**ЛАБОРАТОРНЫЙ ПРАКТИКУМ**

**«Информатика и программирование»**

**Часть третья – численные алгоритмы.**

**Введение.**

*Численные алгоритмы – решение алгебраических и трансцендентных уравнений, решение систем линейных алгебраических уравнений, решение систем обыкновенных дифференциальных уравнений, решение уравнений в частных производных, оптимизация, обработка числовых данных, численное интегрирование, численное дифференцирование. В данной части практикума рассмотрим только ряд задач, решаемых с помощью достаточно простых алгоритмов.*

Практикум базируется на работе в лаборатории кафедры «Электрофизические установки» НИЯУ МИФИ (сервер *accel.ru*).

Ниже представлены ***обязательные******требования*** к написанию и оформлению программ (в самом кратком и минимальном объёме), практические советы.

**Требования к написанию и оформлению программ.**

Практическую работу при выполнении практикума студенту следует начинать с построения файловой структуры выполняемых лабораторных работ в своём личном каталоге. Все необходимые для этого знания получены в первой части практикума - работе в операционной среде UNIX.

1. Каждая лабораторная работа должны быть оформлена в отдельной папке.
2. В этой папке, в конечной стадии работы, должны находиться только окончательные варианты исходных, исполняемых и текстовых файлов (с результатами расчётов). *Все файлы пробных и промежуточных вариантов должны быть удалены.*
3. Имена исходных и исполняемых файлов данной лабораторной работы должны совпадать и, по возможности, отражать её смысловое содержание.

*Пример: Лабораторная работа по нахождению корня н/л уравнения*.

Папка с именем «ROOT». Поскольку методов несколько, то для каждого типа создать свою папку, например, «ROOT/ M\_1» и т.п. Имена исходных файлов могут быть, например, «root\_m\_1.f95», а исполняемых файлов просто «root\_m\_1». Текстовый файл (с результатами расчётов) - «root\_m\_1.txt».

1. Результаты расчётов должны быть оформлены в удобочитаемом виде с необходимыми текстовыми пояснениями и выведены в соответствующий текстовый файл (описание входных и выходных параметров с использованием *format*).

*Практический совет:* Программа должна быть легко читаема, а для этого:

1. Должны быть даны в начале программы краткие, но содержательные комментарии, определяющие суть задачи и производимых действий.
2. Все используемые данные должны быть явно объявлены и сгруппированы по типам.
3. Особое внимание именам переменных и массивов. Они должны отражать физическую или математическую суть переменной.

*Пример:* Напряженность электрического поля – *E*. Ускорение свободного падения – *g*. Для математических выражений нужно использовать символы близкие к написанию математических формул. Если есть выражение , то нужно использовать имена близкие по смыслу - и т.п. Для индексов нужно использовать принятые в математике символы - и их вариации - и т.п. Желательно не употреблять длинных и безликих имён, что затрудняет чтение текста.

1. Для записи управляющих конструкций использовать *правило рельефа* –операторы, вложенные внутри конструкции, записываются правее операторов, образующих эту конструкцию.

*<операторы\_0>*

*do i=0,n*

*<операторы\_1>*

*if <условие> then*

*<операторы\_2>*

*else*

*<операторв\_3>*

*endif*

*<операторы\_1\_1>*

*enddo*

Результаты расчётов – выходные данные – должны быть записаны в удобочитаемом виде с соответствующими краткими текстовыми пояснениями. Выходная информация может быть двух видов промежуточная и конечная. Промежуточная информация необходима как для отладки программы, так и для изучения работы тех или иных алгоритмов, реализованных в данной программе.

**Внешние файлы.**

*Внешний файл* – файл, существующий в среде, внешней по отношению к выполняемой программе. Внешние файлы должны быть открыты оператором OPEN.

Оператор OPEN создаёт устройство ввода/вывода с номером ***n*** и подсоединяет к нему внешний файл ***file*** (полное имя файла с путём доступа). Оператор имеет целый ряд спецификаторов, рассмотрим только три из них.

OPEN (*n*, file=’<*имя\_файла*>’, STATUS=’<*статус*>’).

* *n* – устройство внешнего файла, к которому подсоединяется файл - <*имя\_файла>.*
* <*имя\_файла>* - имя файла с расширением ***.txt*** с путём доступа (от домашнего каталога).
* <*статус*> - ‘unknown’ . Если файл существует, то он открывается,
  + - * + если нет – создаётся.

*Пример:*

Open (2, file='путь доступа\writ.txt', status='unknown')

**Лабораторные работы и задания.**

**Работы на семестр:**

* Л.р. №1\_1, одна из (Л.р. №1\_2, Л.р. №1\_3, Л.р. №1\_4 ) – корень н/л уравнения,
* Л.р. №2\_1, Л.р. №2\_2 – интерполяция,
* Л.р. №3\_1 - интегрирование,
* Л.р. №4\_1 - дифференцирование,
* Л.р. №5\_1, Л.р. №5\_2 – решение системы алгебраических уравнений,
* Л.р. №6\_1, Л.р. №6\_2, №6\_3- нахождение минимума функции.

*Примечание: Основное внимание к программной реализации математических методов и оценки их эффективности. Работы имеют различный уровень сложности программирования -*

* *Простое программирование формул (анализ математических методов) - Л.р. №3\_1, Л.р. №4\_1.*
* *Простые циклы - Л.р. №2\_1, Л.р. №5\_2.*
* *Сложные циклы - Л.р. №2\_2, Л.р. №5\_1 (дан код)*
* *Простые циклы с условиями - Л.р. №1\_1 - Л.р. №1\_4, №6\_1 - Л.р. №6\_3*

1. **Поиск корня алгебраического и трансцендентного уравнения.**

Задача нахождения корней нелинейных уравнений вида

(1)

часто встречается при решении различных физических задач. Нелинейные уравнения можно разделить на два класса:

1. Алгебраические – содержат только алгебраические функции (например, многочлены).
2. Трансцендентные - содержат тригонометрические, показательные, логарифмические функции и др.

Методы решения нелинейных уравнений делятся на *прямые* и *итерационные*. Прямые методы позволяют записать решение в виде некоторого конечного соотношения (формулы).

Итерационные методы предполагают два этапа решения – отыскание приближённого значения корня (начального приближения) и уточнение приближённого значения корня до некоторой степени точности.

Итерационный процесс состоит в построении последовательности приближённых значений корня , которая сходится к истинному значению корня

Общая постановка задачи:

Функция определена и непрерывна на некотором конечном или бесконечном отрезке и удовлетворяет уравнению (1). В некоторых алгоритмах поиска корня необходимы дополнительные условия – существования и непрерывности первой производной или даже второй производной . Всякое значение , обращающее в нуль, (т.е. ) является корнем уравнения или нулём функции .

