Государственный комитет Российской Федерации
по высшему образованию

Санкт -Петербургский государственный
электротехнический университет

Методические указания
к лабораторным работам по дисциплине
"Вычислительная математика"
(для направления "Информатика и
вычислительная техника" и специальности 22.04)
Часть 1

Санкт-Петербург
1996

УДК 519.7

Методические указания к лабораторным работам по дисциплине "Вычислительная математика" (для направления "Информатика и вычислительная техника" и специальности 22.04), часть 1 / Сост. В.Н. Кафтасьев, А.Р. Лисс, М.С. Титов; ГЭТУ. - С.-Пб.; 1996. -32с.

Использование лабораторных работ и методических указаний по их выполнению, представленных в первой части сборника, при проведении занятий должно обеспечить овладение основными понятиями и методами классических разделов вычислительной математики. Указания содержат формулировку заданий курсовых работ и необходимые для их исполнения пояснения. При разработке лабораторных работ особое внимание уделялось вопросам исследования точности и обусловленности применяемых методов машинных вычислений.

Предназначены для студентов ФАВТ направления "Информатика и вычислительная техника" и специальности 22.04, а также для студентов других специальностей, изучающих численные методы решения задач на ЭВМ.

Утверждено

редакционно-издательским советом университета в качестве методических указаний.

ВВЕДЕНИЕ

Цикл лабораторных работ предназначен для студентов второго курса (четвертый семестр), изучающих дисциплину "Вычислительная математика" и работающих в компьютерных классах на базе ЭВМ типа IBM PC AT-286 и выше, снабженных компилятором языка C (C++). Первые две работы цикла посвящены особенностям машинной арифметики, точности вычислений на ЭВМ и обусловленности вычислительной задачи. Они служат практической иллюстрацией вводной части теоретического курса дисциплины. Каждая из последующих работ ставит целью изучение либо конкретного численного метода, либо набора численных процедур (схем, формул), позволяющих реализовать выполнение одной из классических задач вычислительной математики: решение нелинейных уравнений, интерполирование функций, численное интегрирование, решение систем линейных алгебраических уравнений. Порядок расположения лабораторных работ в методических указаниях соответствует последовательности изложения лекционного материала. Для углубленного изучения перечисленных выше задач и методов их решения целесообразно воспользоваться литературой [1-10], приведенной в библиографическом списке.

Выполнение каждой лабораторной работы следует осуществлять поэтапно в следующем порядке:

- подготовка к решению задачи на персональной ЭВМ (ПЭВМ);
- проведение вычислительного эксперимента на ПЭВМ;
- анализ результатов вычислений; оформление отчета.

Подготовка к решению задачи на ПЭВМ производится как вне компьютерного класса, так и непосредственно на ПЭВМ. Она включает:

- ознакомление с описанием работы и заданием для выполнения;
- составление программных модулей, содержащих определенные заданием и персональным вариантом вычислительной процедуры, и/или ввод исходных данных;
- компиляцию разработанных программных модулей, их отладку и сопряжение с имеющимся для большинства работ программами-функциями, реализующими конкретные численные методы;
- планирование вычислительного эксперимента на ПЭВМ в рамках выполняемого задания.

Программы - функции, предназначенные для применения в процессе вычислений, представлены в виде библиотеки модулей на языке программирования C++. Это предопределяет ориентацию цикла работ на студентов, владеющих данным языком и навыками программирования в необходимом объеме.

Проведение вычислительного эксперимента на ПЭВМ осуществляется в соответствии с порядком выполнения работы и заданием на исследование указанных зависимостей и обусловленности изучаемого метода.

Анализ результатов вычислений заключается в построении исследуемых зависимостей и сравнительной оценке метода (вычислительной процедуры) по характерным для данной группы методов параметрам, например, скорости сходимости, степени обусловленности, достижимой точности и т.п.

Анализ результатов и оформление отчета производится вне компьютерного класса.

Отчет должен содержать:

- постановку задачи;
- тексты разработанных программ;
- результаты вычислений, их теоретический и экспериментальный анализ в виде таблиц и графиков, снабженных необходимыми комментариями;
- развернутые выводы по лабораторной работе.

В методических указаниях приведено также задание на курсовую работу, которая ориентирована на исследование четырех методов решения нелинейных уравнений.

1. ОСОБЕННОСТИ МАШИННОЙ АРИФМЕТИКИ, ТОЧНОСТЬ ВЫЧИСЛЕНИЙ НА ЭВМ

(Лабораторная работа №1)

В фундаменте математического анализа прочно утвердилась система действительных чисел. Однако, как бы она не упрощала анализ, практические вычисления вынуждены обходиться без нее [9].

Обычным способом аппроксимации системы действительных чисел в ЭВМ посредством конкретных математических представлений являются числа с плавающей точкой. Множество F чисел с плавающей точкой характеризуется четырьмя параметрами: основанием b, точностью t и интервалом показателей [L,M]. Каждое число x с плавающей точкой, принадлежащее F, имеет значение

$$x = \pm\left(d_1/b + d_2/b^2 + \Lambda + d_t/b^t\right)b^n,$$

где целые числа d_1, d_2, K, d_t удовлетворяют неравенствам $0 \le d_i \le b-1 \quad (i = 1,2,\text{K}\ t)$ и $L \le n \le M$. Если для каждого ненулевого x из F справедливо $d_1 \neq 0$, то система F называется нормализованной. Целое число n называется показателем, а число $f = \left(d_1/b + d_2/b^2 + \Lambda + d_t/b^t\right)$ - дробной частью. Обычно целое число $b^n f$ хранится по той или иной схеме представления, принятой для целых чисел, например, величины со знаком, дополнения до единицы или дополнения до двух. Если принять $-N \le n \langle N$, $N = 2^{(m-1)}$, то переходим к общепринятой терминологии, при которой t - разрядность мантиссы, m - разрядность порядка.

Действительная машинная реализация представлений чисел с плавающей точкой может отличатся в деталях от рассматриваемой идеальной, однако различия несущественны, и на практике их почти всегда можно игнорировать, анализируя основные проблемы ошибок округления. Величина b^{1-t} является оценкой относительной точности плавающей арифметики, которая характеризуется посредством машинного эпсилон, т.е. наименьшего числа с плавающей точкой ε, такого, что 1+ε>1. Точное значение машинного эпсилон зависит не только то указанных выше параметров, но и от принятого способа округления.

В вычислительных машинах используются различные системы чисел с плавающей точкой, причем в некоторых ЭВМ несколько систем. Так, для современных ПЭВМ характерно применение двух систем, которые называются обычной точностью и удвоенной точностью.

Рассматриваемое множество F не является континуумом или даже бесконечным множеством. Оно содержит ровно $2(b-1)b^t(M-L+1)+1$ чисел, которые расположены неравномерно (равномерность расположения имеет место лишь при фиксированном показателе). В силу того, что F - конечное множество, не представляется возможным сколь-нибудь детально отобразить континуум действительных чисел. Например, действительные числа модулей, большим максимального элемента из F, вообще не могут быть отображены, причем последнее справедливо также в отношении ненулевых действительных чисел, меньших по абсолютной величине по сравнению с наименьшим положительным числом из F, и, наконец, каждое число из F должно представлять целый интервал действительных чисел, для которой, как и для любой модели, присущи допущения и ограничения.

