ФЕДЕРВЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБПАЗОВАТЕЛЬНОЕ

УЧРЕЖДКНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«СЕВЕРО - ОСЕТИНСКИЙ НГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИМЕНИ К.Л.ХЕТАГУРОВА

Факультет Математики и информационных технологий

Направление подготовки бакалавра

01.03.02 Прикладная математика и информатика

УЧЕБНАЯ ПРАКТИКА:

ПРАКТИКА ПО ПОЛУЧЕНИЮ ПЕРВИЧНЫХ ПРОФЕССИОНАЛЬНЫХ УМЕНИИ И НАВЫКОВ

Выполнил:

Студент Цховребов Антон

Руководитель: Цахоева А.Ф.

Заведующий кафедрой: доцент, к.ф.-м.н._____Худалов М.З

Владикавказ 2018 г.

Введение

Численные методы (вычислительные методы, методы вычислений) — раздел вычислительной математики, изучающий приближенные способы решения типовых математических задач, которые либо не решаются, либо трудно решаются точными аналитическими методами (вычислительная математика вузком смысле). Примерами типовых задач являются численное решение уравнений, численные дифференцирование и интегрирование и др. Кроме численных методов, к вычислительной математике относят круг вопросов, связанных с использованием компьютеров и с программированием. Деление методов вычислений на аналитические и численные несколько условно. Следует подчеркнуть компьютерно-ориентированный характер численных методов – в конечном итоге их реализация связана с применением вычислительной техники и программирования. Аналитическое решение той или иной задачи в виде отдельных формульных соотношений является скорее исключением, нежели правилом в силу сложного и приближенного характера исследуемых моделей. Вот почему численный анализ математических моделей – метод,

алгоритм, программа, вычислительный эксперимент – является в настоящее время актуальным и наиболее эффективным аппаратом конструктивного исследования прикладных проблем.

Численное интегрирование

Задача численного интегрирования состоит в замене исходной подынтегральной функции *f(x)*, для которой трудно или невозможно записать первообразную в аналитике, некоторой аппроксимирующей функцией *φ(x)*. Такой функцией обычно является полином (кусочный полином). То есть:

$$I = \int_a^b f(x)dx = \int_a^b \varphi(x)dx + R \quad \phi(x) = \sum_{i=1}^{n} c_i \varphi_i(x),$$

где $R = \int_a^b r(x)dx$ – *априорная погрешность метода* на интервале интегрирования,

а *r(x)* – априорная погрешность метода на отдельном шаге интегрирования.

Метод прямоугольников

Различают метод левых, правых и средних прямоугольников. Суть метода ясна из рисунка 1. На каждом шаге интегрирования функция аппроксимируется полиномом нулевой степени – отрезком, параллельным оси абсцисс.

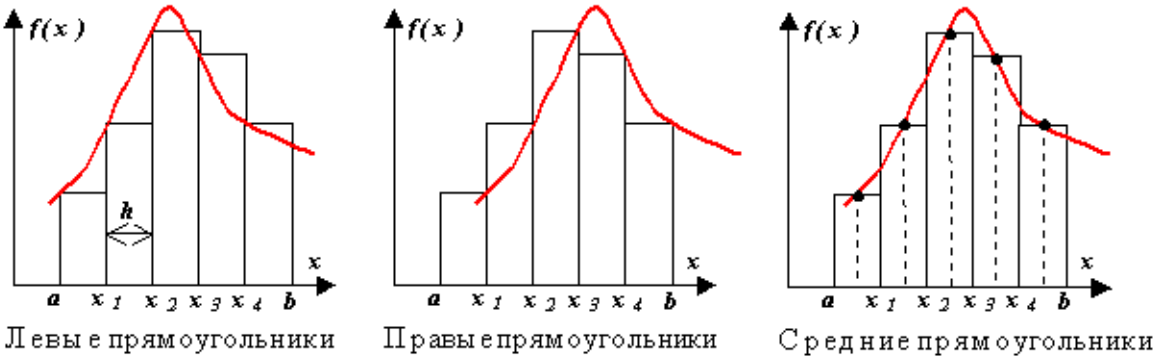


Рисунок 1

Выведем формулу метода прямоугольников из анализа разложения функции *f(x)* в ряд Тейлора вблизи некоторой точки $x = x_i$.

$$f(x)\big|_{x=x_i} = f(x_i) + (x - x_i) f'(x_i) + \frac{(x - x_i)^2}{2!} f''(x_i) + \dots$$

Рассмотрим диапазон интегрирования от x_i до $x_i + h$, где *h* – шаг интегрирования.

