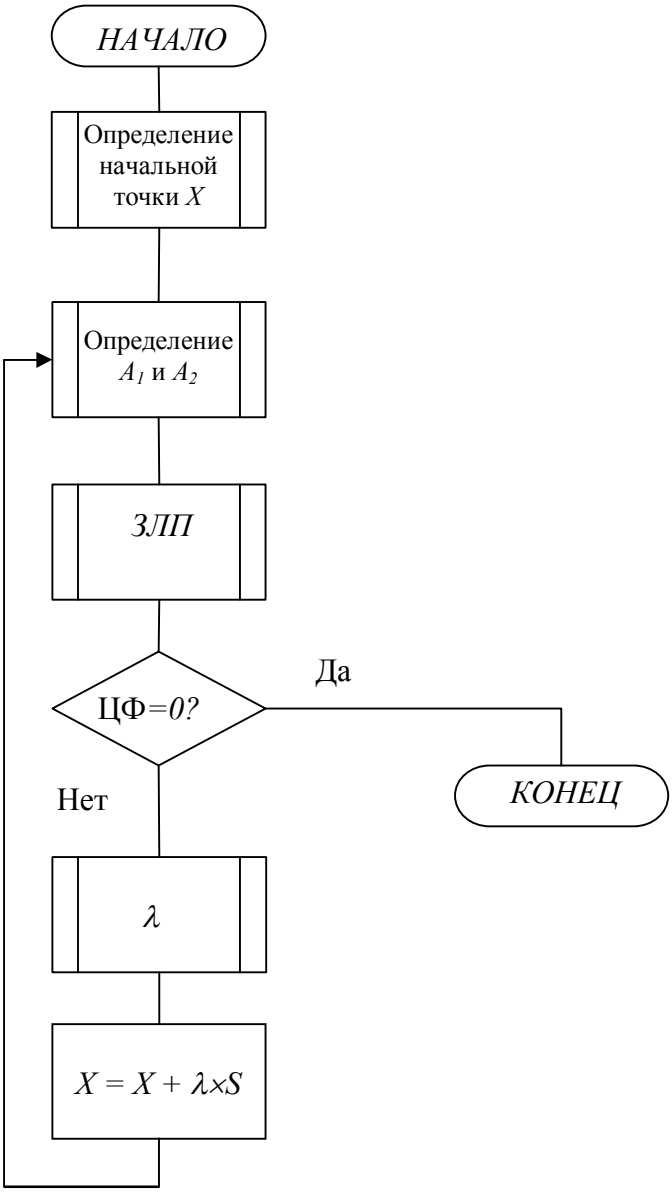

Рисунок 1 – Граф-схема алгоритма Зойтендейка

5. Сравнить между собой по точности и по числу итераций результаты работы версии алгоритмов и результатов выполнения предыдущих лабораторных работ.

6. Оформить отчет с приложением результатов вычислений, необходимой графической информации. Сделать ***содержательный*** вывод, оценивающий полученные результаты.

7. Защитить результаты выполнения лабораторной работы.

4. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какое направление называется направлением возможного спуска?
2. Что такое активное ограничение?
3. О чём говорит неразрешимость построенной эквивалентной задачи линейного программирования?
4. Зачем вычислять параметр λ?
5. Что реально обозначают условия, используемые в эквивалентной задаче линейного программирования?
6. Что понимается под термином точка Куна-Таккера?
7. Как Вы думаете, возможна ли неразрешимость задачи квадратичного программирования?
8. Что служит причиной неразрешимости задачи квадратичного программирования?
9. Возможна ли безрезультативная работа алгоритма метода?
10. Поменяется ли алгоритм, если из постановки задачи убрать ограничения вида «равенство»?

ЛАБОРАТОРНАЯ РАБОТА № 7
ИССЛЕДОВАНИЕ АЛГОРИТМА РЕШЕНИЯ ЗАДАЧ НЕЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ МЕТОДОМ РОЗЕНА

Без светоча науки и
с нефтью будут потёмки
Д.И. Менделеев

1. ЦЕЛЬ РАБОТЫ

1. Освоить алгоритм Розена решения задач нелинейного программирования.
2. Освоить и закрепить способы применения функции `nonlcon`, `linprog`, `fmincon` и `fminbnd` пакета `Optimization toolbox` среды `MatLab`.

3. Сравнить результаты с данными, полученными аналитически, для функций среды `MatLab` и рассчитанными в предыдущих работах.

4. Получить практические навыки в решении задач нелинейного программирования квазиньютоновскими методами.

2. ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

Метод Розена предназначен для решения задач нелинейного программирования с линейной системой ограничений [2, 8, 9]

$$\min f(X)$$
$$\begin{cases} AX \le b, \\ HX = h, \end{cases}$$

где $f(X)$ – нелинейная и ***дифференцируемая*** функция, A – матрица размером $[m \times n]$, H – $[l \times n]$-мерная матрица , а b – m-мерный вектор, а h – l-мерный вектор системы линейных ограничений.

Розен предложил направление спуска строить по правилу

$$S = - P \times \nabla f(X),$$

где P – матрица проецирования или проецирующая (проектирующая) матрица, которая бы гарантировала сохранение допустимости текущей точки.

Пусть в допустимой точке часть неравенств активна, а другая часть – нет, то допускается представление системы ограничений по признаку активности в виде блочной матрицы:

$$\begin{aligned} &A_1 X = b_1, \quad A^T = \begin{bmatrix} A_1^T & A_2^T \end{bmatrix}, \\ &A_2 X < b_2, \quad b^T = \begin{bmatrix} b_1^T & b_2^T \end{bmatrix}. \end{aligned} \tag{1}$$

Матрица проецирования P должна удовлетворять условию $P \times \nabla f(X) \neq 0$ и блокировать возможные направления в сторону активных ограничений. Это будет соблюдаться, когда проектирующая матрица определена как

$$P = I - M^T \times (M \times M^T)^{-1} \times M, \tag{2}$$

где I – единичная матрица, $M^T = [A_1{}^T H^T]$ – невырождена. Отметим, что выполняется тождество $M \times P = 0$, то есть $A_1 \times P = 0$ и $H \times P = 0$, таким образом, спуск в направлении ограничений, выполняемых как равенство, не производится.

Рассмотрим точку, в которой выполняется условие $P \times \nabla f(X) = 0$. Имеем

$$[I - M^T \times (M \times M^T)^{-1} \times M] \times \nabla f(X) = \nabla f(X) + M^T \times W = 0,$$

где $W^T = [U^T V^T]$, а

$$\nabla f(X) + A_1{}^T \times U + H^T \times V = 0.$$

Если компоненты вектора U неотрицательны, то X является точкой Куна-Таккера. В противном случае, можно построить новое направление спуска

$$S = -\widetilde{P} \times \nabla f(X),$$

где $\widetilde{P} = I - \widetilde{M}^T \times (\widetilde{M} \times \widetilde{M}^T)^{-1} \times \widetilde{M}$, в котором $\widetilde{M}^T = [\widetilde{A}_1^T H^T]$, а $\widetilde{A}_1$ получено из A_1 путём вычёркивания строк, соответствующих компонентам $u_j < 0$.

Схема алгоритма показана на рисунке 1.

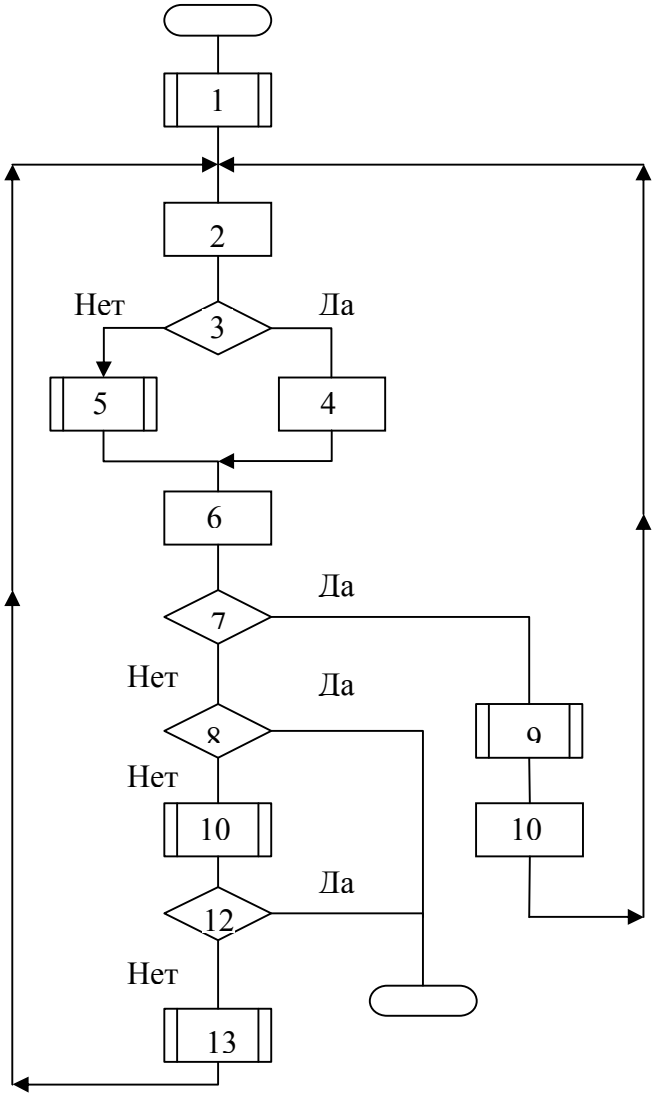

Рисунок 1 – Граф-схема алгоритма Розена

Блок № 1 выполняет выбор начальной точки X, удовлетворяющей системе ограничений задачи, отыскивает активные ограничения и разложение ограничений с неравенствами (1).

В блоке № 2 конструируется матрица $M^T = [A_1{}^T H^T]$, которая проверяется на равенство нулю всех её элементов условием блока № 3: $M = 0?$, если матрица вся нулевая (выход «да» блока № 3), то проецирующая матрица полагается равной единичной (блок № 4).

В противном случае (выход «нет» блока № 3), выполняется расчёт проецирующей матрицы в блоке № 5 по формуле (2).

После определения элементов проецирующей матрицы, рассчитывается возможное направление спуска $S = -P \times \nabla f(X)$ в блоке № 6.

Блок № 7 выполняет проверку: $S \neq 0?$, то есть, не является ли текущая точка точкой Куна-Таккера.

Если все компоненты вектора возможного направления нулевые (выход «нет» блока № 7), то это, возможно, искомое решение. Чтобы убедиться в этом, проверяется равенство нулю всех компонентов блочной матрицы M (блок № 8, $M = 0?$), и если это так (выход «да» блока № 8), то решение успешно заканчивается.

Если среди компонентов матрицы M есть ненулевые элементы (выход «нет» блока № 8), то ищется разложение $W^T = [U^T V^T]$, блок № 10.

Если компоненты подматрицы U неотрицательны, что проверяется блоком № 12 ($\forall_i u_i \geq 0$), то X является точкой Куна-Таккера (выход «да»).

Если компоненты U отрицательны (выход «нет» блока № 12), то переопределяется матрицы $A_1 = \widetilde{A}_1$ и блочная матрица $\widetilde{M}^T = [\widetilde{A}_1^T H^T]$ (блок № 13).

Если не все элементы вектора возможного направления нулевые (выход «да» блока № 7), рассчитывается новая текущая точка. Сперва выбирается параметр λ решением задачи одномерной минимизации Коши (блок № 9): $\min\limits_{0 \leq \lambda \leq \lambda_{\max}} f(X + \lambda S)$, в которой граничное значение параметра λ определяется, как и в методе Зойтендейка:

$$\lambda_{\max} = \begin{cases} \min\limits_k \left\{ \dfrac{\widetilde{b}_k}{\widetilde{s}_k} \middle| \widetilde{s}_k > 0 \right\}, & \exists_k \widetilde{s}_k > 0, \\ \infty, & \forall_k \widetilde{s}_k \leq 0. \end{cases} \quad \text{где } \widetilde{b} = b_2 - A_2 X, \qquad \widetilde{S} = A_2 S .$$

Значение очередной текущей точки рассчитывается в блоке № 11 по рекуррентной формуле $X = X + \lambda \times S$.

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Реализовать две версии алгоритма в части нахождения λ^*:

- с применением аналитического нахождения шага и
- с применением численного нахождения шага используя алгоритм одномерной минимизации `fminbnd`.

Коды представлены в приложении В.8.

2. Для решения задачи линейного программирования воспользоваться процедурой `linprog`.

3. Выполнить расчёт тестового примера с использованием `fmincon`.

4. Для разделения ограничений на ограничения равенства и строгие неравенства применить `nonlcon`.

5. Сравнить между собой по точности и по числу итераций результаты работы версии алгоритмов и результатов выполнения предыдущих лабораторных работ.

6. Оформить отчет с приложением результатов вычислений, необходимой графической информации. Сделать ***содержательный*** вывод, оценивающий полученные результаты.

7. Защитить результаты выполнения лабораторной работы.

4. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Каково назначение проектирующей матрицы?

2. Из каких соображений строится проектирующая матрица?

3. О чём свидетельствует нулевое значение компонентов вектора направления после проецирования?

4. Будет ли получен результат, если начальное приближение недопустимая точка?

5. Можно ли применять сей алгоритм, если все ограничения системы представляются в виде равенств?

6. Каков смысл и геометрическая интерпретация ситуации $P \cdot \nabla f(X_k) = 0$?

7. Каков смысл и геометрическая интерпретация ситуации $P \cdot \nabla f(X_k) = \nabla f(X_k)$?

8. Каков смысл и геометрическая интерпретация ситуации $P \cdot \nabla f(X_k) \neq 0$?

9. Что будет, если в некой точке решения все неравенства окажутся активными?

10. Что будет, если в некой точке решения не будет ни одного активного неравенства?

ЛАБОРАТОРНАЯ РАБОТА № 8
ИССЛЕДОВАНИЕ АЛГОРИТМА РЕШЕНИЯ ЗАДАЧ НЕЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ МЕТОДОМ ШТРАФНЫХ ФУНКЦИЙ ВНУТРЕННЕЙ ТОЧКИ

... математика самый короткий путь
к самостоятельному мышлению
В.А. Каверин

1. ЦЕЛЬ РАБОТЫ

1. Освоить алгоритмы методов внутренней точки решения задач нелинейного программирования.
2. Закрепить способы применения функциq `fminunc` и `fseminf` пакета `Optimization toolbox` среды `MatLab`.
3. Сравнить результаты с данными, полученными аналитически, для функций среды `MatLab` и рассчитанными в предыдущих работах.
4. Получить практические навыки в решении задач методами штрафных функций.

2. ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

Методы штрафных функций основаны на практике перехода от задачи *условной минимизации* к задаче *безусловной минимизации* путём построения специальной штрафной функции[9, 10].

$$P(X,\rho) = f(X) + \sum_{i=1}^{m} \rho_i H[h_i(X)] + \sum_{i=m+1}^{l} \rho_i G[g_i(X)],$$

где $P(X,\rho)$ – штрафная функция; ρ_i, $i = \overline{1,l}$ – весовые коэффициенты, отражающие значимость соблюдения того или иного ограничения; $H[h_i(X)]$ и $G[g_i(X)]$ – некоторые функционалы.

Функционал $H[h_i(X)]$ должен сделать невыгодным любое отклонение аргумента X от поверхности $h_i(X)=0$, то есть

$$\lim_{h_i(X)\to 0} H[h_i(X)] = 0, \qquad i = \overline{1,m}.$$

Поэтому в качестве функционала выбирают чётную степенную функцию

$$H[y] = y^r,\ r = 2, 4, \ldots \text{ либо } H[y] = |y|^r,\ r = 1, 2, \ldots$$

Для метода внутренней точки функционал таков:

$$\lim_{g_i(X)\to 0^+} G[g_i(X)] = \infty, \qquad i = m+1, \bar{l} \text{ для } g_i > 0.$$

Алгоритм (рисунок 1) относится к группе методов внутренней точки, используется для решения задач вида

$$\begin{cases} \min f(X); \\ g_i(X) \ge 0, & i = 1, \overline{m}; \\ x_j \ge 0, & j = 1, \overline{n}. \end{cases}$$

По условиям задачи строится штрафная функция вида

$$P(X, r, \Omega) = f(X) + r \cdot \sum_{i=1}^{m} \omega_i G[g_i(X)]. \qquad (3)$$

В формуле (3) использованы: параметр $r > 0$, убывающий по мере вычислений и Ω – вектор положительных весовых коэффициентов, учитывающих важность (значимость) ограничений модели.