На множестве F определены арифметические операции в соответствии с тем, как они выполняются ЭВМ. Эти операции, в свою очередь моделируются в машине посредством приближений, называемых плавающими операциями. Для плавающих операций сложения, вычитания, умножения и деления существует возможность возникновения ошибок округления, переполнения и появления машинного нуля. Следует отметить, что операции плавающего сложения и умножения коммутативны, но не ассоциативны, и дистрибутивный закон для них также не выполняется. Невыполнение указанных алгебраических законов, имеющих фундаментальное значение для математического анализа, приводит к сложности анализа плавающих вычислений и возникающих при этом ошибок.

Целью лабораторной работы №1 является изучение особенностей вычислений с плавающей точкой.

В работе предлагается, используя готовые программы, выполнить исследования машинной арифметики и точности вычислений на ПЭВМ. Программы для удобства пользователя объединены в одном исполняемом модуле lab1.exe, запускаемом на выполнение просто указанием имени и нажатием клавиши "Enter". В модуле приведены аннотация к лабораторной работе, развернутые теоретические пояснения к каждому заданию (программе), собственно задания и рекомендации по вариации значений параметров при проведении расчетов. Программы в процессе работы выдают подсказки по действиям пользователя и запросы на ввод значений параметров. Это позволяет студентам осуществлять многосторонние исследования, одновременно усваивая теоретические положения вводной части курса дисциплины "Вычислительная математика".

Работа в соответствии с числом используемых расчетных программ содержит следующие четыре задания:

1) исследуйте распределение нормализованных чисел с плавающей точкой на вещественной оси для различных значений параметров b, t, m (из-за ограниченности ресурсов ПЭВМ не рекомендуется задавать большие значения параметров: b=2, t≤7, m≤4);
2) вычислите значения величины машинного эпсилон ε(c) для различных значений константы c, меняющихся от 0 до 2^{15}, постройте график этой зависимости и объясните полученные результаты;
3) исследуйте абсолютные и относительные ошибки округления при вычислениях с плавающей точкой сумм чисел (N чисел вида 1/N) при различных значениях шага суммирования;
4) исследуйте проявления ошибок округления, возникающих при вычислении показательной функции e^x для чисел с плавающей точкой для двух вариантов алгоритма вычислений, также скорость сходимости обоих вариантов.

В последнем задании при первом варианте алгоритма значение функции e^x вычисляется как сумма сходящегося бесконечного ряда, а при втором число x разлагается на целую и дробную части [9].

Студентам требуется выполнить все перечисленные задания, при этом индивидуальных вариантов не предусматривается, - значения необходимых параметров выбираются по усмотрению каждого исследователя. Порядок выполнения заданий может быть произвольным, однако рекомендуется придерживаться приведенной в их перечислении последовательности 1),2),3),4).

2. ИЗУЧЕНИЕ ПОНЯТИЯ ОБУСЛОВЛЕННОСТИ ВЫЧИСЛИТЕЛЬНОЙ ЗАДАЧИ

(Лабораторная работа №2)

Под обусловленностью вычислительной задачи понимают чувствительность ее решения к малым погрешностям входных данных.

Задачу называют хорошо обусловленной, если малым погрешностям входных данных отвечают малые погрешности решения, и плохо обусловленной, если возможны сильные изме-

нения решения. Количественной мерой степени обусловленности вычислительной задачи является число обусловленности, которое можно интерпретировать как коэффициент возможного возрастания погрешностей в решении по отношению к вызвавшим их погрешностям входных данных. Пусть между абсолютными погрешностями входных данных X и решения У установлено неравенство

$$\Delta(y^*) \le \nu_\Delta \Delta(x^*),$$

где x^* и y^* - приближенные входные данные и приближенное решение.

Тогда величина ν_Δ называется абсолютным числом обусловленности. Если же установлено неравенство

$$\delta(y^*) \le \nu_\delta \delta(x^*)$$

между относительными ошибками данных и решения, то величину ν_δ называют относительным числом обусловленности. Для плохо обусловленной задачи $\nu >> 1$. Грубо говоря, если $\nu = 10^N$, где ν - относительное число обусловленности, то порядок N показывает число верных цифр, которое может быть утеряно в результате по сравнению с числом верных цифр входных данных.

Ответ на вопрос о том, при каком значении ν задачу следует признать плохо обусловленной, зависит, с одной стороны, от предъявляемых требований к точности решения и, с другой, - от уровня обеспечиваемой точности исходных данных. Например, если требуется найти решение с точностью 0.1%, а входная информация задается с точностью 0.02%, то уже значение $\nu = 10$ сигнализирует о плохой обусловленности. Однако, при тех же требованиях к точности результата, гарантия, что исходные данные задаются с точностью не ниже 0.0001%, означает, что при $\nu = 10^3$ задача хорошо обусловлена.

Если рассматривать задачу вычисления корня уравнения Y=f(X), то роль числа обусловленности будет играть величина

$$\nu_\Delta = \frac{1}{\left| f'(x^0) \right|}$$

где x^0 - корень уравнения.

В работе предлагается, используя программы - функции BISECT и Round из файла metods.cpp (файл заголовков metods.h, директория LIBR1), исследовать обусловленность задачи нахождения корня уравнения $f(x) = 0$ для линейной функции $f(x) = c(x - d)$. Значения функции $f(x)$ следует вычислить приближенно с точностью Delta, варьируемой в пределах от 0.1 до 0.000001.

Порядок выполнения работы должен быть следующим:

1) Графически или аналитически отделить корень уравнения $f(x)=0$, т.е. найти отрезки [Left, Right], на которых функция $f(x)$ удовлетворяет условиям применимости метода бисекции (см. Подразделы 3.1 и 3.2).

2) Составить подпрограмму вычисления функции $f(x)=c(x-d)$ для параметров c и d, вводимых с клавиатуры. Предусмотреть округление вычисленных значений функции $f(x)$ с использованием программы-функции Round с точностью Delta, также вводимой с клавиатуры.

3) Составить головную программу, вычисляющую корень уравнения с заданной точностью Eps и содержащую обращение к подпрограмме f(x), программам-функциям BISECT, Round и представление результатов.

4) Провести вычисления по программе, варьируя значения параметров c (тангенс угла наклона прямой), Eps (точность вычисления корня) и Delta (точность задания исходных данных).

5) Проанализировать полученные результаты и обосновать выбор точности Eps вычисления корня. Сопоставить полученные теоретические результаты с экспериментальными данными.

Значение параметра d выбирается каждым студентом самостоятельно и согласовывается с преподавателем.

Текст программы для исследования обусловленности задачи нахождения корня уравнения $f(x)=0$ представлен ниже.

```
/*****************************************************************/
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <methods.h>
#include <conio.h>
double delta,c,d;
void main()
{
  int k;
  long int s;
```

```
 float a1,b1,c1,d1,eps1,delta1;
 double a,b,eps,x;
 double F(double);
 printf("Введите eps:");
 scanf("%f",&eps1);
 eps = eps1;
 printf("Введите c:");
 scanf("%f",&c1);
 c = c1;
 printf("Введите d:");
 scanf("%f",&d1);
 d = d1;
 printf("Введите a:");
 scanf("%f",&a1);
 a = a1;
 printf("Введите b:");
 scanf("%f",&b1);
 b = b1;
 printf("Введите delta:");
 scanf("%f",&delta1);
 delta = delta1;
 x = BISECT(a,b,eps,k);
 printf("x=%f    k=%d\n",x,k);
}
double F(double x)
{
 extern double c,d,delta;
 double s;
 long int S;
 s = c*(x - d);
 if( s/delta < 0 )
 S = s/delta - .5;
```

```
else
S = s/delta + .5;
s = S*delta;
s = Round( s,delta );
return(s);
}
/*********************************************************/
```

3. РЕШЕНИЕ НЕЛИНЕЙНЫХ УРАВНЕНИЙ

Задача нахождения корней нелинейных уравнений вида $f(x)=0$ (где $f(x)$ - некоторая непрерывная функция) встречается в различных областях инженерной и научной деятельности [1-10]. Нелинейные уравнения делятся на два класса - алгебраические и трансцендентные. Алгебраическими называются уравнения, содержащие только алгебраические функции (целые, рациональные, иррациональные). Уравнения, которые содержат другие функции (тригонометрические, показательные, логарифмические и т.п.), называются трансцендентными.