Полагаем, что уравнение (1) имеет только изолированные корни, т.е. для каждого корня уравнения (1) существует некоторая окрестность, не содержащая других корней этого уравнения. Нахождение изолированных корней состоит из двух этапов:

1. Отделение корней – выделение некоторых интервалов , в которых содержится один и только один корень уравнения (1).
2. Уточнения приближенных значений корней методом последовательных приближений до необходимой степени точности.

Если непрерывная функция принимает значения разных знаков на конце интервала и существует первая производная , которая сохраняет свой знак ( или ), то на этом интервале существует только один корень и .

В данном разделе практикума функции заданы на отрезке  и имеют только один корень .

**Варианты функций.**

Аргумент определен в диапазоне . Для алгоритмов, в которых используются первые производные функции приведены их выражения.

**Задание.**

Найти корень нелинейного уравнения тестовой функции и одного из приведённых выше вариантов.

Тестовая функция , интервал поиска . корень уравнения 414.

**Лабораторная работа № 1\_1. Деление отрезка пополам.**

Сначала проверяется условие – если , то поиск завершён, иначе организуется итерационный процесс, на каждой итерации которого интервал поиска корня функции уменьшается вдвое. Критерием выбора той или иной половины рассматриваемого интервала для дальнейшего поиска является условие, что непрерывная функция принимает значения разных знаков на концах подынтервалов - либо , либо .



Рис.1. Алгоритм «Деление отрезка пополам». Затенённые области исключаются из последующих итераций, границы поиска либо , либо перемещаются на место в зависимости от знака произведения .

***Алгоритм решения:***

**Шаг 0:** Задать интервал поиска , задать малые числа δ и ε – точность приближения к допустимому значению интервала поиска и вычисления функции. Исследуемую функцию оформить как *function f(x).*

**Шаг 1:** Вычислить значения функции и на краях интервала.

Вычислить среднюю точку и значение .

Если , то поиск не проводится, т.к. корень найден (**выход**), иначе переход к **шагу 2**.

**Шаг 2:** Организовать цикл (с предусловием) с проверкой условия (

**Шаг 3:** Вычислить произведение –

Если , то присваиваем ,

иначе присваиваем

Вычислить и значение . 

**Шаг 4:** Конец цикла. Если достигнута заданная точность вычислений, то перейти **к шагу 5**, иначе перейти **к шагу 3**.

**Шаг 5:** Вычисляем и .

Вывести значение корня, значение функции, число циклов, число обращений к вычислению функции.

*Примечание:*

Эффективностьалгоритма определяется количеством обращений к вычислению функции и числом проходов в цикле.

*Примерный вид выходной информации.*

**------корень н/л уравнения---------деление отрезка пополам----------**

интервал [a,b], точность del,eps

a=0.000000 b=3.000000 del=0.010000 eps=0.010000

-----границы интервала----функция---

a=0.000000 b=1.500000

fxsr= -0.4375000

-----границы интервала----функция---

a=0.750000 b=1.500000

fxsr= 0.2656250

. . . . . . . . . . . . . . . . . . . . . . . .

. . . . . . . . . . . . . . . . . . . . . . . .

-----границы интервала----функция---

a=0.996094 b=1.007813

fxsr= 0.0039101

-----границы интервала----функция---

a=0.996094 b=1.001953

fxsr= -0.0019522

----------РЕШЕНИЕ------

xsr=0.999023 fxsr= -0.1952171E-02 циклы n= 9 функции m=30

**Лабораторная работа № 1\_2. Метод хорд.**

Более быстрый способ нахождения корня уравнения на отрезке  заключается в замене криволинейной функции прямолинейным отрезком  - хордой, соединяющей точки и (Рис.1). Уравнение прямой, проходящей через две точкии записывается как

(1)

Для доказательства сходимости процесса поиска корня, считаем, что он отделён и вторая производная сохраняет знак на отрезке . Кривая функции выпукла вниз и, следовательно расположена ниже своей хорды .

Отсюда, полагая , определяется первое приближение корня (пересечение хордой оси абсцисс).

(2)

Затем вычисляется значение функции в точке - и фиксируется точка . Через точки и проводится новая хорда .

Определяется второе приближение корня

(3)

Далее повторяется процесс построения новых хорд и определения последующих приближений значений корня. Обобщая (3) можно записать

(4)

Таким образом, строится монотонно возрастающая последовательность





Опорная точка *B*

Рис.1. Поиск корня уравнения . Алгоритм: метод хорд.

Опорная точка .

Приближенное значение корня уравнения определяем как

(5)

***Алгоритм решения:***

**Шаг 0:** Задать интервал поиска , задать малое число ε – точность схождения последовательности приближённых значений корня. Исследуемую функцию и её первую и вторую производные оформить как *function.*

**Шаг 1:** Вычислить значения функции на концах интервала - , и интервал между двумя соседними приближёнными значениями корня (изначально это ).

**Шаг 2:** Используя уравнение (2) определить приближённое значение корня и запомнить его в ячейке . Переменная цикла .

**Шаг 3:** Организовать цикл с предусловием Используя формулу (4) определить текущее приближённое значение корня (запомнить его в ячейке ) и интервал между двумя соседними приближёнными значениями корня. Присвоить и .

**Шаг 4:** Конец цикла. Если , то выход, иначе переход к **шагу 3.**

**Шаг 5:** Выход.

*Примечание:*

Эффективностьалгоритма определяется количеством обращений к вычислению функции и числом проходов в цикле. Ввести счётчики этих параметров и вывести их конечные значения.

*Примерный вид выходной информации.*

**------корень н/л уравнения---------метод хорд----------**

интервал - [a,b], точность - eps

a= 0.0000 b= 5.0000 eps=0.001000

корень-x, функция-f

x( 1)=0.38461539 f=-.85207099

корень-x, функция-f

x( 2)=0.54285717 f=-.70530611

. . . . . . . . . . . . . . . . . . . . . . . . . . . .

. . . . . . . . . . . . . . . . . . . . . . . . . . . .

корень-x, функция-f

x(16)=0.99797213 f=-.00405163

корень-x, функция-f

x(17)=0.99864763 f=-.00270291

----РЕШЕНИЕ-----

корень=0.99864763 функция=-.00270291

**Лабораторная работа № 1\_3. Метод касательных (метод Ньютона).**

Функция определена и непрерывна на некотором конечном или бесконечном отрезке . Первая производная вторая производная непрерывны и сохраняют знаки на отрезке .



Рис.1. Поиск корня уравнения . Алгоритм: метод касательных.

Опорная точка B.

По своей сути этот метод похож на метод хорд и отличается только способом построения линейных функций, с помощью которых определяется очередное приближённое значение корня уравнения .

Пусть некое приближение корня (- малая величина).

Пользуясь формулой Тейлора в окрестности точки можно записать функцию в виде ряда (высшие члены ряда не учитываются).