Вычислим $\int_{x_i}^{x_i+h} f(x)dx = x \cdot f(x_i)\Big|_{x_i}^{x_i+h} + \frac{(x - x_i)^2}{2} f'(x_i)\Big|_{x_i}^{x_i+h} + \frac{(x - x_i)^3}{3 \cdot 2!} f''(x_i)\Big|_{x_i}^{x_i+h} + \dots =$

$= f(x_i)h + \frac{h^2}{2} f'(x_i) + O(h^3) = \boxed{f(x_i)h + r_i}$. Получили формулу *правых (или левых) прямоугольников* и априорную оценку погрешности *r* на отдельном шаге интегрирования. Основной критерий, по которому судят о точности алгоритма – степень при величине шага в формуле априорной оценки погрешности.

В случае равного шага *h* на всем диапазоне интегрирования общая формула имеет вид

$$\int_a^b f(x)dx = h\sum_{i=0}^{n-1} f(x_i) + R$$

.

Здесь *n* – число разбиений интервала интегрирования, $R = \sum_{i=0}^{n-1} r_i = \frac{h}{2} \cdot h\sum_{i=0}^{n-1} f'(x_i) = \frac{h}{2}\int_a^b f'(x)dx$. Для справедливости существования этой оценки необходимо существование непрерывной *f'(x)*.

Метод средних прямоугольников. Здесь на каждом интервале значение функции считается в точке $\overline{x} = x_i + \frac{h}{2}$, то есть $\int_{x_i}^{x_i+h} f(x)dx = hf(\overline{x}) + r_i$. Разложение функции в ряд Тейлора показывает, что в случае средних прямоугольников точность метода существенно выше:

$$r = \frac{h^3}{24} f''(\overline{x}),\ R = \frac{h^2}{24}\int_a^b f''(x)dx$$

.

Метод трапеций

Аппроксимация в этом методе осуществляется полиномом первой степени. Суть метода ясна из рисунка 1. На единичном интервале

$$\int_{x_i}^{x_i+h} f(x)dx = \frac{h}{2}\left(f(x_i) + f(x_i+h)\right) + r_i$$

.

В случае равномерной сетки (*h* = const)

$$\int_a^b f(x)dx = h\left(\frac{1}{2}f(x_0) + \sum_{i=1}^{n-1} f(x_i) + \frac{1}{2}f(x_n)\right) + R$$

При этом $r_i = -\frac{h^3}{12}f''(x_i)$, а $R = -\frac{h^3}{12}\int_a^b f''(x)dx$. Погрешность метода трапеций в два раза выше, чем у метода средних прямоугольников! Однако на практике найти среднее значение на элементарном интервале можно только у функций, заданных аналитически (а не таблично), поэтому использовать метод средних прямоугольников удается далеко не всегда. В силу разных знаков погрешности в формулах трапеций и средних прямоугольников истинное значение интеграла обычно лежит между двумя этими оценками.

Метод Симпсона

Подынтегральная функция *f(x)* заменяется интерполяционным полиномом второй степени *P(x)* – параболой, проходящей через три узла, например, как показано на рисунке ((1) – функция, (2) – полином).