Методы решения нелинейных уравнений подразделяются на прямые и итерационные. Прямые методы позволяют записать корни в виде некоторого конечного соотношения. Такие методы для решения ряда трансцендентных, а также простейших алгебраических уравнений известны из школьного курса алгебры. Однако встречающиеся на практике уравнения не удается решить столь простыми методами. Для их решения применяются итерационные методы, при которых алгоритм нахождения корня уравнения в общем случае включает два этапа:

- отыскания приближенного значения корня или содержащего его отрезка;
- уточнения приближенного значения до некоторой заданной степени точности.

3.1. Общие сведения

Пусть задана непрерывная функция f вещественного аргумента x и требуется численным методом решить уравнение $f(x)=0$, т.е. найти приближение x^* к вещественному корню этого уравнения. Если уравнение имеет несколько вещественных корней, то сначала производят

их отделение (изоляцию), а затем уточняют положение отдельного корня. Считается, что отделение корня произведено, если выделен такой интервал $[a_0, b_0]$ области определения функции f, на концах которого значения функции $f(a_0)$ и $f(b_0)$ имеют разные знаки и внутри которого имеется ровно один корень уравнения $f(x) = 0$. Для уточнения метода используют итерационные методы, такие как метод бисекции (половинного деления), метод хорд (секущих или ложного положения), метод Ньютона (касательных), метод итераций (последовательных приближений). В указанных методах вычисляются либо последовательность значений границ сужающихся интервалов $a_0, b_0, a_1, b_1, \ldots, a_n, b_n, \ldots$, содержащих корень, либо последовательность приближений к корню $x_0, x_1, x_2, \ldots, x_n, \ldots$[2,7,8,11].

В первом случае итерационный процесс заканчивается, как только длина текущего интервала становится достаточно малой (например, $|b_n-a_n|<\varepsilon$). Во втором случае условием остановки вычислений является малость очередного приращения $h_n=x_n-x_{n-1}$, $|h_n|<\varepsilon$. В обоих случаях параметр ε определяет момент остановки вычислений. Иногда в качестве условия остановки используют условие $|f(x_n^*)|<\varepsilon$, где x_n^* - текущее приближение к корню, например, $x_n^*=1/2(a_n + b_n)$ в методе бисекции. Выполнение этого условия свидетельствует о малости значения функции в точке x_n^*, т.е. позволяет считать, что $f(x_n^*)\approx 0$.

Для каждого итерационного метода можно указать некоторые условия сходимости. Однако не всегда легко проверить или гарантировать выполнение этих условий. Кроме того необходимо учесть особенности машинных вычислений при реализации итерационных методов. На практике эти затруднения обходят, вводя ограничение n_{max} на число итераций. Такое ограничение предохраняет от "зацикливания" метода, а также позволяет выявить практическое отсутствие сходимости вычислительного процесса.

Целью лабораторных работ, приводимых в данном разделе, является изучение перечисленных выше четырех итерационных методов приближенного решения нелинейных уравнений, при этом каждая работа посвящена одному из них. Для выполнения работ предлагается использовать набор программ - функций, реализующих конкретные численные методы, а также программу - функцию Round, позволяющую моделировать ошибки в исходных данных. Указанные программы (язык С) размещаются в директории LIBR1.

3.2. Метод бисекции

(Лабораторная работа №3)

Если найден отрезок [a,b], такой, что $f(a)\,f(b)<0$, существует точка c, в которой значение функции равно нулю, т.е. $f(c)=0$, $c\in(a,b)$. Метод бисекции состоит в построении последовательности вложенных друг в друга отрезков, на концах которых функция имеет разные знаки. Каждый последующий отрезок получается делением пополам предыдущего. Процесс построения последовательности отрезков позволяет найти нуль функции $f(x)$ (корень уравнения $f(x)=0$ с любой заданной точностью.

Рассмотрим один шаг итерационного процесса. Пусть на (n-1)-м шаге найден отрезок $[a_{n-1}, b_{n-1}]\subset[a, b]$, такой, что $f(a_{n-1})\,f(b_{n-1})<0$. Разделим его пополам точкой $\xi=(a_{n-1}+b_{n-1})/2$ и вычислим $f(\xi)$. Если $f(\xi)=0$, то $\xi=(a_{n-1}+b_{n-1})/2$- корень уравнения. Если $f(\xi)\neq0$, то из двух половин отрезка выбирается та, на концах которой функция имеет противоположные знаки, поскольку искомый корень лежит на этой половине, т.е.

$a_n=a_{n-1}$, $b_n=\xi$, если $f(\xi)\,f(a_{n-1}) < 0$;

$a_n=\xi$, $b_n= b_{n-1}$, если $f(\xi)\,f(a_{n-1}) > 0$.

Если требуется найти корень с точностью ε, то деление пополам продолжается до тех пор, пока длина отрезка не станет меньше 2ε. Тогда координата середины отрезка есть значение корня с требуемой точностью ε.

Метод бисекции является простым и надежным методом поиска простого корня уравнения $f(x)=0$ (простым называется корень x=c дифференцируемой функции $f(x)$, если $f(c)$ и $f'(c)\neq0$). Этот метод сходится для любых непрерывных функций $f(x)$, в том числе недифференцируемых. Скорость его сходимости невысока. Для достижения точности ε необходимо совершить $N\approx\log_2(b-a)/\varepsilon$ итераций. Это означает, что для получения каждых трех верных десятичных знаков необходимо совершить около 10 итераций.

В лабораторной работе №3 предлагается, используя программы - функции BISECT и Round из файла methods.cpp (файл заголовков metods.h, директория LIBR1), найти корень уравнения $f(x)=0$ методом бисекции с заданной точностью Eps, исследовать зависимость числа

итераций от точности Eps при изменении Eps от 0.1 до 0.000001, исследовать обусловленность метода (чувствительность к ошибкам в исходных данных).

Выполнение работы осуществляется по индивидуальным вариантам заданий (нелинейных уравнений), приведенным в подразделе 3.6. Номер варианта для каждого студента определяется преподавателем.

Порядок выполнения работы должен быть следующим:

1) Графически или аналитически отделить корень уравнения $f(x)=0$ (т.е. найти отрезки [Left, Right], на которых функция $f(x)$ удовлетворяет условиям теоремы Коши).
2) Составить подпрограмму вычисления функции $f(x)$.
3) Составить головную программу, содержащую обращение к подпрограмме f(x), BISECT, Round и индикацию результатов.
4) Провести вычисления по программе. Построить график зависимости числа итераций от Eps.
5) Исследовать чувствительность метода к ошибкам в исходных данных. Ошибки в исходных данных моделировать с использованием программы Round, округляющей значения функции с заданной точностью Delta.