 (1)

Полагая , получаем

(2)

Очередное приближённое значение корня определяется как

 ( 3)

Графически метод Ньютона эквивалентен замене части дуги касательной к некоторой точке .

Уравнение касательной в точке имеет вид

(4)

При и , получаем

(5)

Значение производной вычисляется каждый раз заново для точек .

Метод касательных является условно сходящимся методом, то есть для его сходимости

должно быть выполнено следующее условие в области поиска корня

(6)

При произвольном нулевом приближении итерации будут сходиться, если всюду будет выполнено приведённое выше условие. В противном случае сходимость будет лишь в некоторой окрестности корня.

***Алгоритм решения:***

**Шаг 0:** Оформитьвычисление функции и её первой производной как *function.* **Шаг 1:** Задать интервал поиска , задать малое число ε – точность схождения последовательности приближённых значений корня.

**Шаг 2:** Задать начальную точку поиска , задать начальные значения последовательности приближённых значений корня и , вычислить начальное значение интервала между соседними приближёнными значениями корня . Задать начальное значение переменной цикла .

**Шаг 3:** Циклс предусловием . Вычислить очередные значение корня по формуле (5) и интервала , задать .

Если , то выход из цикла.

*Примечание:*

Эффективностьалгоритма определяется количеством обращений к вычислению функции и числом проходов в цикле. Ввести счётчики этих параметров и вывести их конечные значения.

*Примерный вид выходной информации.*

**------корень н/л уравнения---------метод касательных----------**

интервал - [a,b] нач.точка tochoct - eps

a=0.000000 b=3.000000 x0= 2.0000 eps=0.010000

корень-x, функция-f

x( 1)= 1.25000000 f=0.56250000

корень-x, функция-f

x( 2)= 1.02499998 f=0.05062495

корень-x, функция-f

x( 3)= 1.00030482 f=0.00060973

корень-x, функция-f

x( 4)= 1.00000000 f=0.00000000

----РЕШЕНИЕ-----

корень=1.00000000 функция=0.00000000

**Лабораторная работа № 1\_4. Метод секущих.**

Этот метод является модификацией метода касательных. Для случая, когда значения производной на всём отрезке незначительно отличаются друг от друга по величине, достаточно вычислить значение производной в опорной точке и использовать это значения для построения последовательности секущих линий и определения последовательности приближенных значений корня уравнения .



Рис.3.7. Поиск корня уравнения . Алгоритм: метод секущих.

Опорная точка .

***Алгоритм решения:***

**Шаг 0:** Оформитьвычисление функции и её первой производной как *function.*

**Шаг 1:** Задать интервал поиска , задать малое число ε – точность схождения последовательности приближённых значений корня.

**Шаг 2:** Задать начальную точку поиска , задать начальные значения последовательности приближённых значений корня и , вычислить начальное значение интервала между соседними приближёнными значениями корня . Вычислить начальное значение первой производной в точке . Задать начальное значение переменной цикла .

**Шаг 3:** Циклс предусловием . Вычислить очередные значение корня [по формуле (10) л.р. № 4\_3 заменив в ней на ))] и интервала , задать .

Если , то выход из цикла.

*Примечание:* Эффективностьалгоритма определяется количеством обращений к вычислению функции и числом проходов в цикле. Ввести счётчики этих параметров и вывести их конечные значения.

*Примерный вид выходной информации.*

**------корень н/л уравнения---------метод касательных----------**

интервал - [a,b] нач.точка tochoct - eps

a=0.000000 b=3.000000 x0= 2.0000 eps=0.010000

корень-x, функция-f

x( 1)= 1.25000000 f=0.56250000

корень-x, функция-f

x( 2)= 1.10937500 f=0.23071289

. . . . . . . . . . . . . . . . . . . . . . . . . . . . ..

. . . . . . . . . . . . . . . . . . . . . . . . . . . . .

корень-x, функция-f

x( 5)= 1.01243162 f=0.02501779

корень-x, функция-f

x( 6)= 1.00617719 f=0.01239253

----РЕШЕНИЕ-----

корень=1.00617719 функция=0.01239253

**2. Приближение функций. Понятие приближения функций.**

В своей практической деятельности мы часто сталкиваемся с ситуацией, когда те или иные параметры системы или объекта можно определить только в конечном числе точек (узлов). А для проведения численного моделирования необходимо знание значений параметров системы во всей области определения. Наиболее часто исходная функция

приближается многочленами.



Рис.1. Представление табличной функции

В общем случае задачу можно сформулировать следующим образом: Данную функцию требуется приближённо заменить (аппроксимировать) некоторой функцией , так чтобы отклонение от во всей области определения было наименьшим.

**Интерполяция** – приближённая замена табличной функции аналитической**, значения которой в узлах таблицы совпадают со значениями табличной функции.**

Существует несколько схем, реализующих подобную замену. Они отличаются друг от друга как формой исходных таблиц (равноотстоящие узлы или произвольно расположенные), так и по способу выбора узлов, содержащих информацию, используемую для построения интерполирующей функции. Интерполирующая функция может строиться сразу для всего рассматриваемого интервала аргумента или отдельно для разных частей этого интервала.

Одним из основных понятий, связанных с табличными функциями, является понятие конечной разности.

Пусть- заданная функция, а - фиксированная величина приращения аргумента (шаг). Тогда выражение

называется первой конечной разностью функции **.**

В приведённых формулах используется разностный оператор - **.**

- разность значений переменной  в двух соседних узлах таблицы.

- разность значений табличной функции в двух соседних узлах таблицы.

Аналогично определяются конечные разности функции  высших порядков.





Рис.2. Графическое представление конечных разностей и узлов табличной функции.

Таблица 1. Табличная функция и конечные разности. Горизонтальная таблица.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Аргумент** | **Функция** | **1-я к.раз** | **2-я к.раз** | **3-я к.раз** | **4-я к.раз** | **5-я к.раз** |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

Из таблицы 1. видно, что разность более высокого прядка строится на основе разностей предшествующего порядка. Следует обратить внимание на индексацию разностей. Нижний индекс означает, к какому узлу таблицы принадлежит данная разность. Верхний индекс () означает порядок конечной разности (а не возведение в степень *i***).** Для таблицы, содержащей *n* узлов можно построить конечные разности до (*n-1*) порядка включительно.

Горизонтальная таблица разностей используется при построении интерполяционного многочлена Ньютона (первая и вторая интерполяционные формулы Ньютона).

**Варианты функций.**

Аргумент *x*определен в диапазоне *[0÷1]*

Для приведённых функций построить интерполирующие многочлены по Ньютону и Лагранжу.

Тестовая функция , интервал аргумента .