Функции $G[g_i(X)]$ выбирается из альтернатив

$$G[Y] = Y^{-1} \text{ или } G[Y] = -\ln Y^{-1}.$$

Легко видеть, что $\lim_{Y\to 0^+} G[Y] = \infty$, то есть функция барьера, при приближении к нему изнутри области, неограниченно возрастает. Штрафная добавка к функции цели определяется формулой

$$\Delta = r \cdot \sum_{i=1}^{m} \omega_i G[g_i(X)]$$

Функционирование алгоритма представлено на рисунке 1, в его работе использована уменьшающая константа $0 < \beta < 1$, точность расчётов задаётся константой ε.

Замечания:

- область ограничений должна быть непустой: $X_{\text{opt}} \in \{X: g_i(X) > 0,\ i = 1, m\}$;
- начальное значение X должно быть допустимой точкой;

- метод критичен по отношению к параметру r: если значение r мало, то будет отыскан оптимум функции, не совпадающий с истинным, а если значение r велико, то возрастёт число итераций;
- аналогично, близкие к единице значения β могут привести к преждевременному прекращению вычислений.

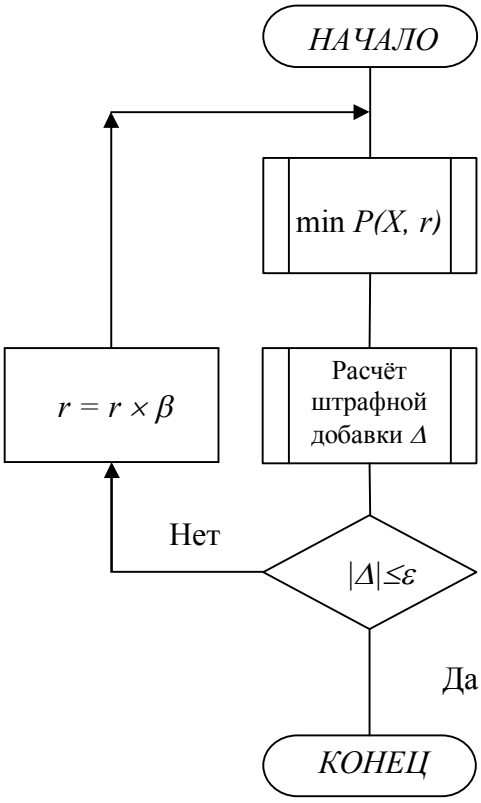

Рисунок 1 – Схема алгоритма МБП

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. По условию варианта составить штрафную функцию.

2. Реализовать алгоритм расчёта, для проведения минимизации на каждом шаге воспользоваться процедурой `fminunc` или `fseminf`. Коды представлены в приложении В.9.

3. Сравнить между собой по точности и по числу итераций результаты работы версии алгоритмов и результатов выполнения предыдущих лабораторных работ.

6. Оформить отчет с приложением результатов вычислений, необходимой графической информации. Сделать ***содержательный*** вывод, оценивающий полученные результаты.

7. Защитить результаты выполнения лабораторной работы.

4. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Почему метод называется методом “внутренней точки”?
2. Что вкладывается в понятие функции барьера?
3. Для чего введены множители ω_i в записи штрафной функции?
4. Каков назначение множителя r в записи штрафной функции метода внутренней точки?
5. Почему множитель β следует выбирать меньше единицы?
6. Каким требованиям должен удовлетворять функционал G в методах внутренней точки?
7. Насколько критичен алгоритм метода к выбору начальной точки?
8. Всегда ли задача разрешима данным методом?
9. Что произойдёт, если в области ограничений будет сразу несколько оптимальных решений?
10. Повлияют ли значения множителей ω_i на найденное оптимальное решение?

ЛАБОРАТОРНАЯ РАБОТА № 9. ИССЛЕДОВАНИЕ АЛГОРИТМА РЕШЕНИЯ ЗАДАЧ НЕЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ МЕТОДОМ ШТРАФНЫХ ФУНКЦИЙ ВНЕШНЕЙ ТОЧКИ

Мыслю – следовательно, существую.
Р. Декарт

1. ЦЕЛЬ РАБОТЫ

1. Освоить алгоритмы методов внешней точки решения задач нелинейного программирования.
2. Закрепить способы применения функций `fminunc` и `fseminf` пакета `Optimization toolbox` среды `MatLab`.
3. Сравнить результаты с данными, полученными аналитически, для функций среды `MatLab` и рассчитанными в предыдущих работах.
4. Получить практические навыки в решении задач методами штрафных функций.

2. ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

Метод внешней точки ориентируется на решение НП-задач в общей постановке [2, 9, 10]

$$\min f(X);$$
$$\begin{cases} h_i(X) = 0, & i = \overline{1,m}; \\ g_i(X) \geq 0, & i = \overline{m+1,l}. \end{cases}$$

Штрафная функция описывается выражением

$$P(X,r) = f(X) + r\mathrm{L}(X),$$

где r – коэффициент штрафа, а величина штрафа в текущей точке X есть

$$L(X) = \sum_{i=1}^{m} H[h_i(X)] + \sum_{i=m+1}^{l} G[g_i(X)].$$

Используются функционалы:

$$G[Y] = [\max\{0; -Y\}]^P,\ P > 0, \text{ целое},$$
$$H[Y] = |\mathrm{Y}|^P,\ P > 0, \text{ целое}.$$

Легко видеть, что функционал $G[Y]$ обращается в нуль при попадании текущей точки в область ограничений.

Алгоритм метода полностью дублирует алгоритм метода барьерных поверхностей, рассмотренный в предыдущей лабораторной работе, но, в отличие от него, используется параметр увеличения штрафа β больше единицы.

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. По условию варианта составить штрафную функцию.

2. Реализовать алгоритм расчёта, для проведения минимизации на каждом шаге воспользоваться процедурой `fminunc` или `fseminf`.
Коды представлены в приложении В.10.

3. Сравнить между собой по точности и по числу итераций результаты работы версии алгоритмов и результатов выполнения предыдущих лабораторных работ.

6. Оформить отчет с приложением результатов вычислений, необходимой графической информации. Сделать ***содержательный*** вывод, оценивающий полученные результаты.

7. Защитить результаты выполнения лабораторной работы.

4. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Почему метод называется методом “внешней точки”?
2. За счёт чего будет происходить минимизация при возрастании r?
3. Каким требованиям должен удовлетворять функционал G в методах внешней точки?
4. Каким требованиям должен удовлетворять функционал H в методах штрафных функций?
5. Почему множитель β следует выбирать много больше единицы?
6. Каков назначение множителя r в записи штрафной функции метода внешней точки?
7. Обязательно ли начальное приближение является недопустимой точкой?
8. Что произойдёт, если область ограничений пуста?
9. Всегда ли достижимо оптимальное решение при использовании указанного метода?
10. Может ли целевая функция в исходной постановке задачи определяться кусочно-непрерывно?

ЗАКЛЮЧЕНИЕ

“Аппарат нелинейного программирования оказывается весьма полезным в автоматическом регулировании, космических исследованиях, строительной механике и многих других областях человеческой деятельности” [12]

“Специалист, желающий решить свою практическую задачу с помощью математических методов оптимизации, должен располагать некоторым запасом реализуемых моделей, среди которых может выбирать … адекватную модель” [9].

“… а математику уже затем учить надо, что она ум в порядок приводит” [М.В. Ломоносов].

Заручившись поддержкой сонма столь выдающихся авторитетов, повествование сие завершаем, в пику персонажу М.Е. Салтыкова-Щедрина, архистратигу Перехват-Залихватскому, коей въехал в г. Глупов на белом коне, сжёг гимназию и упразднил науки. На Святой Руси имя подобным

деятелям – легион. А “…дорога из города Глупова в город Умнов лежит через город Буянов, а не через манную кашу…”. Михаил Евграфович знал, о чём писал.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Банди Б. Методы оптимизации. Вводный курс / Б. Банди. - М. : Радио и Связь, 1988. – 128 с.
2. Бейко И. В. Методы и алгоритмы решения задач оптимизации / И. В. Бейко, Б. Н. Бублик, П. Н. Зинько. – Киев : Вища школа, 1983. - 512 с.
3. Введение в нелинейное программирование / под. ред. К.-Х. Эльстера. – М. : Наука, 1985. – 263 с.
4. Гантмахер Ф. Р. Теория матриц / Ф. Р. Гантмахер, М. : Наука, ГРФМЛ, 1988. – 552 с.
5. Зайченко Ю. П. Исследование операций. : учебное пособие. / Ю. П. Зайченко. – Киев : Вища школа, 1979. -392 с.
6. Кюнци Г. П. Нелинейное программирование / Г. П. Кюнци, В. Крелле. - М. : Советское радио, 1965. - 303 с.
7. Фиакко А. В. Нелинейное программирование. Методы последовательной безусловной оптимизации / А. В. Фиакко, Г. П. Мак-Кормик –М. : Мир, 1972. – 240 с.
8. Химмельблау Д. Прикладное нелинейное программирование / Д. Химмельблау. - М. : Мир, 1975. – 534 с.

Электронные издания

9. Ашманов, С. А. Теория оптимизации в задачах и упражнениях : учебное пособие / С. А. Ашманов, А. В. Тимохов. — 2-е изд., стер. — Санкт-Петербург : Лань, 2022. — 448 с. — URL: https://e.lanbook.com/book/210911 (дата обращения: 09.04.2024). — Режим доступа: для авториз. пользователей. — ISBN 978-5-8114-1366-9. — Текст : электронный.
10. Гончаров, В. А. Методы оптимизации : учебное пособие для вузов / В. А. Гончаров. — Москва : Юрайт, 2024. — 191 с. — URL: https://urait.ru/bcode/534423 (дата обращения: 09.04.2024). — (Высшее образование). — ISBN 978-5-9916-3642-1. — Текст : электронный.
11. Горлач, Б. А. Исследование операций : учебное пособие / Б. А. Горлач. — Санкт-Петербург : Лань, 2022. — 448 с. — URL: https://e.lanbook.com/book/211085 (дата обращения: 09.04.2024). — Режим доступа: для авториз. пользователей. — ISBN 978-5-8114-1430-7. — Текст : электронный.

12. Есипов, Б. А. Методы исследования операций : учебное пособие / Б. А. Есипов. — 2-е изд., испр. и доп. — Санкт-Петербург : Лань, 2022. — 304 с. — URL: https://e.lanbook.com/book/212204 (дата обращения: 09.04.2024). — Режим доступа: для авториз. пользователей. — ISBN 978-5-8114-0917-4. — Текст : электронный.
13. Кудрявцев, К. Я. Методы оптимизации : учебное пособие для вузов / К. Я. Кудрявцев, А. М. Прудников. — 2-е изд. — Москва : Юрайт, 2024. — 140 с. — URL: https://urait.ru/bcode/541315 (дата обращения: 09.04.2024). — (Высшее образование). — ISBN 978-5-534-08523-5. — Текст : электронный.
14. Сухарев, А. Г. Методы оптимизации : учебник и практикум для бакалавриата и магистратуры / А. Г. Сухарев, А. В. Тимохов, В. В. Федоров. — 3-е изд., испр. и доп. — Москва : Юрайт, 2022. — 367 с. — URL: https://urait.ru/bcode/507818 (дата обращения: 09.04.2024). — (Бакалавр и магистр. Академический курс). — ISBN 978-5-9916-3859-3. — Текст : электронный.
15. Токарев, В. В. Методы оптимизации : учебное пособие для вузов / В. В. Токарев. — Москва : Юрайт, 2024. — 440 с. — URL: https://urait.ru/bcode/539567 (дата обращения: 09.04.2024). — (Высшее образование). — ISBN 978-5-534-04712-7. — Текст : электронный.
16. Методы оптимизации: теория и алгоритмы : учебное пособие для вузов / А. А. Черняк, Ж. А. Черняк, Ю. М. Метельский, С. А. Богданович. — 2-е изд., испр. и доп. — Москва : Юрайт, 2024. — 357 с. — URL: https://urait.ru/bcode/539155 (дата обращения: 09.04.2024). — (Высшее образование). — ISBN 978-5-534-04103-3. — Текст : электронный.
17. Колбин, В. В. Специальные методы оптимизации : учебное пособие / В. В. Колбин. — Санкт-Петербург : Лань, 2022. — 384 с. . — URL: https://e.lanbook.com/book/211448 (дата обращения: 09.04.2024). — Режим доступа: для авториз. пользователей. — ISBN 978-5-8114-1536-6. — Текст : электронный.

ПРИЛОЖЕНИЕ А
(обязательное)

ВАРИАНТЫ СИСТЕМ ОГРАНИЧЕНИЙ ДЛЯ ИЗУЧЕНИЯ АЛГОРИТМОВ РЕШЕНИЯ ЗАДАЧ НЕЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ

00.
$$\begin{cases} 6\cdot x_1 + 3\cdot x_2 \le 18; \\ 3\cdot x_1 + 5\cdot x_2 \le 15; \\ 2\cdot x_1 + 2\cdot x_2 \ge 4; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

01.
$$\begin{cases} 7\cdot x_1 + 5\cdot x_2 \le 35; \\ 3\cdot x_1 + 6\cdot x_2 \le 18; \\ 7\cdot x_1 + 2\cdot x_2 \ge 14; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

02.
$$\begin{cases} 4\cdot x_1 + 5\cdot x_2 \le 20; \\ 3\cdot x_1 + 7\cdot x_2 \le 21; \\ 5\cdot x_1 + 3\cdot x_2 \ge 15; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

03.
$$\begin{cases} 7\cdot x_1 + 6\cdot x_2 \le 42; \\ 4\cdot x_1 + 6\cdot x_2 \le 24; \\ 5\cdot x_1 + 3\cdot x_2 \ge 15; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

04.
$$\begin{cases} 6\cdot x_1 + 3\cdot x_2 \le 18; \\ 7\cdot x_1 + 2\cdot x_2 \ge 14; \\ 3\cdot x_1 + 3\cdot x_2 \ge 9; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

05.
$$\begin{cases} 1\cdot x_1 + 6\cdot x_2 \le 6; \\ 6\cdot x_1 + 4\cdot x_2 \le 24; \\ 2\cdot x_1 + 2\cdot x_2 \ge 4; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

06.
$$\begin{cases} 5\cdot x_1 + 2\cdot x_2 \le 10; \\ 2\cdot x_1 + 4\cdot x_2 \le 8; \\ 4\cdot x_1 + 1\cdot x_2 \ge 4; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

07.
$$\begin{cases} 5\cdot x_1 + 7\cdot x_2 \le 35; \\ 7\cdot x_1 + 2\cdot x_2 \le 14; \\ 3\cdot x_1 + 4\cdot x_2 \ge 12; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

08.
$$\begin{cases} 5\cdot x_1 + 6\cdot x_2 \le 30; \\ 4\cdot x_1 + 6\cdot x_2 \le 24; \\ 5\cdot x_1 + 3\cdot x_2 \ge 15; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

09.
$$\begin{cases} 6\cdot x_1 + 5\cdot x_2 \le 30; \\ 2\cdot x_1 + 6\cdot x_2 \le 12; \\ 5\cdot x_1 + 2\cdot x_2 \ge 10; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

10.
$$\begin{cases} 5\cdot x_1 + 5\cdot x_2 \le 25; \\ 7\cdot x_1 + 3\cdot x_2 \le 21; \\ 3\cdot x_1 + 2\cdot x_2 \ge 6; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

11.
$$\begin{cases} 6\cdot x_1 + 4\cdot x_2 \le 24; \\ 3\cdot x_1 + 5\cdot x_2 \le 15; \\ 4\cdot x_1 + 3\cdot x_2 \ge 12; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