Текст программы-функции BISECT, предназначенной для решения уравнения $f(x)=0$ методом бисекции, представлен в подразделе 3.7.

3.3. Метод хорд

(Лабораторная работа №4)

Пусть найден отрезок [a, b], на котором функция $f(x)$ меняет знак. Для определенности положим f (a)>0, f (b)<0. В методе хорд процесс итераций состоит в том, что в качестве приближений к корню уравнения $f(x)=0$ принимаются значения c_0, c_1, . . . точек пересечения хорды с осью абсцисс, как это показано на рис.1.

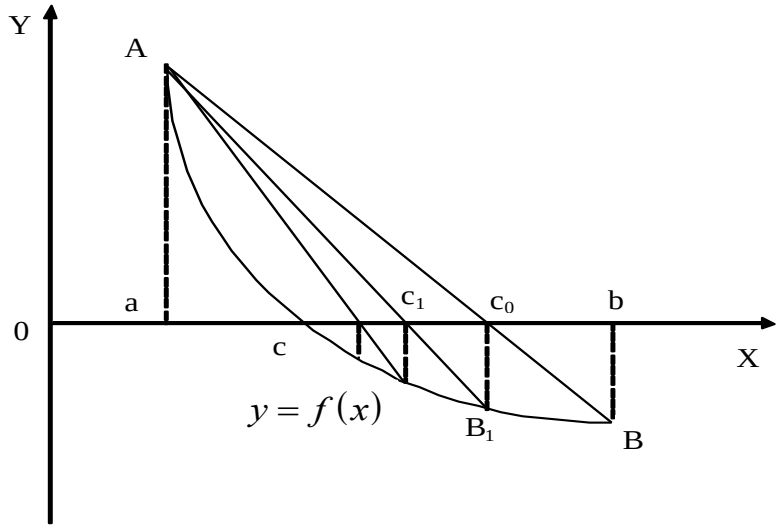

Рис. 1

Сначала находится уравнение хорды AB:

$$\frac{y - f(a)}{f(b) - f(a)} = \frac{x - a}{b - a}.$$

Для точки пересечения ее с осью абсцисс ($x=c_0$, $y=0$) получается уравнение

$$c_0 = a - \frac{b - a}{f(b) - f(a)} f(a).$$

Далее сравниваются знаки величин f (a) и f (c_0) и для рассматриваемого случая оказывается, что корень находится в интервале (a, c_0), так как f (a) f (c_0)<0. Отрезок [c_0,b] отбрасывается. Следующая итерации состоит в определении нового приближения c_1 как точки пересечения хорды AB_1 с осью абсцисс и т.д. Итерационный процесс продолжается до тех пор, пока значение f (c_n) не станет по модулю меньше заданного числа ε (см. подраздел 3.1).

Алгоритмы методов бисекции и хорд похожи, однако метод хорд в ряде случаев дает более быструю сходимость итерационного процесса, причем успех его применения, как и метода бисекции, гарантирован.

В лабораторной работе №4 предлагается, используя программы - функции HORDA и Round из файла methods.cpp (файл заголовков metods.h, директория LIBR1), найти корень уравнения $f(x) = 0$с заданной точностью Eps методом хорд, исследовать скорость сходимости и обусловленности метода.

Для данной работы, как и для лабораторной работы №3 задаются индивидуальные варианты нелинейных уравнений (см. подраздел 3.6).

Порядок выполнения лабораторной работы №4:

1) Графически или аналитически отделить корень уравнения $f(x)=0$ (т.е. найти отрезки [Left, Right], на которых функция $f(x)$ удовлетворяет условиям применимости метода).
2) Составить подпрограмму - функцию вычисления функции $f(x)$, предусмотрев округление значений функции с заданной точностью Delta с использованием программы Round.
3) Составить головную программу, вычисляющую корень уравнения $f(x)=0$ и содержащую обращение к подпрограмме f(x), HORDA, Round и индикацию результатов.
4) Провести вычисления по программе. Теоретически и экспериментально исследовать скорость сходимости и обусловленность метода.

В подразделе 3.7 приводится текст программы - функции HORDA, предназначенной для решения уравнения $f(x)=0$ методом хорд.

3.4. Метод Ньютона

(Лабораторная работа № 5)

В случае, когда известно хорошее начальное приближение решения уравнения $f(x)=0$, эффективным методом повышения точности является метод Ньютона. Он состоит в построении итерационной последовательности $x_{n+1}=x_n-f(x_n)/f'(x_n)$, сходящейся к корню уравнения $f(x)=0$. Достаточные условия сходимости метода формулируются теоремой, приведенной в [1,7].

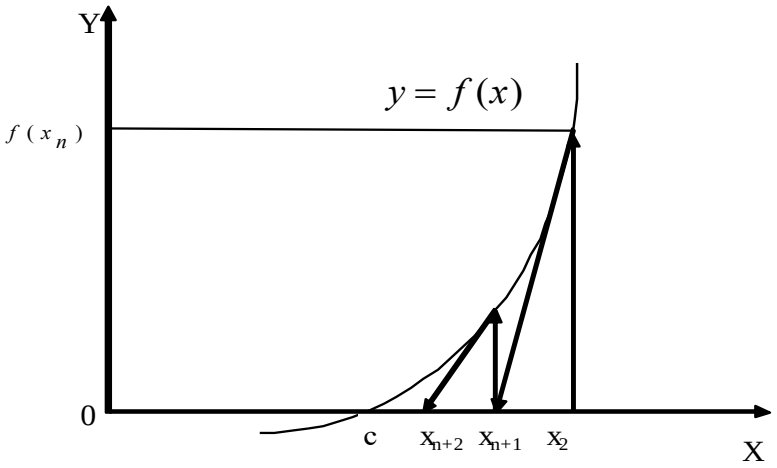

Рис.2

Метод Ньютона допускает простую геометрическую интерпретацию (рис. 2). Если через точку с координатами $(x_n; f(x_n))$ провести касательную, то абсцисса точки пересечения этой касательной с осью Ox будет очередным приближением x_{n+1} корня уравнения $f(x) = 0$.

Для оценки погрешности n-го приближения корня предлагается пользоваться неравенством

$$|c - x_n| \le \frac{M_2}{2m_1}|x_n - x_{n-1}|^2,$$

где M_2-наибольшее значение модуля второй производной $|f''(x)|$ на отрезке [a,b]; m_1-наименьшее значение модуля первой производной $|f'(x)|$ на отрезке [a,b]. Таким образом, если $|x_n - x_{n-1}| < \varepsilon,$ то $|c - x_n| \le \frac{M_2 \varepsilon^2}{2m_1}$. Это означает, что при хорошем начальном приближении корня после каждой итерации число верных десятичных знаков в очередном приближении удваивается, т.е. процесс сходится очень быстро (имеет место квадратическая сходимость). Из указанного следует, что при необходимости нахождения корня с точностью ε итерационный процесс можно прекращать, когда

$$|x_n - x_{n-1}| < \varepsilon_0 = \sqrt{2m_1 \varepsilon / M_2}. \quad (3.1)$$

Рассмотрим один шаг итераций. Если на (n-1)-м шаге очередное приближение x_{n-1} не удовлетворяет условию окончания процесса, то вычисляются величины $f(x_{n-1}), f'(x_{n-1})$ и следующие приближение корня $x_n = x_{n-1} - f(x_{n-1}) / f'(x_{n-1}).$ При выполнении условия (3.1) величина x_n принимается за приближенное значение корня с, вычисленное с точностью ε.