**Лабораторная работа № 2\_1. Интерполирование. Первая интерполяционная**

**формула Ньютона.**

Интерполяционная функция (по Ньютону) представляется как многочлен вида

(1)

где  - координаты узлов табличной функции, а - коэффициенты, значения которых определяются исходя из значений табличной функции в соответствующих узлах. В зависимости от количества членов многочлена получаем различные приближения интерполирующей функции к табличной функции. Так первые два члена представляют линейную интерполяцию между двумя узлами , первые три члена – квадратичную интерполяцию, первые четыре члена – кубическую интерполяцию, и т.д. Если функция задана в узле, то можно построить многочлен -ой степени.

Рассмотрим процесс вычисления коэффициентов . Полагая в (1) , получаем , т.е. .

Для определения  вычислим конечную разность многочлена в точках  и 

(2)

Пологая в (2) , получаем

(3)

Последовательно, продолжая этот процесс, получим

(4)

Следует отметить, что формулы ( 2 и 3 ) записаны для интерполяции на интервале для «опорного» нулевого узла . Для -го узла в формулах нужно произвести замену индексов при переменных , , и т.д. на , , и т.д.

*x2*

X

Y

0

*x*

1

*x*

*x*

Рис. 3. Выбор узлов для квадратичной интерполяции на участке .

Первой интерполяционной формуле Ньютона соответствуют конечные разности, расположенные в горизонтальных рядах Таблицы 1. Интерполирующая функция как бы «прижимается» к левой границе табличной функции. Для построения интерполирующей функции на правой границе табличной функции используется вторая интерполяционная формула Ньютона.



**Задание.**

Используя первую интерполяционную формулу Ньютона построить интерполяционный многочлен второй степени для таблично заданной функции.

***Алгоритм решения.***

* Для тестовой функции в интервале построить табличную функцию , где – количество узлов, отстоящих друг от друга на величину , т.е. создать массив узлов и соответствующий массив значений функции в этих узлах . В тестовых расчётах положить . *Индексы узлов начинаются с «0»*
* Вычислить первые и вторые конечные разности (*создать массивы*)
* Вычислить коэффициенты для каждого интервала Нижние индексы (см.табл.) относятся к коэффициентам полинома (2), верхние – к номеру соответствующего узла таблицы.
* Сформировать массив значений аргумента в середине каждого интервала . *[Индексация () означает только, что значение берётся в середине интервала , а сам массив имеет целочисленную индексацию]*
* Для этих значений аргумента вычислить значения тестовой функции и интерполяционного многочлена .
* Оценить разницу полученных значений и . Результаты сравнений значений и представить в виде таблиц
* Провести численный эксперимент для различного числа узлов. Как результат зависит от количества узлов в заданном интервале.
* Провести интерполяцию для одного из вариантов функций.

**Лабораторная работа № 2\_2. Интерполирование.**

**Интерполяционная формула Лагранжа. (Глобальная интерполяция)**

Для интерполирования табличной функции с неравноотстоящими узлами используется формула Лагранжа.

На отрезке даны () различных значений аргумента - и известны значения функции в этих узлах - . Нужно построить многочлен степени не выше , такой, что для



Рис. 4. Табличная функция и многочлен Лагранжа.

Для построения такого многочлена сначала нужно решить частную задачу - построить вспомогательные многочлены , такие, что при и , т.е.

(1)

*x*

*y*

1



Рис.5. Вспомогательные многочлены.

Так как вспомогательный многочлен обращается в нуль в  узлах

(кроме *i-го*), то он может быть записан как

(2)

где  - постоянный коэффициент.

Полагая в (6 2) и учитывая, что , получаем

(3)

и, следовательно, многочлен (6 2) запишется как

(4)

Чтобы выполнилось исходное требование , интерполяционный многочлен должен иметь вид

(5)

Следует отметить, что интерполяционный многочлен Лагранжа, в отличие от других интерполяционных функций, содержит в явном виде значения , что при решении некоторых задач может оказаться важным фактором.

**Задание.**

Используя интерполяционную формулу Лагранжа построить интерполяционный многочлен для таблично заданной функции.

***Алгоритм решения.***

Для тестовой функции в интервале построить табличную функцию , где – количество узлов, отстоящих друг от друга на величину , т.е. создать массив узлов и соответствующий массив значений функции в этих узлах . В общем случае расстояние между узлами может быть различным.

* В тестовых расчётах положить .
* Сформировать массив значений аргумента .
* Для этих значений аргумента вычислить значения тестовой функции.
* Сформировать массив значений аргумента в середине каждого интервала .
* Для этих значений аргумента вычислить значения тестовой функции.
* Построить интерполяционный многочлен для всего интервала

1. Построить вспомогательные многочлены (4) для этого использовать алгоритм трёх вложенных циклов.

Для каждого значения (всего их ) из массива значений аргумента построить вспомогательных многочленов

Внешний цикл – перебор значения по индексам от 0 до

1. Внутренний цикл - перебор значения по индексам от 0 до и формирование матрицы вспомогательных многочленов размерностью

1. Самый внутренний цикл – вычисление произведений в числителе и знаменателе (4) с перебором значений в сомножителях .

* Вычислить значения многочлена (5) в серединах интервалов , используя алгоритм двух вложенных циклов.
* Оценить разницу полученных значений и . Результаты сравнений значений и представить в виде таблиц
* Провести интерполяцию для одного из вариантов функций.

*Студентам: Постарайтесь сначала написать собственный алгоритм.*

*Если это вызывает сложности, воспользуйтесь приведённым кодом.*

real x(0:100),y(0:100),x1(0:100),y1(0:100)

real r(0:100,0:100),p(0:100)

write(\*,\*) 'vvesti a b n'

read(\*,\*) a,b,n

! ----- шаг таблиц, аргумент, функция -----------------

h=(b-a)/n

do i=0,n

x(i)=h\*i

y(i)=f(x(i))

enddo

! ----- аргумент, функция в серединах интервалов-----------------

do i=0,n-1

x1(i)=h\*i+h/2

y1(i)=f(x1(i))

enddo

!----реализация формулы (5)

!------ (5)

do k=0,n-1 *!--цикл по*

do i=0,n *!--цикл по*

u=1

d=1

do j=0,n *!--цикл по в сомножителях: числитель и знаменатель*

if (i.ne.j) then

u=u\*(x1(k)-x(j))

d=d\*(x(i)-x(j))

endif

enddo

r(k,i)=u/d

enddo

enddo

!-----значение многочлена (5) , вычисление суммы

do k=0,n-1

sum=0

do i=0,n

sum=sum+y(i)\*r(k,i)

enddo

p(k)=sum

enddo

!------------------------------------------------------

write(\*,\*) ' x y x1 y1'

write(\*,10) (x(i),y(i),x1(i),y1(i),i=0,n)

write(\*,\*) ' p y1'

write(\*,11) (p(j),y1(j),j=0,n-1)

10 format(4(x,f8.5))

11 format(2(x,e14.7))

end

!------тестовая функция-------------------

real function f(x)

real x

f=sin(x)

return

end

**3. Численное интегрирование функций.**

Пусть функция  непрерывна на отрезке  и известна её первообразная , тогда определённый интеграл от этой функции в пределах от  до  может быть вычислен по формуле Ньютона-Лейбница

(1)

где

Вычисление однократного интеграла численным методом называется *механической квадратурой,* двойного – *механической кубатурой*. Соответствующие формулы называются *квадратурными* и *кубатурными* формулами.