12.
$$\begin{cases} 6\cdot x_1 + 4\cdot x_2 \le 24; \\ 3\cdot x_1 + 7\cdot x_2 \le 21; \\ 4\cdot x_1 + 3\cdot x_2 \ge 12; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

13.
$$\begin{cases} 4\cdot x_1 + 3\cdot x_2 \ge 12; \\ 2\cdot x_1 + 7\cdot x_2 \le 14; \\ 5\cdot x_1 + 1\cdot x_2 \ge 5; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

14.
$$\begin{cases} 1\cdot x_1 + 2\cdot x_2 \le 6; \\ 7\cdot x_1 + 3\cdot x_2 \le 21; \\ 1\cdot x_1 + 1\cdot x_2 \ge 3; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

15.
$$\begin{cases} 1\cdot x_1 + 7\cdot x_2 \le 7; \\ 6\cdot x_1 + 3\cdot x_2 \le 18; \\ 1\cdot x_1 + 1\cdot x_2 \ge 1; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

16.
$$\begin{cases} 5\cdot x_1 + 2\cdot x_2 \le 10; \\ 2\cdot x_1 + 6\cdot x_2 \le 12; \\ 1\cdot x_1 + 1\cdot x_2 \ge 1; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

17.
$$\begin{cases} 4\cdot x_1 + 7\cdot x_2 \le 28; \\ 5\cdot x_1 + 3\cdot x_2 \le 15; \\ 3\cdot x_1 + 1\cdot x_2 \ge 3; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

18.
$$\begin{cases} 3\cdot x_1 + 3\cdot x_2 \le 9; \\ 7\cdot x_1 + 1\cdot x_2 \le 7; \\ 1\cdot x_1 + 7\cdot x_2 \ge 7; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

19.
$$\begin{cases} 4\cdot x_1 + 5\cdot x_2 \le 20; \\ 2\cdot x_1 + 1\cdot x_2 \le 6; \\ 1\cdot x_1 + 3\cdot x_2 \ge 6; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

20.
$$\begin{cases} 2\cdot x_1 + 7\cdot x_2 \le 14; \\ 7\cdot x_1 + 4\cdot x_2 \le 28; \\ 7\cdot x_1 + 1\cdot x_2 \ge 7; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

21.
$$\begin{cases} 3\cdot x_1 + 7\cdot x_2 \le 21; \\ 7\cdot x_1 + 3\cdot x_2 \le 21; \\ 4\cdot x_1 + 3\cdot x_2 \ge 12; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

22.
$$\begin{cases} 6\cdot x_1 + 3\cdot x_2 \le 18; \\ 3\cdot x_1 + 7\cdot x_2 \le 21; \\ 2\cdot x_1 + 1\cdot x_2 \ge 2; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

23.
$$\begin{cases} 7\cdot x_1 + 4\cdot x_2 \le 28; \\ 3\cdot x_1 + 6\cdot x_2 \le 18; \\ 4\cdot x_1 + 2\cdot x_2 \ge 6; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

24.
$$\begin{cases} 5\cdot x_1 + 5\cdot x_2 \le 25; \\ 5\cdot x_1 + 3\cdot x_2 \le 15; \\ 4\cdot x_1 + 6\cdot x_2 \ge 24; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

25.
$$\begin{cases} 2\cdot x_1 + 7\cdot x_2 \le 14; \\ 3\cdot x_1 + 2\cdot x_2 \le 6; \\ 3\cdot x_1 + 1\cdot x_2 \ge 3; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

26.
$$\begin{cases} 6\cdot x_1 + 4\cdot x_2 \le 24; \\ 4\cdot x_1 + 5\cdot x_2 \le 20; \\ 4\cdot x_1 + 3\cdot x_2 \ge 12; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

27.
$$\begin{cases} 3\cdot x_1 + 5\cdot x_2 \le 15; \\ 5\cdot x_1 + 4\cdot x_2 \le 20; \\ 4\cdot x_1 + 2\cdot x_2 \ge 8; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

28.
$$\begin{cases} 7\cdot x_1 + 6\cdot x_2 \le 42; \\ 2\cdot x_1 + 7\cdot x_2 \le 14; \\ 5\cdot x_1 + 2\cdot x_2 \ge 10; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

29.
$$\begin{cases} 4\cdot x_1 + 4\cdot x_2 \le 16; \\ 7\cdot x_1 + 1\cdot x_2 \le 7; \\ 1\cdot x_1 + 1\cdot x_2 \ge 2; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

30.
$$\begin{cases} 1\cdot x_1 + 7\cdot x_2 \le 7; \\ 6\cdot x_1 + 2\cdot x_2 \le 12; \\ 3\cdot x_1 + 2\cdot x_2 \ge 6; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

31.
$$\begin{cases} 6\cdot x_1 + 2\cdot x_2 \le 12; \\ 3\cdot x_1 + 4\cdot x_2 \le 12; \\ 2\cdot x_1 + 1\cdot x_2 \ge 2; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

32.
$$\begin{cases} 5\cdot x_1 + 5\cdot x_2 \le 25; \\ 3\cdot x_1 + 7\cdot x_2 \le 21; \\ 3\cdot x_1 + 1\cdot x_2 \ge 3; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

33.
$$\begin{cases} 6\cdot x_1 + 5\cdot x_2 \le 30; \\ 5\cdot x_1 + 4\cdot x_2 \le 20; \\ 4\cdot x_1 + 5\cdot x_2 \ge 20; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

34.
$$\begin{cases} 5\cdot x_1 + 1\cdot x_2 \le 5; \\ 3\cdot x_1 + 5\cdot x_2 \le 15; \\ 1\cdot x_1 + 2\cdot x_2 \ge 2; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

35.
$$\begin{cases} 1\cdot x_1 + 6\cdot x_2 \le 6; \\ 3\cdot x_1 + 3\cdot x_2 \le 9; \\ 7\cdot x_1 + 1\cdot x_2 \ge 7; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

36.
$$\begin{cases} 3\cdot x_1 + 6\cdot x_2 \le 18; \\ 6\cdot x_1 + 4\cdot x_2 \le 24; \\ 4\cdot x_1 + 2\cdot x_2 \ge 8; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

37.
$$\begin{cases} 5\cdot x_1 + 3\cdot x_2 \le 15; \\ 4\cdot x_1 + 6\cdot x_2 \le 24; \\ 2\cdot x_1 + 1\cdot x_2 \ge 4; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

38.
$$\begin{cases} 1\cdot x_1 + 7\cdot x_2 \le 7; \\ 3\cdot x_1 + 4\cdot x_2 \le 12; \\ 4\cdot x_1 + 1\cdot x_2 \ge 4; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

39.
$$\begin{cases} 5\cdot x_1 + 4\cdot x_2 \le 20; \\ 4\cdot x_1 + 6\cdot x_2 \le 24; \\ 6\cdot x_1 + 2\cdot x_2 \ge 12; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

40.
$$\begin{cases} 3\cdot x_1 + 7\cdot x_2 \le 21; \\ 4\cdot x_1 + 4\cdot x_2 \le 16; \\ 5\cdot x_1 + 1\cdot x_2 \ge 5; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

41.
$$\begin{cases} 5\cdot x_1 + 7\cdot x_2 \le 35; \\ 8\cdot x_1 + 3\cdot x_2 \le 24; \\ 2\cdot x_1 + 1\cdot x_2 \ge 4; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

42.
$$\begin{cases} 7\cdot x_1 + 4\cdot x_2 \le 28; \\ 2\cdot x_1 + 7\cdot x_2 \le 14; \\ 5\cdot x_1 + 1\cdot x_2 \ge 5; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

43.
$$\begin{cases} 1\cdot x_1 + 2\cdot x_2 \le 6; \\ 5\cdot x_1 + 2\cdot x_2 \le 10; \\ 2\cdot x_1 + 1x_2 \ge 4; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

44.
$$\begin{cases} 5\cdot x_1 + 3\cdot x_2 \le 15; \\ 3\cdot x_1 + 7\cdot x_2 \le 21; \\ 2\cdot x_1 + 3\cdot x_2 \ge 6; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

45.
$$\begin{cases} 1\cdot x_1 + 7\cdot x_2 \le 7; \\ 4\cdot x_1 + 4\cdot x_2 \le 16; \\ 6\cdot x_1 + 2\cdot x_2 \ge 12; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

46.
$$\begin{cases} 6\cdot x_1 + 4\cdot x_2 \le 24; \\ 3\cdot x_1 + 7\cdot x_2 \le 21; \\ 7\cdot x_1 + 2\cdot x_2 \ge 14; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

47.
$$\begin{cases} 2\cdot x_1 + 3\cdot x_2 \le 12; \\ 6\cdot x_1 + 1\cdot x_2 \le 6; \\ 3\cdot x_1 + 2\cdot x_2 \ge 6; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

48.
$$\begin{cases} 7\cdot x_1 + 4\cdot x_2 \le 28; \\ 3\cdot x_1 + 6\cdot x_2 \le 18; \\ 5\cdot x_1 + 1\cdot x_2 \ge 5; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

49.
$$\begin{cases} 4\cdot x_1 + 5\cdot x_2 \le 20; \\ 5\cdot x_1 + 3\cdot x_2 \le 15; \\ 2\cdot x_1 + 6\cdot x_2 \ge 12; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

50.
$$\begin{cases} 6\cdot x_1 + 2\cdot x_2 \le 12; \\ 5\cdot x_1 + 3\cdot x_2 \le 15; \\ 7\cdot x_1 + 1\cdot x_2 \ge 7; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

51.
$$\begin{cases} 8\cdot x_1 + 3\cdot x_2 \le 24; \\ 7\cdot x_1 + 7\cdot x_2 \le 49; \\ 4\cdot x_1 + 2\cdot x_2 \ge 8; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

52.
$$\begin{cases} 4\cdot x_1 + 4\cdot x_2 \le 16; \\ 2\cdot x_1 + 5\cdot x_2 \le 10; \\ 5\cdot x_1 + 2\cdot x_2 \ge 10; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

53.
$$\begin{cases} 7\cdot x_1 + 2\cdot x_2 \le 14; \\ 2\cdot x_1 + 6\cdot x_2 \le 12; \\ 1\cdot x_1 + 1\cdot x_2 \ge 1; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

54.
$$\begin{cases} 5\cdot x_1 + 4\cdot x_2 \le 20; \\ 4\cdot x_1 + 6\cdot x_2 \le 24; \\ 2\cdot x_1 + 7\cdot x_2 \ge 14; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

55.
$$\begin{cases} 2\cdot x_1 + 7\cdot x_2 \le 14; \\ 3\cdot x_1 + 3\cdot x_2 \le 9; \\ 4\cdot x_1 + 1\cdot x_2 \ge 4; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

56.
$$\begin{cases} 7\cdot x_1 + 2\cdot x_2 \le 14; \\ 3\cdot x_1 + 5\cdot x_2 \le 15; \\ 2\cdot x_1 + 3\cdot x_2 \ge 6; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

57.
$$\begin{cases} 5\cdot x_1 + 5\cdot x_2 \le 25; \\ 7\cdot x_1 + 2\cdot x_2 \le 14; \\ 4\cdot x_1 + 1\cdot x_2 \ge 4; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

58.
$$\begin{cases} 7\cdot x_1 + 3\cdot x_2 \le 21; \\ 2\cdot x_1 + 7\cdot x_2 \le 14; \\ 7\cdot x_1 + 1\cdot x_2 \ge 7; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

59.
$$\begin{cases} 6\cdot x_1 + 7\cdot x_2 \le 42; \\ 7\cdot x_1 + 5\cdot x_2 \le 35; \\ 7\cdot x_1 + 3\cdot x_2 \ge 21; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

60.
$$\begin{cases} 7\cdot x_1 + 3\cdot x_2 \le 21; \\ 3\cdot x_1 + 7\cdot x_2 \le 215; \\ 4\cdot x_1 + 1\cdot x_2 \ge 4; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

61.
$$\begin{cases} 2\cdot x_1 + 7\cdot x_2 \le 14; \\ 7\cdot x_1 + 4\cdot x_2 \le 28; \\ 3\cdot x_1 + 3\cdot x_2 \ge 9; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

62.
$$\begin{cases} 6\cdot x_1 + 6\cdot x_2 \le 36; \\ 5\cdot x_1 + 7\cdot x_2 \le 35; \\ 7\cdot x_1 + 3\cdot x_2 \ge 21; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

63.
$$\begin{cases} 3\cdot x_1 + 6\cdot x_2 \le 18; \\ 5\cdot x_1 + 5\cdot x_2 \le 25; \\ 3\cdot x_1 + 2\cdot x_2 \ge 6; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

64.
$$\begin{cases} 2\cdot x_1 + 3\cdot x_2 \le 12; \\ 5\cdot x_1 + 4\cdot x_2 \le 20; \\ 3\cdot x_1 + 5\cdot x_2 \ge 15; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

65.
$$\begin{cases} 1\cdot x_1 + 6\cdot x_2 \le 6; \\ 4\cdot x_1 + 2\cdot x_2 \le 8; \\ 7\cdot x_1 + 1\cdot x_2 \ge 7; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

66.
$$\begin{cases} 7\cdot x_1 + 2\cdot x_2 \le 14; \\ 4\cdot x_1 + 5\cdot x_2 \le 20; \\ 2\cdot x_1 + 1\cdot x_2 \ge 2; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

67.
$$\begin{cases} 4\cdot x_1 + 3\cdot x_2 \le 12; \\ 3\cdot x_1 + 5\cdot x_2 \le 15; \\ 5\cdot x_1 + 1\cdot x_2 \ge 5; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

68.
$$\begin{cases} 5\cdot x_1 + 4\cdot x_2 \le 20; \\ 3\cdot x_1 + 4\cdot x_2 \le 12; \\ 2\cdot x_1 + 5\cdot x_2 \ge 10; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

69.
$$\begin{cases} 1\cdot x_1 + 1\cdot x_2 \le 4; \\ 5\cdot x_1 + 1\cdot x_2 \le 5; \\ 1\cdot x_1 + 6\cdot x_2 \ge 6; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

70.
$$\begin{cases} 5\cdot x_1 + 6\cdot x_2 \le 30; \\ 7\cdot x_1 + 3\cdot x_2 \le 21; \\ 7\cdot x_1 + 1\cdot x_2 \ge 7; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

71.
$$\begin{cases} 5\cdot x_1 + 6\cdot x_2 \le 30; \\ 7\cdot x_1 + 4\cdot x_2 \le 28; \\ 1\cdot x_1 + 1\cdot x_2 \ge 3; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

72.
$$\begin{cases} 6\cdot x_1 + 4\cdot x_2 \le 24; \\ 1\cdot x_1 + 5\cdot x_2 \le 5; \\ 3\cdot x_1 + 2\cdot x_2 \ge 6; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

73.
$$\begin{cases} 7\cdot x_1 + 3\cdot x_2 \le 21; \\ 2\cdot x_1 + 5\cdot x_2 \le 10; \\ 6\cdot x_1 + 1\cdot x_2 \ge 6; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

74.
$$\begin{cases} 6\cdot x_1 + 6\cdot x_2 \le 36; \\ 7\cdot x_1 + 4\cdot x_2 \le 28; \\ 2\cdot x_1 + 6\cdot x_2 \ge 12; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

75.
$$\begin{cases} 1\cdot x_1 + 7\cdot x_2 \le 7; \\ 2\cdot x_1 + 5\cdot x_2 \le 10; \\ 3\cdot x_1 + 1\cdot x_2 \ge 3; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

76.
$$\begin{cases} 7\cdot x_1 + 3\cdot x_2 \le 21; \\ 4\cdot x_1 + 5\cdot x_2 \le 20; \\ 1\cdot x_1 + 1\cdot x_2 \ge 2; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

77.
$$\begin{cases} 1\cdot x_1 + 7\cdot x_2 \le 7; \\ 4\cdot x_1 + 5\cdot x_2 \le 20; \\ 7\cdot x_1 + 1\cdot x_2 \ge 7; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