В лабораторной работе № 5 предлагается, используя программы-функции NEWTON и ROUND из файла methods.cpp (файл заголовков methods.h, директория LIBR1), найти корень уравнения $f(x)=0$ с заданной точностью Eps методом Ньютона, исследовать скорость сходимости и обусловленность метода.

Для данной работы вид функции $f(x)$ задается индивидуально каждому студенту преподавателем из числа вариантов, приведенных в подразделе 3.6.

Порядок выполнения лабораторной работы №5.

1) Графически или аналитически отделить корень уравнения $f(x)=0$ (т.е. найти отрезки [Left, Right], на котором функция $f(x)$ удовлетворяет условиям сходимости метода Ньютона).
2) Составить подпрограммы - функции вычисления $f(x)$, $f'(x)$, предусмотрев округление их значений с заданной точностью Delta.
3) Составить головную программу, вычисляющую корень уравнения $f(x)=0$ и содержащую обращение к подпрограммам , $f(x)$, f'(x), Round, NEWTON и индикацию результатов.
4) Выбрать начальное приближение корня x_0 из [Left, Right] так, чтобы $f(x)\ f''(x)>0$.!!!!!!!
5) Провести вычисления по программе. Исследовать скорость сходимости метода и чувствительность метода к ошибкам в исходных данных.

Для приближенного вычисления корней уравнения $f(x)=0$ методом Ньютона предназначена программа - функция NEWTON, текст которой представлен в подразделе 3.7.

3.5. Метод простых итераций

(Лабораторная работа №6)

Метод простых итераций решения уравнения $f(x)=0$ состоит в замене исходного уравнения эквивалентным ему уравнением $x=\varphi(x)$ и построении последовательности $x_{n+1}=\varphi(x_n)$, сходящейся при $n\to\infty$ к точному решению. Достаточные условия сходимости метода простых итераций формулируются теоремой, приведенной [1,2,7].

Рассмотрим один шаг итерационного процесса. Исходя из найденного на предыдущем шаге значения x_{n-1}, вычисляется $y= \varphi(x_{n-1})$. Если $|y - x_{n-1}| > \varepsilon$, то полагается $x_n=y$ и выполняется очередная итерация. Если же $|y - x_{n-1}| < \varepsilon$, то вычисления заканчиваются и за приближенное значение корня принимается величина $x_n=y$. Погрешность результата вычислений зависит от знака производной $\varphi'(x)$: при $\varphi'(x) > 0$ погрешность определения корня составляет $q\varepsilon/1-q$, а при $\varphi'(x) < 0$ погрешность не превышает ε. Здесь q- число, такое, что $|\varphi'(x)| \le q < 1$ на отрезке [a,b]. Существование числа q является условием сходимости метода в соответствии с отмеченной выше теоремой.

Для применения метода простых итераций определяющее значение имеет выбор функции $\varphi(x)$ в уравнении $x = \varphi(x)$, эквивалентном исходному. Функцию $\varphi(x)$ необходимо подбирать так, чтобы $|\varphi'(x)| \le q < 1$. Это обусловливается тем, что если $\varphi'(x) < 0$ на отрезке [a,b], то последовательные приближения $x_n=\varphi(x_{n-1})$ будут колебаться около корня с, если же $\varphi'(x) > 0$, то последовательные приближения будут сходиться к корню с монотонно. Следует также помнить, что скорость сходимости последовательности $\{x_n\}$ к корню с функции $f(x)$ тем выше, чем выше число q.

В лабораторной работе № 6 предлагается, используя программы-функции ITER и Round из файла methods.cpp (файл заголовков methods.h, директория LIBR1), найти корень уравнения $f(x) = 0$ с заданной точностью Eps методом итераций, исследовать скорость сходимости и обусловленность метода.

Для данной работы вид функции $f(x)$ задается индивидуально каждому студенту преподавателем из числа вариантов, приведенных в подразделе 3.6.

Порядок выполнения лабораторной работы №6 должен быть следующим.

1) Графически или аналитически отделить корень уравнения $f(x) = 0$.
2) Преобразовать уравнение $f(x) = 0$ к виду $x = \varphi(x)$ так, чтобы в некоторой окрестности [Left, Right] корня производная $\varphi'(x)$ удовлетворяла условию $|\varphi'(x)| \le q < 1$. При этом следует иметь в виду, что чем меньше величина q, тем быстрее последовательные приближения сходятся к корню.
3) Выбрать начальное приближение, лежащее на [Left, Right].
4) Составить подпрограмму для вычисления значений $\varphi(x), \varphi'(x)$, предусмотрев округление вычисленных значений с точностью Delta.

5) Составить головную программу, вычисляющую корень уравнения и содержащую обращение к программам $\varphi(x)$, $\varphi'(x)$ и ITER и индикацию результатов.

6) Провести вычисления по программе. Исследовать скорость сходимости и обусловленность метода.

Текст программы - функции ITER, позволяющей вычислять корни уравнения x=$\varphi(x)$ для любой функции, которая удовлетворяет достаточным условиям сходимости метода, приводится ниже.

3.6. Курсовая работа по дисциплине и варианты заданий

Лабораторные работы, описанные в настоящем разделе, целесообразно объединять для выполнения курсовой работы, нацеленной на сравнительную оценку различных методов приближенного решения нелинейных уравнений. Типовое задание на курсовую работу формулируется следующим образом.

Задание на курсовую работу по дисциплине "Вычислительная математика".

Используя программы - функции BISECT, NEWTON, HORDA, ITER, Round из файла methods.cpp (файл заголовков methods.h, директория LIBR1), найти корень уравнения $f(x) = 0$с заданной точностью методом бисекции, Ньютона, хорд и итераций соответственно.

Исследуйте обусловленность методов и зависимость числа итераций от точности результата Eps при изменении Eps от 0.0 до 0.000001.

Порядок выполнения курсовой работы

Графически или аналитически отделить корень уравнения $f(x) = 0$ (т.е. найти отрезки [Left, Right], на которых функция $f(x)$ удовлетворяет условиям применимости методов).

Составить подпрограмму- функцию вычисления функции $f(x)$ и ее производной $f'(x)$ (при необходимости), предусмотрев округление их значений с заданной точностью Delta с использованием библиотечной функции Round.

Составить головную программу, содержащую ввод исходных данных, обращение к подпрограммам BISECT, NEWTON, HORDA, ITER вывод результатов.

Выполнить вычисления по программе. Построить графики зависимости числа итераций, необходимых для достижения заданной точности Eps, от величины Eps, а также достижимой точности результатов от точности Delta задания функции $f(x)$.

Теоретически и экспериментально сравнить методы бисекции, Ньютона, хорд и итераций по скорости сходимости и степени обусловленности.

Результаты оформить в виде отчета, содержащего постановку задачи, тексты разработанных программ, результаты теоретического и экспериментального анализа в виде таблиц и графиков, выводы.

Вид функции $f(x)$ определяется вариантом задания. Эти же варианты могут использоваться при выполнении лабораторных работ №№3-6 по отдельности.

