ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ»
(Н И У « Б е л Г У »)

ИНСТИТУТ ИНЖЕНЕРНЫХ ТЕХНОЛОГИЙ И ЕСТЕСТВЕННЫХ НАУК

КАФЕДРА ОБЩЕЙ МАТЕМАТИКИ

ЧИСЛЕННОЕ НАХОЖДЕНИЕ ЭКСТРЕМУМОВ ФУНКЦИИ

Выпускная квалификационная работа

обучающегося по направлению подготовки
01.03.02 Прикладная математика и информатика
очной формы обучения,
группы 07001406
Трана Зуя

Научный руководитель:
к.ф.-м.н., доцент,
Флоринский В.В.

БЕЛГОРОД 2018

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В настоящий время, в различных областях науки и техники, существует большое количество экстремальных задач. Задачи по нахождению экстремумов функций, часто встречаются в науке, экономике и технике. Умение переходить от содержательной постановки задачи к математической, даёт возможность для применения математических методов их анализа и решения. С помощью численных методов можно получить приближенной решение. В зависимости от метода, с помощью числа итерационных исчислений можно определить минимальное значение функции и точность расчета точки минимума.

Актуальность темы данной работы состоит в том, что использование численных методов упрощает алгоритм решения многомерных задач, а также дает возможность найти решение абсолютно всех классов экстремальных задач, появившихся в последнее десятилетие.

Целью данной работы является разработка программ некоторых методов нахождения численных значений экстремумов функций.

В процессе работы будут решены следущие задачи:

- Изучить и исследовать общую задачу;
- проанализировать характеристики различных численных методов, а также дать подробное описание метода половинного деления, метода золотого сечения, градиентного метода, метода Ньютона;
- осуществить программные реализации алгоритмов;
- изучить возможности пакета программирования Matlab по решению экстремальных задач;
- разработать программы в среде пакета программирования Matlab для нахождения экстремумов функции.

Объект исследования – задачи на нахождение экстремумов функции. Предмет исследования – численные методы решения (а именно метод половинного деления, метод золотого сечения, градиентный метод, метод

Ньютона) экстремальных задач, то есть задач на нахождение минимума и максимума функции.

Практическая значимость данной работы заключается в разработке программ в среде пакета программирования Matlab по нахождению экстремумов функции.

Для того чтобы достичь поставленной цели, разбираются некоторые мелкие задачи, которые решаются в процессе написания одной из глав.

В первой главе работы рассматривается теоретический обзор по обзоре состояния вопроса и задачи исследования, включаются в следующие:

- Общая постановка задачи.
- Существующие методы нахождения экстремумов функций.
- Выбор среды разработки ПО.

В второй главе работы рассматривается описание методов минимизации функций одной переменной, включаются в следующие:

- Метод половинного деления.
- Программная реализация метода половинного деления.
- Метод золотого сечения.
- Программная реализация метода золотого сечения.

В третьей главе работы рассматривается описание методов минимизации для функций нескольких переменных, включаются в следующие:

- Градиентный метод.
- Программная реализация градиентного метода.
- Метод Ньютона.
- Программная реализация метода Ньютона.

В четвертой главе представляет собой практическая часть, которая содержит:

- Создание ПО.
- Тестирование созданного ПО.

1. ОБЗОР СОСТОЯНИЯ ВОПРОСА И ЗАДАЧИ ИССЛЕДОВАНИЯ

1.1 Общая постановка задачи

При постановке задачи поиска минимума функций необходимо следущие данные:

- функция $f(x)$, где $x = (x_1, x_2, \ldots, x_n)^T$, определенную на n – мерном евклидовом пространстве R^n, которая называется целевой. Степень достижения цели, которая решается задача, определяется значениями этой функции;
- множество векторов $X \subseteq R^n$, среди элементов которых осуществляется поиск допустимых решений. Данное множество будем называть областью допустимых решений [1].

Задача нахождения минимума целевой функции на области допустимых решений заключается в нахождении вектора x^* из множества векторов $X \subseteq R^n$, для которого целевая функция $f(x)$ достигает минимального значения на этом множестве:

$$f(x^*) = \min_{x \in X} f(x)$$

Чтобы решить задачу на максимума функции $f(x)$, необходимо рассмотреть задачу нахождения минимума функции $-f(x)$, т.е. заменить знак функции на противоположный (рис. 1.1).

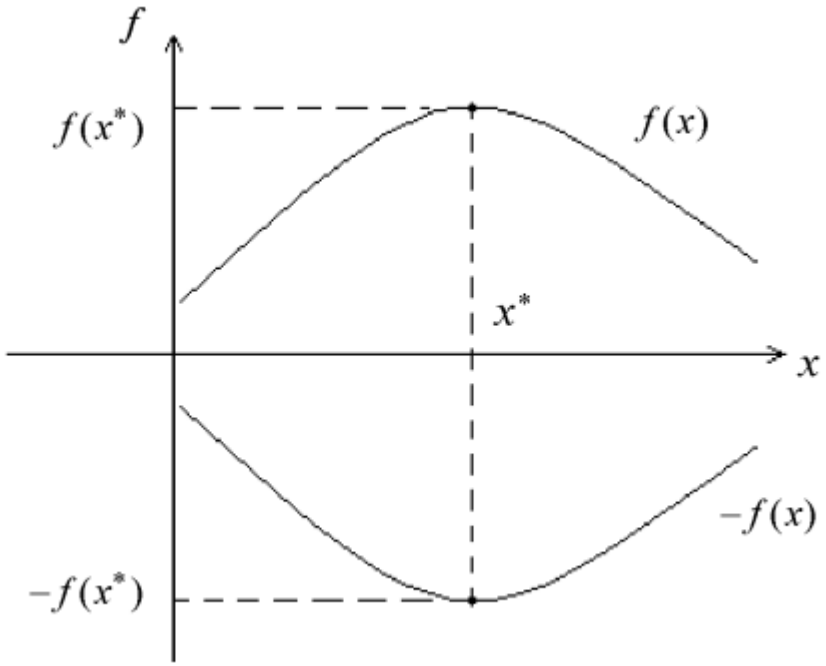

Рис. 1.1

Задача поиска минимума или максимума целевой функции $f(x)$ составляет задачу поиска экстремума:

$$f(x^*) = extr\, f(x).$$
$$x \in X$$

Мы получим задачу условного экстремума, если на множество допустимых решений X наложим ограничения (условия). Таким образом, если $X = R^n$, т.е. ограничения (условия) на множество допустимых решений X отсутствуют, то переходим к решению задачи поиска безусловного экстремума – пара $(x^*, f(x^*))$, включающая точку x^* и значение целевой функции $f(x^*)$ в этой точке.

Обозначим через X^* множество точек минимума (максимума) целевой функции $f(x)$ на множестве X. X^* может содержить конечное количество точек, бесконечное количество точек или быть пустым.

Точка $x^* \in X$ называется точкой глобального (абсолютного) минимума функции $f(x)$ на множестве X, если функция достигает в этой точке своего наименьшего значения, т.е.

$$f(x^*) \le f(x) \qquad \forall\, x \in X.$$

Точка $x^* \in X$ называется точкой локального (относительного) минимума функции $f(x)$ на множестве X, если существует $\varepsilon > 0$, такое, что если $x \in X$ и $\|x - x^*\| < \varepsilon$, то $f(x^*) \leq f(x)$. Здесь $\|x\| = \sqrt{x_1^2 + x_2^2 + x_3^2 + \cdots + x_n^2}$ −евклидова норма вектора x.

Точка x^* глобального минимума сравнивается по величине функции со всеми точками из множества допустимых решений X, а точка x^* локального минимума только с принадлежащими её ε − окрестности (рис. 1.2)[1].

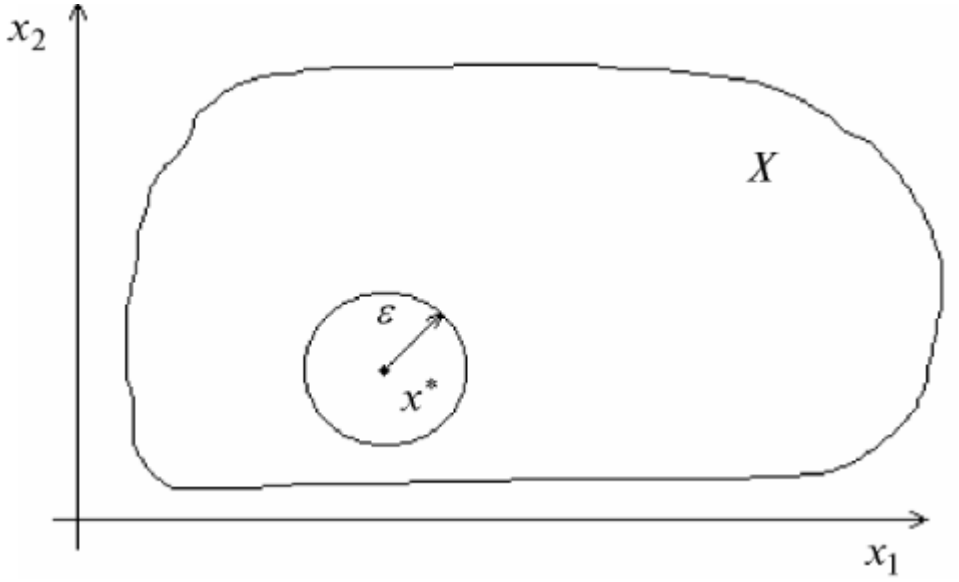

Рис. 1.2

1.2 Существующие методы нахождения экстремумов функций

Под понятием экстремальных задач понимают задачи, которые ориентированы на поиск и нахождение максимального или минимального, т.е. экстремального некоторой определенной функции.

Основным отличием численных методов нахождения экстремума функции от аналитических является то, что с их помощью можно получить только приближенное решение. Точность найденных точек минимума и максимума зависит от точности метода, числа итераций и др.

Численные методы поиска и нахождения экстремумов функции одной переменной можно классифицировать следующим образом:

- методы нулевого порядка, которые используют для нахождения необходимых экстремумов только лишь значения функции и не требуют вычисления ни одной из ее производных;
- методы первого и более высоких порядков, которые требуют для поиска и нахождения максимального и минимального значения функций использование производных.

К главным достоинства нулевых (прямых) методов можно отнести следующее:

- позволяют исследовать и находить решение даже недифференцируемых целевых функций;
- имеют в своей основе достаточно простые алгоритмы и программы оптимизации, которые позволяют легко вычислить любую точку экстремума и значение функции в этой точке;
- требуют относительно малого объема машинной памяти, что позволяет экономить пространство на электронном носителе.

К недостаткам прямых методов относят:

- требуют длительного времени работы компьютера, так как метод и стратегия поиска экстремума является не самой лучшей;
- прямые методы дают очень низкую точность, , для получения высокой точности нужно совершить достаточно большое количество вычислений, в случае ограничения на число вычислений получить высокую точность вычисления становится практически невозможным.

Говорят, что функция имеет локальный (т.е. местный) минимум или максимум в какой-либо известной точке x_0; только при условии существования некоторого δ такого, что для любого x из неравенства $|x - x_0| < \delta$ выполняется равенство:

$$f(x^*) \leq f(x)\ (f(x^*) \geq f(x))$$

В большинстве случаев рассматривают такие задачи по нахождению и поиску условного экстремума, которые требуют дополнительных специфических ограничений. В частности, подобного рода задачами можно считать задачи математического программирования.

Унимодальной на некотором отрезке является функция, для которой существует такая точка, делящая его на различные две части, что в одной части функция убывает, а в другой соответственно возрастает.

Для корректного вычисления решения задач численными методами обязательно условие унимодальности данной функции на всем рассматриваемом отрезке.

Стационарной точка x называется только при выполнение следующего условия:

$$f'(x^*) = 0$$

Стационарная точка является локальной минимальной точкой для дважды дифференцируемой функции при выполнении следующего неравенства:

$$f''(x^*) > 0$$

Ниже рассмотрим основные классы численных методов.

Одним из основных является метод половинного деления (дихотомии). Основная идея метода дихотомии состоит в постепенном делении отрезка на две части, часть, не содержащая минимума или максимума функции, отбрасывается. Следует отметить, что для использования данного метода необходимо, чтобы функция была унимодальна на всей длине данного отрезка.

Стоит отметить, что уменьшение длины данного отрезка должно проходить методом выбора двух точек x_1 и x_2, которые располагаются симметрично по отношению к середине отрезка

$$x = \frac{a+b}{2}$$

Абсциссы этих точек находятся по формулам:

$$x_1 = x - \frac{\delta}{2};\ x_2 = x + \frac{\delta}{2}$$

δ – это величина различимости точек (δ < ε). Наиболее подробно данный метод рассматривается в следующем разделе [2].

Метод равномерного поиска (метод перебора) можно отнести к пассивным стратегиям. Для начала необходимо задать начальный интервал $L_0 = [a_0, b_0]$ неопределенности, а также число N вычисляемых значений функции на этом отрезке. Потом разделим интервал L_0 на $N + 1$ равных отрезков N точками. Далее происходит процесс сравнения величин, а затем находится искомая точка, в которой значение функции принимает экстремальное значение (как минимальное, так и максимальное).

Метод деления пополам даёт возможность исключать в точности половину интервала на этапе абсолютно каждой итерации. В редких случаях этот метод называют трехточечным поиском на равных интервалах, так как его осуществление происходит на как можно более точном выборе трех пробных точек, которые относительно равномерно распределены в определенном интервале поиска. Наиболее подробно данный метод рассматривается в следующем разделе [3].

Метод поразрядного поиска

Метод поразрядного поиска имеет вид усовершенствованного метода перебора. С помощью переменного шага, можно найти точку минимума функции.

Изначально шаг предполагается очень большим и сложно найти отрезок, в котором присутствует точка минимума. Но с заданием высокой точности, возможно отыскать точку минимума на данном отрезке.

Данную идею возможно реализовать с помощью метода поразрядного поиска таким образом: выбор точек лежащих на отрезке осуществляется с шагом:

$$\Delta = x_{i+1} - x_i > \varepsilon$$

до того момента, когда, начнет увеличиваться функция, то есть условие:

$$f(x_{i+1}) \geq f(x_i)$$

не выполнится, или же нет совпадения первой точки с правым концом отрезка [a, b], на котором происходит поиск минимума функции. Затем шаг становится меньше, а выбор точек происходит справа налево, до того момента, когда значения функции не начнут увеличиваться.

Процесс можно считать законченным только в том случае, если выбор точек по заданному направлению завершен, а значение шага Δ не превышает значение точности ε.

1.3 Выбор среды разработки ПО

На сегодняшнее время язык программирования C# является самым мощным, быстро развивающимся и востребованным языком в ИТ–отрасли. В настоящее время на этом языке C# пишут многие приложения: от небольших десктопных программок до крупных веб-порталов и веб-сервисов, обслуживающих ежедневно миллионы пользователей [28].

C# язык подобен синтаксисом с C, близок к C++ и Java. Это облегчает работу с C#.