78.
$$\begin{cases} 4\cdot x_1 + 3\cdot x_2 \le 12; \\ 2\cdot x_1 + 7\cdot x_2 \le 14; \\ 4\cdot x_1 + 1\cdot x_2 \ge 5; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

79.
$$\begin{cases} 6\cdot x_1 + 5\cdot x_2 \le 30; \\ 1\cdot x_1 + 6\cdot x_2 \le 6; \\ 4\cdot x_1 + 3\cdot x_2 \ge 12; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

80.
$$\begin{cases} 7\cdot x_1 + 3\cdot x_2 \le 21; \\ 5\cdot x_1 + 5\cdot x_2 \le 25; \\ 6\cdot x_1 + 3\cdot x_2 \ge 18; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

81.
$$\begin{cases} 5\cdot x_1 + 6\cdot x_2 \le 30; \\ 6\cdot x_1 + 3\cdot x_2 \le 18; \\ 2\cdot x_1 + 7\cdot x_2 \ge 14; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

82.
$$\begin{cases} 6\cdot x_1 + 4\cdot x_2 \le 24; \\ 2\cdot x_1 + 5\cdot x_2 \le 10; \\ 2\cdot x_1 + 3\cdot x_2 \ge 6; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

83.
$$\begin{cases} 6\cdot x_1 + 6\cdot x_2 \le 36; \\ 3\cdot x_1 + 7\cdot x_2 \le 21; \\ 5\cdot x_1 + 2\cdot x_2 \ge 10; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

84.
$$\begin{cases} 1\cdot x_1 + 1\cdot x_2 \le 6; \\ 7\cdot x_1 + 3\cdot x_2 \le 21; \\ 1\cdot x_1 + 3\cdot x_2 \ge 6; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

85.
$$\begin{cases} 2\cdot x_1 + 7\cdot x_2 \le 14; \\ 7\cdot x_1 + 3\cdot x_2 \le 21; \\ 3\cdot x_1 + 1\cdot x_2 \ge 3; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

86.
$$\begin{cases} 7\cdot x_1 + 3\cdot x_2 \le 21; \\ 2\cdot x_1 + 7\cdot x_2 \le 14; \\ 3\cdot x_1 + 2\cdot x_2 \ge 6; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

87.
$$\begin{cases} 3\cdot x_1 + 5\cdot x_2 \le 15; \\ 4\cdot x_1 + 3\cdot x_2 \le 12; \\ 2\cdot x_1 + 1\cdot x_2 \ge 2; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

88.
$$\begin{cases} 2\cdot x_1 + 7\cdot x_2 \le 14; \\ 7\cdot x_1 + 5\cdot x_2 \le 35; \\ 1\cdot x_1 + 4\cdot x_2 \ge 4; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

89.
$$\begin{cases} 2\cdot x_1 + 7\cdot x_2 \le 14; \\ 5\cdot x_1 + 3\cdot x_2 \le 15; \\ 3\cdot x_1 + 2\cdot x_2 \ge 6; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

90.
$$\begin{cases} 2\cdot x_1 + 5\cdot x_2 \le 10; \\ 6\cdot x_1 + 4\cdot x_2 \le 25; \\ 3\cdot x_1 + 1\cdot x_2 \ge 3; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

91.
$$\begin{cases} 5\cdot x_1 + 6\cdot x_2 \le 30; \\ 2\cdot x_1 + 1\cdot x_2 \le 6; \\ 2\cdot x_1 + 2\cdot x_2 \ge 4; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

92.
$$\begin{cases} 5\cdot x_1 + 7\cdot x_2 \le 35; \\ 6\cdot x_1 + 3\cdot x_2 \le 18; \\ 7\cdot x_1 + 1\cdot x_2 \ge 7; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

93.
$$\begin{cases} 3\cdot x_1 + 7\cdot x_2 \le 21; \\ 6\cdot x_1 + 4\cdot x_2 \le 24; \\ 4\cdot x_1 + 3\cdot x_2 \ge 12; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

94.
$$\begin{cases} 7\cdot x_1 + 4\cdot x_2 \le 28; \\ 5\cdot x_1 + 7\cdot x_2 \le 35; \\ 3\cdot x_1 + 5\cdot x_2 \ge 15; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

95.
$$\begin{cases} 1\cdot x_1 + 6\cdot x_2 \le 6; \\ 5\cdot x_1 + 4\cdot x_2 \le 20; \\ 3\cdot x_1 + 2\cdot x_2 \ge 6; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

96.
$$\begin{cases} 7\cdot x_1 + 3\cdot x_2 \le 21; \\ 5\cdot x_1 + 6\cdot x_2 \le 30; \\ 2\cdot x_1 + 1\cdot x_2 \ge 2; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

97.
$$\begin{cases} 5\cdot x_1 + 5\cdot x_2 \le 25; \\ 3\cdot x_1 + 4\cdot x_2 \le 12; \\ 7\cdot x_1 + 2\cdot x_2 \ge 14; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

98.
$$\begin{cases} 6\cdot x_1 + 2\cdot x_2 \le 12; \\ 2\cdot x_1 + 4\cdot x_2 \le 8; \\ 4\cdot x_1 + 1\cdot x_2 \ge 4; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

99.
$$\begin{cases} 6\cdot x_1 + 5\cdot x_2 \le 30; \\ 3\cdot x_1 + 6\cdot x_2 \le 18; \\ 5\cdot x_1 + 3\cdot x_2 \ge 15; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

ПРИЛОЖЕНИЕ Б
(обязательное)
ВАРИАНТЫ ФУНКЦИЙ ЦЕЛИ

Функция цели задаётся в виде элементов матрицы C и вектора b, используемых в записи $f(X) = b^T X + \frac{1}{2} X^T C X$

№	$c_{11} = c_{22}$	$c_{12} = c_{21}$	b_1	b_2
(1)	(2)	(3)	(4)	(5)
0.	— 0.60	0.55	0.40	0.55
1.	— 0.35	0.33	0.45	— 0.30
2.	— 0.40	0,30	— 0.10	— 0.20
3.	— 0.70	0.60	0.70	0.35
4.	— 0.55	0.45	— 0.65	0.40
5.	— 0,40	0.20	— 0.10	2.00
6.	— 0.80	0.30	0.60	0.45
7.	— 0.65	0.35	0.55	— 0.70
8.	— 0.50	0.40	0.30	— 0.65
9.	— 0.55	0.45	— 0.40	0.60
10.	— 0,50	0,30	— 0.50	0.50
11.	— 0.45	0.43	0.35	— 0.55
12.	— 0.37	0.35	— 0.30	0.30
13.	— 0.61	0.40	— 0.60	0.55
14.	— 0.55	0.50	— 0.65	0.60
15.	— 0.80	0,10	1.00	2.50
16.	— 0.65	0.30	— 0.70	0.55
17.	— 0.65	0.35	— 0.75	0.30
18.	— 0.80	0.40	— 0.80	0.35
19.	— 0.55	0.45	0.85	— 0.40
20.	— 0.80	0.70	— 0.50	0.45
21.	— 1.00	0.80	1.00	2.50
22.	— 0.85	0.65	0.70	0.70
23.	— 0.50	0.47	— 0.65	0.65
24.	— 0.57	0.53	— 0.60	0.60
25.	— 0.75	0.30	— 0.55	0.50
26.	— 0.45	0.35	0.30	0.70

(1)	(2)	(3)	(4)	(5)
27.	— 0.80	0.75	0.80	0.80
28.	— 0.48	0.40	— 0.35	— 0.65
29.	— 0.88	0.45	0.40	0.60
30.	— 0.73	0.70	0.45	— 0.55
31.	— 0.67	0.65	— 0.70	0.30
32.	— 0.77	0.60	0.65	— 0.75
33.	— 1,25	1.75	— 1.50	— 0.50
34.	— 0.75	0.55	— 0.80	0.80
35.	— 0.83	0.30	0.85	0.85
36.	— 0.65	0.35	— 0.50	0.50
37.	— 0.53	0.40	0.55	— 0.45
38.	— 0.55	0.41	0.45	0.25
39.	— 0.87	0.80	— 0.50	0.50
40.	— 0.85	0.70	— 0.40	0.20
41.	— 0.62	0.40	— 0.85	0.15
42.	— 0,85	0.81	— 2.25	2.50
43.	— 0.85	0.63	— 0.30	0.40
44.	— 0.95	0.55	0.60	— 0.75
45.	— 1,25	0,75	— 1.00	0.50
46.	— 0.88	0.33	0.65	— 0.35
47.	— 0.61	0.60	— 0.70	0.50
48.	— 0.52	0.46	— 0.50	0.70
49.	— 0.95	0.35	0.25	0.40

Приложение В
(обязательное)
ТЕКСТЫ ПРОГРАММ

В.1. РЕШЕНИЕ ЗАДАЧИ КВАДРАТИЧНОГО ПРОГРАММИРОВАНИЯ НА ОСНОВАНИИ quadprog

```
% решение задачи квадратичного программирования путём
вызова
% стандартной процедуры x = quadprog(C,b,A,A0)
clear;
c=-[-0.5 0.25; 0.25 -0.5]; b=-[2; 3];
% минусы потому что на максимум, а процедура “заточена”
на поиск минимума
a=[2 5; 3 2; 1 1];
a0=[10; 6; 4];
x_opt=quadprog(c,b,a,a0);
f_opt=b'*x_opt+0.5*(x_opt'*c*x_opt);

% графики
[x1,x2] = meshgrid(0:.25:4);
f=-2*x1-3*x2 -0.25*(x1.*x2)+0.25*(x1.^2+x2.^2);
figure (1)
mesh(x1,x2,f);
hold
g1=2*x1+5*x2-10;
mesh(x1,x2,g1);
g2=3*x1+2*x2-6;
mesh(x1,x2,g2);
g3=1*x1+1*x2-4;
mesh(x1,x2,g3);

axis([0 4 0 4 -10 20])

figure (2)
contour(x1,x2,f,10,'r:','LineWidth',2);
hold
plot([0 5],[2 0],'g:','LineWidth',2);
plot([0 2],[3 0],'b:','LineWidth',2);
plot([0 4],[4 0],'k:','LineWidth',2);
grid;
plot(x_opt(1),x_opt(2),'kd-','LineWidth',5);
plot(x_opt(1),x_opt(2),'kd-','LineWidth',5);
title(sprintf('f= %g',-f_opt));
```

```
xlabel(sprintf('x1= %g',x_opt(1)));
ylabel(sprintf('x2= %g',x_opt(2)));
```

Пример решения

Матричные компоненты условия задачи

$$A=\begin{pmatrix}2 & 5\\ 3 & 2\\ 1 & 1\end{pmatrix},\ A_0=\begin{pmatrix}10\\ 6\\ 4\end{pmatrix},\quad b^T=\begin{pmatrix}2 & 3\end{pmatrix},\ C=\begin{pmatrix}-\frac{1}{2} & \frac{1}{4}\\ \frac{1}{4} & -\frac{1}{2}\end{pmatrix}.$$

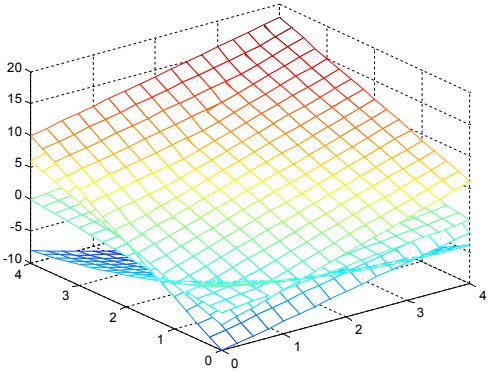

Рисунок В.1 – Пространственная картина

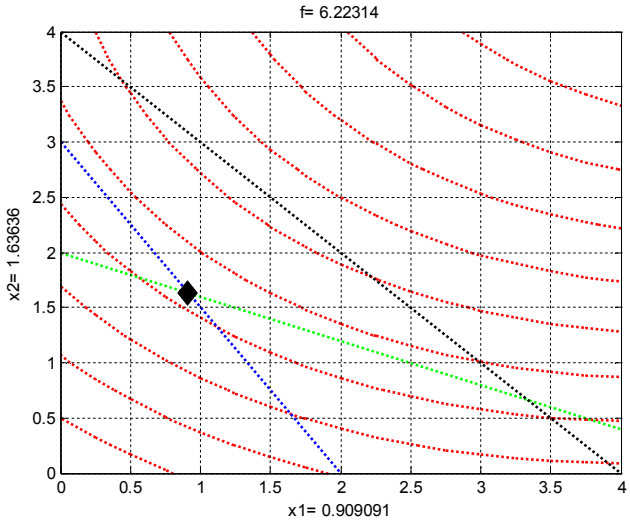

Рисунок В.2 – Линии равного уровня и точка оптимального решения

В.2. РЕШЕНИЕ ЗАДАЧИ КВАДРАТИЧНОГО ПРОГРАММИРОВАНИЯ НА ОСНОВАНИИ ТЕОРЕМЫ, ПОЗВОЛЯЮЩЕЙ ПОСТРОИТЬ ЭКВИВАЛЕНТНУЮ ЗАДАЧУ ЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ С ИСПОЛЬЗОВАНИЕМ ФУНКЦИИ `linprog`

```
% решение задачи линейного программирования путём вызова
% стандартной процедуры x = linprog(fs,as,bs,aeq,beq)
clear;
fs=[0;0;0;0;0;0;0;0;0;1;1;1;1;1];

aeq=[0.5 -0.25 2 3 1 -1  0 0 0 0 1 0 0 0 0; ...
    -0.25 0.5  5 2 1  0 -1 0 0 0 0 1 0 0 0; ...
     2.0  5.0  0 0 0  0  0 1 0 0 0 0 1 0 0; ...
     3.0 -2.0  0 0 0  0  0 0 1 0 0 0 0 1 0; ...
     1.0   1.0 0 0 0  0  0 0 0 1 0 0 0 0 1];
beq=[2;3;10; 6; 4];
x_opt = linprog(fs,aeq,beq);
f_opt=fs'*x_opt;
figure (1)
plot([0 5],[2 0],'g:','LineWidth',2);
hold
plot([0 2],[3 0],'b:','LineWidth',2);
plot([0 4],[4 0],'r:','LineWidth',2);
grid;
plot(x_opt(1),x_opt(2),'kd-','LineWidth',5);
title(sprintf('f= %g',f_opt));
xlabel('X_1');
ylabel('X_2');
```

В.3. ПРОГРАММА, РЕАЛИЗУЮЩАЯ МЕТОД ТРАЕКТОРИЙ (ХУКА-ДЖИВСА)

Головная

```
format compact
format shortG

options = optimset('PlotFcns', @optimplotfval);
fun = @(x) -1/2*x(1)^2 * 0.65-1/2*x(2)^2 *
0.65+0.35*x(1)*x(2)+0.55*x(1)+0.7*x(2);
fun2 = @(x) -fun(x);
X0 = [1 1];
```

```
disp('Usage of function fminsearch');
[x fval exitflag output] = fminsearch(fun2, X0,
options)
grid on;
disp(' ');
disp(' ');

disp('Usage of written funcion (hook_jeeves)');
step = 0.1;
acceleration = 1;
eps = 0.0001;
S = eye(2);

[X fval iterations] = hook_jeeves(fun2, X0, step,
acceleration, eps, S)
```

Собственно реализация метода

```
function [X, fval, iterations] = hook_jeeves(fun, X0,
step, acceleration, eps, S)

    Y = X0;
    iterations = 0;

    while true
        moved = false;
        for j = 1:size(S, 1)
            if fun(Y + step * S(j, :)) < fun(Y)
                Y = Y + step * S(j, :);
            else
                if fun(Y - step * S(j, :)) < fun(Y)
                    Y = Y - step * S(j, :);
                else
                    continue;
                end
            end

            step = 2 * step;
            moved = true;
        end

        if moved
            Y = X0 + acceleration * (Y - X0);
            X0 = Y;
        else
```

```
            if eps >= step
                break
            else
                step = step / 2;
            end
        end

        iterations = iterations + 1;
    end

    X = Y;
    fval = fun(X);
end
```

Пример решения.

$$f(x_1,x_2)=\begin{pmatrix}-0{,}55 & -0{,}7\end{pmatrix}\cdot\begin{pmatrix}x_1\\x_2\end{pmatrix}+\begin{pmatrix}x_1 & x_2\end{pmatrix}\cdot\begin{pmatrix}0{,}65 & -0{,}35\\-0{,}35 & 0{,}65\end{pmatrix}\cdot\begin{pmatrix}x_1\\x_2\end{pmatrix}$$

Результат: $X^T = [2{,}0083;\ 2{,}1584]$, $f_{opt} = -1{,}3087$; $Iter = 41 - 78$.