№ п/п	$f(x)$	№ п/п	$f(x)$
1	$f(x)=x^2-5.0*\sin(x);$	19	$f(x)=\ln(x)*\ln(x)-1/x$
2	$f(x)=\arccos(x^2)-x;$	20	$f(x)=e^{-x}-x^3;$
3	$f(x)=\ln(x)-1/(1+x^2);$	21	$f(x)=arctg(x)-1/x;$
4	$f(x)=\ln(\ln(x))-e^{(-x^2)};$	22	$f(x)=\ln((1+x)/(1-x))-\cos(x^2);$
5	$f(x)=arctg(1/x)-x^2;$	23	$f(x)=\sinh(x)-x+1;$
6	$f(x)=tg(x)-1/x;$	24	$f(x)=arctg(x)-\ln(x);$
7	$f(x)=x^4-13x^2+36-1/x;$	25	$f(x)=\cosh(x)-4x^3/(1+x^2);$
8	$f(x)=2x^2-x^4-1-\ln(x);$	26	$f(x)=1/(3+2\cos(x))-x^3;$
9	$f(x)=x^2-x^3-1/(4+x^2);$	27	$f(x)=\ln((1+x)/(1-x))-1/x;$
10	$f(x)=x^3-3x-2e^{(-x)};$	28	$f(x)=e^x-3-\cos(x);$
11	$f(x)=2^{x^2}-1/x;$	29	$f(x)=e^{-x}-arctg(x);$
12	$f(x)=(1+\cos(x))/(3-\sin(x))-x;$	30	$f(x)=2^{e^x}+x;$
13	$f(x)=\sin(x^2)-6x+1;$	31	$f(x)=\arccos(x^2)-x^3;$
14	$f(x)=\cos(x^2)-10x;$	32	$f(x)=tg(x)-2/x;$
15	$f(x)=\arccos(1-x^2)/(1+x^2)-x;$	33	$f(x)=2tg(x)-1/x;$
16	$f(x)=\arcsin(2x/(1+x^2))-e^{(-x^2)};$	34	$f(x)=x-\ln(x-1+\sqrt{(x-1)^2+1});$
17	$f(x)=e^x-\arccos(\sqrt{x});$	35	$f(x)=e^{-x^2}-\sqrt{x};$
18	$f(x)=e^{1/x^2}-\ln(x);$	36	$f(x)=\lg\ln(x)-1/(1+x^2);$

3.7. Программы для решения нелинейных уравнений

Текст заголовочного файла methods.h.

/***/

```
extern double F(double);
/*****************************************************************/
/*          Функция F (X) , задаваемая пользователем             */
/*****************************************************************/

#ifdef __NEWTON
extern double F1(double);
/*****************************************************************/
/*     Ïðîèçâîäíàÿ ôóíêöèè F (X) , çàäàâàåìàÿ ïîëüçîâàòåëåì       */
/*****************************************************************/
#endif

double Round (double X,double Delta);
/*****************************************************************/
/*   Ôóíêöèÿ Round (X,Delta) , ïðåäíàçíà÷åíà äëÿ îêðóãëåíèÿ       */
/*                X ñ òî÷íîñòüþ Delta                              */
/*****************************************************************/

double BISECT(double Left,double Right,double Eps,int &N);
/*****************************************************************/
/*    Ôóíêöèÿ BISECT ïðåäíàçíà÷åíà äëÿ ðåøåíèÿ óðàâíåíèÿ F(X)=0   */
/*    ìåòîäîì äåëåíèÿ îòðåçêà ïîïîëàì. Èñïîëüçîâàíû îãðíà÷åíèÿ:  */
/*      Left - ëåâûé êîíåö ïðîìåæóòêà                              */
/*      Right - ïðàâûé êîíåö ïðîìåæóòêà                            */
/*      Eps - ïîãðåøíîñòü âû÷èñëåíèÿ êîðíÿ óðàâíåíèÿ;              */
/*      N - ÷èñëî èòåðàöèé                                          */
/*****************************************************************/

double ITER(double X0,double Eps,int &N);
/*****************************************************************/
/*    Ôóíêöèÿ ITER ïðåäíàçíà÷åíà äëÿ ðåøåíèÿ óðàâíåíèÿ F(X)=X     */
/*      ìåòîäîì ïðîñòîé èòåðàöèè. Èñïîëüçîâàíû îãðíà÷åíèÿ:       */
```

```
/*      X0 - íà÷àëüíîå ïðèáëèæåíèå êîðíÿ                    */
/*      Eps - ïîãðåøíîñòü âû÷èñëåíèÿ êîðíÿ óðàâíåíèÿ;       */
/*      N - ÷èñëî èòåðàöèé                                   */
/*********************************************************************/

double HORDA(double Left,double Right,double Eps,int &N);
/*********************************************************************/
/*    Ôóíêöèÿ HORDA ïðåäíàçíà÷åíà äëÿ ðåøåíèÿ óðàâíåíèÿ F(x)=0    */
/*      ìåòîäîì õîðä. Èñïîëüçîâàíû îãðàíè÷åíèÿ:              */
/*      Left - ëåâûé êîíåö ïðîìåæóòêà                       */
/*      Right - ïðàâûé êîíåö ïðîìåæóòêà                     */
/*      Eps - ïîãðåøíîñòü âû÷èñëåíèÿ êîðíÿ óðàâíåíèÿ;       */
/*      N - ÷èñëî èòåðàöèé                                   */
/*********************************************************************/

double NEWTON (double X,double Eps,int &N);
/*********************************************************************/
/*  Ôóíêöèÿ NEWTON ïðåäíàçíà÷åíà äëÿ ðåøåíèÿ óðàâíåíèÿ F(X)=0     */
/*       ìåòîäîì êàñàòåëüíûõ. Èñïîëüçîâàíû îãðàíè÷åíèÿ:       */
/*      X - íà÷àëüíîå ïðèáëèæåíèå êîðíÿ                     */
/*      Eps - ïîãðåøíîñòü âû÷èñëåíèÿ êîðíÿ óðàâíåíèÿ;       */
/*      N - ÷èñëî èòåðàöèé                                   */
/*********************************************************************/

#include "methods.cpp"
/*************************************************************************/
```