Java и C++ дали возможность быть объектно-ориентированным C#. К примеру, C# поддерживает статическую типизацию, полиморфизм, перегрузку операторов, наследование. Объектно-ориентированные подходы позволяют решить задачи по построению крупных, но в тот же момент гибких, расширяемых и масштабируемых приложений. И язык C# активно развивается, и каждая новая версия имеет больше интересных возможностей и функций, таких как асинхронные методы, динамическое связывание, и т.д.[30].

Язык C# имеет массу преимуществ: объектная ориентированность, простота, типовая защищенность, поддержка совместимости версий, «сборка мусора» и многое другое. Это всё дает возможность для быстроты и легкости разработки приложений. Учитывая достижения многих других языков программирования: C++, C, Java, Visual Basic авторы C# создали проще, удобнее и современнее язык, который не уступает C++, но при этом существенно повышает продуктивность разработок [28].

На C# программа содержит один или несколькие файлы. Каждый файл может состоить из одного или нескольких пространств имен. Каждое пространство имен может содержать вложенные пространства имени типы, такие как классы, структуры, интерфейсы, перечисления и делегаты - функциональные типы.

На языке C# возможно создать приложения для Windows Forms. Windows Forms дает очень простые и в тоже время мощные механизмы для управления графикой. Форма имеет вид экранного объекта, обычно в прямоугольной форме, который позволяет предоставлять информацию пользователю и вводить информацию от пользователя. Формы представляют собой следующие виды: стандартное диалоговое окно, много документного интерфейса (MDI) или поверхности для отображения графической информации. Разместить компоненты управления на поверхности формы – самый простой способ задания поверхности [30].

Форма – это объект, который содежит свойство для определения их внешного вида, методы для определения их вида, событие для определения их взаимодействия с пользователем. Форма имеет два окна программы для процесса создания приложения: окно дизайнера форм для отображения графического, которого представляет визуальных компонента формы и окно кода программы, в котором все данные вашей программы хранятся [28].

Дальше будем рассматривать еще одно хорошее программное обеспечение, которое используется по всему миру.

Matlab – это системный пакет программ, использующийся в инженерии и в научных вычислениях. В нем присутствует программы, связанные с различными математическими вычислениями. В среде Matlab возможна визуализация процессов и объектов исследования, построение графиков различных функций. Данная среда служит для построения математических моделей различных процессов. Система Matlab может применятся в следующих научных областях:

- математика и вычисления;
- анализ данных, исследование и визуализация результатов;
- вычислительный эксперимент, имитационное моделирование, макетирование;
- разработка алгоритмов;
- разработка приложений, включая графический интерфейс пользователя;
- научная и инженерная графика.

Программирование в среде Matlab осуществляется на основе векторов, представляющих собой массив данных различной размерности. Т.е. существует возможность решать вычислительные вычислительные задачи с данными, заданными в векторно-матричном виде.

Система Matlab имеет 2 вида м–файлов:

- скрипты – это последовательность команд (представляют собой процедуры);
- Function –это функция с входными аргументами и выходными параметрами (значениями функции).

При решении других проблем может быть сложно визуализировать процесс, то есть переменная динамически изменяется в процессе решения задачи.

Все эти и другие трудности могут быть решены с помощью графического интерфейса пользователя. (GUI-GraphicalUserInterface).

При использовании GUI дает возможность для создания программы более универсальной.

После создания интерфейса появляются два файла: fig-файл и m-файл. fig-файл – это «фигура» самого интерфейса и m-файл создается самим Matlab и содержит программный код всех элементов интерфейса.

В составе MatLab имеет среда GUIDE, которая позволяет создать приложения с графическим интерфейсом пользователя. Работа в GUIDE достаточно проста – компоненты управления (кнопки, выпадающие списки и т. д.) размещаются с помощью мыши, а затем события программируются, которые возникают, когда пользователь обращается к этим элементам управления.

Преимущество среды GUIDE заключается в том, что легко сделать простой графический интерфейс. Весь код для интерфейса создается самим Matlab. Для работы с частью программы GUI вам необходимо изучить принцип обмена данными при использовании команд setappdata и getappdata (который является стандартным методом обмена данными между различными элементами GUI).

Процесс построения графического интерфейса пользователя представляет собой три этапы:

- Постановка задачи.
- Создание формы интерфейса и создание на неё компонентов управления.
- Написание кода программы и кода обработки событий.

Таким образом, для разработки программы пользовательского интерфейса, с которой решены назначенные задачи в этой работе, обе программы, описанные выше подходят [29].

2. МЕТОДЫ МИНИМИЗАЦИИ ФУНКЦИЙ ОДНОЙ ПЕРЕМЕННОЙ

2.1. Метод половинного деления

Метод половинного деления – это самый простой для нахождения решения нелинейных уравнений численный метод. Данный метод предполагает, что функция $J(u)$ непрерывна. Основа данного метода располагается на теореме о промежуточных значениях. Таким образом, если необходимо найти значение минимума или максимума, то необходимо, чтобы функция имела на концах заданного отрезка противоположные знаки. Разделив данный отрезок пополам и взяв ту из частей, на концах которой значение функции также имеет противоположный знак. Если значение функции в точке, которая находится в середине, является искомым экстремумом, то процесс можно считать завершенным [1].

Опишем этот метод, пусть минимизируемая функция $J(u)$ унимодальна на отрезке [a, b]. Нахождение минимума $J(u)$ на отрезке [a, b] начинается с выбора двух точек $u_1 = \frac{a+b-\delta}{2}$ и $u_2 = \frac{a+b+\delta}{2}$, где $\delta -$ постоянный параметр метода, $0 < \delta < b - a$. Значение параметра δ выбирается произвольно и может быть определено соответствующим числом правильных десятичных цифр при задании аргумента. Ясно, что δ всегда больше машинного нуля ЭВМ, используемой при решении рассматриваемой задачи. Точки u_1и u_2 симметрично расположены на [a, b] относительно его середины и делят его почти пополам при малых $\delta -$ это объясняет название метода [4].

После выбора точек u_1 и u_2, значения $J(u_1),\ J(u_2)$ вычисляются и сравниваются друг с другом. Если $J(u_1) \leq J(u_2)$, то пусть $a_1 = a,\ b_1 = u_2$; если $J(u_1) > J(u_2)$, то предположим что $a_1 = u_1,\ b_1 = b$. Необходимо, что функция $J(u)$ была унимодальна на отрезке [a, b], то ясно, что отрезок [a, b] имеет общую точку с множеством U_* точек минимума $J(u)$ на [a, b] и его длину равно

$$b_1 - a_1 = \frac{b-a-\delta}{2} + \delta.$$

Предположим, что отрезок $[a_{k-1}, b_{k-1}]$, имеет непустое пересечение с U_*, уже известен, и пусть $b_{k-1} - a_{k-1} = \frac{b-a-\delta}{2^{k-1}} + \delta > \delta (k \geq 2)$. Тогда возьмём точки $u_{2k-1} = \frac{a_{k-1}+b_{k-1}-\delta}{2}$, $u_{2k} = \frac{a_{k-1}+b_{k-1}+\delta}{2}$, расположенные на отрезке $[a_{k-1}, b_{k-1}]$, симметрично относительно его середины, и вычислим значения $J(u_{2k-1})$, $J(u_{2k})$. Если $J(u_{2k-1}) \leq J(u_{2k})$ то полагаем $a_k = a_{k-1}, b_k = u_{2k}$, если же $J(u_{2k-1}) > J(u_{2k})$, то полагаем $a_k = u_{2k-1}, b_k = b_{k-1}$. Длина получившегося отрезка $[a_k, b_k]$ равна $b_k - a_k = \frac{b-a-\delta}{2^k} + \delta > \delta$ и $[a_k, b_k] \cap U_* \neq \emptyset$ [22].

Если число вычислений значений минимизированной функции не ограничено, то описанный процесс деления интервала пополам может быть продолжать до тех пор, что не будет получить интервал $[a_k, b_k]$ длины $b_k - a_k < \varepsilon$, где $\varepsilon -$ заданная точность, $\varepsilon > \delta$. Отсюда имеем, что $k > log_2(\frac{b-a-\delta}{\varepsilon-\delta})$. Так как каждое деление пополам требует двух вычислений значений функции, то для достижения точности $b_k - a_k < \varepsilon$ нужно всего $n = 2k > 2log_2(\frac{b-a-\delta}{\varepsilon-\delta})$ таких вычислений [23].

После нахождения отрезка $[a_k, b_k]$ в качестве приближения к множеству U_* можно взять точку $\bar{u}_n = u_{2k-1}$ при $J(u_{2k-1}) \leq J(u_{2k})$и $\bar{u}_n = u_{2k}$ при $J(u_{2k-1}) > J(u_{2k})$, а значение $J(\bar{u}_n)$ может служить приближением для $J_* = \inf J(u)$.

$u \in [a, b]$

При этом выборе приближения для U_* будет допущена погрешность $\rho(\bar{u}_n, U_*) \leq \max\{b_k - \bar{u}_n; \bar{u}_n - a_k\} = \frac{b-a-\delta}{2^k}$. Если не будем требовать того, чтобы значение функции, приниятой в качестве приближения к J_*, вычислялось в той же точке, которая служит приближением к U_*, то вместо

$\bar{u}_n$ можем взять точку $v_n = \frac{a_k+b_k}{2}$ с меньшей погрешностью $\rho(\bar{u}_n, U_*) \leq \frac{b_k-a_k}{2} = \frac{b-a-\delta}{2^{k+1}} + \frac{\delta}{2}$ ($k = \frac{n}{2}$ и δ достаточно мало) [24].

2.2. Программная реализация метода половинного деления

Метод половинного деления относится к последовательным стратегиям. Задаются пользователем начальный интервал и точности. Алгоритм основан на анализ значений функции в двух точках. Чтобы найти их, необходимо делить данный интервал неопределенности пополам и по обе стороны от середины сохраняется в $\frac{\varepsilon}{2}$, где $\varepsilon -$ небольшое положительное число. Если длина текущего интервала неопределенности меньше установленного величины, то процесса поиска стандартные: поиск заканчивается [10].

Схема решения задачи методом половинного деления:

Этап 1. Задается начальный интервал неопределенности $L_0 = [a_0, b_0]$, $\varepsilon -$ малое число, $l > 0$ −точность.

Этап 2. Примем $k = 0$.

Этап 3. Вычислим $y_k = \frac{a_k+b_k-\varepsilon}{2}$, $f(y_k), z_k = \frac{a_k+b_k+\varepsilon}{2}$, $f(z_k)$.

Этап 4. Сравним $f(y_k)$ и $f(z_k)$:

1) если $f(y_k) \leq f(z_k)$, то положим $a_{k+1} = a_k, b_{k+1} = z_k$ (рис. 2.1, а) и перейдём к этапу 5;

2) если $f(y_k) > f(z_k)$, то положим $a_{k+1} = y_k, b_{k+1} = b_k$ (рис. 2.1, б).

Этап 5. Вычислим $\Delta = |b_{k+1} - a_{k+1}|$ и проверим условие окончания:

а) если $\Delta \leq l$, процесс поиска завершается. Точка минимума $x^* \in [a_{k+1}, b_{k+1}]$. Значение приближенного решения определяется по формуле:

$$x^* \cong \frac{a_{k+1}+b_{k+1}}{2};$$

б) если $\Delta > l$, положим $k = k + 1$ и перейдём к этапу 3[11].

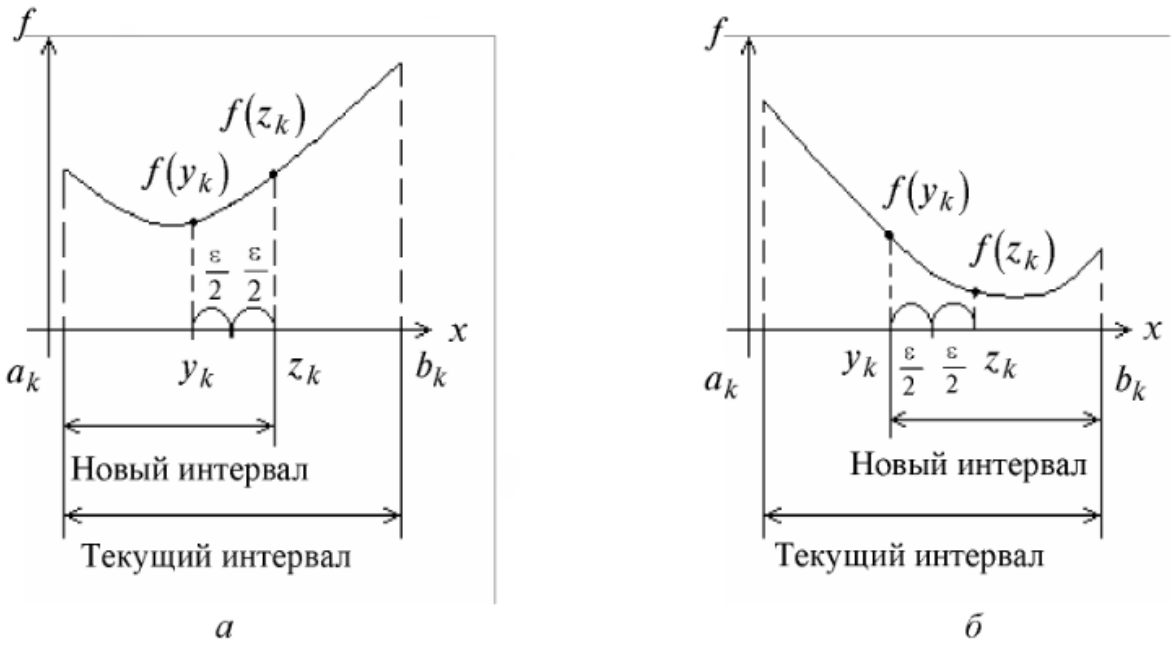

Рис. 2.1

2.3. Метод золотого сечения

Идея метода с золотого сечения остоит в постепенном делении отрезка на две неравные части так, чтобы отношение длины всего отрезка к длине большей части равнялось отношению длины большей части к длине меньшей части отрезка.

Известно, что золотое сечение интервала [a, b] производится двумя точками:

$$u_1 = a + \frac{3-\sqrt{5}}{2}(b-a) = a + 0.3819\ldots(b-a);$$

$$u_2 = a + \frac{\sqrt{5}-1}{2}(b-a) = a + 0.6180\ldots(b-a),$$

расположенными симметрично относительно от середины интервала, причем $a < u_1 < u_2 < b, \frac{b-a}{b-u_1} = \frac{b-u_1}{u_1-a} = \frac{b-a}{u_2-a} = \frac{u_2-a}{b-u_2} = \frac{\sqrt{5}+1}{2} = 1.618033989\ldots$

Замечательно здесь то, что точка и, в свою очередь производит золотое сечение интервала $[a, u_2]$, так как $u_2 - u_1 < u_1 - a = b - u_2$ и $\frac{u_2-a}{u_1-a} = \frac{u_1-a}{u_2-u_1}$. Аналогично точка u_2 производит золотое сечение отрезка $[u_1, b]$. На

основании этого свойства золотого сечения, мы можем предложить следующий метод для минимизации функции унимодального $J(u)$ на интервале [a, b] [21].