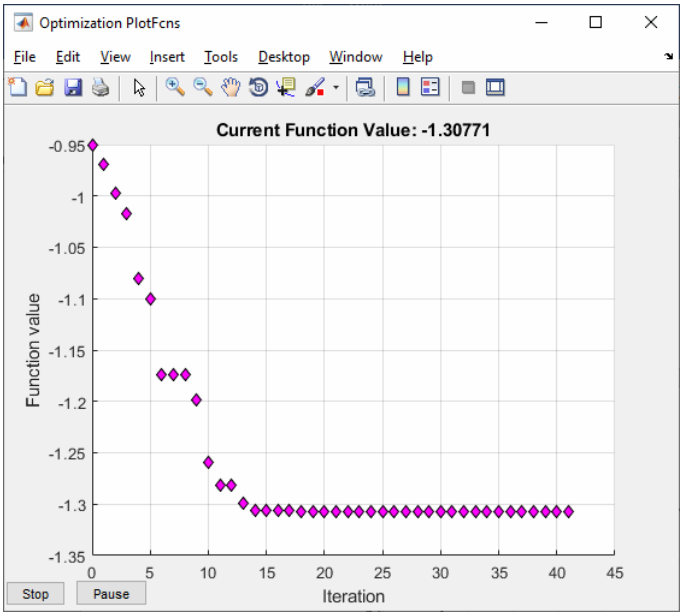

Рисунок В.3 – Итерационный процесс

В.4. МЕТОД НАИСКОРЕЙШЕГО СПУСКА

Главный модуль

```
close all
clear
clc
format compact
format shortG

options = optimset('PlotFcns', @optimplotfval);
%fun = @(x) -1/2*x(1)^2 - 1/2*x(2)^2
+0.8*x(1)*x(2)+x(1)+2.5*x(2);
fun = @(x) -1/2*x(1)^2 * 0.65-1/2*x(2)^2 *
0.65+0.35*x(1)*x(2)+0.55*x(1)+0.7*x(2);
fun2 = @(x) -fun(x);
%gradient_fun2 = @(X) [
    %X(1) - 0.8 * X(2) - 1
    %-0.8*X(1) + X(2) - 2.5]';
gradient_fun2 = @(X) [
    0.65*X(1) - 0.35 * X(2) - 0.55
    -0.35*X(1) + 0.65*X(2) - 0.7]';
X0 = [1 1];
eps = 10e-4;

disp('Usage function fminunc')
[x,fval,exitflag] = fminunc(fun2, X0, options)
grid
disp(' ');

disp('Lambda search with by using fminbnd')
tic
[x,fval, iterations, all_from_1] =
steepest_descent_method_1(fun2, gradient_fun2, X0,
eps);
toc
x,fval, iterations
disp(' ');

disp('Analytical lamgda search (by solving differential
equation)')
tic
[x,fval, iterations, all_from_2] =
steepest_descent_method_2(fun2, gradient_fun2, X0,
eps);
```

```
toc
x,fval, iterations


X_min = [8.33, 9.17];
show_result_by_plot(fun2, X0, X_min, all_from_2,
all_from_1, 'Lambda by solve', 'Lambda by fminbnd',
'Steepest descent method');
```

Процедура визуализации

```
function show_result_by_plot(fun, X0, X_min,
x_from_method1, x_from_method2, method1_name,
method2_name, titleName)

    figure
    title(titleName)
    hold on;
    grid on;

    h(1) = plot(X0(1), X0(2), 's', 'MarkerSize', 15,
'LineWidth', 4, 'DisplayName', 'X0');
    h(2) = plot(X_min(1), X_min(2), 'x', 'MarkerSize',
15, 'LineWidth', 4, 'DisplayName', 'min');

    [X, Y] = meshgrid(-6:.4:10);
    F = arrayfun(@(x1, x2) fun([x1 x2]), X, Y);
    contour(X,Y,F, 'ShowText', 'on', 'LineWidth', 2);


    h(3) = plot(x_from_method1(:, 1), x_from_method1(:,
2), '-o', 'LineWidth', 3, 'DisplayName',
sprintf('%s\n(%d iterations)', method1_name,
length(x_from_method1)-1));
    h(4) = plot(x_from_method2(:, 1), x_from_method2(:,
2), '--hr', 'DisplayName', sprintf('%s\n(%d
iterations)', method2_name, length(x_from_method2)-1));
    legend(h(:));

end
```

Метод 1

```
function [ X, fval, iterations, all_x ] =
steepest_descent_method_1(fun, gradient, X0, eps )
```

```
    X = X0;
    all_x(1, :) = X;

    iterations = 0;
    while ~all(abs(gradient(X)) <= eps)
        s = -gradient(X);

        step_fun = @(step) fun(X + step * s);
        step = fminbnd(step_fun, -100, 100);

        X = X + step * s;

        iterations = iterations + 1;
        all_x(iterations+1, :) = X;
    end
    fval = fun(X);

end
```

Метод 2

```
function [ X, fval, iterations, all_x ] =
steepest_descent_method_2(fun, gradient, X0, eps )

    X = X0;
    all_x(1, :) = X;
    iterations = 0;
    syms L;

    syms L x1 x2 real;
    Xk = [x1 x2];
    S = -gradient(Xk);
    diff_eq = diff(fun(Xk + L * S), L) == 0;
    sol = solve(diff_eq, L);
    pretty(simplify(sol));

    while ~all(abs(gradient(X)) <= eps)
        s = -gradient(X);

        f = fun(X + L * s);
        %step = double(solve(diff(f, L) == 0));
        step = double(subs(sol, [x1 x2], X));

        X = X + step * s;
```

```
        iterations = iterations + 1;
        all_x(iterations+1, :) = X;
    end
    fval = fun(X);

end
```

Пример решения.

Условие: $f(x_1,x_2)=\begin{pmatrix}-0{,}55 & -0{,}7\end{pmatrix}\cdot\begin{pmatrix}x_1\\x_2\end{pmatrix}+\begin{pmatrix}x_1 & x_2\end{pmatrix}\cdot\begin{pmatrix}0{,}65 & -0{,}35\\-0{,}35 & 0{,}65\end{pmatrix}\cdot\begin{pmatrix}x_1\\x_2\end{pmatrix}$

Градиент $\nabla f(X) = [0.65 \bullet x_1 - 0.35 \bullet x_2 - 0.55;\ -0.35 \bullet x_1 + 0.65 \bullet x_2 - 0.7]^T$

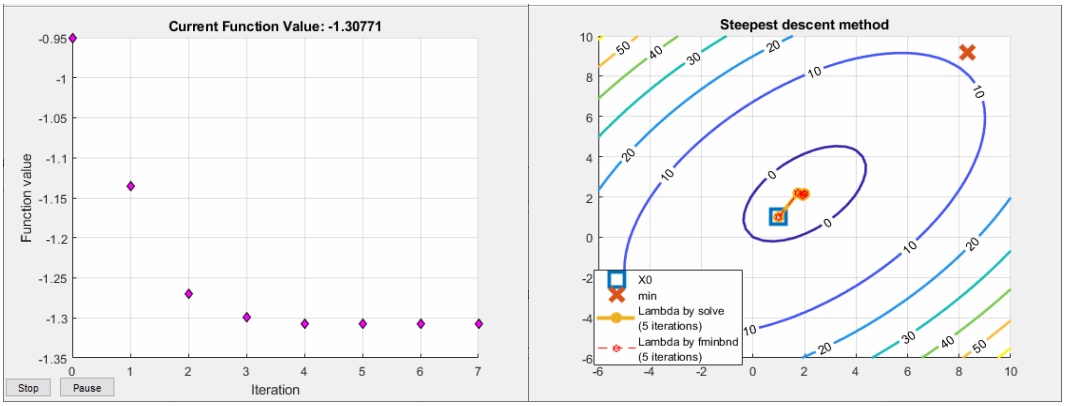

Рисунок В.4 – Графическая интерпретация решения

В.5. МЕТОД СОПРЯЖЁННОГО ГРАДИЕНТА

Главная программа

```
close all
clear
clc
format compact
format shortG

fun = @(x) -1/2*x(1)^2 * 0.65-1/2*x(2)^2 *
0.65+0.35*x(1)*x(2)+0.55*x(1)+0.7*x(2);
fun2 = @(x) -fun(x);
gradient_fun2 = @(X) [
    0.65*X(1) - 0.35 * X(2) - 0.55
    -0.35*X(1) + 0.65*X(2) - 0.7]';
X0 = [1 1];
```

```
eps = 10e-4;

syms L;
step_fun1 = @(fun, X, s) double(solve(diff(fun(X + L *
s), L) == 0));
step_fun2 = @(fun, X, s) fminbnd(@(step) fun(X + step *
s), -100, 100);


conjugate_gradient_method1 = @(fun, gradient, X0, eps)
conjugate_gradient_method(fun, gradient, X0, eps,
step_fun1);
conjugate_gradient_method2 = @(fun, gradient, X0, eps)
conjugate_gradient_method(fun, gradient, X0, eps,
step_fun2);


disp('Analytical lambda search (by solving differential
equation)')
[x,fval, iterations, all_x_from_method1] =
conjugate_gradient_method1(fun2, gradient_fun2, X0,
eps);
x,fval, iterations
disp(' ');

disp('Lambda search with by using fminbnd')
[x,fval, iterations, all_x_from_method2] =
conjugate_gradient_method2(fun2, gradient_fun2, X0,
eps);
x,fval, iterations


X_min = [8.33, 9.17];
show_result_by_plot(fun2, X0, X_min,
all_x_from_method1, all_x_from_method2, 'Lambda by
solve', 'Lambda by fminbnd', 'Conjugate gradient
methods');
```

Реализация метода

```
function [ X, fval, iterations, allX ] =
conjugate_gradient_method(fun, gradient, X0, eps,
step_fun )

    X_pred = X0;
```

```
    X = X_pred;
    allX(1, :) = X;
    s = -gradient(X);
    iterations = 0;

    while ~all(abs(gradient(X)) <= eps)
        step = step_fun(fun, X, s);

        X = X + step * s;

        omega = norm(gradient(X))^2 /
norm(gradient(X_pred))^2;
        s = -gradient(X) + omega * s;

        X_pred = X;
        iterations = iterations + 1;
        allX(iterations+1, :) = X;
    end

    fval = fun(X);

end
```

Функция демонстрации результатов

```
function show_result_by_plot(fun, X0, X_min,
x_from_method1, x_from_method2, method1_name,
method2_name, titleName)

    figure
    title(titleName)
    hold on;
    grid on;

    h(1) = plot(X0(1), X0(2), 's', 'MarkerSize', 15,
'LineWidth', 4, 'DisplayName', 'X0');
    h(2) = plot(X_min(1), X_min(2), 'x', 'MarkerSize',
15, 'LineWidth', 4, 'DisplayName', 'min');

    [X, Y] = meshgrid(-6:.4:10);
    F = arrayfun(@(x1, x2) fun([x1 x2]), X, Y);
    contour(X,Y,F, 'ShowText', 'on', 'LineWidth', 2);
```

```
    h(3) = plot(x_from_method1(:, 1), x_from_method1(:,
2), '-o', 'LineWidth', 3, 'DisplayName',
sprintf('%s\n(%d iterations)', method1_name,
length(x_from_method1)-1));
    h(4) = plot(x_from_method2(:, 1), x_from_method2(:,
2), '--hr', 'DisplayName', sprintf('%s\n(%d
iterations)', method2_name, length(x_from_method2)-1));
    legend(h(:));

end
```

Пример решения

Условие задачи

$$f(x_1,x_2)=\begin{pmatrix}-0{,}55 & -0{,}7\end{pmatrix}\cdot\begin{pmatrix}x_1\\x_2\end{pmatrix}+\begin{pmatrix}x_1 & x_2\end{pmatrix}\cdot\begin{pmatrix}0{,}65 & -0{,}35\\-0{,}35 & 0{,}65\end{pmatrix}\cdot\begin{pmatrix}x_1\\x_2\end{pmatrix}$$

Градиент $\nabla f(X) = [0.65 \bullet x_1 - 0.35 \bullet x_2 - 0.55;\ -0.35 \bullet x_1 + 0.65 \bullet x_2 - 0.7]^T$

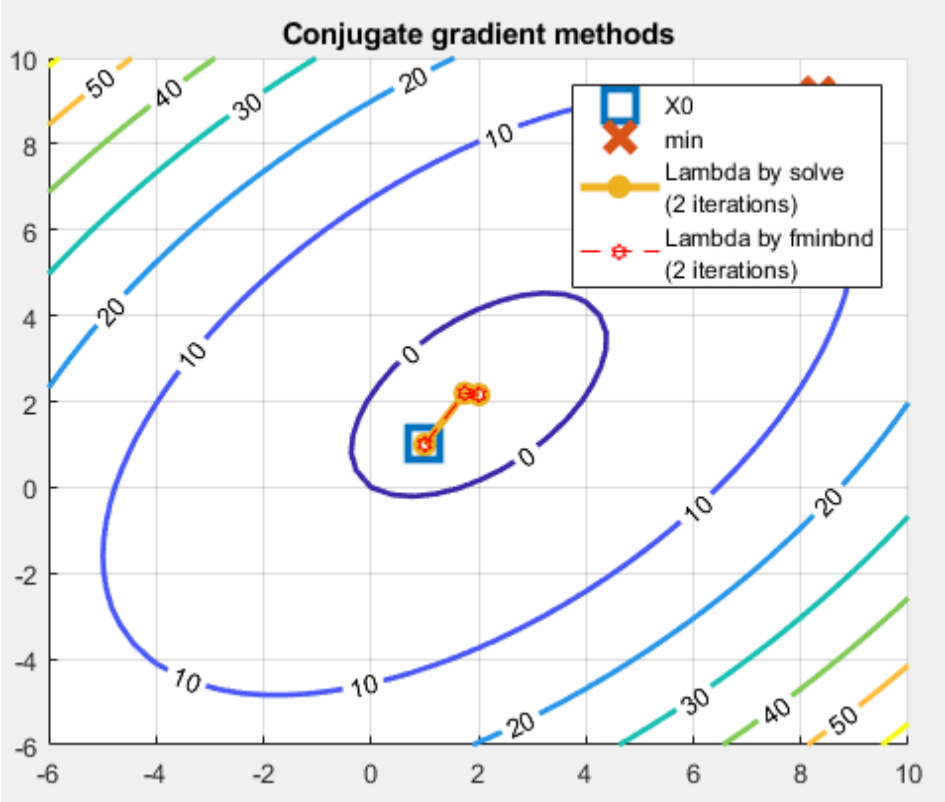

Рисунок В.5 – Результат решения методом сопряжённого градиента

В.6. РЕШЕНИЕ ОПТИМИЗАЦИОННОЙ ЗАДАЧИ НА ОСНОВАНИИ КВАЗИНЬЮТОНОВСКИХ МЕТОДОВ

Главный модуль

```
close all
clear
clc
format compact
format shortG

fun = @(x) -1/2*x(1)^2 * 0.65-1/2*x(2)^2 *
0.65+0.35*x(1)*x(2)+0.55*x(1)+0.7*x(2);
fun2 = @(x) -fun(x);
gradient_fun2 = @(X) [
    0.65*X(1) - 0.35 * X(2) - 0.55
    -0.35*X(1) + 0.65*X(2) - 0.7]';
X0 = [1 1];
eps = 10e-4;


broyden_strategy = @(delta_x, delta_g, D) deal(1,
delta_x - D * delta_g, delta_x - D * delta_g);
davidon_fletcher_powell_strategy = @(delta_x, delta_g,
D) deal(1, delta_x, D * delta_g);

broyden_method = @(fun, gradient, X0, eps)
quasi_newtonian_method(fun, gradient, X0, eps,
broyden_strategy);
davidon_fletcher_powell_method = @(fun, gradient, X0,
eps) quasi_newtonian_method(fun, gradient, X0, eps,
davidon_fletcher_powell_strategy);


disp('Broyden C.G. method')
[x,fval, iterations, all_from_broyden] =
broyden_method(fun2, gradient_fun2, X0, eps);
x,fval, iterations
disp(' ');

disp('DFP method')
[x,fval, iterations, all_from_dfp] =
davidon_fletcher_powell_method(fun2, gradient_fun2, X0,
eps);
x,fval, iterations
```

```
X_min = [2.01, 2.16];
show_result_by_plot(fun2, X0, X_min, all_from_dfp,
all_from_broyden, 'DFP', 'Broyden', 'Quasi newtonian
methods');
```