Основным инструментом численного интегрирования является представление подынтегральной функции интерполирующим полиномом. Такая аппроксимация позволяет приближённо заменить определённый интеграл конечной суммой

(2)

где - значения функции в узлах интерполяции,  - числовые коэффициенты. Это выражение называется квадратурной формулой, а правая часть – квадратурной суммой. В зависимости от способа вычисления этой суммы реализуются различные численные методы - методы прямоугольников, трапеций, парабол, сплайнов и др.

Квадратурную сумму можно вычислить как

(3)

где - приближённое значение площади элементарной криволинейной трапеции, соответствующей элементарному отрезку .

В качестве точек можно выбрать левые точки или правые точки элементарного интервала . Обозначая и получаем формулы метода прямоугольников для левых и правых сумм

(4)

(5)



Рис.1. Левые суммы.



Рис.2. Правые суммы.

Более точным приближение метода прямоугольников является приближение с использованием средних значений функции в элементарных прямоугольниках (в полуцелых узлах).

(6)



Рис.3. Центральные суммы.

Следующим методом, использующим линейную интерполяцию, является метод трапеций.

График функциипредставляется в виде ломанной линии, соединяющей точки . Площадь всей фигуры приближённо складывается их суммы площадей прямолинейных трапеций (Рис.4). Площадь каждой трапеции определяется как

 (7)



Рис.4. Метод трапеций. Фрагмент – малый интервал.

Таким образом, полная площадь под ломанной кривой (т.е. приближённое значение интеграла запишется как)

(8)

Для таблицы с равноотстоящими узлами 

формулы прямоугольников и трапеций принимают соответственно вид

(9) (10)

Повышение точности вычислений достигается за счёт повышения степени интерполяционных многочленов – квадратичной интерполяции (метод Симпсона) и интерполирование с помощью сплайнов.



Рис.5. Метод Симпсона. Фрагмент – малый интервал.

**Метод Симпсона.**

Пусть функция  определена на интервале , который разобьём на чётное число  равных частей с шагом . На каждом отрезке подынтегральную функцию заменим многочленом второй степени

при (11)

Коэффициенты  можно вычислить из условия прохождения многочлена через узлы табличной функции.

Рассмотрим интерполяционный многочлен второй степени Лагранжа, проходящий через точки

(12)

Учитывая равенства можно записать сумму площадей  и  как определенный интеграл

(13)

Проводя вычисления на всех отрезках , получаем формулу Симпсона

(14)

**Варианты определённых интегралов**

*;*

*;*

*a и b –* целые числа *,* - пределы интегрирования,

красным цветом выделены первообразные интегралов ().

**Лабораторная работа № 3. Интегрирование.**

**Задание.**

Используя приведённые выше методы интегрирования (4, 5, 9, 10, 14) вычислить определённые интегралы и сравнить их значения между собой, а также со значениями аналитической первообразной.

***Алгоритм решения***

1. В заданном диапазоне сформировать массив аргументов с шагом , число интервалов от 10 до 50.
2. Для заданного массива аргументов сформировать массив значений подынтегральной функции.
3. Вычислить значения интегралов и первообразной для двух вариантов функции.
4. Изменяя величину шага, оценить его влияние на точность вычислений.
5. Результаты представить в виде таблицы.
6. **Численное дифференцирование.**

Для численного определения значений производных можно использовать представление их через конечные разности

(1)

Это представление является *аппроксимацией производной с помощью конечных разностей.*



Узлы табличной функции

Рис.6. Графическое представление табличной функции.

Для табличной функции возможны представления производной с помощью левых разностей, правых разностей и центральных разностей.

Рассмотрим табличную функцию, определённую на сетке с постоянным шагом (Рис.6), первая конечная разность . Определим первую производную как

(2)

Таким образом, производную в точке  выразили через разность значений функции в этой точке и в точке слева от нее , т. е. через левые разности. Аналогично можно поступить, выразив производную с помощью правых разностей

(3)

или с помощью центральных разностей

(4)

По такому же принципу можно определить производные более высокого порядка:

(5)

***Такие же соотношения можно получить с помощью ряда Тейлора***

(6)

Пусть функция записана в виде таблицы

При ряд Тейлора с точностью до членов порядка запишется как

(7)

Отсюда производная в точке определяется как

(8)

Это выражение совпадает с ранее полученным выражением аппроксимации первой левосторонней производной первого порядка . Также можно получить выражение для правосторонней производной.

Рассмотрим выражения для первой и второй производных второго порядка погрешности.

(9)

Вычитая из первого уравнения второе, получим

(10)

что совпадает с аппроксимацией центральными разностями. Погрешность имеет второй порядок.

Складывая уравнения, оценим погрешность аппроксимации производной второго порядка

(11)

Данное выражение также совпадает с аппроксимацией центральными разностями.

***Для численного определения производных можно использовать различные интерполяционные полиномы с различным количеством членов***. На практике более удобным оказывается использование полиномов Лагранжа, т.к. в них в явном виде присутствуют значения табличной функции в узлах. В зависимости от количества используемых узлов получаем выражения для производных функции с различной погрешностью вычислений.

Рассмотренный источник погрешностей – погрешность аппроксимации, которая определяется величиной остаточного члена. При уменьшении шага таблиц эта погрешность, как правило, уменьшается.

Другие погрешности – неточность значений функции в узлах и погрешность округления вычислений. Эти погрешности возрастают при уменьшении шага .

Полином Лагранжа имеет вид (см. лаб.р. №3\_2 (5))

Рассмотрим случай трёх узлов интерполирования ().

Расписав многочлен как сумму трёх (соответствующих) слагаемых и приведя их к общему знаменателю, для таблиц с постоянным шагом получим следующие выражения для многочлена и первой производной

(12)

где - значение производной третьего порядка в некоторой внутренней точке

Запишем значения производных в точках

(13)

Подобным образом можно вычислить производные для любого количества узлов аппроксимации. При четных значениях  наиболее простые выражения получаются для центральных точек. Вариант с чётным значением  - *аппроксимация с помощью центральных разностей.* Он более удобен и при вычислении вторых производных. Для четырёх узлов производные

Для пяти узлов

Аналогичным образом можно получить выражения для вторых производных, дважды продифференцировав соответствующий многочлен Лагранжа.