Положить $a_1 = a, b_1 = b$. На интервале $[a_1,\ b_1]$ взять точки u_1 и u_2, которые производят золотое сечение, и вычислить значения $J(u_1)$, $J(u_2)$. Далее, если $J(u_1) \leq J(u_2)$, то положить $a_2 = a_1,\ b_2 = u_2,\ \bar{u}_2 = u_1$; если $J(u_1) > J(u_2)$, то положить $a_2 = u_1,\ b_2 = b_1,\ \bar{u}_2 = u_2$. Поскольку функция $J(u)$ унимодальна на интервале[a, b], то интервал $[a_2,\ b_2]$ имеет по крайней мере одну общую точку с множеством U_* точек минимума функции $J(u)$ на [a, b]. Кроме того, $b_2 - a_2 = \frac{\sqrt{5}-1}{2}(b-a)$ и важно то, что внутри $[a_2,\ b_2]$ была точка $\bar{u}_2$ с вычисленным значением $J(\bar{u}_2) = \min\{J(u_1), J(u_2)\}$, которая производит золотое сечение отрезка $[a_2,\ b_2]$ [18].

Пусть уже определены точки $u_1, u_2, \dots, u_{n-1}$ вычислены значения $J(u_1)$, ..., $J(u_{n-1})$, найден отрезок $[a_{n-1},\ b_{n-1}]$ такой, что $[a_{n-1},\ b_{n-1}] \cap U_* \neq \emptyset$, $b_{n-1} - a_{n-1} = \left(\frac{\sqrt{5}-1}{2}\right)^{n-2}(b-a)$, и известна точка $\bar{u}_{n-1}$, производящая золотое сечение отрезка $[a_{n-1},\ b_{n-1}]$ и такая, что $J(\bar{u}_{n-1}) = \min\limits_{1 \leq i \leq n-1} J(u_i) (n \geq 2)$.

Тогда в качестве следующей точки возьмем точку $u_n = a_{n-1} + b_{n-1} - \bar{u}_{n-1}$, также производящую золотое сечение отрезка $[a_{n-1},\ b_{n-1}]$ вычислим значение $J(u_n)$.

Предположим для определенности, что $a_{n-1} < u_n < \bar{u}_{n-1} < b_{n-1}$ (случай $\bar{u}_{n-1} < u_n$ рассматривается аналогично). Если $J(u_n) \leq J(\bar{u}_{n-1})$, то полагаем $a_n = a_{n-1},\ b_n = \bar{u}_{n-1}, \bar{u}_n = u_n$; если $J(u_n) > J(\bar{u}_{n-1})$, то полагаем $a_n = u_n, b_n = b_{n-1}, \bar{u}_n = \bar{u}_{n-1}$. Новый отрезок $[a_n,\ b_n]$ таков, что $[a_n,\ b_n] \cap U_* \neq \emptyset, b_n - a_n = \left(\frac{\sqrt{5}-1}{2}\right)^{n-1}(b-a)$, точка $\bar{u}_n$, производит золотое сечение $[a_n,\ b_n]$ и $J(\bar{u}_{n-1}) = \min\{J(u_n); J(\bar{u}_{n-1})\} = \min J(u_i)\ (1 \leq i \leq n-1)$ [16].

Если количество вычислений значений $J(u)$ не ограничено заранее, то описанный процесс можно продолжить, например, до тех пор, пока не выполнено неравенство $b_n - a_n < \varepsilon$, где $\varepsilon -$ требуемая точность. Если количество вычислений значении функции $J(u)$ предварительно фиксировано и равно n, то процесс заканчивается и как решение проблемы второго типа, мы можем взять пару $J(\bar{u}_n), \bar{u}_n$, где $J(\bar{u}_n)$ является приближением для $J_* = inf \quad J(u)(u \in [a,b])$, а точка $\bar{u}_n$ служит приближением для множества U_* с погрешностью

$\rho(\bar{u}_n, U_*) \leq \max\{b_n - \bar{u}_n; \bar{u}_n - a_n\} = \frac{1}{2}(\sqrt{5}-1)(b_n - a_n) =$

$= \left(\frac{\sqrt{5}-1}{2}\right)^n (b-a) = A_n.$

Ранее было показано, с помощью метода половинного деления за $n = 2k$ вычислений значений функции $J(u)$ в аналогичном случае было получить точку $\bar{u}_n$ с погрешностью

$$\rho(\bar{u}_n, U_*) \leq 2^{-n/2}(b-a-\delta) < 2^{-n/2}(b-a) = B_n.$$

Отсюда имеем $A_n/B_n = (\frac{2\sqrt{2}}{\sqrt{5}+1})^n \approx (0.87\ldots)^n -$ видно, что уже при небольших n преимущество метода золотого сечения перед методом деления отрезка пополам становится ощутимым [22].

Рассмотрим возможности численной реализации метода золотого сечения на компьютере. Заметим, что число $\sqrt{5}$ в компьютере неизбежно будет задаваться приближенно, поэтому первая точка $u_1 = a + \frac{3-\sqrt{5}}{2}(b-a)$ будет найдена с некоторой погрешностью. Посмотрим, как повлияет эта погрешность на результаты последующих шагов метода золотого сечения. Обозначим $\Delta_n = b_n - a_n = \left(\frac{\sqrt{5}-1}{2}\right)^{n-1}(b-a)$. Нетрудно проверим, что Δ_n является решением конечно-разностного уравнения $\Delta_{n-2} = \Delta_{n-1} + \Delta_n$, или

$\Delta_n = \Delta_{n-2} - \Delta_{n-1} (n = 3, 4, \ldots (1).$

с начальными условиями $\Delta_1 = b - a, \Delta_2 = b - u_1.$

Как известно, линейно независимые частные решения этого уравнения имеют τ_1^n и τ_2^n ($n = 1, 2, ...$), где $\tau_1 = \frac{\sqrt{5}-1}{2}$, $\tau_2 = \frac{\sqrt{5}+1}{2}$ – корни характеристического уравнения $\tau^2 + \tau - 1 = 0$, а любое решение ура (2) я (1) представимо в виде

$$\Delta_n = A\tau_1^n + B\tau_n^2,\ n = 1, 2, ...$$

где постоянные A и B однозначно определяются начальными условиями из линейной системы [2]

$$A\tau_1 + B\tau_2 = \Delta_1, A\tau_1^2 + B\tau_2^2 = \Delta_2. \quad (3)$$

2.4. Программная реализация метода золотого сечения

Метод относится к последовательным стратегиям. Задаются пользователем начальный интервал и точность. Алгоритм уменьшения интервала основан на анализе значения функции в двух точках. Точки золотого сечения выбираются в качестве точек вычисления функции Затем, принимая во внимание свойства золотого сечения на каждой итерации, кроме первой, требуется только один новый расчет значения функции. Условия окончания процесса поиска стандартные: поиск заканчивается, когда длина текущего интервала неопределенности оказывается меньше установленной величины [1].

Схема решения задачи методом золотого сечения:

Этап 1. Задается начальный интервал неопределенности $L_0 = [a_0, b_0], \varepsilon > 0$ – точность.

Этап 2. Примем $k = 0$.

Этап 3. Вычислим

$$y_0 = a_0 + \frac{3-\sqrt{5}}{2}(b_0 - a_0); z_0 = a_0 + b_0 - y_0.$$

Этап 4. Вычислим $f(y_k)$, $f(z_k)$.

Этап 5. Сравним $f(y_k)$ и $f(z_k)$:

а) если $f(y_k) \leq f(z_k)$, положим $a_{k+1} = a_k, b_{k+1} = z_k$ и $y_{k+1} = a_{k+1} + b_{k+1} - y_k, z_{k+1} = y_k$ (рис. 2.2, а) и перейдём к этапу 6;

б) если $f(y_k) > f(z_k)$, то положим $a_{k+1} = y_k, b_{k+1} = b_k$ и $y_{k+1} = z_k, z_{k+1} = a_{k+1} + b_{k+1} - z_k$ (рис. 2.2, б).

Этап 6. Вычислим $\Delta = |b_{k+1} - a_{k+1}|$ и проверим условие окончания:

1) если $\Delta \leq \varepsilon$, процесс поиска завершается. Точка минимума $x^* \in [a_{k+1}, b_{k+1}]$. Значение приближенного решения определяется по формуле:

$$x^* \cong \frac{a_{k+1}+b_{k+1}}{2};$$

2) если $\Delta > \varepsilon$, то положим $k = k + 1$ и перейдем к этапу 4 [23].

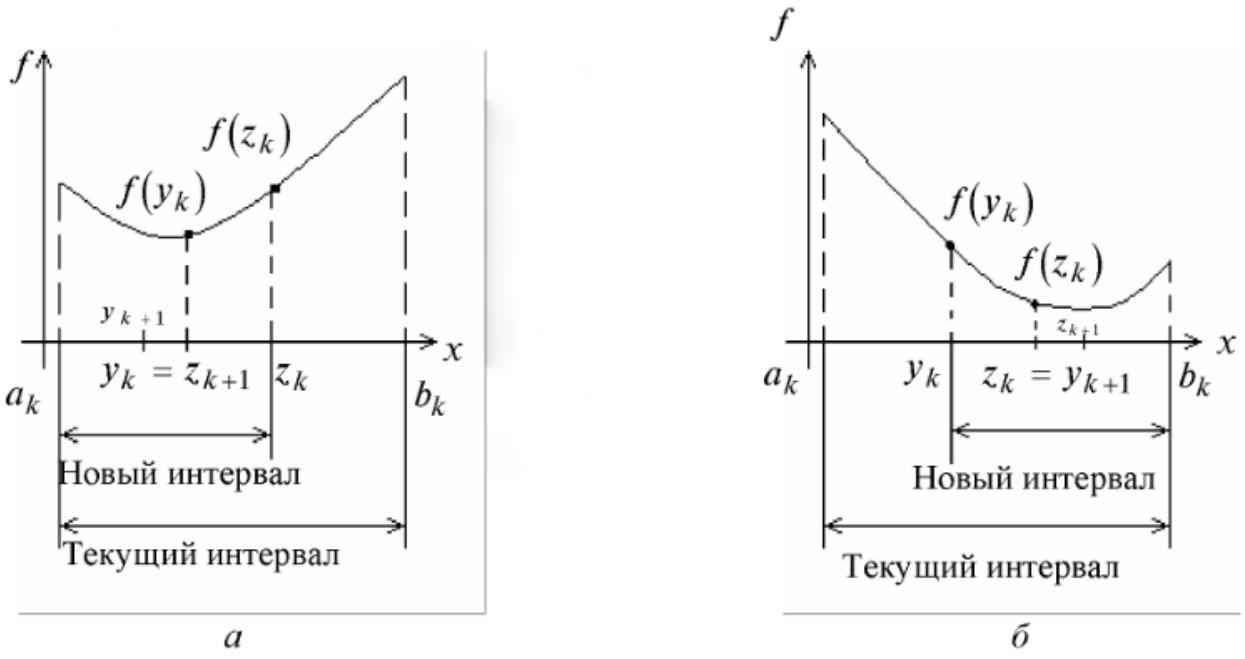

Рис. 2.2

3. МЕТОДЫ МИНИМИЗАЦИИ ДЛЯ ФУНКЦИЙ НЕСКОЛЬКИХ ПЕРЕМЕННЫХ

3.1. Градиентный метод.

Рассматрим задачу

$$J(u) \to inf; u \in U \equiv E^n,$$

предполагать, что функция J(u) непрерывно дифференцируема на E^n, т. е. $J(u) \in C^1(E^n)$. Согласно дифференцируемой фупкцни

$$J(u + h)\text{-}J(u) = \langle J'(u), h\rangle + o(h; u), \qquad (1)$$

где $\lim_{|h| \to 0} o(h; u)|h|^{-1} = 0$. Если $J'(u) \neq 0$, то при достаточно малых $|h|$ главная часть приращения (1) определяется дифференциалом функции $dJ(u) = \langle J'(u), h\rangle$. Справедливо неравенство Коши − Буняковского

$$-|J'(u)|.|h| \leq \langle J'(u), h\rangle \leq |J'(u)|.|h|,$$

причем если $J'(u) \neq 0$, то правое неравенство превращается в равенство только при $h = \alpha J'(u)$, а левое неравенство − только при $J'(u) \neq 0$, где $a = const \geq 0$. Отсюда ясно, что при $J'(u) \neq 0$ направление наибыстрейшего возрастания функции $J(u)$ в точке и совпадает с направлением градиента $J'(u)$, а направление наибыстреншего убывания − с направлением антпградиента $\left(-J'(u)\right)$ [2].

Это свойство градиента, которое лежит в основе ряда итерационных методов минимизации функций. Один из этих методов называется градиентным методом. Этот метод, как и все итерационные методы, предполагают выбор начального приближения − некоторой точки u_0. Общих правил выбора точки u_0 в методе градиента, как, к сожалению, другими методами. В тех случаях, когда априорная информация о местоположении точки (или точек) минимума может быть получена из геометрических,

физических или любых других соображений, начальное приближение u_0 стараются выбрать поближе к этой области [15].

Предположим, что некоторая начальная точка u_0 уже выбрана. Тогда метод градиента состоит в построении последовательности $\{u_k\}$ по правилу

$$u_{k+1} = u_k - \alpha_k J'(u_k), \alpha_k > 0, k = 0,1,2, \ldots \tag{2}$$

Число α_k из (2) часто называют длиной шага или просто шагом градие метода. Если $J'(u_k) \neq 0$, то шаг $\alpha_k > 0$ можно выбрать так, чтобы $J(u_{k+1}) < J(u_k)$. В самом деле, из равенства (1) имеем

$$J(u_{k+1}) - J(u_k) = \alpha_k \left[-|J'(u_k)|^2 + \mathrm{o}(\alpha_k)\alpha_k^{-1}\right] < 0$$

при всех достаточно малых $\alpha_k > 0$. Если $J'(u_k) = 0$, то $u_k -$ стационарная точка. В этом случае процесс (2) прекращается, и при необходимости проводится дополнительное исследование поведения функции в окрестности точки ик для выяснения того, достигается ли в точке u_k минимум функции $J(u)$ или не достигается. В частности, если $J(u) -$ выпуклая функция, то в стационарной точке всегда достигается минимум [7].

Существовать многие способы выбора величины α_k в методе (2). В зависимости от способа выбора α_k можно получить различные варианты градиентного метода.

1) На луче $\{u \in E^n : u = u_k - \alpha_k J'(u_k), \alpha \geq 0\}$, направленном по антиградиенту, введем функцию одной переменной

$$f_k(\alpha) = J(u_k - \alpha_k J'(u_k)), \alpha \geq 0,$$

и определим α_k из условий

$$f_k(\alpha_k) = \inf_{\alpha \geq 0} f_k(\alpha) = f_{k*}, \alpha_k > 0 \tag{3}$$

Метод (2), (3) принято называть методом скорейшего спуска. При $J'(u_k) \neq 0$ согласно $f_k'(0) = -|J'(u_k)|^2 < 0$, поэтому нижняя грань в (3) может достигаться лишь при $\alpha_k > 0$ [14].

2) На практике нередко довольствуются нахождением какого-либо $\alpha_k > 0$, обеспечивающего условие монотонности: $(u_{k+1}) < J(u_k)$. С этой целью задаются какой-либо постоянной $\alpha > 0$ и в методе (2) на каждой итерации берут $\alpha_k = \alpha$. При этом для каждого $k \geq 0$ проверяют условие монотонности, и в случае его нарушения $\alpha_k = \alpha$ дробят до тех пор, пока не восстановится монотонность метода. Время от времени полезно пробовать увеличить α с сохранением условия монотонности.