Реализация метода

```
function [ X, fval, iterations, allX ] =
quasi_newtonian_method(fun, gradient, X0, eps,
strategy)

    D = eye(2);
    X_pred = X0';
    X = X_pred;
    allX(1, :) = X;

    iterations = 0;
    while ~all(abs(gradient(X)) <= eps)

        s = -gradient(X);
        step_fun = @(step) fun(X' + step * s);
        step = fminbnd(step_fun, -100, 100);

        X = X_pred - step * D * gradient(X_pred)';
        delta_g = (gradient(X) - gradient(X_pred))';
        delta_X = X - X_pred;

        [omega, Y, Z] = strategy(delta_X, delta_g, D);

        delta_D = 1 / omega * (delta_X * Y' / (delta_g'
* Y)) - D * delta_g * Z' / (delta_g' *  Z);
        D = omega * (D + delta_D);

        X_pred = X;
        iterations = iterations + 1;
        allX(iterations+1, :) = X;
    end
    fval = fun(X);

end
```

Построение графиков

```
function show_result_by_plot(fun, X0, X_min,
x_from_method1, x_from_method2, method1_name,
method2_name, titleName)

    figure
    title(titleName)
    hold on;
    grid on;

    h(1) = plot(X0(1), X0(2), 's', 'MarkerSize', 15,
'LineWidth', 4, 'DisplayName', 'X0');
    h(2) = plot(X_min(1), X_min(2), 'x', 'MarkerSize',
15, 'LineWidth', 4, 'DisplayName', 'min');

    [X, Y] = meshgrid(-6:.4:10);
    F = arrayfun(@(x1, x2) fun([x1 x2]), X, Y);
    contour(X,Y,F, 'ShowText', 'on', 'LineWidth', 2);


    h(3) = plot(x_from_method1(:, 1), x_from_method1(:,
2), '-o', 'LineWidth', 3, 'DisplayName',
sprintf('%s\n(%d iterations)', method1_name,
length(x_from_method1)-1));
    h(4) = plot(x_from_method2(:, 1), x_from_method2(:,
2), '--hr', 'DisplayName', sprintf('%s\n(%d
iterations)', method2_name, length(x_from_method2)-1));
    legend(h(:));

end
```

Пример решения.

Условие: $f(x_1,x_2)=\begin{pmatrix}-0{,}55 & -0{,}7\end{pmatrix}\cdot\begin{pmatrix}x_1\\x_2\end{pmatrix}+\begin{pmatrix}x_1 & x_2\end{pmatrix}\cdot\begin{pmatrix}0{,}65 & -0{,}35\\-0{,}35 & 0{,}65\end{pmatrix}\cdot\begin{pmatrix}x_1\\x_2\end{pmatrix}.$

Результаты:

- Алгоритм Бройдена: Количество итераций – 3.
- Алгоритм Давидона-Флетчера-Пауэлла: Количество итераций 3.

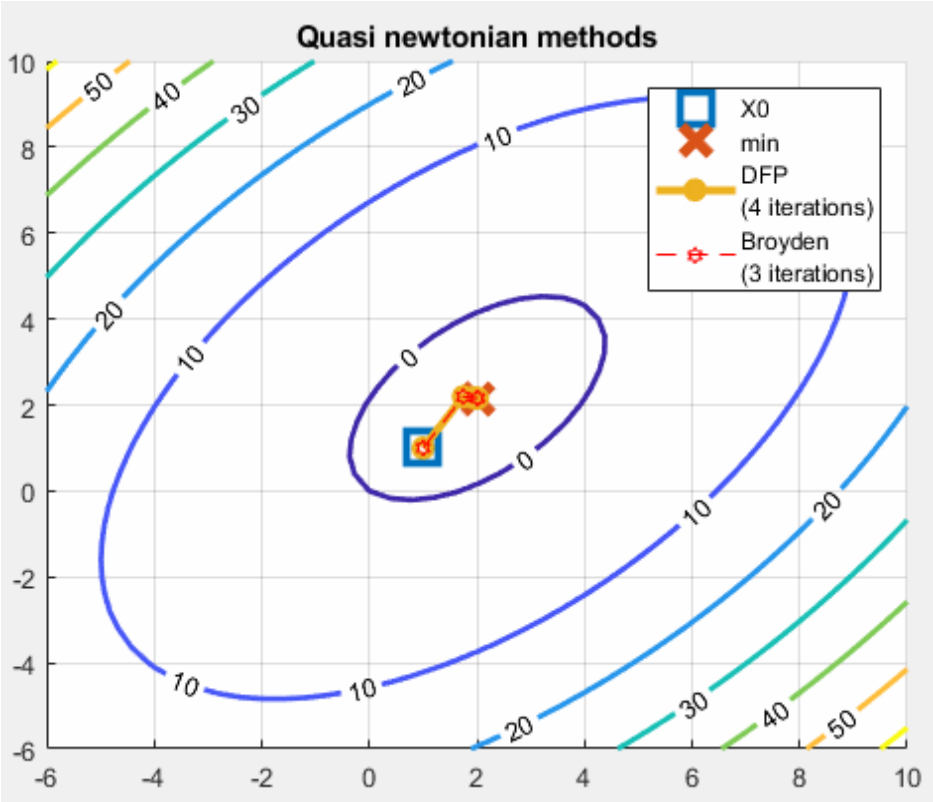

Рисунок В.6 – Результат оптимизации квазиньютоновским методом

В.7. МЕТОД ЗОЙТЕНДЕЙКА

Главная программа

```
start_and_input;

A = [5 7
       7 2
       -3 -4
       -0.65 0
       0 -0.65
];
H = A;
b = [35 14 -12 0 0]';
%X0 = [3 0];
%  X0 = [2.1 2.1];
%    X0 = [2.45 1];
%  X0 = [2.98 0.04];
% X0 = [1.11 2.52];
X0 = [-1 -1];
```

```
disp('Usage function fmincon')
options = optimset('PlotFcns', @optimplotfval);
options.TolFun = eps;
options.TolCon = eps;
options.TolX = eps;
options.ActiveConstrTol = eps;

nonlconfun = @(x) deal(A*x'- b, []);
[x,fval,exitflag, out] = fmincon(fun2, X0, [],
[],[],[],zeros(1,2),[], nonlconfun, options);
x,fval,exitflag, out.iterations
grid
disp(' ');

disp('Usage of Zoytendeyka method')
[x,fval, iterations, all_from_zoytendeyka] =
zoytendeyka_method(fun2, gradient_fun2, A, b, X0, eps);
x,fval, iterations


X_min = [2.01, 2.16];
show_result_with_constraints(fun2, X0, X_min,
all_from_zoytendeyka, 'Zoytendeyka method', [-6 10]);
show_result_with_constraints(fun2, X0, X_min,
all_from_zoytendeyka, 'Zoytendeyka method', [-1 4],
12);
```

Модуль задания данных

```
lose all
clear
clc
format compact
format shortG

fun   =   @(x)   -1/2*x(1)^2   *   0.65-1/2*x(2)^2   *
0.65+0.35*x(1)*x(2)+0.55*x(1)+0.7*x(2);
fun2 = @(x) -fun(x);
gradient_fun2 = @(X) [
    0.65*X(1) - 0.35 * X(2) - 0.55
    -0.35*X(1) + 0.65*X(2) - 0.7]';
X0 = [1 1];
eps = 10e-4;
```

Собственно реализация метода

```
function [ X, fval, iterations, allX ] =
zoytendeyka_method(fun, gradient, A, b, X0, eps)

    X = X0;
    allX(1, :) = X;

    iterations = 0;
    while true
        active_constrate_ids = find(abs(A * X' - b) <=
eps);
        ids = 1:length(A);
        A1 = A(active_constrate_ids, :);
        A2 = A(~ismember(ids, active_constrate_ids),
:);
        b1 = b(active_constrate_ids);
        b2 = b(~ismember(ids, active_constrate_ids));


        % 1)
        [S, val] = linprog(gradient(X),A1,zeros(1,
length(b1)),[],[], [-1 -1], [1 1], [],
optimset('Display', 'none'));
        if abs(val) <= eps
             break
        end


        S_modifired = A2*S;
        b_modifired = b2 - A2*X';

        if any(S_modifired > 0)
            positive_s_indexes = find(S_modifired > 0);
            step_max =
min(b_modifired(positive_s_indexes) ./
S_modifired(positive_s_indexes));
        elseif all(S_modifired <= 0)
            step_max = Inf;
        end

        step_fun = @(step) fun(X' + step * S);
        step = fminbnd(step_fun, 0, step_max);
```

```
        X = X + step * S';


        iterations = iterations + 1;
        allX(iterations+1, :) = X;
    end

    fval = fun(X);

end
```

Модули визуализации

```
function show_result_with_constraints(fun, X0, X_min,
x_from_method1, title_name, axis_range, contour_levels)
    method1_name = title_name;

    figure
    title(title_name)
    hold on;
    grid on;
    h = [];

    [min, max] = deal(axis_range(1), axis_range(2));
    axis([min max min max])
    [X, Y] = meshgrid(min:0.1:max);


    g1_x2 = @(x1) (35 - 5* x1) / 7;
    g2_x2 = @(x1) (35 - 7*x1) / 2;
    g3_x2 = @(x1) (-12 + 3*x1) / -4;

    X1 = X(1,:);
    h(end+1) = plot(X1, g1_x2(X1), '--', 'Color', [0.3
0.7 0.21], 'LineWidth', 1.8, 'DisplayName', 'g_1: 5x_1
+ 7x_2 \leq 35');
    h(end+1) = plot(X1, g2_x2(X1), '-.', 'Color', [0.7
0.3 0.21], 'LineWidth', 1.8,  'DisplayName', 'g_2: 7x_1
+ 2x_2 \leq 14');
    h(end+1) = plot(X1, g3_x2(X1), ':', 'Color', [1 0
0.3], 'LineWidth', 1.8, 'DisplayName', 'g_3: 3x_1 +
4x_2 \geq 12');
    h(end+1) = line([0 0], [min max], 'LineStyle', '--
', 'Color', [1 0 0], 'LineWidth', 1.8, 'DisplayName',
'g_4: x_1 \geq 0');
```

```
    h(end+1) = line([min max], [0 0], 'LineStyle', '--
', 'Color', [0 1 0], 'LineWidth', 1.8, 'DisplayName',
'g_4: x_2 \geq 0');

    F = arrayfun(@(x1, x2) fun([x1 x2]), X, Y);
    if exist('contour_levels')
        contour(X,Y,F, contour_levels, 'ShowText',
'on', 'LineWidth', 1.2);
    else
        contour(X,Y,F, 'ShowText', 'on', 'LineWidth',
1.2);
    end

    h(end+1) = plot(x_from_method1(:, 1),
x_from_method1(:, 2), '-h', 'Color', [1 0.8 0],
'LineWidth', 3, 'DisplayName', sprintf('%s\n(%d
iterations)', method1_name, length(x_from_method1)-1),
'MarkerEdgeColor', 'r', 'MarkerSize', 8);
    h(end+1) = plot(X0(1), X0(2), 's', 'MarkerSize',
15, 'LineWidth', 4, 'DisplayName', 'X_0', 'Color',
'blue');
    h(end+1) = plot(X_min(1), X_min(2), 'x',
'MarkerSize', 15, 'LineWidth', 4, 'DisplayName',
'extremum');

    legend(h(:));

end

function     show_result_by_plot(fun,     X0,     X_min,
x_from_method1,       x_from_method2,       method1_name,
method2_name, titleName)

    figure
    title(titleName)
    hold on;
    grid on;

    h(1) = plot(X0(1), X0(2), 's', 'MarkerSize', 15,
'LineWidth', 4, 'DisplayName', 'X0');
    h(2) = plot(X_min(1), X_min(2), 'x', 'MarkerSize',
15, 'LineWidth', 4, 'DisplayName', 'min');

    [X, Y] = meshgrid(-6:.4:10);
    F = arrayfun(@(x1, x2) fun([x1 x2]), X, Y);
```

```
    contour(X,Y,F, 'ShowText', 'on', 'LineWidth', 2);


    h(3) = plot(x_from_method1(:, 1), x_from_method1(:, 2), '-o', 'LineWidth', 3, 'DisplayName', sprintf('%s\n(%d iterations)', method1_name, length(x_from_method1)-1));
    h(4) = plot(x_from_method2(:, 1), x_from_method2(:, 2), '--hr', 'DisplayName', sprintf('%s\n(%d iterations)', method2_name, length(x_from_method2)-1));
    legend(h(:));

end
```

Условие:

$$f(x_1,x_2)=\begin{pmatrix}-0{,}55 & -0{,}7\end{pmatrix}\cdot\begin{pmatrix}x_1\\x_2\end{pmatrix}+\begin{pmatrix}x_1 & x_2\end{pmatrix}\cdot\begin{pmatrix}0{,}65 & 0{,}35\\-0{,}35 & 0{,}65\end{pmatrix}\cdot\begin{pmatrix}x_1\\x_2\end{pmatrix},$$

$$\begin{cases}5\cdot x_1+7\cdot x_2\le 35,\\7\cdot x_1+2\cdot x_2\le 14,\\3\cdot x_1+4\cdot x_2\ge 12,\\x_1\ge 0, x_2\ge 0.\end{cases}$$

Результат: X^T = [1.4892; 1.7875], $f(X)$ = – 1.2428, 2 итерации.