Для трёх узлов

Для четырёх узлов

Для пяти узлов

**Варианты функций**

1. *;*
2. *;*
3. *;*

**Лабораторная работа № 4. Дифференцирование.**

**Задание.**

Для функции в интервале построить табличную функцию , где ( – количество узлов, отстоящих друг от друга на величину .

Используя выражения первых и вторых производных, полученных для табличной функции из интерполяционной формулы Лагранжа, вычислить их значения в узлах и оценить точность вычислений в зависимости от числа узлов , используемых для построения многочлена Лагранжа.

* Рассмотреть варианты и (нумерация узлов начинается «0»), т.е. рассмотреть узлы (узлы с номерами большими не рассматривать).
* Сравнить результаты между собой, а также сравнить со значениями аналитической производной.
* Провести численный эксперимент для двух значений количества узлов .

***Алгоритм решения.***

* Задать количество узлов . .
* Сформировать массив значений аргумента . Индекс от 0 до .
* Сформировать массив значений функции .
* Используя интерполяционную формулу Лагранжа для таблиц с постоянным шагом записать выражения производных в схемах с четырьмя ) и пятью

узлами. Вычислить значения производных в этих узлах.

* Вычислить аналитические значения производных в рассматриваемых узлах.
* Для узла с индексом «1» вычислить значения левосторонней, центральной и правосторонней производных и сравнить их со значениями, полученными из многочлена Лагранжа.
* Провести численный эксперимент с варьированием величины шага.
* Вычислить значения вторых производных в схемах с четырьмя и пятью узлами.
* Результаты работы представить в соответствующей форме, удобной для анализа (см. пример вывода результатов ниже).

*Примерный вид выходной информации.*

**------дифференцирование----------**

=0-й= =1-й= =2-й= =3-й= =4-й=

==== первые производные===

0.1000000E+01 0.9950042E+00 0.9800666E+00 0.9553365E+00 0.9210610E+00

==== 4 узла

0.1000030E+01 0.9949925E+00 0.9800798E+00 0.9552920E+00

==== 5 узлов

0.9999807E+00 0.9950089E+00 0.9800633E+00 0.9553415E+00 0.9210409E+00

==== 2\_л = 2\_ц = 2 \_п =

0.9983342E+00

0.9933466E+00

0.9883591E+00

==== вторые производные=====

0.0000000E+00 -0.9983342E-01 -0.1986693E+00 -0.2955202E+00 -0.3894183E+00

==== 4 узла

-0.1000613E-02 -0.9975135E-01 -0.1985021E+00 -0.2972528E+00

==== 5 узлов

0.8106232E-03 -0.9991601E-01 -0.1986668E+00 -0.2954416E+00 -0.3902406E+00

1. **Решение СЛАУ**

**Лабораторная работа № 5\_1.**

Способы решения систем линейных алгебраических уравнений можно разделить на две группы:

1. Точные методы – представляют собой конечные алгоритмы для вычисления корней системы (правило Крамера, метод Гаусса, метод главных элементов, метод квадратных корней и др.)
2. Итерационные методы – построенные на сходящихся бесконечных процессах, обеспечивающих заданную точность вычисления корней системы (метод итераций, метод Зейделя, метод релаксации и др.)

Полученные результаты в обоих случаях являются приближёнными из-за округления численных результатов и погрешностей методов.

Рассмотрим систему  линейных уравнений с  неизвестными

(1)

Введем обозначения:

* матрица коэффициентов
* вектор-столбец правых частей системы и вектор-столбец неизвестных

, .

Система (24) в матричном виде запишется как

(2)

Необходимым и достаточным условием существования единственного решения системы является условие неравенства нулю детерминанта матрицы .

(3)

В противном случае система называется *вырожденной* и может иметь либо множество решений, либо не иметь ни одного решения.

При  система является *плохо обусловленной*. Это условие является необходимым, но недостаточным для определения плохо обусловленной системы.

Прямые методы – расчёт ведётся по конечным соотношениям (формулам) для определения неизвестных. Число операций заранее известно. Достоинством прямых методов является их относительная простота и универсальность. Недостатком является необходимость хранить в памяти компьютера сразу всю матрицу, что при большой размерности системы приводит к неэффективности вычислений, особенно, если матрица содержит много нулевых элементов – в этом случае производятся ненужные операции с нулями. Другим недостатком является накопление погрешности результатов, т.к. последующие операции используют данные, полученные на предшествующих операциях (это особенно опасно для плохо обусловленных матриц).

Итерационные методы – методы последовательных приближений. Для работы процедуры необходимо задать начальное приближение, т.е. задать произвольное значение вектору неизвестных. Это стартовое значение позволяет провести один цикл вычислений (итерацию). Затем вычисления повторяются до достижения нужной точности результатов. Алгоритмы итерационных методов, как правило, более сложные, чем алгоритмы прямых методов. Их достоинством является отсутствие необходимости хранить в памяти машины целиком все матрицы, достаточно хранить несколько векторов с  компонентами. Погрешности вычислений не накапливаются, т.к. на текущей итерации используются результаты только предшествующей итерации, а не всех предшествующих вычислений. Более того, случайный сбой вычислений означает, что расчёт начинается с нового произвольного начального приближения.

**Прямые методы решения СЛАУ.**

**Метод Гаусса**

Этот метод основан на приведении матрицы к треугольному виду. Это достигается последовательным исключением неизвестных из уравнений системы. С помощью первого уравнения исключается  из всех последующих уравнений. Преобразованная система содержит  уравнение относительно  неизвестных. Процесс повторяется - с помощью первого уравнения преобразованной системы исключается  из всех последующих уравнений этой системы. Процесс продолжается до получения одного уравнения относительно неизвестного . Таким образом, из исходной системы и преобразованных систем уравнений формируется матрица треугольного вида. Этот процесс называется *прямым ходом метода Гаусса.*

Например, для системы из трёх уравнений

(4)

Формирование матриц в процессе прямого хода можно отобразить следующим образом

Разделив первое уравнение системы (4) на , получим уравнение

(5)

которое умножим на 

(6)

Вычтем из второго уравнения (4) уравнение (6), получим

(7)

Умножим уравнение (5) на и вычтем это уравнение из третьего уравнения (4)

(8)

В результате этих действий пришли к системе из двух уравнений, не содержащих переменную

(9)

где , , (10)

, ,

Проведём подобное преобразование для системы из двух уравнений (9) и получим одно уравнение с одним неизвестным .

(11)

На этом шаге процедура прямого хода заканчивается, треугольная матрица сформирована.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

(12)

В процессе формирования треугольной матрицы (исключение неизвестных) производится деления на коэффициенты и т.д. Поэтому они должны быть отличными от нуля. В случае необходимости нужно переставить уравнения системы местами, чтобы выполнить это требование. На этом принципе исключения переменных основаны несколько алгоритмов.