3) Если функция $J(u) \in C^{1,1}(E^n)$, т. е. $J(u) \in C^1(E^n)$, и градиент $J'(u)$ удовлетворяет условию

$$J'(u) - J'(v) \leq L|u - v|, u, v \in E^n$$

причем константа L известна, то в (2) в качестве α_k может быть взято любое число, удовлетворяющее условиям

$$0 < \varepsilon_0 \leq \alpha_k \leq 2/(L + 2\varepsilon), \tag{4}$$

где $\varepsilon_0, \varepsilon -$ положительные числа, являющиеся параметрами метода. В частности, при $\varepsilon = L/2$, $\varepsilon_0 = 1/L$ получим метод (2) с постоянным шагом $\alpha_k = 1/L$. Отсюда ясно, что если константа L большая или получена с помощью слишком грубых оценок, то шаг α_k в (2) будет маленьким [13].

4) Возможен выбор α_k из условия

$$J(u_k) - J\big(u_k - \alpha_k J'(u_k)\big) \geq \varepsilon\alpha_k |J'(u_k)|^2, \varepsilon > 0. \tag{5}$$

Для удовлетворения условия (5) сначала обычно берут некоторое число а $\alpha_k = \alpha > 0$ (одно и то жо на всех итерациях), а затем при необходимости дробят его, т. е. изменяют по закону $\alpha_k = \omega^i \alpha$ ($i = 0,1,2, \ldots, 0 < \omega < 1$) до тех пор, пока впервые не выполнится условие (5).

Из рис. 3.1 и 3.2 можно попять, что чем ближе линии уровня $J(u) = const$ к окружности, тем лучше сходится метод скорейшего спуска. Это же явление можно усмотреть и чем ближе μ/L к единице (для функции $J(u) =$

$|u|^2$, у которой линиями уровня являются окружности (сферы), как раз имеем $\mu/L = 1$, тем ближе q к нулю и тем лучше сходимость [3].

Те же рис. 3.1 и 3.2 показывают, а теоретические исследования и численные эксперименты подтверждают, что метод скорейшего спуска и другие варианты градиентного метода медленно

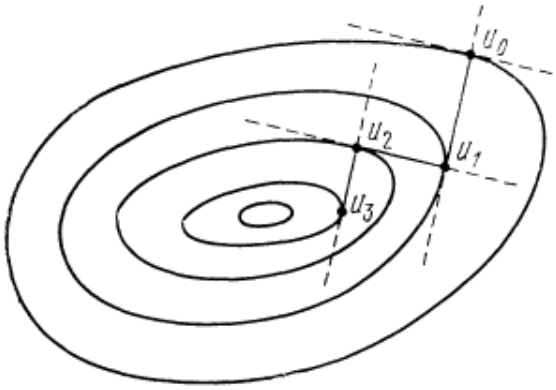

Рис. 3.1

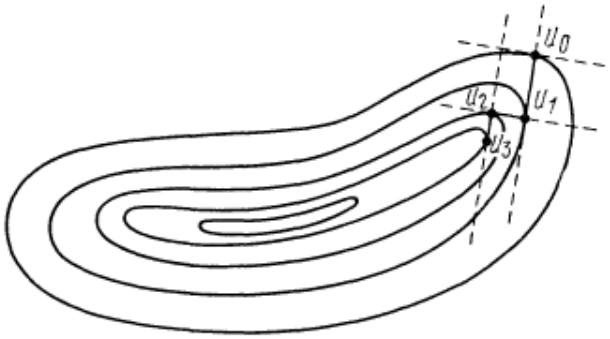

Рис. 3.2

сходятся в тех случаях, когда поверхности уровня функции $J(u)$ сильно вытянуты и функция имеет так называемый «овражный» характер. Это означает, что небольшое изменение некоторых переменных приводит к резкому изменению значений функции – эта группа переменных характеризует «склон оврага», а по остальным переменным, задающим направление «дна оврага», функция меняется незначительно (на рис. 3.2 и 3.3 изображены линии уровня «овражной» функции двух переменных) [18].

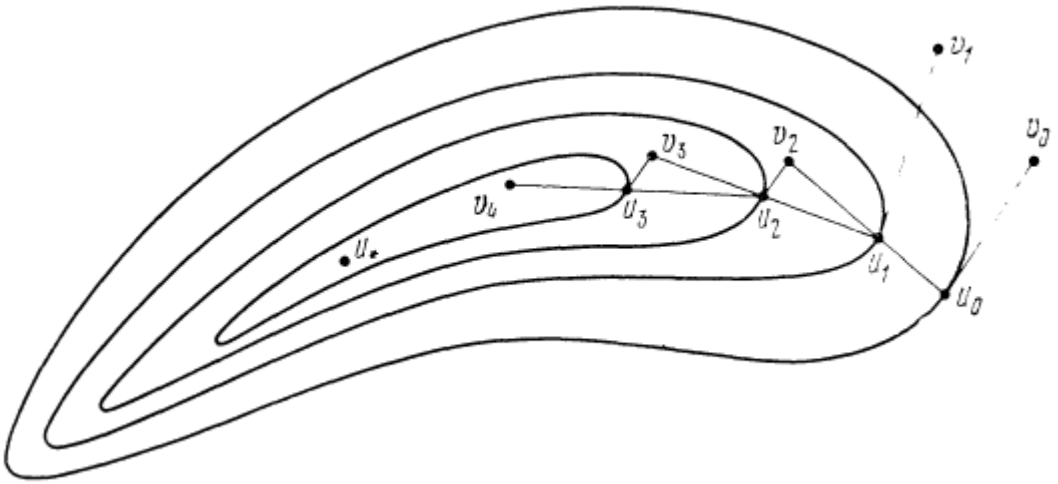

Рис. 3.3

Если точка лежит на «склоне оврага», то направление спуска из этой точки будет почти перпендикулярным к направлению «дна оврага», и в результате аппроксимации $\{u_k\}$, полученной метод градиента, будет попеременно располагаться на одном «склоне оврага». Если «склоны оврага» достаточно круты, то такие скачки «со склона на склон» точек u_k могут значительно замедлить сходимость метода градиента.

Чтобы ускорить сближение этого метода при поиске минимума функции «овражной», мы можем предложить следующий эвристический метод, называемый методом оврага. Сначала мы опишем простейшую версию этого метода. В начале поиска задаются две точки v_0, v_1 из которых производят спуск с помощью некоторого варианта метода градиента, и получить две точки u_0, u_1 на «дне оврага». Затем полагают

$$v_2 = u_1 - (u_1 - u_0)|u_1 - u_0|^{-1} h \operatorname{sign}\big(J(u_1) - J(u_0)\big),$$

где $h -$ положительная постоянная, называемая овражным шагом. Из точки v_2, которая, вообще говоря, находится на «склоно оврага», производят спуск с помощью градиентного метода и определяют следующую точку u_2 на «дне оврага» [17].

Если уже известны точки $u_0, u_1, \dots, u_k (k \geq 2)$, то из точки

$$v_{k+1} = u_k - (u_k - u_{k-1})|u_k - u_{k-1}|^{-1} h \operatorname{sign}\big(J(u_k) - J(u_{k-1})\big),$$

совершают спуск с помощью градиентного метода и находят следующую точку u_{k+1} на «дне оврага» (см. рис. 3.3; спуск из точки v_k в точку u_k, состоящий, быть может, из нескольких итерационных шагов градиентного метода, условно изображен отрезком прямой, соединяющей точки v_k, u_k, $k = 0, 1, 2, \ldots$).

3.2. Программная реализация градиентного метода

Стратегия решения задачи состоит в построении последовательности точек $\{x^k\}$, $k = 0, 1, 2, \ldots$, таких, что $f(x^{k+1}) < f(x^k), k = 0, 1, 2, \ldots$.Точки последовательности $\{x^k\}$ вычисляются по правилу

$$x^{k+1} = x^k - t_k \nabla f(x^k)$$

где точка x^0 задается пользователем; $\nabla f(x^k)$ – градиент функции $f(x)$, вычисленный в точке x^k; величина шага t_k задается пользователем и остается постоянной до тех пор, пока функция убывает в точках последовательности, что контролируется путем проверки выполнения условия:

$f(x^{k+1}) - f(x^k) < 0$ или $f(x^{k+1}) - f(x^k) < -\varepsilon\left\|\nabla f(x^k)\right\|^2, 0 < \varepsilon < 1.$

Построение последовательности $\{x^k\}$ заканчивается в точке x^k, для которой $\left\|\nabla f(x^k)\right\| < \varepsilon_1$, где ε_1 – заданное малое положительное число, или $k \geq M$, где $M -$ предельное число итерации, или при двукратном одновременном выполнении двух неравенств:

$\left\|x^{k+1} - x^k\right\| < \varepsilon_2$ и $\left|f(x^{k+1}) - f(x^k)\right| < \varepsilon_2$, где ε_2 – малое положительное число.

Схема решения задачи градиентным методом

Этап 1. Задается x^0, $0 < \varepsilon < 1$, $\varepsilon_1 > 0$, $\varepsilon_2 > 0, M -$ предельное количество итераций. Вычислим градиент функции в произвольной точке:

$$\nabla f(x) = \left(\frac{\partial f(x)}{\partial x_1}, \ldots, \frac{\partial f(x)}{\partial x_n}\right)$$

Этап 2. Примем $k = 0$.

Этап 3. Вычислим $\nabla f(x^k)$

Этап 4. Проверим выполнение критерия окончания $\left\|\nabla f(x^k)\right\| < \varepsilon_1$:

1) если $\left\|\nabla f(x^k)\right\| < \varepsilon_1$, процесс поиска закончен, $x^* = x^k$;

2) если $\left\|\nabla f(x^k)\right\| \geq \varepsilon_1$, то перейдём к этапу 5.

Этап 5. Проверим $k \geq M$:

1) если $k \geq M$, то процесс поиска окончен: $x^* = x^k$;

2) если $k < M$, то перейдём к этапу 6.

Этап 6. Задается величину шага t_k.

Этап 7. Вычислим $x^{k+1} = x^k - t_k \nabla f(x^k)$.

Этап 8. Проверим выполнение условия

$f(x^{k+1}) - f(x^k) < 0$ (или $f(x^{k+1}) - f(x^k) < -\varepsilon\left\|\nabla f(x^k)\right\|^2$):

1) если условие выполнено, то перейдём к этапу 9;

2) если условие не выполнено, положим $t_k = \frac{t_k}{2}$ и перейдём к этапу 7.

Этап 9. Проверим выполнение условий

$$\left\|x^{k+1} - x^k\right\| < \varepsilon_2 \text{ и } \left|f(x^{k+1}) - f(x^k)\right| < \varepsilon_2:$$

1) если два условия выполнены при текущем значении k и $k = k - 1$, то процесс поиска окончен, $x^* = x^{k+1}$;

2) если хотя бы одно из условий не выполнено, положим $k = k + 1$ и перейдём этапу 3 [19].

Геометрическая интерпретация метода для $n = 2$ приведена на рис. 3.4.

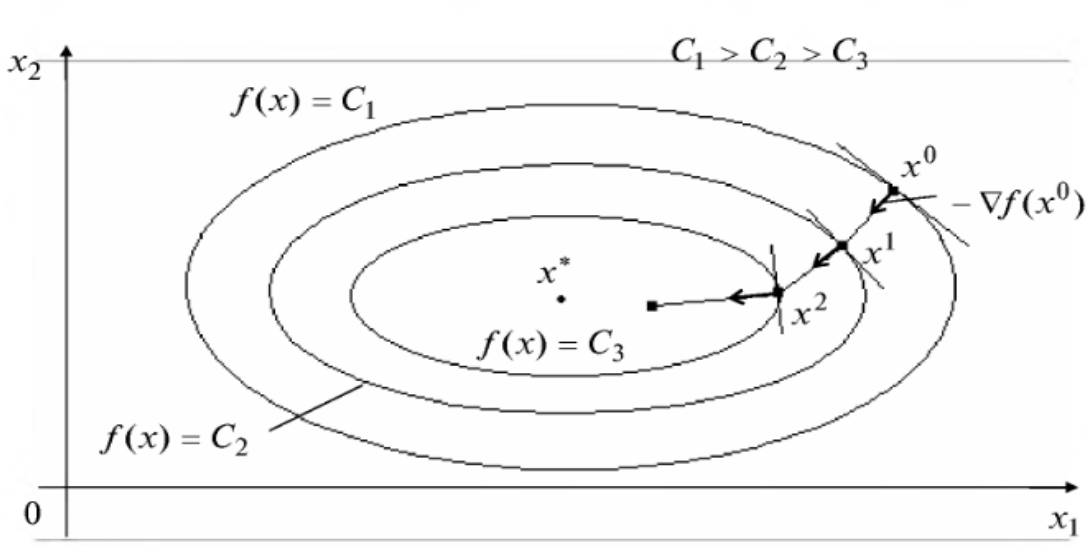

Рис 3.4

3.3. Метод Ньютона

Опишем метод Ньютона для задачи

$$J(u) \to inf; u \in U, \tag{1}$$

где $J(u) \in C^2(U)$, $U -$ выпуклое замкнутое множество из E^n. Пусть $u_0 \in U -$ некоторое начальное приближение. Если известно $k - e$ приближение u_k, то приращение функции $J(u) \in C^2(U)$ в точке u_k можно представить в виде

$$J(u) - J(u_k) = \langle J'(u_k), u - u_k \rangle + \frac{1}{2}\langle J''(u_k)(u - u_k), u - u_k \rangle + o(|u - u_k|^2).$$

Возьмем квадратичную часть этого приращения

$$J_k(u) \equiv \langle J'(u_k), u - u_k \rangle + \frac{1}{2}\langle J''(u_k)(u - u_k), u - u_k \rangle \tag{2}$$

и определим вспомогательное приближение $\bar{u}_k$ из условий

$$\bar{u}_k \in U, J_k(\bar{u}_k) = \inf_U J_k(u). \tag{3}$$

Следующее $(k + 1) - e$ приближение будем искать в виде

(4)

$$u_{k+1} = u_k + \alpha_k(\bar{u}_k - u_k), 0 \le \alpha_k \le 1.$$

В зависимости от способа выбора величины α_k в (4) можно получить различные варианты метода (2) − (4), называемого методом Ньютона. Укажем несколько наиболее употребительных способов выбора α_k [20].