Текст программного модуля methods.cpp.

```
/*****************************************/
#include <stdio.h>
#include <math.h>
```

```
#include <stdlib.h>
extern double F(double);
double BISECT(double Left,double Right,double Eps,int &N)
{
  double E = fabs(Eps)*2.0;
  double FLeft = F(Left);
  double FRight = F(Right);
  double X = (Left+Right)/2.0;
  double Y;
  if (FLeft*FRight>0.0) {puts("Неверное задание интервала\n");exit(1);}
  if (Eps<=0.0) {puts("Неверное задание точности\n");exit(1);}
  N=0;
  if (FLeft==0.0)  return Left;
  if (FRight==0.0) return Right;
  while ((Right-Left)>=E)
  {
          X = 0.5*(Right + Left);     /* вычисление середины отрезка       */
          Y = F(X);
          if (Y == 0.0) return (X);
       if (Y*FLeft < 0.0)
         Right=X;
       else
        { Left=X; FLeft=Y; }
       N++;
  };
  return(X);
}
double Round (double X,double Delta)
{
 if (Delta<=1E-9) {puts("Неверное задание точности округления\n");exit(1);}
 if (X>0.0) return (Delta*(long((X/Delta)+0.5)));
    else    return (Delta*(long((X/Delta)-0.5)));
```

```
}
double ITER(double X0,double Eps,int &N)
{
 if (Eps<=0.0) {puts("Неверное задание точности\n");exit (1);}
 double X1=F(X0);
 double X2=F(X1);
 N = 2;
 while( (X1 - X2)*(X1 - X2) > fabs((2*X1-X0-X2)*Eps) )
 {
          X0 = X1;
          X1 = X2;
          X2 = F(X1);
      N++;
 }
  return(X2);
}
#ifdef __NEWTON
double NEWTON (double X,double Eps,int &N)
{
 extern double F1 (double);
 double Y,Y1,DX;
 N=0;
 do
 {
   Y  = F(X);
   if (Y==0.0) return (X);
   Y1 = F1(X);
   if (Y1==0.0) {puts("Производная обратилась в ноль\n");exit(1);}
   DX=Y/Y1; X=X-DX; N++;
 }
 while (fabs(DX)>Eps);
 return (X);
```

```
}
#endif
double HORDA(double Left,double Right,double Eps,int &N)
{
 double FLeft = F(Left);
 double FRight = F(Right);
 double X,Y;
 if (FLeft*FRight>0.0) {puts("Неверное задание интервала\n");exit(1);}
 if (Eps<=0.0) {puts("Неверное задание точности\n");exit(1);}
 N=0;
 if (FLeft==0.0)  return Left;
 if (FRight==0.0) return Right;
 do
 {
        X = Left-(Right-Left)*FLeft/(FRight-FLeft);
        Y = F(X);
        if (Y == 0.0) return (X);
     if (Y*FLeft < 0.0)
      { Right=X; FRight=Y; }
     else
      { Left=X; FLeft=Y; }
     N++;
 }
 while ( fabs(Y) >= Eps );
 return(X);
}
/*********************************************************/
```

4. ЧИСЛЕННОЕ ИНТЕГРИРОВАНИЕ

4.1. Составные формулы прямоугольников, трапеций, Симпсона.

(Лабораторная работа №6)

Повышения точности численного интегрирования добиваются путем применения составных формул. Для этого при нахождении определенного интеграла отрезок $[a,b]$ разбивают на четное $n=2m$ число отрезков длины $h=(b-a)/n$ и на каждом из отрезков длины $2h$ применяют соответствующую формулу. Таким образом получают составные формулы прямоугольников, трапеций и Симпсона.

На сетке $x_i=a+ih$, $y=f(x_i)$, $i=0,1,2,\ldots,2m$, составные формулы имеют следующий вид:

формула прямоугольников

$$\int_a^b f(x)dx = h\sum_{i=0}^{n-1}\left(f\left(x_i+\frac{h}{2}\right)\right)+R_1;$$

формула трапеций

$$\int_a^b f(x)dx = \frac{h}{2}\sum_{i=0}^{n-1}\left(f(x_i)+f(x_{i+1})\right)+R_2;$$

формула Симпсона

$$\int_a^b f(x)dx = \frac{h}{3}\sum_{i=0}^{m-1}\left(f(x_{2i})+4f(x_{2i+1})+f(x_{2i+2})\right)+R_3,$$

где R_1,R_2,R_3 - остаточные члены. При $n\to\infty$ приближенные значения интегралов для всех трех формул (в предположении отсутствия погрешностей округления) стремятся к точному значению интеграла [1,7,8].

Для практической оценки погрешности квадратурной можно использовать правило Рунге. Для этого проводят вычисления на сетках с шагом h и $h/2$, получают приближенные значения интеграла I_h и $I_{h/2}$ и за окончательные значения интеграла принимают величины:

$I_{h/2}+\left|I_{h/2}-I_h\right|/3$ - для формулы прямоугольников;

$I_{h/2}-\left|I_{h/2}-I_h\right|/3$ - для формулы трапеций;

$I_{h/2}-\left|I_{h/2}-I_h\right|/15$ - для формулы Симпсона.

За погрешность приближенного значения интеграла для формул прямоугольников и трапеций тогда принимают величину $\left|I_{h/2} - I_h\right| / 3$, а для формулы Симпсона $\left|I_{h/} - I_h\right| / 15$.

В лабораторной работе №6 требуется, используя квадратурные формулы прямоугольников, трапеций и Симпсона, вычислить значения заданного интеграла и, применив правило Рунге, найти наименьшее значение n (наибольшее значение шага h), при котором каждая из указанных формул дает приближенное значение интеграла с погрешностью ε, не превышающей заданную.

Порядок выполнения лабораторной работы №6.

1) Составить программы-функции для вычисления интегралов по формулам прямоугольников, трапеций и Симпсона.
2) Составить программу-функцию для вычисления подынтегральной функции.
3) Составить головную программу, содержащую оценку по Рунге погрешности каждой из перечисленных выше квадратурных формул, удваивающих n до тех пор, пока погрешность не станет меньше ε, и осуществляющих печать результатов: значения интеграла и значения n для каждой формулы.
4) Провести вычисления по программе, добиваясь, чтобы результат удовлетворял требуемой точности.
5) Результаты работы оформить в виде краткого отчета, содержащего сравнительную оценку применяемых для вычисления формул.

Варианты заданий приведены в таблице 4.1 (ε = 0.01; 0.001; 0.0001).

4.2. Формула Гаусса. (Лабораторная работа №7)

В квадратурной формуле Гаусса

$$\int_{-1}^{1} f(x)dx \approx \sum_{i=0}^{n} A_i f\left(x_i\right)$$

узлы $x_1, x_2, \ldots, x_n$ и коэффициенты $A_1, A_2, \ldots, A_n$ подобраны так, чтобы формула была точна для всех многочленов степени $2n-1$. Для приближенного вычисления интеграла по конечному отрезку $[a,b]$ выполняется замена переменной $t = (a+b)/2 + (b-a)x/2$; тогда квадратурная формула Гаусса принимает вид [2,8,12]

Таблица 4.1

1	$\int_0^1 \cos\left(x+x^3\right)dx$	2	$\int_0^1 \sin\left(x^4+2x^3+x^2\right)dx$	3	$\int_0^1 \exp(\sin x)dx$
4	$\int_0^1 \sin x \exp\left(-x^2\right)dx$	5	$\int_0^1 \exp(\cos x)dx$	6	$\int_0^1 ch\ \ x^2 dx$
7	$\int_0^1 \cos\ x^2 dx$	8	$\int_0^1 \sin\left(x+x^3\right)dx$	9	$\int_0^1 \cos x \exp\left(-x^2\right)dx$
10	$\int_1^2 \sin 2x \exp\left(-x^2\right)dx$	11	$\int_0^1 \exp\left(-(x+1/x)\right)dx$	12	$\int_1^2 \ln x(x+1)^{-1} dx$
13	$\int_{\pi/2}^{\pi} \sqrt{x}\ \exp\left(-x^2\right)dx$	14	$\int_0^1 \cos\ x^3 dx$	15	$\int_0^1 \cos\ x^2 dx$
16	$\int_{\pi/4}^{\pi/2} \ln\sin x dx$	17	$\int_0^{\pi} \cos(2\sin x)dx$	18	$\int_0^{\pi} x^2\ \exp\left(-x^2\right)dx$
19	$\int_0^{\pi} x^4\ \exp\left(-x^2\right)dx$	20	$\int_{\pi/2}^{\pi} \cos\left(x+x^3\right)dx$	21	$\int_1^2 \sin x^3 dx$
22	$\int_1^2 x^{-1} \ln x(x+1)\ \ dx$	23	$\int_1^2 x^{-1} \exp x dx$	24	$\int_1^2 shx^2 dx$
25	$\int_0^{\pi} x \sin x^3 dx$	26	$\int_0^{\pi/4} \ln(1+\cos x)\ \ dx$	27	$\int_0^{\pi/4} x\cos x^3 dx$
28	$\int_{0.1}^2 \cos x / \sqrt{x} dx$	29	$\int_{0.1}^2 \sin x / \sqrt{x} dx$	30	$\int_0^{\pi/2} \sin(2\cos x)dx$
31	$\int_0^{\pi/4} \ln(1+\sin x)\ \ dx$	32	$\int_0^{\pi/2} \cos(2\sin x)dx$	33	$\int_0^1 \cos(x-1)/x^2 dx$