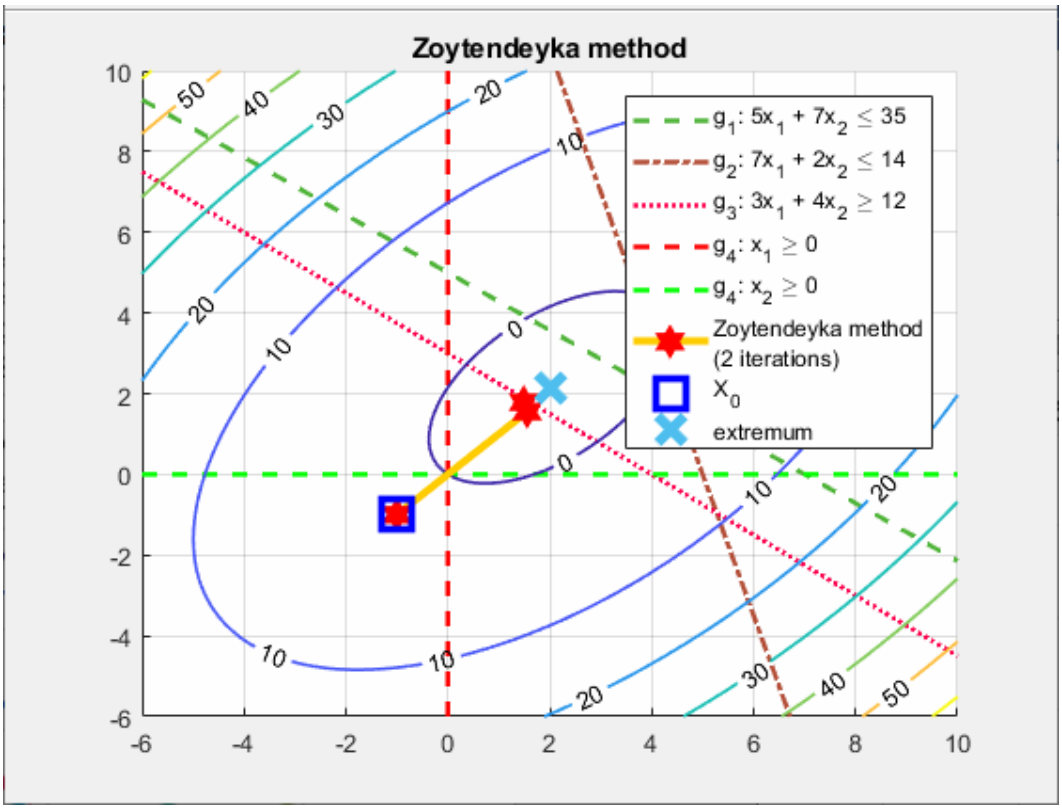

Рисунок В.7 – Результат оптимизации методом Зойтендейка

В.8. МЕТОД ПРОЕКЦИЙ РОЗЕНА

Главная программа

```
start_and_input;

A = [5 7
      7 2
      -3 -4
      -0.65 0
      0 -0.65
];
H = A;
b = [35 14 -12 0 0]';
X0 = [-1 -1];


disp('Usage of standart function fmincon')
options = optimset('PlotFcns', @optimplotfval);
options.TolFun = eps;
options.TolCon = eps;
options.TolX = eps;
options.ActiveConstrTol = eps;

nonlconfun = @(x) deal(A*x'- b, []);
```

Настройки

```
close all
clear
clc
format compact
format shortG

fun = @(x) -1/2*0.65*x(1)^2 - 1/2*0.65*x(2)^2
+0.35*x(1)*x(2)+0.55*x(1)+0.7*x(2);
fun2 = @(x) -fun(x);
gradient_fun2 = @(X) [
    0.65*X(1) - 0.35 * X(2) - 0.55
    -0.35*X(1) + 0.65*X(2) - 0.8]'
X0 = [1 1];
eps = 10e-4;
```

Реализация метода

```
function [ X, fval, iterations, allX ] =
rosen_method(fun, gradient, A, b, H, X0, eps)

    X = X0;
    allX(1, :) = X;
    I = eye(length(X));

    iterations = 0;

    active_constrate_ids = find(abs(A * X' - b) <=
eps);
    ids = 1:length(A);
    A1 = A(active_constrate_ids, :);
    A2 = A(~ismember(ids, active_constrate_ids), :);
    b1 = b(active_constrate_ids);
    b2 = b(~ismember(ids, active_constrate_ids));


    while true

        M = [A1' H']';

        is_M_equal_zero = all(abs(M(:)) <= eps);
        if is_M_equal_zero
            P = I;
        else
            % modification 1
            % before: P = I - M' * (M * M')^-1 * M;
            % (M * M')^-1
            % cause: Warning: Matrix is close to
singular or badly scaled. Results may be inaccurate.
RCOND =  3.568574e-18.
            % cause: det = 0
            P = I - M' * pinv(M * M') * M;
        end

        S = -P * gradient(X)';

        if ~all(abs(S(:)) <= eps)

            % deprecated modification 2
            A2 = [];
            b2 = [];
            for i = 1:size(A,1)
```

```
                if all(ismember(A(i, :), A1, 'rows') ==
0)
                    A2(end+1, :) = A(i, :);
                    b2(end+1, :) = b(i, :);
                end
            end

            S_modifired = A2*S;
            b_modifired = b2 - A2*X';

            if any(S_modifired > 0)
                positive_s_indexes = find(S_modifired >
0);
                [step_max, index] =
min(b_modifired(positive_s_indexes) ./
S_modifired(positive_s_indexes));
                constraint_index =
positive_s_indexes(index);
            elseif all(S_modifired <= 0)
                step_max = Inf;
            end

            step_fun = @(step) fun(X' + step * S);
            step = fminbnd(step_fun, 0, step_max);

            % modification 2
            % before: null
            if abs(abs(step) - abs(step_max)) <= eps
                A1 = [A1; A2(constraint_index, :)];
                rows = find(1:size(A2, 1) ~=
constraint_index);
                A2 = A2(rows, :);
                b2 = b2(rows);
            end

            X = X + step * S';

        else
           if is_M_equal_zero
               break
           else
               % modfication 3
               % before: W = -(M')^-1 * gradient(X)';
               % cause: Inputs must be a scalar and a
square matrix.
```

```
                W = -pinv(M') * gradient(X)';

                U = W;

                % deprecated modification 4
%                 negative_U = U(find(U < 0));
%                 max_abs_value = max(abs(negative_U));
%                 is_element_not_max_abs = abs( abs(U) -
max_abs_value) > eps;
%                 non_max_abs_indexes =
find(is_element_not_max_abs);
%
%                 A1_tilda = A1(non_max_abs_indexes, :);

                A1_tilda = A1(find(U >= 0), :);

                if all(U(:) >= eps)
                    break
                else
                    A1 = A1_tilda;
                end
            end
        end

        iterations = iterations + 1;
        allX(iterations+1, :) = X;
    end

    fval = fun(X);

end
```

Визуализация

```
function show_result_by_plot(fun, X0, X_min,
x_from_method1, x_from_method2, method1_name,
method2_name, titleName)

    figure
    title(titleName)
    hold on;
    grid on;

    h(1) = plot(X0(1), X0(2), 's', 'MarkerSize', 15,
'LineWidth', 4, 'DisplayName', 'X0');
```

```
    h(2) = plot(X_min(1), X_min(2), 'x', 'MarkerSize',
15, 'LineWidth', 4, 'DisplayName', 'min');

    [X, Y] = meshgrid(-6:.4:10);
    F = arrayfun(@(x1, x2) fun([x1 x2]), X, Y);
    contour(X,Y,F, 'ShowText', 'on', 'LineWidth', 2);


    h(3) = plot(x_from_method1(:, 1), x_from_method1(:,
2), '-o', 'LineWidth', 3, 'DisplayName',
sprintf('%s\n(%d iterations)', method1_name,
length(x_from_method1)-1));
    h(4) = plot(x_from_method2(:, 1), x_from_method2(:,
2), '--hr', 'DisplayName', sprintf('%s\n(%d
iterations)', method2_name, length(x_from_method2)-1));
    legend(h(:));

end

function show_result_with_constraints(fun, X0, X_min,
x_from_method1, title_name, axis_range, contour_levels)
    method1_name = title_name;

    figure
    title(title_name)
    hold on;
    grid on;
    h = [];

    [min, max] = deal(axis_range(1), axis_range(2));
    axis([min max min max])
    [X, Y] = meshgrid(min:0.1:max);


    g1_x2 = @(x1) (35 - 5 * x1) / 7;
    g2_x2 = @(x1) (14 - 7*x1) / 2;
    g3_x2 = @(x1) (-12 + 3*x1) / -4;

    X1 = X(1,:);
    h(end+1) = plot(X1, g1_x2(X1), '--', 'Color', [0.3
0.7 0.21], 'LineWidth', 1.8, 'DisplayName', 'g_1: 5x_1
+ 7x_2 \leq 35');
    h(end+1) = plot(X1, g2_x2(X1), '-.', 'Color', [0.7
0.3 0.21], 'LineWidth', 1.8,  'DisplayName', 'g_2: 7x_1
+ 2x_2 \leq 14');
```

```
    h(end+1) = plot(X1, g3_x2(X1), ':', 'Color', [1 0
0.3], 'LineWidth', 1.8, 'DisplayName', 'g_3: 3x_1 +
4x_2 \geq 12');
    h(end+1) = line([0 0], [min max], 'LineStyle', '--
', 'Color', [1 0 0], 'LineWidth', 1.8, 'DisplayName',
'g_4: x_1 \geq 0');
    h(end+1) = line([min max], [0 0], 'LineStyle', '--
', 'Color', [0 1 0], 'LineWidth', 1.8, 'DisplayName',
'g_4: x_2 \geq 0');

    F = arrayfun(@(x1, x2) fun([x1 x2]), X, Y);
    if exist('contour_levels')
        contour(X,Y,F, contour_levels, 'ShowText',
'on', 'LineWidth', 1.2);
    else
        contour(X,Y,F, 'ShowText', 'on', 'LineWidth',
1.2);
    end

    h(end+1) = plot(x_from_method1(:, 1),
x_from_method1(:, 2), '-h', 'Color', [1 0.8 0],
'LineWidth', 3, 'DisplayName', sprintf('%s\n(%d
iterations)', method1_name, length(x_from_method1)-1),
'MarkerEdgeColor', 'r', 'MarkerSize', 8);
    h(end+1) = plot(X0(1), X0(2), 's', 'MarkerSize',
15, 'LineWidth', 4, 'DisplayName', 'X_0', 'Color',
'blue');
    h(end+1) = plot(X_min(1), X_min(2), 'x',
'MarkerSize', 15, 'LineWidth', 4, 'DisplayName',
'extremum');

    legend(h(:));

end
```

Условие:

$$f(x_1,x_2)=\begin{pmatrix}-0{,}55 & -0{,}7\end{pmatrix}\cdot\begin{pmatrix}x_1\\x_2\end{pmatrix}+\begin{pmatrix}x_1 & x_2\end{pmatrix}\cdot\begin{pmatrix}0{,}65 & 0{,}35\\-0{,}35 & 0{,}65\end{pmatrix}\cdot\begin{pmatrix}x_1\\x_2\end{pmatrix},$$

$$\begin{cases}5\cdot x_1+7\cdot x_2\le 35,\\7\cdot x_1+2\cdot x_2\le 14,\\3\cdot x_1+4\cdot x_2\ge 12,\\x_1\ge 0, x_2\ge 0.\end{cases}$$

Результат X^T = [1.4545 1.9091], $f(X)$ = –1.2361, 2 итерации.

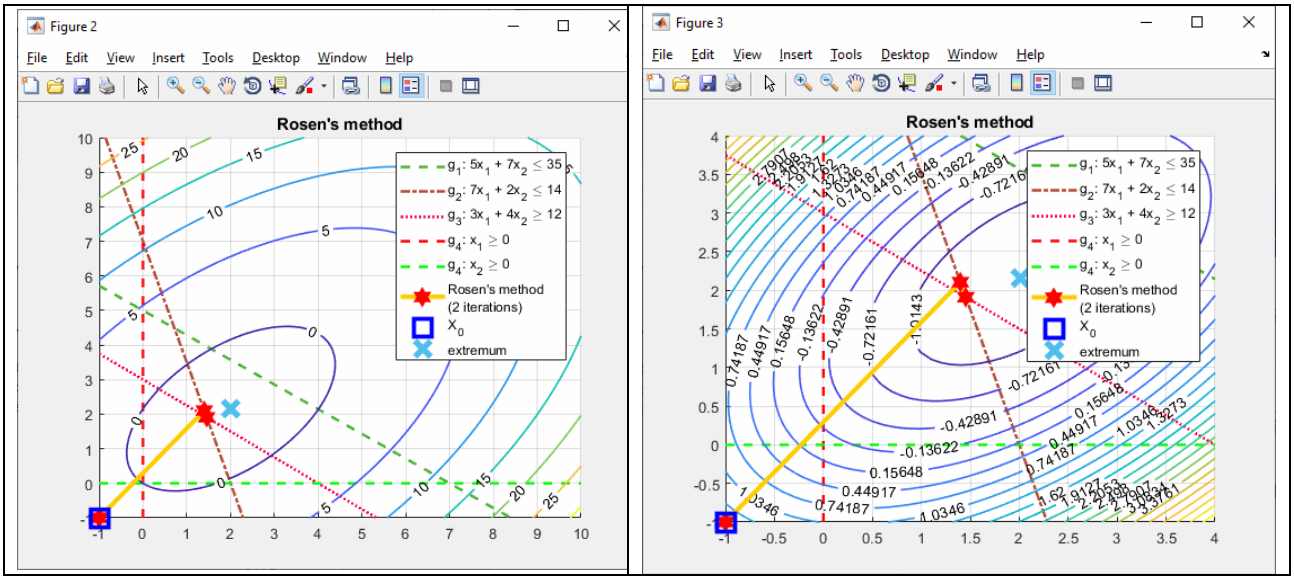

Рисунок В.8 – Результат оптимизации методом проекций Розена

В.9. МЕТОД ВНУТРЕННЕЙ ТОЧКИ

Главная программа

```
start_and_input;

A = [5 7
      7 2
      -3 -4
      -1 0
      0 -1
];
b = [35 14 -12 0 0]';

%G = @(y) 1 ./ y;
G = @(y) -log(y);

g = {};
g{end+1} = @(x) -(3*x(1) + 7*x(2) -21);
g{end+1} = @(x) -(7*x(1) + 3*x(2) -21);
g{end+1} = @(x) 4*x(1) + 3*x(2) -12;
g{end+1} = @(x) x(1);
g{end+1} = @(x) x(2);

X0 = [2.45 1];
```

```
weight = ones(1, length(g));
beta = 0.01;
r0 = 1;

disp('Usage of standart function fmincon')
options = optimset('PlotFcns', @optimplotfval);
options.TolFun = eps;
options.TolCon = eps;
options.TolX = eps;
options.ActiveConstrTol = eps;

nonlconfun = @(x) deal(A*x'- b, []);
[x,fval,exitflag, out] = fmincon(fun2, X0, [],
[],[],[],zeros(1,2),[], nonlconfun, options);
x,fval,exitflag, out.iterations
grid
disp(' ');

disp('Usage of Interior penalty method')
[x,fval, iterations, all_from_zoytendeyka] =
interior_penalty_method(fun2,weight, G, g, beta, r0,
X0, eps);
x,fval, iterations

X_min = [2.01, 2.16];
show_result_with_constraints(fun2, X0, X_min,
all_from_zoytendeyka, 'Interior penalty method', [-1
10]);
show_result_with_constraints(fun2, X0, X_min,
all_from_zoytendeyka, 'Interior penalty method', [-1
4], 22);
```

Настройки

```
close all
clear
clc
format compact
format shortG

fun = @(x) -1/2*0.65*x(1)^2 - 1/2*0.65*x(2)^2
+0.35*x(1)*x(2)+0.55*x(1)+0.7*x(2);
fun2 = @(x) -fun(x);
gradient_fun2 = @(X) [
```

```
    0.65*X(1) - 0.35 * X(2) - 0.55
    -0.35*X(1) + 0.65*X(2) - 0.8]'
X0 = [1 1];
eps = 10e-4;
```

Реализация метода

```
function [ X, fval, iterations, allX ] =
interior_penalty_method(fun, weight, G, g, beta, r0,
X0, eps)

    penalty_add_fun = @(X, r) r * sum(weight .*
G(cellfun(@(fun) fun(X), g)));
    penalty_fun = @(X, r) fun(X) + penalty_add_fun(X,
r);

    X = X0;
    r = r0;
    allX(1, :) = X;
    iterations = 0;

    while true
        X = fminunc(@(X) penalty_fun(X, r), X);
        penalty = penalty_add_fun(X, r);


        iterations = iterations + 1;
        allX(iterations+1, :) = X;

        if abs(penalty) <= eps
            break
        end

        r = r * beta;
    end

    fval = fun(X);

end
```