1) В (4) можно принять

$$\alpha_k = 1, k = 0,1,2, \ldots \quad (5)$$

В этом случае, как следует из (4), $u_{k+1} = u_k$ $(k = 0,1,2, \ldots)$, т. е. условие (3) сразу определяет следующее $(k + 1)$-е приближение. Иначе говоря,

$$u_{k+1} \in U, J_k(u_{k+1}) = \inf J_k(u). \quad (6)$$

В частности, когда $U = E^n$, в точке минимума функции $J_k(u)$ ее производная $J_k'(u)$ обращается в нуль, т. е.

$$J_k'(u_{k+1}) = J'(u_k) + J''(u_k)(u_{k+1} - u_k) = 0 \quad (7)$$

Это значит, что на каждой итерации метода (2) − (5) пли (6) нужно решать линейную алгебраическую систему уравнении (7) относительно неизвестной разпости $u_{k+1} - u_k$. Если матрица этой системы $J''(u_k)$ − невырожденная, то из (7) имеем

$$u_{k+1} = u_k - \left(J''(u_k)\right)^{-1} J'(u_k), k = 0,1,2, \ldots \quad (8)$$

Широко известный метод Ньютона для решения системы уравнений

$$F(u) = \{F_1(u), \ldots, F_n(u)\} = 0, u \in E^n,$$

представляет собой итерационный процесс

$$u_{k+1} = u_k - \left(F'(u_k)\right)^{-1} F(u_k) \quad (9)$$

где $F'(u_k)$ − матрица, i-я строка которой равна $F_i' = (F_{iu^1}, \ldots, F_{iu^n})$. Сравнение формул (8) и (9) показывает, что метод (8) решения задачи (1) в случае $U = E^n$ представляет собой известный метод Ньютона для решения

уравнения $J'(u) = 0$, определяющего стационарные точки функции $J(u)$. Отсюда происходит название метода (2) − (4) и в общем случае [23].

2) В качестве α_k в (4) можно принять $\alpha_k = \omega^{i_0}$, где i_0 − минимальный среди $i \geq 0$ номер, для которых выполняется неравенство

$$J(u_k) - J(u_k + \omega^i(\bar{u}_k - u_k)) \geq \varepsilon\omega^i|J_k(\bar{u}_k)|,$$

Где ω, ε − параметры метода, $0 < \omega; \varepsilon < 1$.

3) Возможен выбор α_k в (4) из условий [3]

$$0 \leq \alpha_k \leq 1, f_k(\alpha_k) = \min f_k(\alpha), f_k(\alpha) = J\big(u_k + \alpha(\bar{u}_k - u_k)\big).$$
$$0 \leq \alpha \leq 1$$

3.4. Программная реализация метода Ньютона

Стратегия метода Ньютона (Newton) состоит в построении последовательности точек $\{x^k\}$, $k = 0, 1, 2, ...$, таких, что $f(x^{k+1}) < f(x^k), k = 0, 1, 2, ...$.Точки последовательности $\{x^k\}$ вычисляются по правилу

$$x^{k+1} = x^k + d^k, \qquad k = 0, 1, 2, ...,$$

где точка x^0 задается пользователем; а направление спуска d^k определяется для каждого значения k по формуле

$$d^k = -H^{-1}(x^k)\nabla f(x^k) \qquad (3.1)$$

Выбор d^k по формуле (3.1) гарантирует выполнение требования $f(x^{k+1}) < f(x^k)$, при условии, что $H(x^k) > 0$. Формула (3.1) получена из следующих соображений:

1) функция $f(x)$ аппроксимируется в каждой точке последовательности $\{x^k\}$ квадратичной функцией $F_k = f(x^k) + (\nabla f(x^k), d^k) + \frac{1}{2}(d^k, H(x^k)d^k)$;

2) направление d^k определяется из необходимого условия экстремума первого порядка: $\frac{dF_k}{dd^k} = 0$. Таким образом, при выполнении требования $H(x^k) > 0$ последовательность является последовательностью точек минимумов квадратичных функций F_k, $k = 0, 1, 2, ...$(рис.3.5). Чтобы обеспечить выполнение требования $f(x^{k+1}) < f(x^k)$, $k = 0, 1, 2, ...$ даже в тех

случаях, когда для каких-либо значений матрица Гессе $H(x^k)$ не окажется положительно определенной, рекомендуется для соответствующих значений k вычислить точку x^{k+1} по методу градиентного спуска $x^{k+1} = x^k - t_k \nabla f(x^k)$ с выбором величины шага t_k из условия $f\left(x^k - t_k \nabla f(x^k)\right) < f(x^k)$ [1].

Построение последовательности $\{x^k\}$ заканчивается в точке x^k, для которой $\left\|\nabla f(x^k)\right\| < \varepsilon_1$, где $\varepsilon_1 -$ заданное малое положительное число, или $k \geq M$, где $M -$ предельное число итерации, или при двукратном одновременном выполнении двух неравенств:
$\left\|x^{k+1} - x^k\right\| < \varepsilon_2$ и $\left|f(x^{k+1}) - f(x^k)\right| < \varepsilon_2$, где $\varepsilon_2 -$ малое положительное число.

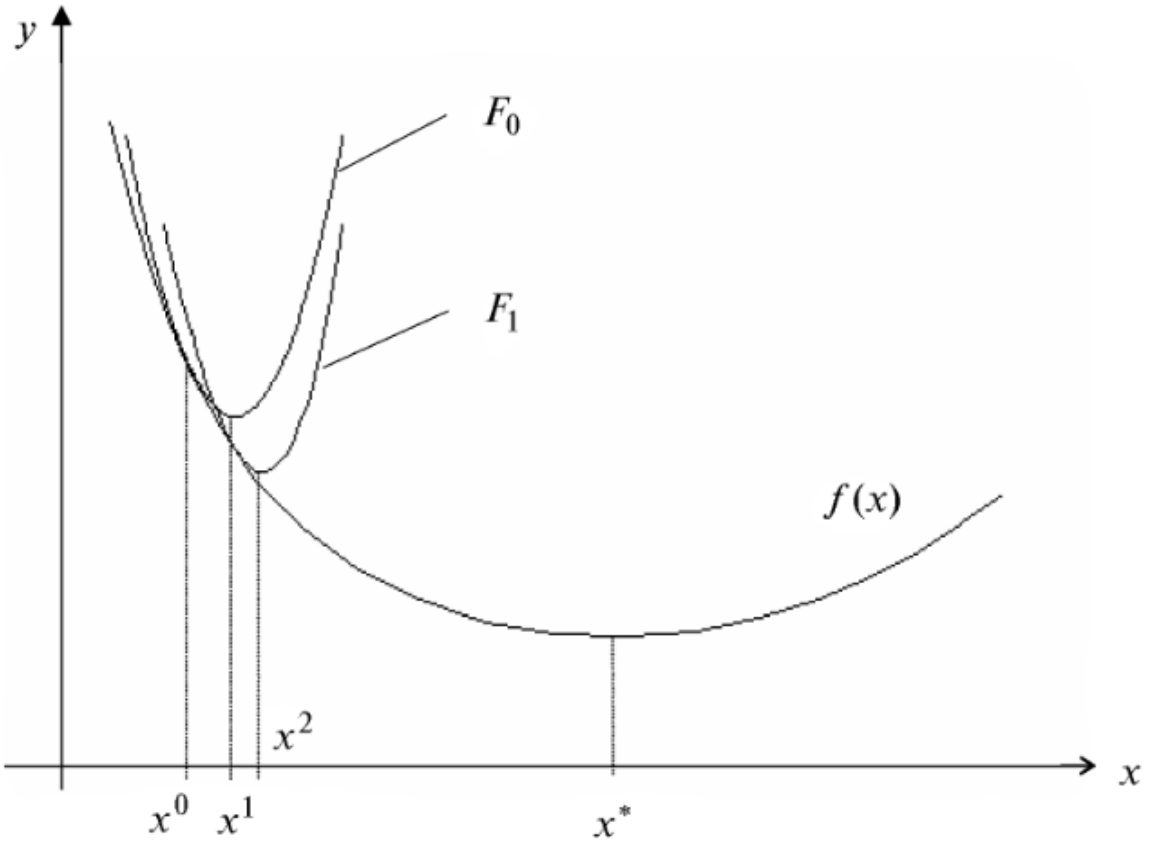

Рис. 3.5

Схема решения задачи методом Ньютона:

Этап 1. Задается x^0, $\varepsilon_1 > 0$, $\varepsilon_2 > 0, M -$ предельное число итераций. Найти градиент функции $\nabla f(x)$ и матрицу Гессе $H(x)$.

Этап 2. Примем $k = 0$.

Этап 3. Вычислим $\nabla f(x^k)$.

Этап 4. Проверим выполнение условия окончания $\left\|\nabla f(x^k)\right\| < \varepsilon_1$:

1) если условие выполнено, процесс поиска закончен, $x^* = x^k$;

2) если условие не выполнено, то перейдём к этапу 5.

Этап 5. Проверим $k \geq M$:

1) если $k \geq M$, то процесс поиска окончен: $x^* = x^k$;

2) если $k < M$, то перейдём к этапу 6.

Этап 6. Вычислим элементы матрицы $H(x^k)$.

Этап 7. Вычислим обратную матрицу $H^{-1}(x^k)$.

Этап 8. Проверим выполнение условия $H^{-1}(x^k) > 0$:

1) если условие выполнено, то перейти к этапу 9;

2) если нет, то перейдём к этапу 10, положив $d^k = -\nabla f(x^k)$.

Этап 9. Вычислим $d^k = -H^{-1}(x^k)\nabla f(x^k)$.

Этап 10. Найти точку $x^{k+1} = x^k + t_k d^k$, положив $t_k = 1$, если $d^k = -H^{-1}(x^k)\nabla f(x^k)$, или выбрав t_k из условия $f(x^{k+1}) < f(x^k)$, если $d^k = -\nabla f(x^k)$

Этап 11. Проверим выполнение условий

$$\left\|x^{k+1} - x^k\right\| < \varepsilon_2 \text{ и } \left|f(x^{k+1}) - f(x^k)\right| < \varepsilon_2:$$

1) если одва условия выполнены при текущем значении k и $k = k - 1$, то процесс поиска окончен, $x^* = x^{k+1}$;

2) если хотя бы одно из условий не выполнено, положим $k = k + 1$ и перейдём к этапу 3 [1].

4. ТЕСТИРОВАНИЕ ПО И ПРОВЕДЕНИЕ ВЫЧИСЛИТЕЛЬНЫХ ЭКСПЕРИМЕНТОВ

Далее по приведенным выше алгоритмам было создано ПО с графическим пользовательским интерфейсом GUI MATLAB(см. на рис. 4.1 и на рис. 4.2).

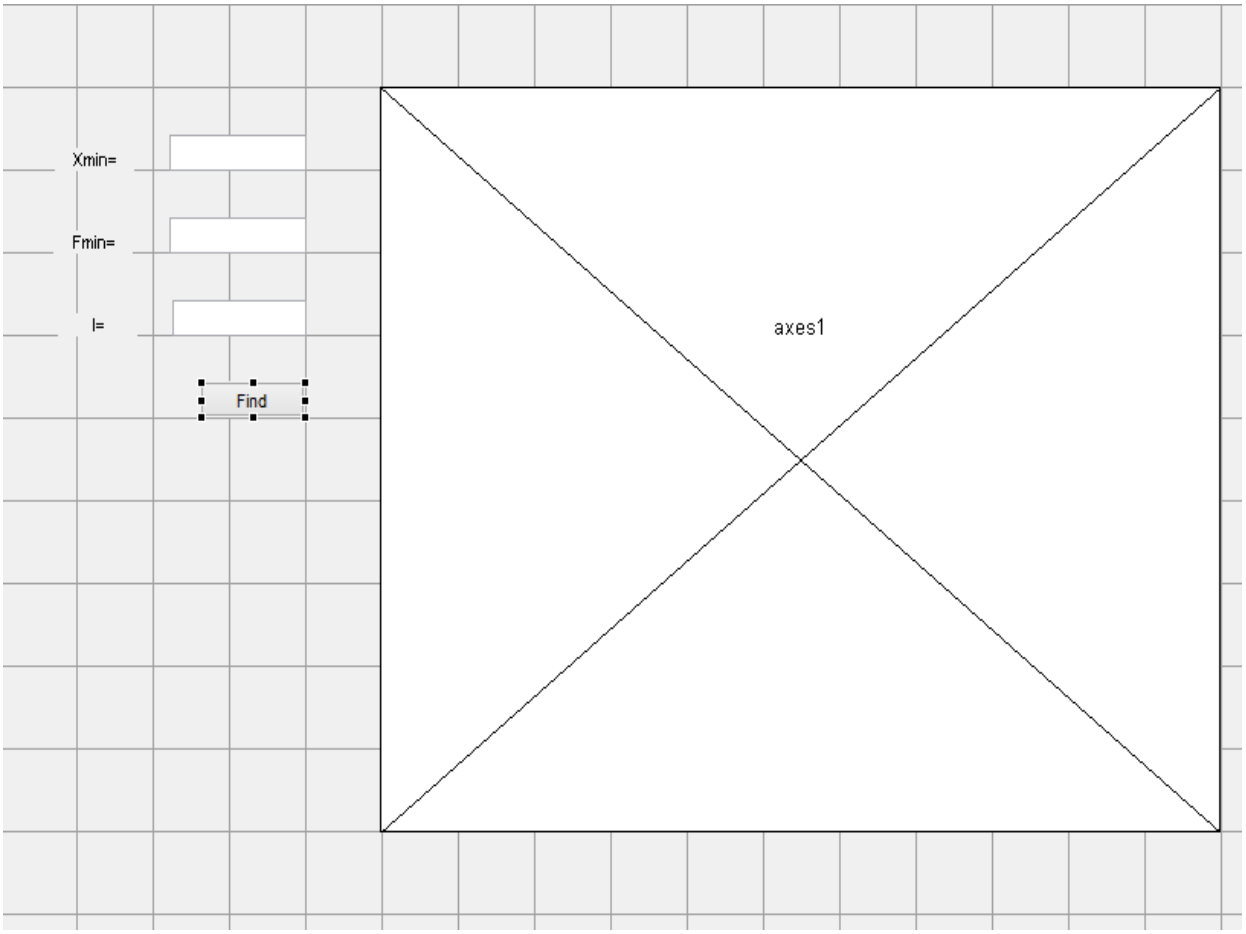

Рис. 4.1 Окно редактирования формы для методов минимизации функций одной переменной