Приведённый ниже алгоритм не требует хранения промежуточных данных (пересчёта коэффициентов промежуточных этапов).В общем виде вычисление коэффициентов можно записать как

(13)

На следующем этапе производится вычисление неизвестных, начиная с

В результате обратного хода

(14 )

Последовательно вычисляются все неизвестные. Этот алгоритм часто называют схемой единственного деления, а элемент - ведущим элементом.

**Задание.**

Используя зависимости (13) и (14) найти решение тестовой системы уравнений

***Алгоритм решения.*** *(псевдокод).*

Для (*цикл последовательного исключения переменных*)

Для

Для

Студентам: Постарайтесь сначала написать собственный алгоритм.

Если это вызывает сложности, воспользуйтесь приведённым кодом.

**Код программы**

real a(5,5),t(5,5),b(5),x(5)

write(\*,\*) 'vvesti n'

read(\*,\*) n

do i=1,n

do j=1,n

if (i.eq.j) then

a(i,j)=i

else

a(i,j)=0.1

endif

enddo

b(i)=i

enddo

write(\*,\*) 'исходная матрица a(n\*n) и вектор b(n)'

write(\*,10) ((a(i,j),j=1,5),b(i),i=1,5)

do k=1,n-1

do i=k+1,n

t(i,k)=a(i,k)/a(k,k)

b(i)=b(i)-t(i,k)\*b(k)

do j=k+1,n

a(i,j)=a(i,j)-t(i,k)\*a(k,j)

enddo

enddo

! промежуточная печать на k-ой итерации

write(\*,\*) ' матрица a(n\*n) и вектор b(n) k=',k

write(\*,10) ((a(i,j),j=1,5),b(i),i=1,5)

enddo

! прямой ход

write(\*,\*) 'результирующая матрица и вектор’

write(\*,10) ((a(i,j),j=1,5),b(i),i=1,5)

! обратный ход

x(n)=b(n)/a(n,n)

sum=0

do k=n-1,1,-1

do j=k+1,n

sum=sum+a(k,j)\*x(j)

enddo

x(k)=(b(k)-sum)/a(k,k)

enddo

write(\*,\*) 'результат'

write(\*,10) (x(k),k=1,n)

10 format (5e12.5,4x,e12.5)

end

*Примерный вид выходной информации.*

**------решение СЛАУ метод Гаусса----------**

Размерность матрицы n=5

'исходная матрица a(n\*n) и вектор b(n)'

0.10000E+01 0.10000E-01 0.10000E-01 0.10000E-01 0.10000E-01 0.10000E+01

0.10000E-01 0.20000E+01 0.10000E-01 0.10000E-01 0.10000E-01 0.20000E+01

0.10000E-01 0.10000E-01 0.30000E+01 0.10000E-01 0.10000E-01 0.30000E+01

0.10000E-01 0.10000E-01 0.10000E-01 0.40000E+01 0.10000E-01 0.40000E+01

0.10000E-01 0.10000E-01 0.10000E-01 0.10000E-01 0.50000E+01 0.50000E+01

==== решение x(1) - x(5)====

0.99601E+00 0.99800E+00 0.99867E+00 0.99900E+00 0.99920E+00

===== проверка====

1.0000000 1.9999999 3.0000000 4.0000000 5.0000000

**Лабораторная работа № 5\_2.**

**Итерационные методы.**

**Метод простой итерации.**

При большом числе уравнений метод Гаусса может оказаться достаточно сложным. В этом случае более удобно пользоваться не точными, а приближёнными численными методами. Систему уравнений  представим в развёрнутом виде

(1)

Полагая решим первое уравнение относительно , второе — относительно и т. д. Получаем эквивалентную систему

(2)

где , (3)

и

при

Введя матрицы

и (4)

и систему уравнений можно записать в матричном виде

(5)

Эта система решается методом последовательных приближений. За нулевое приближение принимается, например, столбец свободных членов .

Затем строится первое приближение

(6)

потом — второе приближение

(7)

и т. д. Общий алгоритм решения

(8)

В результате получаем последовательность, предел которой (если он !!!!! !!!!! существует)

является решением системы

Формулы приближений можно представить в виде

(9)

**Задание.**

Систему уравнений из предшествующего задания решить итерационным методом и сравнить результаты.

***Алгоритм решения.***

* Задать матрицу и вектор .

;

* Сформировать матрицу и вектор .
* За нулевое приближение принять столбец свободных членов .
* В цикле организовать процесс (9). Завершение процесса – значения вектора на двух соседних итерациях отличаются на малую величину (точность вычислений)

размерность СЛАУ n= 5

исходный массив А(n\*n) b(n)

0.10000E+01 0.10000E-02 0.10000E-02 0.10000E-02 0.10000E-02 0.10000E+01

0.10000E-02 0.20000E+01 0.10000E-02 0.10000E-02 0.10000E-02 0.20000E+01

0.10000E-02 0.10000E-02 0.30000E+01 0.10000E-02 0.10000E-02 0.30000E+01

0.10000E-02 0.10000E-02 0.10000E-02 0.40000E+01 0.10000E-02 0.40000E+01

0.10000E-02 0.10000E-02 0.10000E-02 0.10000E-02 0.50000E+01 0.50000E+01

==== решение ====

0.99601E+00 0.99800E+00 0.99867E+00 0.99900E+00 0.99920E+00

===== проверка ====

1.0000000 1.9999999 3.0000000 4.0000000 5.0000000

1. **Нахождения минимума функции.**

Задачи оптимизации, как правило, связаны с нахождением минимума многомерной целевой функции.

В данном разделе практикума рассматриваются наиболее простые методы поиска минимума одномерной функции. Алгоритмы предполагают наличие только одного минимума функции (унимодальная функция) на исследуемом интервале изменений аргумента. В понятие минимизации функции следует включить не только поиск её экстремума, но и оценку минимального значения функции на краях исследуемого интервала, т.е. функция может быть монотонно возрастающей или убывающей.

Рассматриваемые алгоритмы нахождения локального минимума относятся к методам *сокращения интервала поиска*. Все алгоритмы построены на анализе значений минимизируемой функции в пробных точках, расположенных внутри интервала поиска. Различие методов заключается в принципе выбора места расположения этих пробных точек.

Принцип симметричного расположения пробных точек реализуется несколькими способами – методом дихотомии, методом деления отрезка пополам, методом золотого сечения. Алгоритмы различаются быстродействием, которое, в основном, связано с количеством обращений к вычислению минимизируемой функции. В большинстве случаев этот критерий является основным, т.к. вычисление самой функции составляет, как правило, основное время счёта. При этом нужно учитывать, что процесс поиска минимума функции определяется задаваемой точностью расчётов, т.е. конечным размером интервала .

**Задание.**

Произвести поиск минимума функции при различных значениях точности вычисления. Сравнить методы (л/р. 6\_1, 6\_2, 6\_3) по быстродействию – количеству обращений к вычислению функции при одинаковой точности вычисления. Значения варьируемых параметров (границы интервала поиска минимума функции и точность вычисления) задавать с клавиатуры, используя операторы ввода данных.