Демонстрация результатов

```
function show_result_by_plot(fun, X0, X_min,
x_from_method1, x_from_method2, method1_name,
method2_name, titleName)
```

```
    figure
    title(titleName)
    hold on;
    grid on;

    h(1) = plot(X0(1), X0(2), 's', 'MarkerSize', 15,
'LineWidth', 4, 'DisplayName', 'X0');
    h(2) = plot(X_min(1), X_min(2), 'x', 'MarkerSize',
15, 'LineWidth', 4, 'DisplayName', 'min');

    [X, Y] = meshgrid(-6:.4:10);
    F = arrayfun(@(x1, x2) fun([x1 x2]), X, Y);
    contour(X,Y,F, 'ShowText', 'on', 'LineWidth', 2);


    h(3) = plot(x_from_method1(:, 1), x_from_method1(:,
2), '-o', 'LineWidth', 3, 'DisplayName',
sprintf('%s\n(%d iterations)', method1_name,
length(x_from_method1)-1));
    h(4) = plot(x_from_method2(:, 1), x_from_method2(:,
2), '--hr', 'DisplayName', sprintf('%s\n(%d
iterations)', method2_name, length(x_from_method2)-1));
    legend(h(:));

end

function show_result_with_constraints(fun, X0, X_min,
x_from_method1, title_name, axis_range, contour_levels)
    method1_name = title_name;

    figure
    title(title_name)
    hold on;
    grid on;
    h = [];

    [min, max] = deal(axis_range(1), axis_range(2));
    axis([min max min max])
    [X, Y] = meshgrid(min:0.1:max);


    g1_x2 = @(x1) (35 - 5 * x1) / 7;
    g2_x2 = @(x1) (14 - 7*x1) / 2;
    g3_x2 = @(x1) (-12 + 3*x1) / -4;
```

```
    X1 = X(1,:);
    h(end+1) = plot(X1, g1_x2(X1), '--', 'Color', [0.3
0.7 0.21], 'LineWidth', 1.8, 'DisplayName', 'g_1: 5x_1
+ 7x_2 \leq 35');
    h(end+1) = plot(X1, g2_x2(X1), '-.', 'Color', [0.7
0.3 0.21], 'LineWidth', 1.8,  'DisplayName', 'g_2: 7x_1
+ 2x_2 \leq 14');
    h(end+1) = plot(X1, g3_x2(X1), ':', 'Color', [1 0
0.3], 'LineWidth', 1.8, 'DisplayName', 'g_3: 3x_1 +
4x_2 \geq 12');
    h(end+1) = line([0 0], [min max], 'LineStyle', '--
', 'Color', [1 0 0], 'LineWidth', 1.8, 'DisplayName',
'g_4: x_1 \geq 0');
    h(end+1) = line([min max], [0 0], 'LineStyle', '--
', 'Color', [0 1 0], 'LineWidth', 1.8, 'DisplayName',
'g_4: x_2 \geq 0');

    F = arrayfun(@(x1, x2) fun([x1 x2]), X, Y);
    if exist('contour_levels')
        contour(X,Y,F, contour_levels, 'ShowText',
'on', 'LineWidth', 1.2);
    else
        contour(X,Y,F, 'ShowText', 'on', 'LineWidth',
1.2);
    end

    h(end+1) = plot(x_from_method1(:, 1),
x_from_method1(:, 2), '-h', 'Color', [1 0.8 0],
'LineWidth', 3, 'DisplayName', sprintf('%s\n(%d
iterations)', method1_name, length(x_from_method1)-1),
'MarkerEdgeColor', 'r', 'MarkerSize', 8);
    h(end+1) = plot(X0(1), X0(2), 's', 'MarkerSize',
15, 'LineWidth', 4, 'DisplayName', 'X_0', 'Color',
'blue');
    h(end+1) = plot(X_min(1), X_min(2), 'x',
'MarkerSize', 15, 'LineWidth', 4, 'DisplayName',
'extremum');

    legend(h(:));

end
```

Пример решения

Условие задачи

$$f(x_1, x_2) = \begin{pmatrix} -0{,}55 & -0{,}7 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} x_1 & x_2 \end{pmatrix} \cdot \begin{pmatrix} 0{,}65 & 0{,}35 \\ -0{,}35 & 0{,}65 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix},$$

$$\begin{cases} 5 \cdot x_1 + 7 \cdot x_2 \le 35, \\ 7 \cdot x_1 + 2 \cdot x_2 \le 14, \\ 3 \cdot x_1 + 4 \cdot x_2 \ge 12, \\ x_1 \ge 0, x_2 \ge 0. \end{cases}$$

Стартовая точка $X^T{}_0 = [2{,}45;\ 1{,}0]$.

Результаты расчётов при разных функционалах штрафов
$G(y) = 1 / y$: $X^T = [1.6479;\ 1.8028]$, $f(X) = -1.2693$, 7 итераций.
$G(y) = -\ln y$: $X^T = [1.9892;\ 2.1351]$, $f(X) = -1.3076$, 3 итерации.

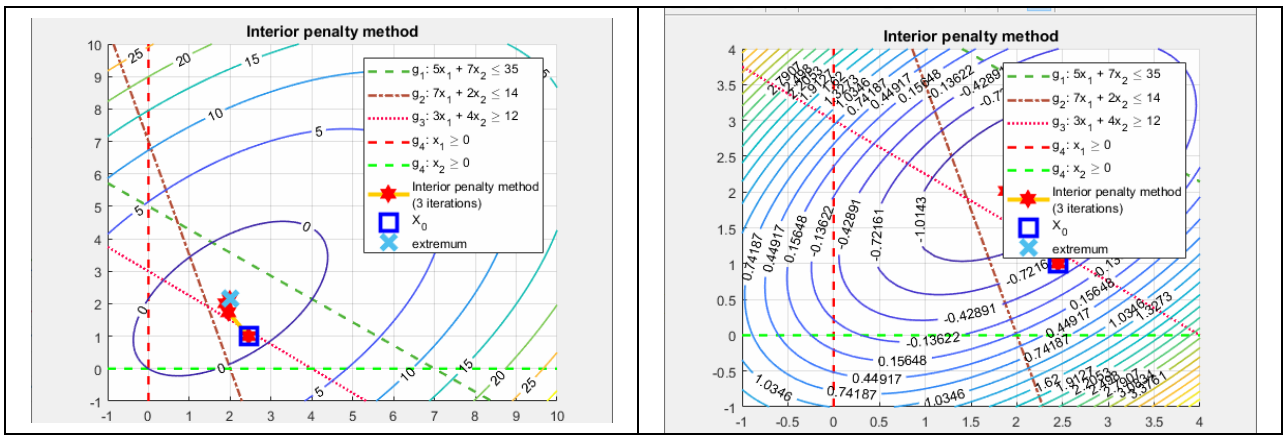

Рисунок В.9 – Результат оптимизации методом внутренней точки

В.10. МЕТОД ВНЕШНЕЙ ТОЧКИ

Главная программа

```
start_and_input;

A = [5 7
     7 2
     -3 -4
     -1 0
     0 -1
];
b = [35 14 -12 0 0]';

G = @(y) max(0, -y) .^ 2;
```

```
g = {};
g{end+1} = @(x) -(5*x(1) + 7*x(2) -35);
g{end+1} = @(x) -(7*x(1) + 2*x(2) -14);
g{end+1} = @(x) 3*x(1) + 4*x(2) -12;
g{end+1} = @(x) x(1);
g{end+1} = @(x) x(2);

X0 = [2.45 1];

weight = ones(1, length(g));
beta = 2;
r0 = 1;

disp('Usage of standart function fmincon')
options = optimset('PlotFcns', @optimplotfval);
options.TolFun = eps;
options.TolCon = eps;
options.TolX = eps;
options.ActiveConstrTol = eps;

nonlconfun = @(x) deal(A*x'- b, []);
[x,fval,exitflag, out] = fmincon(fun2, X0, [],
[],[],[],zeros(1,2),[], nonlconfun, options);
x,fval,exitflag, out.iterations
grid
disp(' ');

disp('Usage of Interior penalty method')
[x,fval, iterations, all_from_zoytendeyka] =
external_penalty_method(fun2,weight, G, g, beta, r0,
X0, eps);
x,fval, iterations

X_min = [8.33, 9.17];
show_result_with_constraints(fun2, X0, X_min,
all_from_zoytendeyka, 'External penalty method', [-1
10]);
show_result_with_constraints(fun2, X0, X_min,
all_from_zoytendeyka, 'External penalty method', [-1
4], 22);
```

Настройки

```
close all
clear
```

```
clc
format compact
format shortG

fun = @(x) -1/2*0.65*x(1)^2 - 1/2*0.65*x(2)^2
+0.35*x(1)*x(2)+0.55*x(1)+0.7*x(2);
fun2 = @(x) -fun(x);
gradient_fun2 = @(X) [
    0.65*X(1) - 0.35 * X(2) - 0.55
    -0.35*X(1) + 0.65*X(2) - 0.8]'
X0 = [1 1];
eps = 10e-4;
```

Реализация метода

```
function [ X, fval, iterations, allX ] =
external_penalty_method(fun, weight, G, g, beta, r0,
X0, eps)

    penalty_add_fun = @(X, r) r * sum(weight .*
G(cellfun(@(fun) fun(X), g)));
    penalty_fun = @(X, r) fun(X) + penalty_add_fun(X,
r);

    X = X0;
    r = r0;
    allX(1, :) = X;
    iterations = 0;

    while true
        X = fminunc(@(X) penalty_fun(X, r), X);
        penalty = penalty_add_fun(X, r);


        iterations = iterations + 1;
        allX(iterations+1, :) = X;

        if abs(penalty) <= eps
            break
        end

        r = r * beta;
    end

    fval = fun(X);
```

```
end
```

Демонстрация результатов

```
function show_result_by_plot(fun, X0, X_min,
x_from_method1, x_from_method2, method1_name,
method2_name, titleName)

    figure
    title(titleName)
    hold on;
    grid on;

    h(1) = plot(X0(1), X0(2), 's', 'MarkerSize', 15,
'LineWidth', 4, 'DisplayName', 'X0');
    h(2) = plot(X_min(1), X_min(2), 'x', 'MarkerSize',
15, 'LineWidth', 4, 'DisplayName', 'min');

    [X, Y] = meshgrid(-6:.4:10);
    F = arrayfun(@(x1, x2) fun([x1 x2]), X, Y);
    contour(X,Y,F, 'ShowText', 'on', 'LineWidth', 2);


    h(3) = plot(x_from_method1(:, 1), x_from_method1(:,
2), '-o', 'LineWidth', 3, 'DisplayName',
sprintf('%s\n(%d iterations)', method1_name,
length(x_from_method1)-1));
    h(4) = plot(x_from_method2(:, 1), x_from_method2(:,
2), '--hr', 'DisplayName', sprintf('%s\n(%d
iterations)', method2_name, length(x_from_method2)-1));
    legend(h(:));

end

function show_result_with_constraints(fun, X0, X_min,
x_from_method1, title_name, axis_range, contour_levels)
    method1_name = title_name;

    figure
    title(title_name)
    hold on;
    grid on;
    h = [];
```

```
    [min, max] = deal(axis_range(1), axis_range(2));
    axis([min max min max])
    [X, Y] = meshgrid(min:0.1:max);


    g1_x2 = @(x1) (35 - 5 * x1) / 7;
    g2_x2 = @(x1) (14 - 7*x1) / 2;
    g3_x2 = @(x1) (-12 + 3*x1) / -4;

    X1 = X(1,:);
    h(end+1) = plot(X1, g1_x2(X1), '--', 'Color', [0.3
0.7 0.21], 'LineWidth', 1.8, 'DisplayName', 'g_1: 5x_1
+ 7x_2 \leq 35');
    h(end+1) = plot(X1, g2_x2(X1), '-.', 'Color', [0.7
0.3 0.21], 'LineWidth', 1.8,  'DisplayName', 'g_2: 7x_1
+ 2x_2 \leq 14');
    h(end+1) = plot(X1, g3_x2(X1), ':', 'Color', [1 0
0.3], 'LineWidth', 1.8, 'DisplayName', 'g_3: 3x_1 +
4x_2 \geq 12');
    h(end+1) = line([0 0], [min max], 'LineStyle', '--
', 'Color', [1 0 0], 'LineWidth', 1.8, 'DisplayName',
'g_4: x_1 \geq 0');
    h(end+1) = line([min max], [0 0], 'LineStyle', '--
', 'Color', [0 1 0], 'LineWidth', 1.8, 'DisplayName',
'g_4: x_2 \geq 0');

    F = arrayfun(@(x1, x2) fun([x1 x2]), X, Y);
    if exist('contour_levels')
        contour(X,Y,F, contour_levels, 'ShowText',
'on', 'LineWidth', 1.2);
    else
        contour(X,Y,F, 'ShowText', 'on', 'LineWidth',
1.2);
    end

    h(end+1) = plot(x_from_method1(:, 1),
x_from_method1(:, 2), '-h', 'Color', [1 0.8 0],
'LineWidth', 3, 'DisplayName', sprintf('%s\n(%d
iterations)', method1_name, length(x_from_method1)-1),
'MarkerEdgeColor', 'r', 'MarkerSize', 8);
    h(end+1) = plot(X0(1), X0(2), 's', 'MarkerSize',
15, 'LineWidth', 4, 'DisplayName', 'X_0', 'Color',
'blue');
```

```
    h(end+1) = plot(X_min(1), X_min(2), 'x',
'MarkerSize', 15, 'LineWidth', 4, 'DisplayName',
'extremum');

    legend(h(:));

end
```

Пример решения

Условие

$$f(x_1,x_2)=\begin{pmatrix}-0{,}55 & -0{,}7\end{pmatrix}\cdot\begin{pmatrix}x_1\\x_2\end{pmatrix}+\begin{pmatrix}x_1 & x_2\end{pmatrix}\cdot\begin{pmatrix}0{,}65 & 0{,}35\\-0{,}35 & 0{,}65\end{pmatrix}\cdot\begin{pmatrix}x_1\\x_2\end{pmatrix},$$

$$\begin{cases}5\cdot x_1+7\cdot x_2\le 35,\\7\cdot x_1+2\cdot x_2\le 14,\\3\cdot x_1+4\cdot x_2\ge 12,\\x_1\ge 0, x_2\ge 0.\end{cases}$$

Решение

$r_0 = 1$, $\beta = 2$; $X^T = [1.9338; 2.1718]$, $f(X) = -6.4951$, 6 итераций.

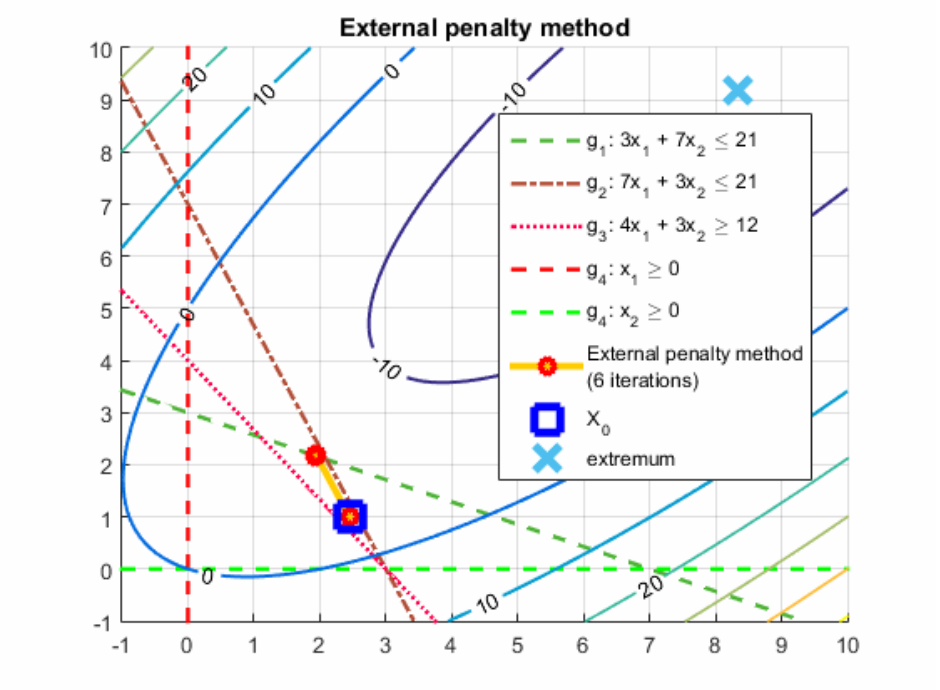

Рисунок В.10 – Результат оптимизации методом внешней точки

МЕТОДЫ ОПТИМИЗАЦИИ
И НЕЛИНЕЙНОЕ ПРОГРАММИРОВАНИЕ

Методические указания
к выполнению лабораторного практикума по дисциплине

Составитель : **Карлусов** Вадим Юрьевич

В авторской редакции

Изд. № 59/2024. Объем 5,5 п.л. Усл. печ. л. 5,11. Уч.-изд. л. 5,39.
Издательский центр ФГАОУ ВО «Севастопольский государственный университет»