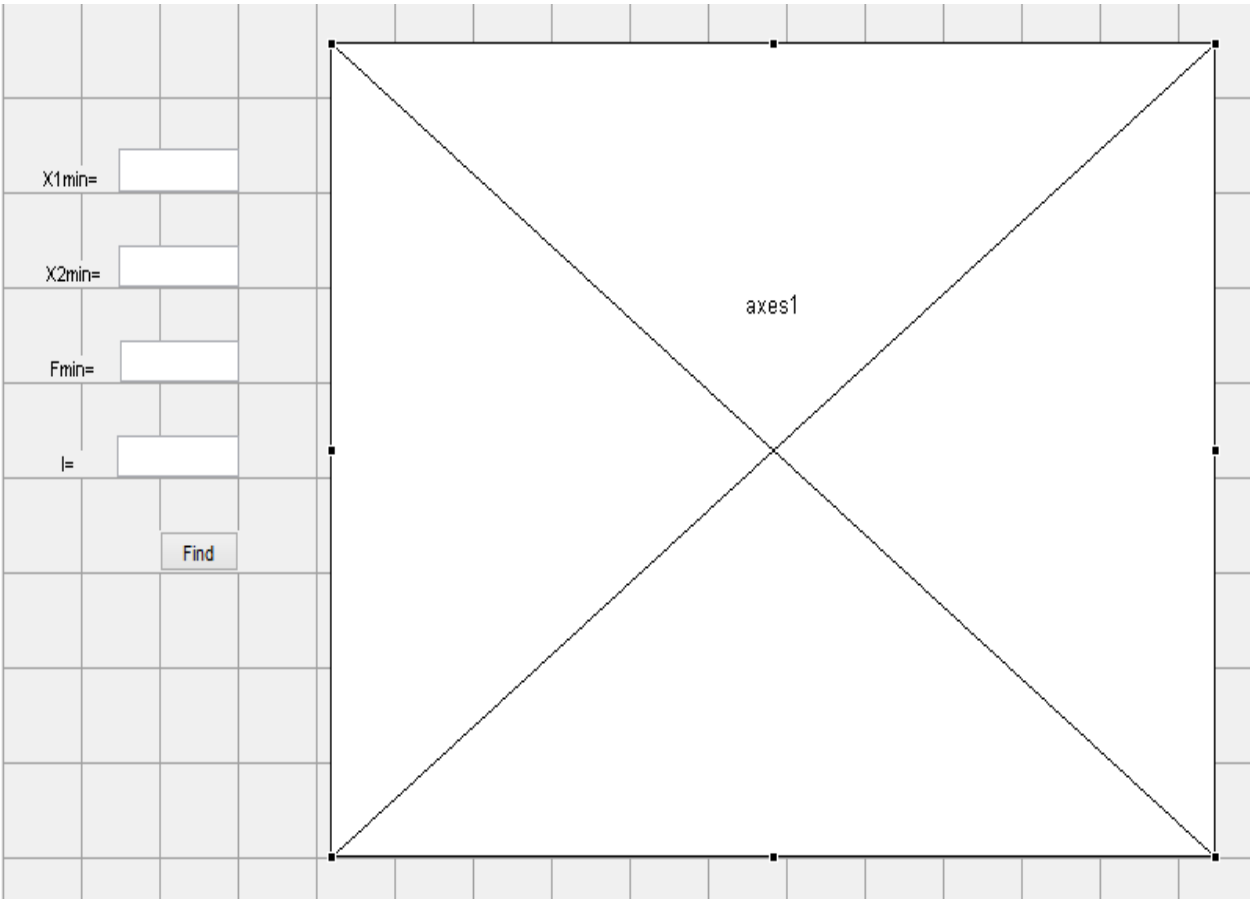

Рис. 4.2 Окно редактирования формы для методов минимизации функций многих переменных

Теперь будем рассмотривать такие главные компоненты, которые отображаются на выше интерфейсе формы (см. на рис. 4.1 и на рис. 4.2).

1) Одна кнопка, созданная в программе

Рис. 4.3 Созданная кнопка в программе.

Кнопка «Find» выполняет поиск минимизации функций и возвращает нахожденные результаты поставленной задачи.

2) Компонент «Static Text» для вывода названия.
3) Компонент «Edit» для выода результатов работы.
4) Область графика

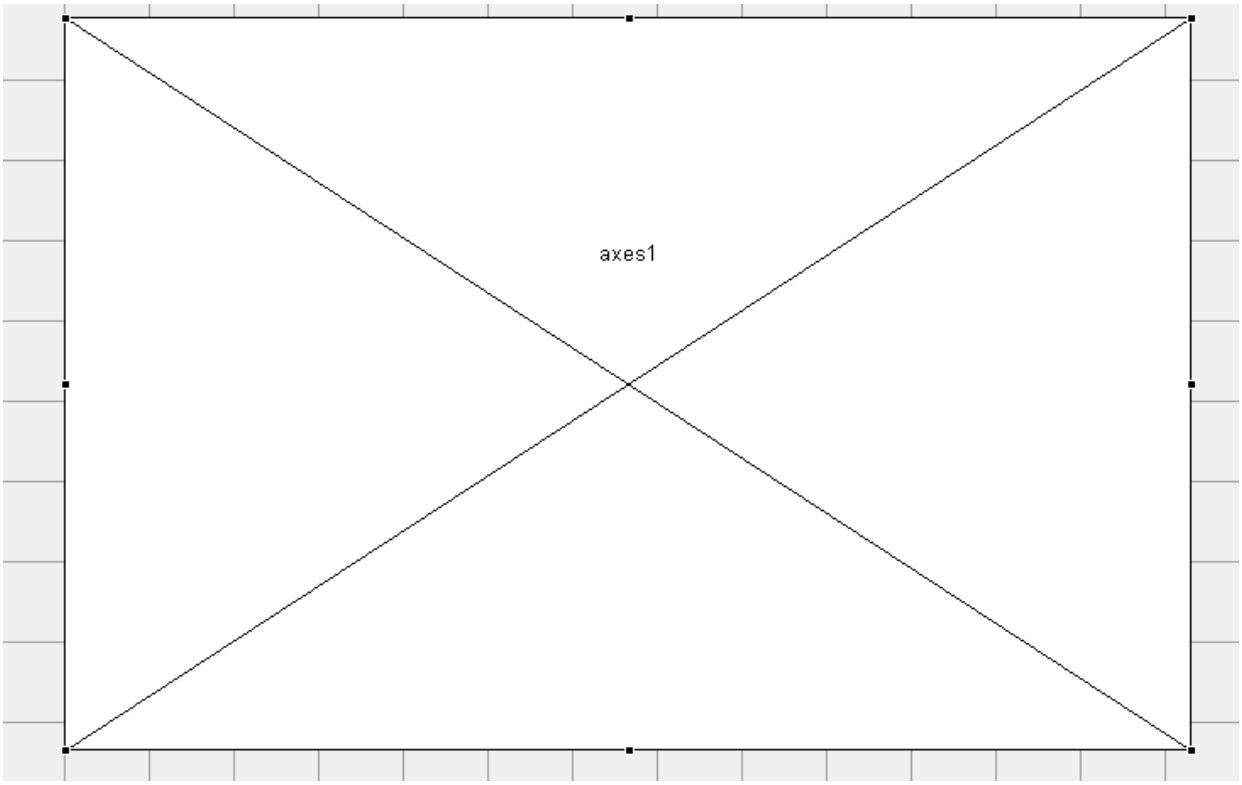

Рис. 4.4 Область для отображения графика.

Протестируем нахождение экстремума функции методом половинного деления.

Задача 1: Найти минимум функции $f(x) = 2x^2 - 12x$ методом половинного деления.

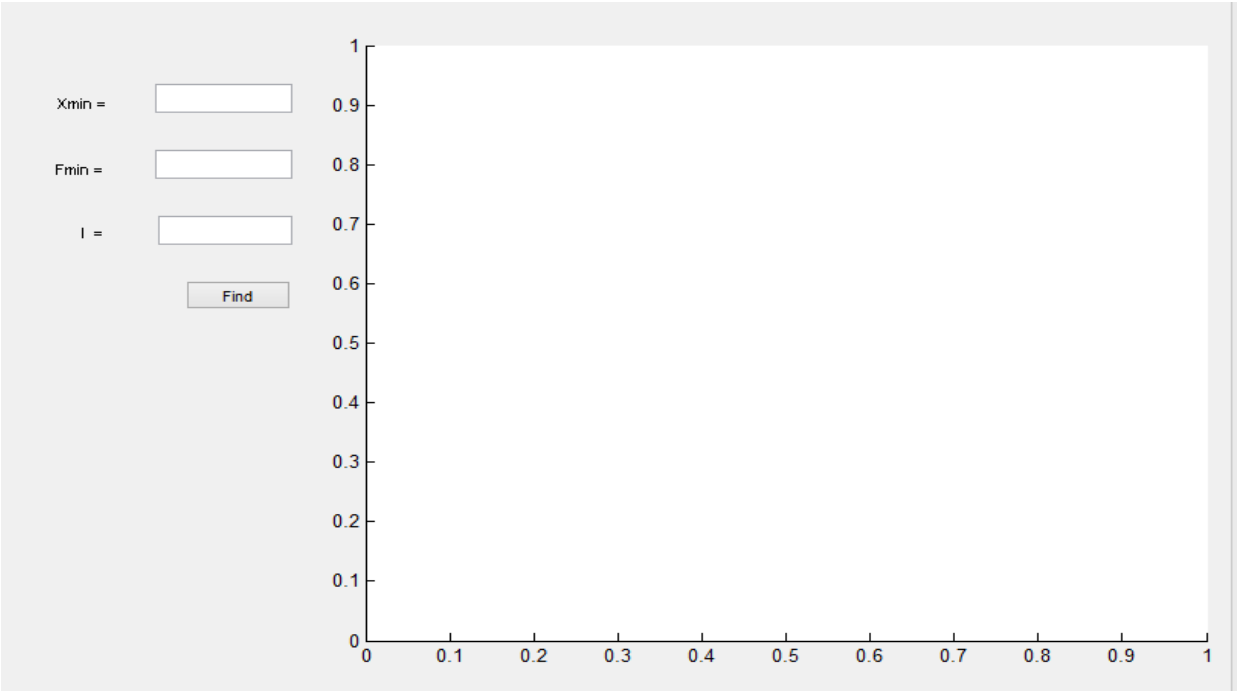

Рис. 4.5 Окно вывода нахождения минимума функции методом половинного деления

Нажимаем на кнопку «Find», после этого полученные результаты отображают в области вывода: $Xmin = 3.05, Fmin = -17.995, I = 55$ и график нарисован, как показано на рис. 4.6.

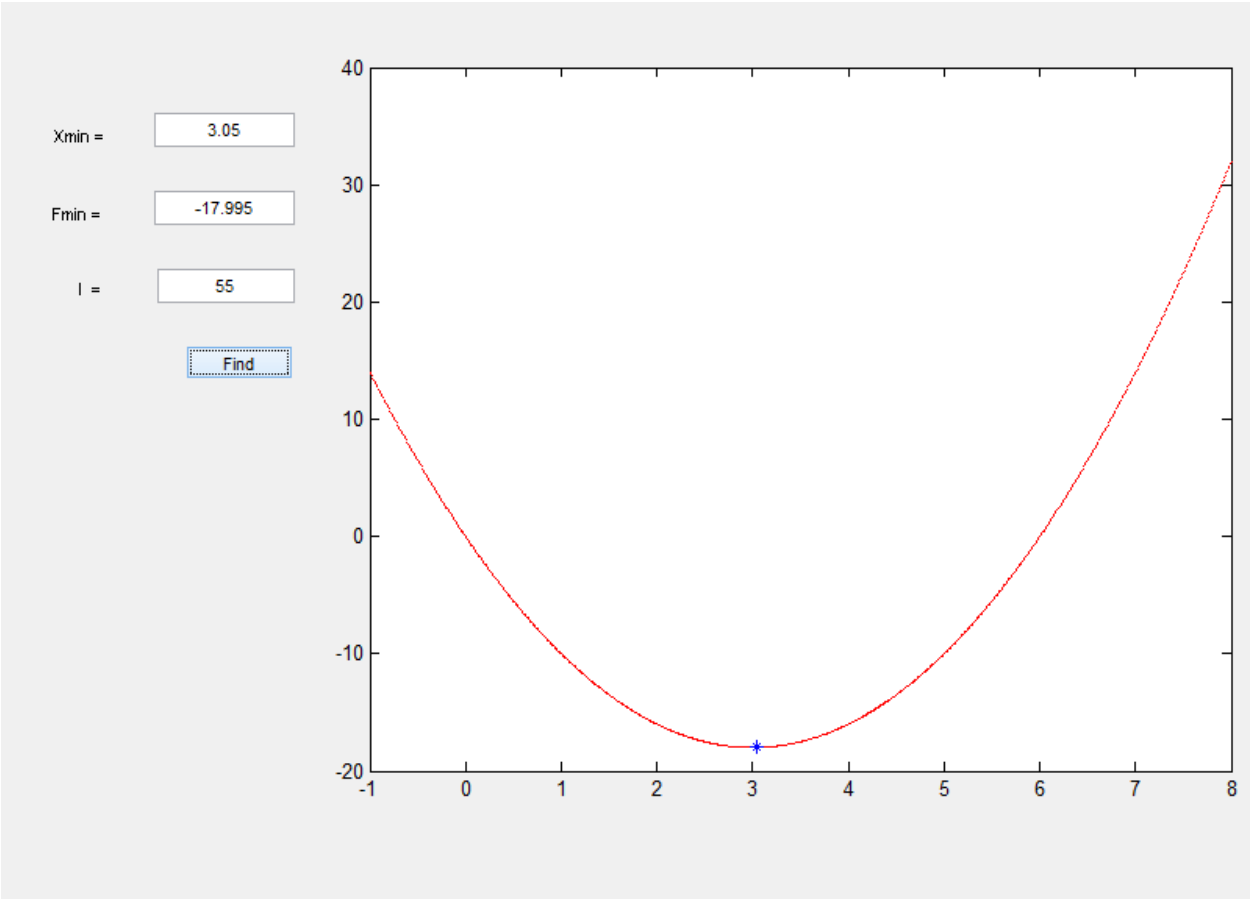

Рис. 4.6 Окно вывода результата

Протестируем нахождение экстремума функции методом золотого сечения.

Задача 2: Найти минимум функции $f(x) = x^4 + 2x^2 + 4x + 1$ методом золотого сечения.

.

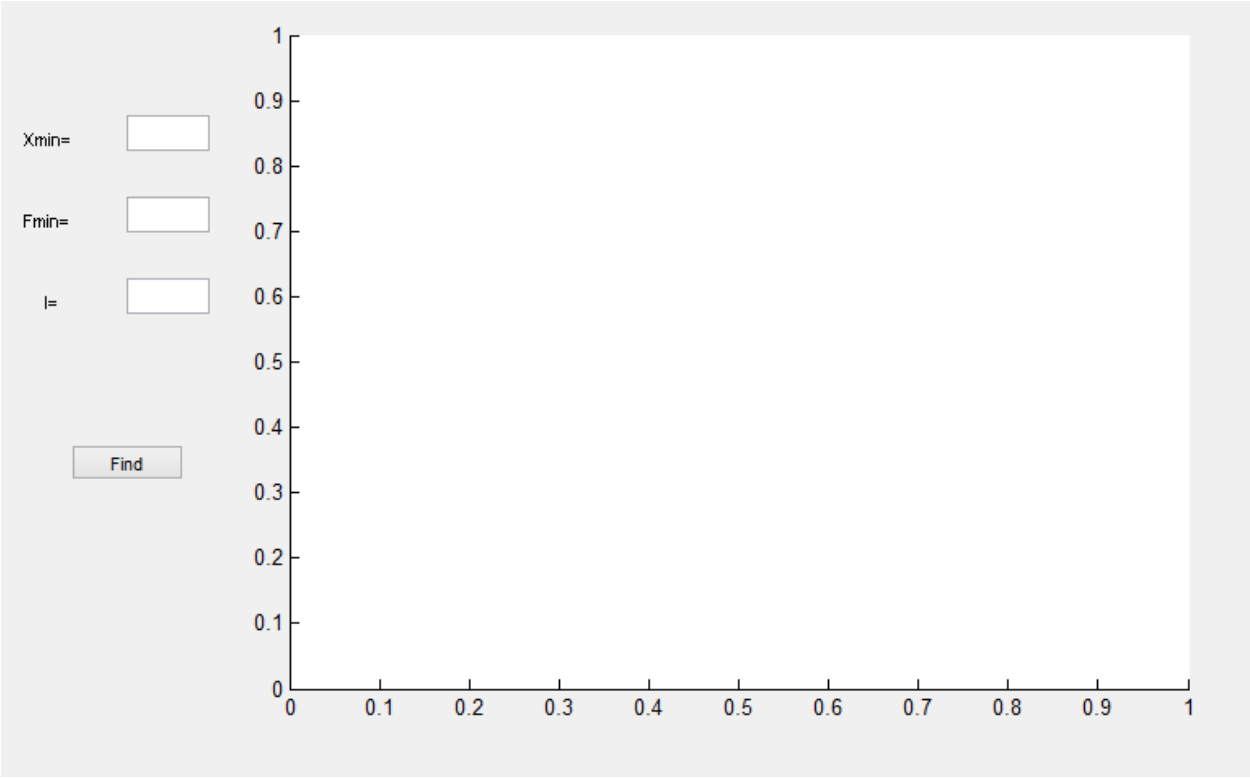

Рис. 4.7 Окно вывода нахождения минимума функции методом золотого сечения.