**Варианты функций**

**Лабораторная работа № 6\_1. Дихотомия.**

Наиболее простой алгоритм, в котором на каждой итерации интервал поиска минимума функции *сокращается ровно вдвое*. Вычисление значения минимизируемой функции на каждой итерации производится дважды. Суть алгоритма заключается в следующем – на текущей итерации определяется середина отрезка и вычисляются значения двух близко лежащих точек и ***,*** где наперёд заданное малое значение аргумента функции (Рис.1). Далее сравниваются значения функции в точка и и выбирается половина интервала, в которой содержится минимум функции.

*f(x)*

Рис.1. Метод дихотомии.

***Алгоритм решения.***

Пусть функция задана на отрезке и имеет только один минимум.

**Шаг0:** Задать интервал поиска , задать малое число ε *-* точностьвычисления функции,задать малое число - приращение координаты (см. рисунок).

**Шаг1:** Начало цикла с постусловием. Определить середину интервала , выделить две симметричные относительно краёв интервала поиска точки на оси аргументов

.

**Шаг2:** Вычислить значения функции и

**Шаг3:** Сравнить значения функции и

Если *,* то присвоить , иначе .

**Шаг4:** Конец цикла. Признак завершения счета. Если и , то перейти к шагу 5, иначе перейти к шагу 1.

**Шаг5:** Вычислить и Вывести результаты поиска - и количество обращений к вычислению функции .

*Примечание : цикл можно организовать как с предусловием, так и с постусловием.*

Для тестирования программы найдите минимум параболы на интервале [-1, 3]. Результат - , .

**Лабораторная работа № 6\_2. Деление отрезка пополам.**

Метод, в котором на каждой итерации интервал поиска минимума функции *сокращается вдвое*. Он основан на выборе трёх пробных точек, расположенных равномерно на интервале поиска минимума целевой функции. На текущей итерации определяется середина отрезка и значения двух симметрично расположенных точек *.* Далее сравниваются значения функции в точках ,. Можно определить, какую часть интервала поиска следует исключить на следующей итерации.

На Рис.2. приведены возможные положения расположения минимума целевой функции, которые могут возникнуть в процессе сокращения интервала поиска. Затенённые области исключаются из дальнейшего рассмотрения.

При *–* исключается правая половина (Рис.2*а*.), при - левая половина (Рис.2*б*.) и при - левая и правая четверти (Рис.2*в*.). Соответствующим образом переносятся границы интервала поиска (Рис.2*а*.) или (Рис.2*б*.) или и (Рис.2*в*.) и формируется очередной интервал ].

Поиск завершается, когда интервал поиска сокращается до заданной малой величины δ.

На первом шаге поиска производится три обращения к вычислению целевой функции в точках *xср*, *x1* и *x2* напоследующих шагах поиска – два обращения (в точках *x1* и *x2*) Общее количество обращений к вычислению целевой функции определяется как (2\**N+1*), где *N* – число шагов поиска.







Рис.2. Возможные положения минимумов целевой функции.

***Алгоритм поиска.***

Пусть функция задана на отрезке имеет только один минимум

**Шаг0:** Задать малые значения *-* точностьвычисления функции,- точностьвычисления интервала поиска.

**Шаг1:** Цикл с постусловием. Определить середину интервала , Вычислить в этой точке значение функции *..*

**Шаг2:** Выделить две симметричные относительно краёв интервала поиска точки на оси аргументов

**Шаг3:** Вычислить значения функции и

**Шаг4:** Сравнить значения функции и *.*

Если *,* то присвоить , вычислить точки

перейти к шагу 6.

Иначе перейти к шагу 5.

**Шаг5:** Сравнить значения функции  и *.*

Если *,* то присвоить ,

вычислить точки *,* перейти к шагу 6. Иначе присвоить *,* , вычислить точки

**Шаг6:** Завершения счета. Если и , то перейти к шагу 7, иначе перейти к шагу 1.

**Шаг7:** Вычислить и Вывести результаты поиска - и количество обращений к вычислению функции

*Примечание : цикл можно организовать как с предусловием, так и с постусловием*

Для тестирования программы найдите минимум параболы на интервале [-1, 3]. Результат - , .

**Лабораторная работа № 6\_3. Золотое сечение.**

Из рассмотрения выше приведенных методов можно сделать выводы позволяющие повысить эффективность поиска:

1. Если количество пробных точек равно двум, то их следует размещать на одинаковых расстояниях от середины интервала.
2. В соответствии с общей минимаксной стратегией пробные точки должны размещаться по симметричной схеме таким образом, чтобы отношение длины исключаемого подынтервала к величине интервала поиска оставалось постоянным.
3. На каждой итерации поиска должна вычисляться только одна пробная точка и значение целевой функции в этой точке.

Геометрическая интерпретации принципа метода золотого сечения заключается в следующем. Дана прямая ACB

A

B

C

Отношение отрезков прямой определяется соотношением AC/AB=CB/AC, т.е. отношение большего отрезка к целому равно отношению меньшего отрезка к большему отрезку. Если принять целый отрезок АВ=1, то . Приближённо АС≈0.62, а СВ≈0.38.

Количество обращений к вычислению целевой функции определяется как (N+1), где N – число шагов поиска.

Для тестирования программы найдите минимум параболы на интервале [-1, 3]. Результат - .





Рис.3. Возможные положения минимума целевой функции.

***Алгоритм поиска.***

Пусть функция задана на отрезке имеет только один минимум

**Шаг0:** Задать малые значения *-* точностьвычисления функции,- точностьвычисления интервала поиска.

Шаг1: Задать коэффициенты =0.38, интервал и

**Шаг2:** Цикл с постусловием. Вычислить длину отрезка . Выделить две точки и . Вычислить в этих точках значения функции и *.*

**Шаг3:** Сравнить значения функции и*.*

Если *,* то присвоить , вычислить

и *,*

иначе присвоить , вычислить ,

и

**Шаг4:** Конец цикла. Если и , то перейти к **шагу 2**,

иначе перейти к шагу 5, завершение поиска.

**Шаг5:** Вычислить и Вывести результаты поиска - и количество обращений к вычислению функции

*Примечание : цикл можно организовать как с предусловием, так и с постусловием.*

Для тестирования программы найдите минимум параболы на интервале [-1, 3]. Результат - , .

**Сравнение эффективности приведённых алгоритмов.**

Пусть длина исходного интервала поиска , а конечного интервала - . Эффективность метода можно оценить по отношению В методах дихотомии и деления отрезка пополам длина конечного интервала составляет , а в методе золотого сечения -. При заданной точности вычислений ε количество обращений к вычислению целевой функции в двух первых методах составляет , а для метода золотого сечения .