$$\int_a^b f(t)dt \approx \frac{b-a}{2}\sum_{i=1}^{n} A_i f\left(t_i\right),$$

где $t_i = (a+b)/2 + (b-a)x_i/2$; x_i - узлы квадратурной формулы Гаусса; A_i - гауссовы коэффициенты $i = 0,1,2,\ldots,n$.

Если подынтегральная функция достаточно гладкая, то формула Гаусса обеспечивает очень высокую точность при небольшом числе узлов.

В лабораторной работе №7 требуется, используя квадратурную формулу Гаусса наивысшего порядка точности, вычислить приближенное значение заданного интеграла.

Интеграл предлагается вычислить по квадратурной формуле Гаусса с восемью узлами:

$X_1 = X_8 = -0.96028986$, $A_1 = A_8 = 0.10122854$;

$X_2 = X_7 = -0.79666648$, $A_2 = A_7 = 0.22238103$;

$X_3 = X_6 = -0.52553242$, $A_3 = A_6 = 0.31370664$;

$X_4 = X_5 = -0.18343464$, $A_4 = A_5 = 0.36268378$.

Порядок выполнения лабораторной работы №7.

1) Составить программу-функцию для вычисления интеграла по формуле Гаусса.
2) Составить программу-функцию для вычисления значений подынтегральной функции.
3) Составить головную программу, содержащую обращение к вычислительным процедурам и осуществляющую печать результатов.
4) Результаты работы оформить в виде краткого отчета, содержащего характеристику используемого метода вычислений, его точности и полученное значение интеграла.

Варианты заданий к лабораторной работе приведены в таблице 4.2.

Библиографический список

1. Амосов А.А., Дубинский Ю.А., Копченова Н.В. Вычислительные методы для инженеров: Уч.пособие.- М.: Высш.шк., 1994. - 544 с.
2. Бахвалов Н.С., Жидков Н.О., Кобельков Г.М. Численные методы. - М.: Наука , 1987. - 600 с.
3. Березин И.С., Жидков Н.П. Методы вычислений. Т.1.- М.: Наука , 1966. - 632.
4. Демидович Б.П., Марон И.А. Основы вычислительной математики. - М.: Наука , 1970. - 664 с.
5. Копченова Н.В., Марон И.А. Вычислительная математика в примерах и задачах. - М.: Наука , 1972. - 367 с.
6. Марчук Г.И. Методы вычислительной математики. - М.: Наука , 1989. - 608 с.

7. Плис А.И., Сливина Н.А. Лабораторный практикум по высшей математики: Учеб. Пособие для втузов. - 2-е изд. - М.: Высш.шк., 1994. - 416 с.
8. Самарский А.А., Гулин А.В. Численные методы. - М.: Наука , 1989. - 432 с.
9. Форсайт Дж., Малькольм М., Моулер К. Машинные методы математических вычислений/Пер. с англ. - М.: Мир , 1980. - 279 с.
10. Хемминг Р. Численные методы/Пер. с англ. - М.: Наука , 1972. - 400 с.
11. Сборник задач по структурному программированию: Учеб. пособие/С.А. Ивановский, Ю.Е. Прокофьев, А.В. Смольянинов / Под ред. В.И. Тимохина. - Л.: ЛЭТИ, 1987. - 64 с.
12. Воробьева Г.Н., Данилова А.Н. Практикум по численным методам. - М.: Высш.шк, 1979. - 184 с.
13. Волков Е.А. Численные методы: Учеб. пособие. - М.: Наука , 1982. - 256 с.
14. Основы математического моделирования. Построение и анализ моделей с примерами на языке MATLAB: Учеб. пособие/ Д.Л. Егоренков, А.Л. Фрадков, В.Ю. Харламов; Под ред. А.Л. Фрадкова. - С.-Пб.: БГТУ, 1994. - 192 с.
15. Компьютерная математика. Методич. указ. к лаборат. работам/Сост. И.А. Назаров. - С.-Пб.: ГЭТУ, 1993. - 32 с.

Таблица 4.2

1	$\int_1^3 x^{-1}\exp x\,dx$	2	$\int_{-1}^3 x\exp(-x)\,dx$	3	$\int_1^3 dx/\ln(1+x)\,dx$
4	$\int_1^2 x^{-2}\exp(-2x)\,dx$	5	$\int_1^2 x^{-2}\exp(-2x)\,dx$	6	$\int_1^2 \exp\left(x^{-2}-x^2\right)dx$
7	$\int_0^1 \cos x/\left(1+x^2\right)dx$	8	$\int_0^1 \sin x/\left(1+x^2\right)dx$	9	$\int_0^1 x\cos x/\left(1+x^2\right)dx$
10	$\int_0^1 x\sin x/\left(1+x^2\right)dx$	11	$\int_0^1 \exp x/\left(1+x^2\right)dx$	12	$\int_0^1 x\exp x/\left(1+x^2\right)dx$
13	$\int_0^1 \sin x/(1+x)\,dx$	14	$\int_0^1 \cos x/(1+x)\,dx$	15	$\int_0^1 \left(x/\left(1+x^2\right)\right)\exp(-x)\,dx$
16	$\int_0^1 \left(1/\sqrt{1+x}\right)\exp(-x)\,dx$	17	$\int_0^1 \left(x/\left(1+x^2\right)\right)\exp(-x)^2\,dx$	18	$\int_0^1 \cos\left(x^2+x+1\right)dx$
19	$\int_0^1 \sin\left(x^2+x+1\right)dx$	20	$\int_0^1 \cos\ x\ln x\,dx$	21	$\int_0^1 x^{3/2}\exp(-x)\,dx$
22	$\int_0^{\pi/2} \exp(-\cos x)\cos(\sin x)\,dx$	23	$\int_{0.1}^1 x^2/\left(\exp x-1\right)dx$	24	$\int_0^{\pi} ch(\cos x)\,dx$
25	$\int_0^{\pi} \exp(\cos x)\cos 2x\,dx$	26	$\int_{0.1}^{0.9} \cos\ x/\sqrt{1-x^2}\,dx$	27	$\int_0^{\pi} \exp\left(\sin^2 x\right)\sin 3x\,dx$
28	$\int_0^{\pi} \cos(x-\sin x)\,dx$	29	$\int_0^{\pi} \sin(x-\cos x)\,dx$	30	$\int_0^{\pi} \exp(-x/\sin x)\,dx$
31	$\int_0^{\pi} \ln(1+\sin x)\,dx$	32	$\int_0^{\pi} \cos(2\sin x)\,dx$	33	$\int_0^{\pi} \cos\left(x+x^5\right)dx$

Содержание