Нажимаем на кнопку «Find», после этого полученные результаты отображают в области вывода: $Xmin = -0.6828, Fmin = -0.5814, I = 10$ и график нарисован, как показано на рис. 4.8.

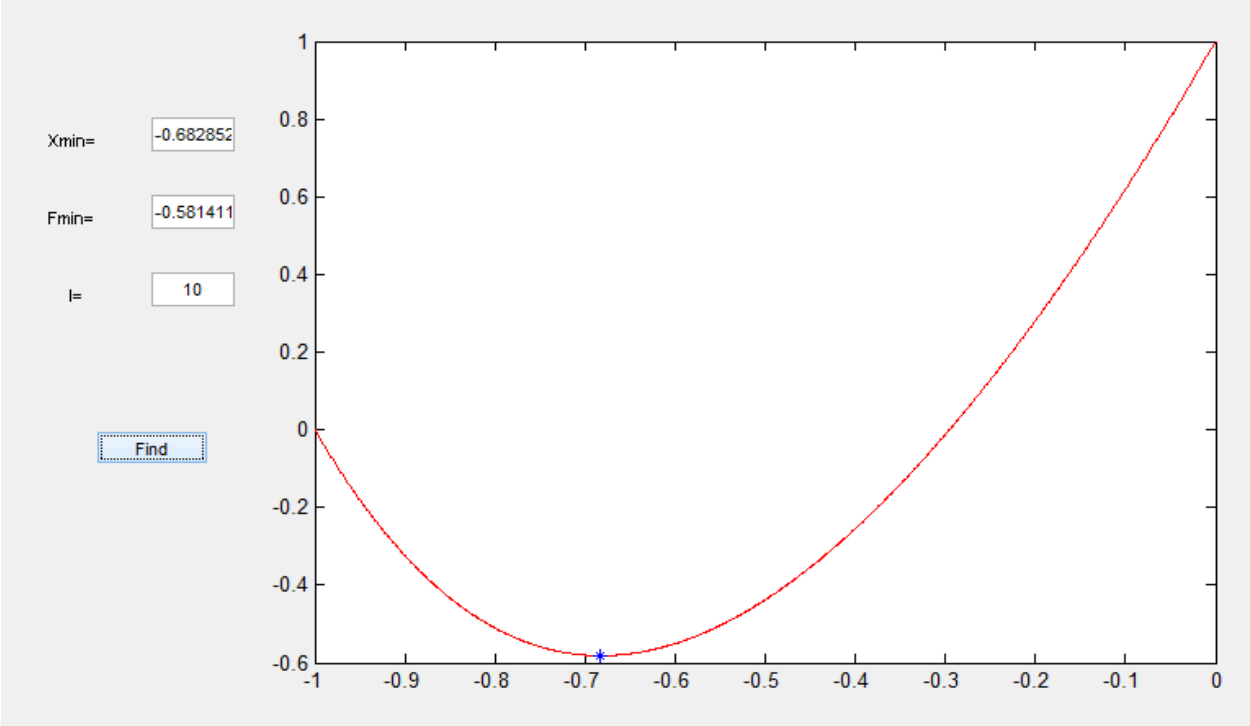

Рис. 4.8 Окно вывода результата

Протестируем нахождение экстремума функции градиентным методом.

Задача 3: Найти локальный минимум функции $f(x) = 2{x_1}^2 + x_1x_2 + {x_2}^2$ градиентным методом.

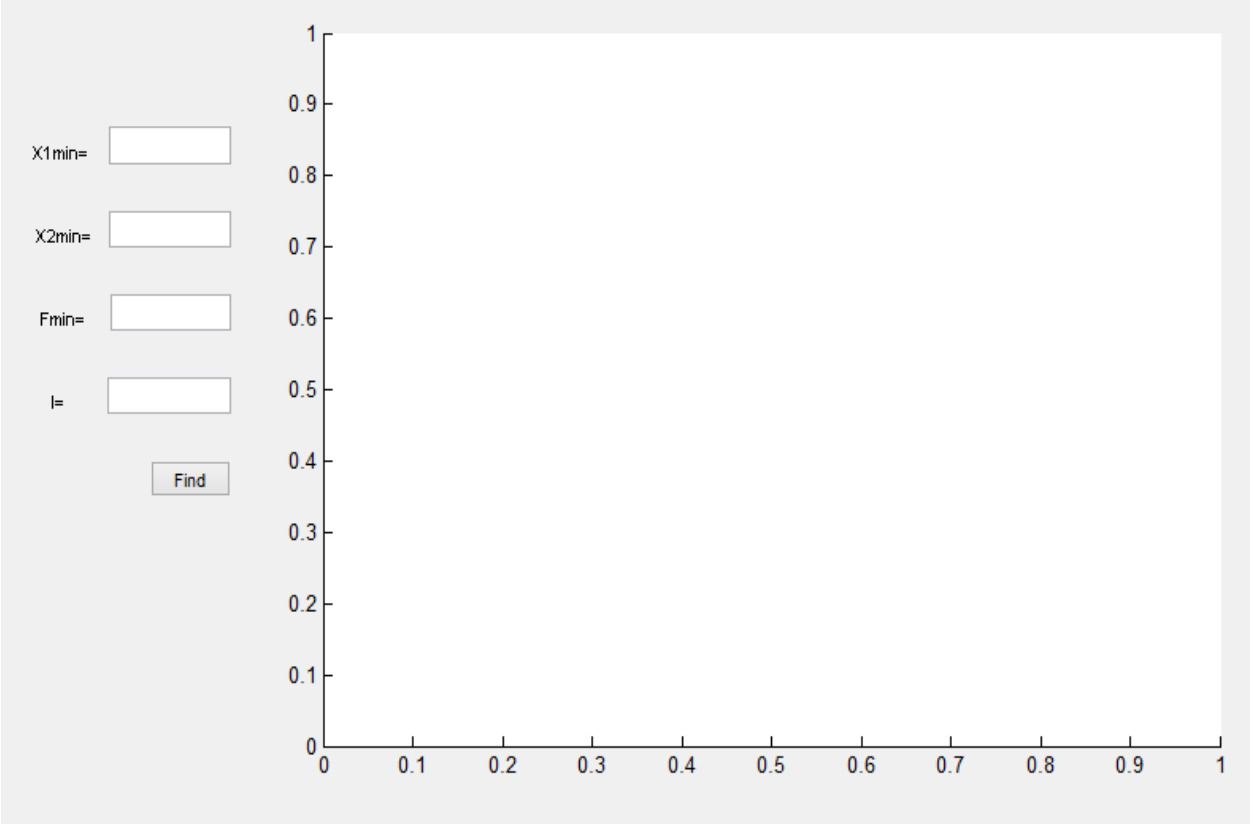

Рис. 4.9 Окно вывода нахождения минимума функции градиентным методом

Нажимаем на кнопку «Find», после этого полученные результаты отображают в области вывода: $X1min = -0.02171, X2min = 0.23807, Fmin = 0.0524515, I = 5$ и график нарисован, как показано на рис. 4.10.

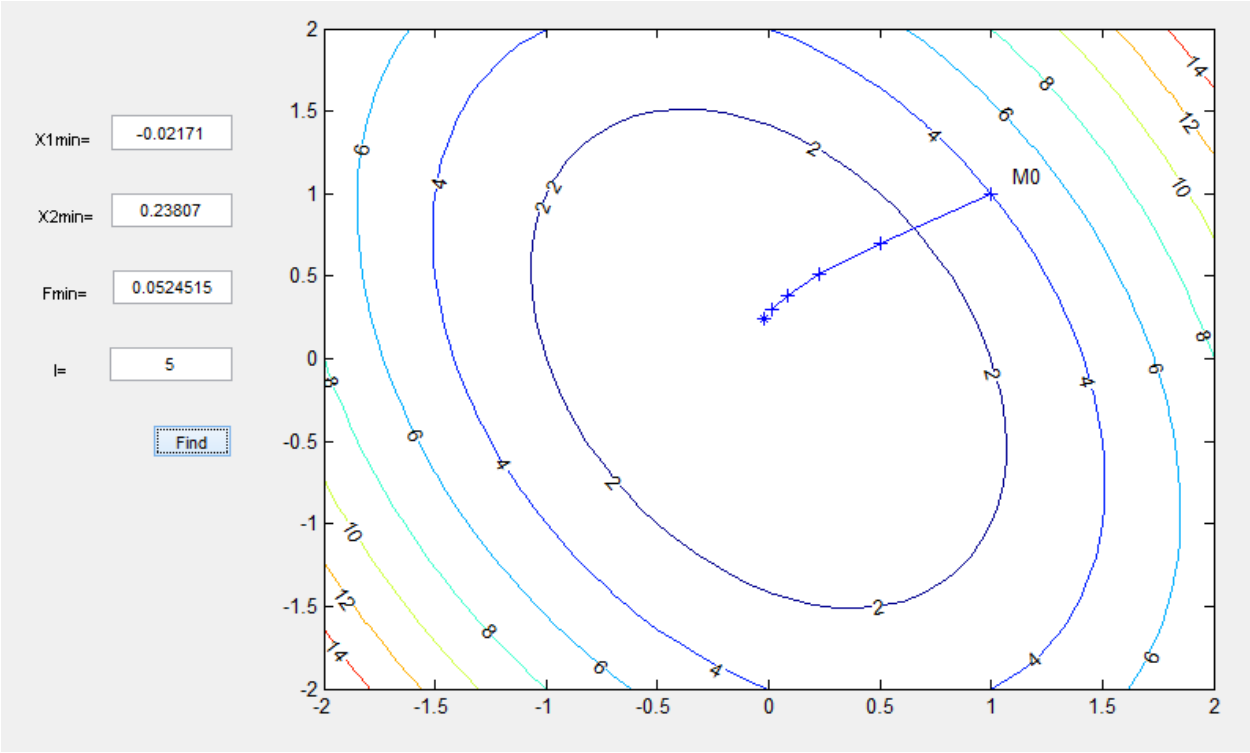

Рис. 4.10 Окно вывода результата

Протестируем нахождение экстремума функции методом Ньютона.

Задача 4: Найти локальный минимум функции $f(x) = 2(1 + x_2)^3 + 3(x_1 - 1)^2$ методом Ньютона.

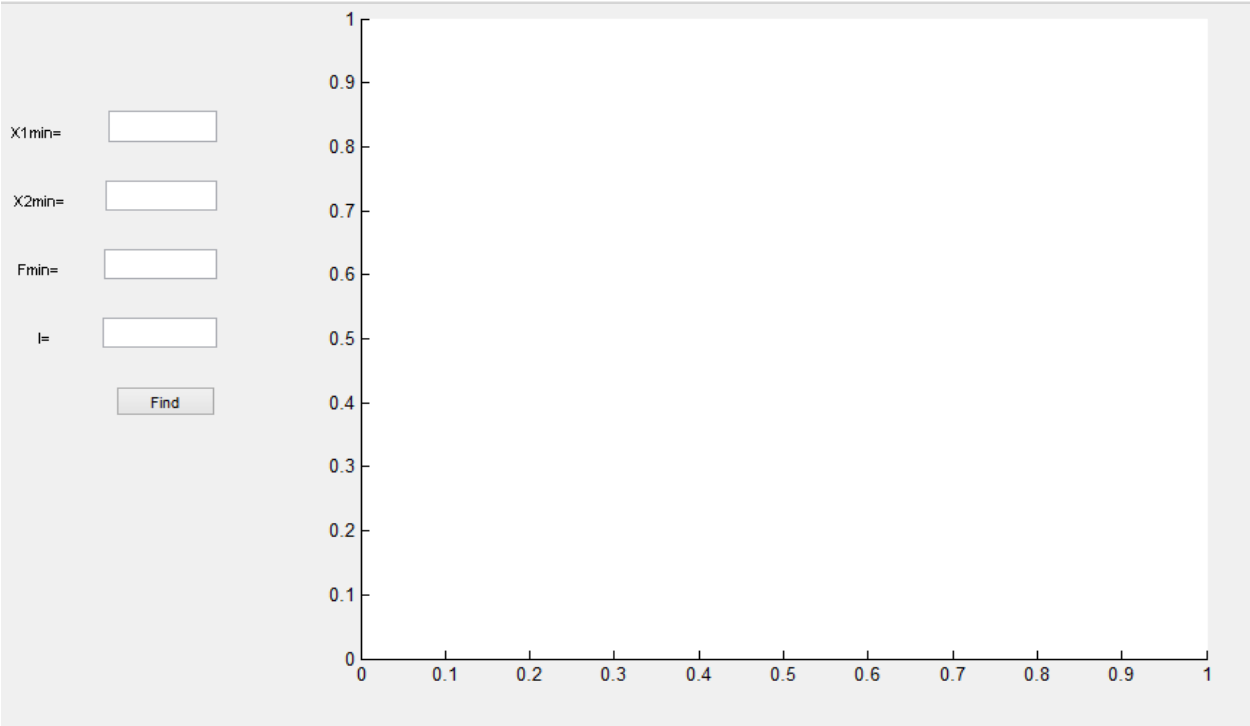

Рис. 4.11 Окно вывода нахождения минимума функции методом Ньютона

Нажимаем на кнопку «Find», после этого полученные результаты отображают в области вывода: $X1min = 1, X2min = -0.875, Fmin = 0.00390625, I = 5$ и график нарисован, как показано на рис. 4.12.

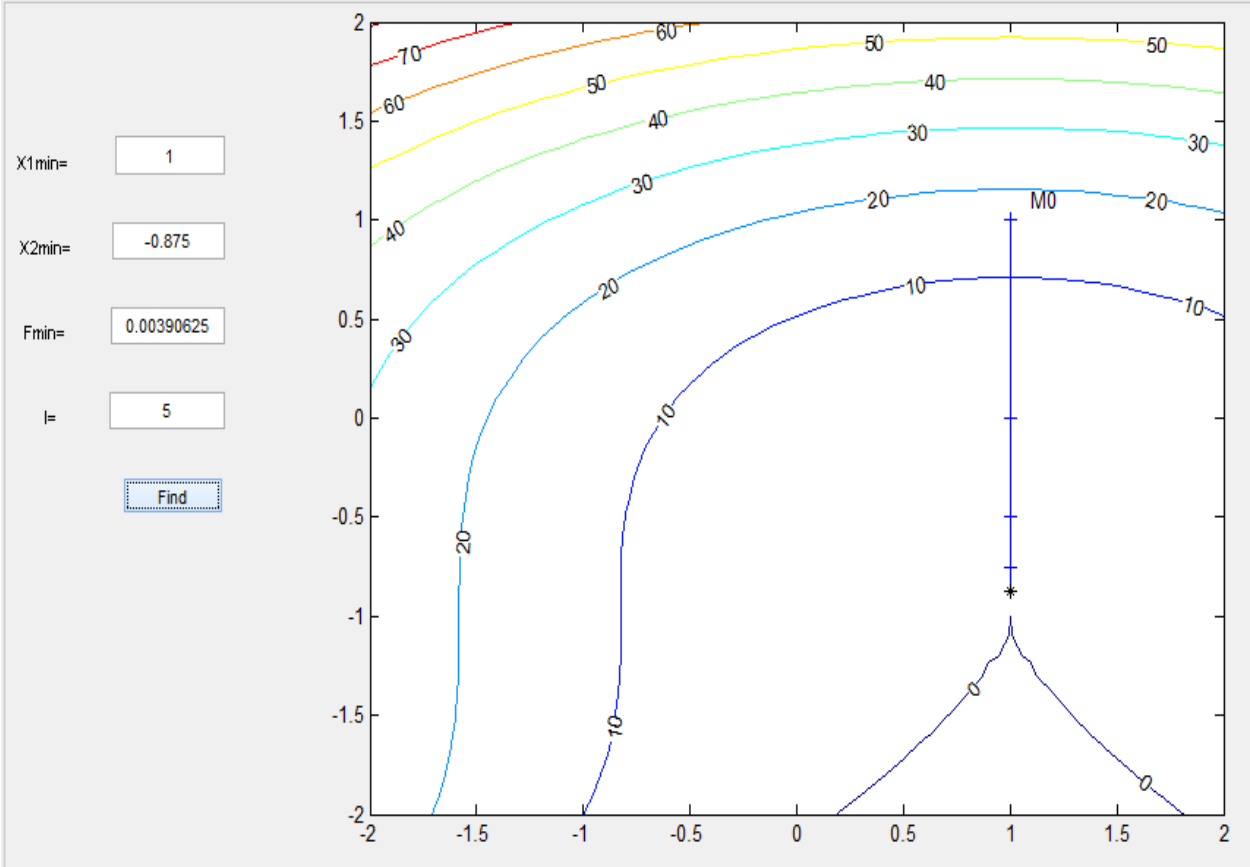

Рис. 4.12 Окно вывода результата

ЗАКЛЮЧЕНИЕ

В выпускной квалификационной работе были получены следующие результаты:

- изучена и определена общая задача;
- исследованы характеристики различных численных методов;
- составлен алгоритм различных численных методов;
- исследованы возможности пакета программирования Matlab по решению экстремальных задач;
- выполнены программные реализации алгоритмы различных численных методов с помощью обеспечением программы GUI MATLAB (Graphical User Interface - MatrixLaboratory) для нахождения экстремумов функции.

Созданное ПО было протестировано на контрольных примерах, и может быть использовано для нахождения экстремумов функции.

В ходе выполнения выпускной квалификационной работы были приобретены практические и теоретические знания и навыки в области математики, а также создания приложения. Таким образом, задачи, поставленные в выпускной квалификационной работе, были выполнены, а цель – достигнута.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Методы оптимизации. Практический курс: учебное пособие с мультимедиа сопровождением / А.В. Пантелеева, Т.А. Летова. – М.: Логос, 2011. – 424 с: ил.
2. Васильева Ф. П. Численные методы решения экстремальных задач: Учеб. пособие для вузов – 2-е изд., перераб. И доп.- М.: Наук, Гл. ред. физ.-мат. лит., 1988-552 с.- ISBN 5-02-013796-0.
3. Аттетков А.В., Зарубин В.С., Канатников А.Н. Введение в методы оптимизации,- М.: Финансы и статистика, Инфра-М, 2008.
4. Киреев В.И., Пантелеев А.В. Численные методы в примерах и задачах. - М.: Высшая школа, 2008.
5. Ильин В. А., Садовничий В. А., Сендов Бл. Х. Математический анализ. Начальный курс.— М.: Изд-во МГУ, 1985.— 660 с.
6. Карманов В.Г. Математическое программирование.— М.: Наука, 1986.— 288 с.
7. Ашманов С.А. Линейное программирование.— М.: Наука, 1981.—
8. 304 с.
9. Бахвалов Н.С., Жидков Н.П., Кобельков Г.М. Численные методы.—М.: Наука, 1987.—600 с.
10. Габасов Р., Кириллова Ф.М. Методы оптимизации.— Минск: Изд-во Б ГУ, 1981.— 352 с.
11. Ильин В.А., Садовничий В.А., Сендов Бл.Х. Математический анализ. Начальный курс.— М.: Изд-во МГУ, 1985.— 660 с.
12. Карманов В.Г. Математическое программирование.— М.: Наука, 1986.— 288 с.
13. Ляшенко И.Н., Карагодова Е.А., Черникова Н. В., Шор Н.З. Линейное и нелинейное программирование.— Киев: Вища школа, 1975.— 372 с.
14. Марчук Г.И. Методы вычислительной математики.— М.: Наука, 1980.— 536 с.
15. Моисеев Н.Н., Иванилов Ю.П., Столярова Е.М. Методы оптимизации.—М.: Наука, 1978.—352 с.
16. Морозов В.В., Сухарев А.Г., Федоров В.В. Исследование операций в задачах и упражнениях,—М.: Высшая школа, 1986.— 287 с.
17. Пшеничный Б.Н., Данилин Ю.М. Численные методы в экстремальных задачах.— М.: Наука, 1975.— 320 с.
18. Самарский А.А., Николаев Е.С. Методы решения сеточных уравнений.—М.: Наука, 1978.— 592 с.

19. Сухарев А.Г., Тимохов А.В., Федоров В.В. Курс методов оптимизации.— М.: Наука, 1986.— 328 с.
20. Тихонов А.Н., Арсенин В.Я. Методы решения некорректных задач.— М.: Наука, 1986.— 288 с.
21. Абрамов Л.М., Капустин В.Ф. Математическое программирование.— Л.: Изд-во ЛГУ, 1981.—328 с.
22. Вилков А.В., Жидков Н.П., Щедрин Б.М. Метод отыскания глобального минимума функции одного переменного Ц Журн. вычислит. матем. и матем. физики.— 1975.— Т. 15, № 4.— С. 1040—1042.
23. Воробьев Н.Н. Числа Фибоначчи.— М.: Наука, 1978.— 144 с.
24. Певный А.Б. Об оптимальных стратегиях поиска максимума функции с ограниченной старшей производной Ц Журн. вычисл. матем. и матем. физики.— 1982.— Т. 22, № 5.— С. 1061—1066.
25. Пиявский С.А. Один алгоритм отыскания абсолютного экстремума функции Ц Журн. вычисл. матем. и матем. физики.— 1972.— Т. 12, № 4.— С. 888—896.
26. Ильин В.А., Позняк Э.Г. Основы математического анализа.— М.: Наука, ч. I, 1971.— 600 с, ч. И, 1973.—448 с.
27. Никольский С.М. Курс математического анализа.— М.: Наука, 1973.— Т. 1.— 432 с. Т. 2.- 392 с.
28. Терёхин В.В. Т- Моделирование в системе MATLAB: Учебное пособие /Кемеровский государственный университет. – Новокузнецк: Кузбассвузиздат, 2004. -376с.
29. Си Шарп: Создание приложений для Windows.Мн.: Харвест, 2003. - 384 с.
30. Фридман, А.Л. Язык программирования С++ / А.Л.Фридман. - М.: Бином, 2006. - 523с.: ил. 4. Лахатин, А.С. Языки программирования. Учеб. пособие / А.С. Лахатин, Л.Ю. Искакова. - Екатеринбург, 1998. - 548с.: ил.

ПРИЛОЖЕНИЕ

1.

Отдельный m-файл g1.m

```
function F1 = g1( x)
F1=2*x*x-12*x;
End
```

```
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
a=3;
b=10;
e=0.1;
k=0;
syms x;
f=2*x.^2-12*x;
while (b-a)> e
    k=k+1;
    u1=(a+b-e)/2;
    u2=(a+b+e)/2;
    f1=g1(u1);
    f2=g1(u2);
   if f1 < f2
      b=u2;
   else
      a=u1;
   end
   xmin=(a+b)/2;
   fmin=g1(xmin);
end
x1=-1:0.01:8;
[X1]=meshgrid(x1');
F=2*X1.^2-12*X1;
plot(X1,F,'r',xmin,fmin,'b*');
set(handles.xmin,'string',xmin);
set(handles.fmin,'string',fmin);
set(handles.i,'string',k);
```

2.

Отдельный m-файл g2.m

```
function F2 = g2( x )
F2=x.^4+2*x.^2+4*x+1;
end
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
a=-1;
b=0;
e=0.01;
k=0;
syms x;
f=x.^4+2*x.^2+4*x+1;
 u1=a+(3-sqrt(5))/2*(b-a);
    u2=a+b-u1;
    f1= g2(u1);
    f2= g2(u2);
while (b-a)>e
    if f1<f2
        b=u2;
        u2=u1;
        f2=f1;
        u1=a+(3-sqrt(5))/2*(b-a);
        f1= g2(u1);
    else a=u1;
        u1=u2;
        f1=f2;
        u2=a+b-u1;
        f2= g2(u2);
    end
     k=k+1;
   xmin=(a+b)/2;
     fmin=g2(xmin);
end
 x1=-1:0.001:0;
  [X1]=meshgrid(x1');
   F=X1.^4+2*X1.^2+4*X1+1;
plot(X1,F,'r',xmin,fmin,'b*');
set(handles.xmin,'string',xmin);
set(handles.fmin,'string',fmin);
%set(handles.t,'string',Nsec);
```

```
set(handles.i,'string',k);
```
3.

Отдельный m-файл g3.m

```
function F3 = g3( x1,x2)
F3=2*x1.^2+x1.*x2+x2.^2;
end

function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
e1 = 0.1;
e2 = 0.1;
x0 = [1;1];
Kmax = 100;
a1=-2;
a2=2;
b1=-2;
b2=2;
alf1=0.1;
alf2=0.1;
x1=[a1:alf1:a2];
x2=[b1:alf2:b2];
f = 2*x1.^2+x1.*x2+x2.^2;
x=x0;
Xm = x;
t0=0.1;
k = 0;
i=1;
while k < Kmax;
    k = k + 1;
    Gr = [4*x(1)+x(2); 2*x(2)+x(1)];
    if sqrt(Gr(1).^2+Gr(2).^2)<e1
       xmin=x;
       fmin=g3(xmin(1),xmin(2));
        break;
    end
    if sqrt(Gr(1).^2+Gr(2).^2)>e1
        h = -t0*Gr;
        f1=g3(x(1),x(2));
        x1=x+h;
        f2= g3(x1(1),x1(2));
             while f2-f1>0
               t0=t0/2;
```

```
                h = -t0*Gr;
                x1=x+h;
                f2_1=  g3(x1(1),x1(2));
                    f2=f2_1;
            end
                if (f2-f1)<0&& sqrt(sum((x1-x).^2 ))<e1 && abs(f2-f1)< e2
                  xmin=x1;
                  fmin=g3(xmin(1),xmin(2));
                  break;
                end
    end
     i = i + 1;
    Xm(:,i) = x1;
              x=x1;
end
x1=[a1:alf1:a2];
x2=[b1:alf2:b2];
[X1,X2] = meshgrid(x1', x2');
F =  2*X1.^2+X1.*X2+X2.^2 ;
   [C, h] = contour(X1, X2, F);
   clabel(C, h)
hold on;
plot(Xm(1,:), Xm(2,:), '-+');
 text(Xm(1,1) + 0.1, Xm(2,1) + 0.1, 'M0');
      plot(xmin(1),xmin(2), '*');
hold off
set(handles.x1min,'string',xmin(1));
set(handles.x2min,'string',xmin(2));
set(handles.fmin,'string',fmin);
set(handles.i,'string',i);
```

4.

Отдельный m-файл g4.m

```
function F4 = g4( x1,x2 )
F4=2*(1+x2).^3+3*(x1-1).^2 ;
end
```

```
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
e1=0.1;
e2=0.1;
K=100;
k=0;
i=1;
x0=[1;1];
x1=-2:0.1:2;
x2=-2:0.1:2;
[X1,X2] = meshgrid(x1', x2');
F =2*(1+X2).^3+3*(X1-1).^2 ;
x=x0;
Xm= x;
while k<K
  k = k + 1;
    Gr = [6*(x(1)-1); 6*(1+x(2)).^2];
    if sqrt(Gr(1).^2+Gr(2).^2)<e1
        xmin=x;

        fmin=g4(xmin(1),xmin(2));
          break;
    end
     if sqrt(Gr(1).^2+Gr(2).^2)>e1
       DFx1x1=6;
       DFx1x2=0;
       DFx2x1=0;
       DFx2x2=12*(x(2)+1);
       G1=zeros(2,2);
       G1(1,1)=DFx1x1(:);
       G1(1,2)=DFx1x2(:);
       G1(2,1)=DFx2x1(:);
       G1(2,2)=DFx2x2(:);
       H=inv(G1);
       if H(1,1)>0 && H(1,1)*H(2,2)-H(2,1)*H(1,2)> 0
          d=-H*Gr;
```

```
            t0=1;
           f1=g4(x(1),x(2));
            x1=x+t0*d;
             f2= g4(x1(1),x1(2));
        else
            d=-Gr;
            t0=0.5;% 0<t<2;
           f1=g4(x(1),x(2));
            x1=x+t0*d;
             f2=g4(x1(1),x1(2));
        end
     if sqrt(sum((x1-x).^2 ))<e1 && abs(f2-f1)< e2
         xmin=x1;
        fmin=g4(xmin(1),xmin(2));
           break;
     end
      end
      i = i + 1;
     Xm(:,i) = x1;
    x=x1;
end
 y1=xmin(1);
 y2=xmin(2);
x1=-2:0.1:2;
x2=-2:0.1:2;
[X1,X2] = meshgrid(x1', x2');
F = 2*(1+X2).^3+3*(X1-1).^2 ;
    [C, h] = contour(X1, X2, F);
    clabel(C, h)
hold on;
plot(Xm(1,:), Xm(2,:), '-+');
 text(Xm(1,1) + 0.1, Xm(2,1) + 0.1, 'M0');
    plot(xmin(1),xmin(2), '*k');
hold off
set(handles.x1min,'string',y1);
set(handles.x2min,'string',y2);
set(handles.fmin,'string',fmin);
set(handles.i,'string',i);
```

Выпускная квалификационная работа выполнена мной совершенно самостоятельно. Все использованные в работе материалы и концепции из опубликованной научной литературы и других источников имеют ссылки на них.

«___» _____ _____ г.

_____ _____

(подпись) *(Ф.И.О.)*