Рассмотрим два шага интегрирования (h = const = $x_{i+1} - x_i$), то есть три узла x_0, x_1, x_2, через которые проведем параболу, воспользовавшись уравнением Ньютона:

$$P(x) = f_0 + \frac{x - x_0}{h}(f_1 - f_0) + \frac{(x - x_0)(x - x_1)}{2h^2}(f_0 - 2f_1 + f_2).$$

Пусть z = x - x_0, тогда

$$P(z) = f_0 + \frac{z}{h}(f_1 - f_0) + \frac{z(z-h)}{2h^2}(f_0 - 2f_1 + f_2) =$$

$$= f_0 + \frac{z}{2h}(-3f_0 + 4f_1 - f_2) + \frac{z^2}{2h^2}(f_0 - 2f_1 + f_2)$$

Теперь, воспользовавшись полученным соотношением, сосчитаем интеграл по данному интервалу:

$$\int_{x_0}^{x_2} P(x)dx = \int_0^{2h} P(z)dz = 2hf_0 + \frac{(2h)^2}{4h}(-3f_0 + 4f_1 - f_2) + \frac{(2h)^3}{6h^2}(f_0 - 2f_1 + f_2) =$$

$$= 2hf_0 + h(-3f_0 + 4f_1 - f_2) + \frac{4h}{3}(f_0 - 2f_1 + f_2) = \frac{h}{3}(6f_0 - 9f_0 + 12f_1 - 3f_2 + 4f_0 - 8f_1 + 4f_2)$$

.

В итоге

$$\boxed{\int_{x_0}^{x_2} f(x)dx = \frac{h}{3}(f_0 + 4f_1 + f_2) + r}$$

.

Для равномерной сетки и четного числа шагов n формула Симпсона принимает вид:

$$\int_a^b f(x)dx = \frac{h}{3}\left(f_0 + 4f_1 + 2f_2 + 4f_3 + 2f_4 + \ldots + 2f_{n-2} + 4f_{n-1} + f_n\right) + R$$

Здесь $r = -\frac{h^5}{90} f^{IV}(x_i)$, а $R = -\frac{h^4}{180}\int_a^b f^{IV}(x)dx$ в предположении непрерывности четвертой производной подынтегральной функции.

Индивидуальный вариант №22

Текст программы

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication6
{
 class Program
 {
  static double Y(double x)
  {
   return x * x;

     // 1. Math.Sqrt(x * x + 0.5) / (2 * x + Math.Sqrt(x * x + 2.5))
     // 2.Math.Cos(x * x + 1) / (2 + Math.Sin(2 * x + 0.5))
  }

  static void LT(double a, double b, double h)
  { //левые
   double sum = 0;

   for (double i = a; i < b - h / 2; i += h)
   {
    sum += Y(i) * h;
   }
   Console.WriteLine("МЛТ " + sum);

  }
  static void RT(double a, double b, double h)
  { //правые
   double sum = 0;
   for (double i = a; i < b - h / 2; i += h)
```

```
    {
      sum += Y(i + h) * h;
    }
    Console.WriteLine("МПТ " + sum);
  }

  static void CT(double a, double b, double h)
  {  //центральные
    double sum = 0;

    for (double i = a; i < b - h / 2; i += h)
    {
      sum += Y(i + h / 2) * h;

    }
    Console.WriteLine("МСТ " + sum);
  }
  static void MS(double a, double b, double h)
  {
    double x, s;
    s = 0; x = a + h;
    while (x < b)
    {
      s = s + 4 * Y(x);
      x = x + h;
      s = s + 2 * Y(x);
      x = x + h;
    }
    s = h / 3 * (s + Y(a) - Y(b));
    Console.WriteLine("МС {0}", s);

  }

  static void MT(double a, double b, double h)
  { //трапеции
    double sum = 0;
    int count = 0;
    for (double i = a; i < b - h / 2; i += h)
    {
      sum += (Y(i) + Y(i + h)) * h / 2;

    }
    Console.WriteLine("МТ " + sum);
  }


  static void Main(string[] args)
  {
    double a = 0;
    double b = 1;
    int n = 10;

    double h = (b - a) / n;
    MS(a, b, h);
    MT(a, b, h);
    LT(a, b, h);
    RT(a, b, h);
    CT(a, b, h);

  }
 }
}
```

Результаты работы программы

	МС 0,333333333333333 МТ 0,335 МЛТ 0,285 МПТ 0,385 МСТ 0,3325

	МС 0,253992825786528 МТ 0,25401714462711 МЛТ 0,25407484479835 МПТ 0,25395944445587 МСТ 0,253980379364562
	МС 0,0579108939062036 МТ 0,0578027442160447 МЛТ 0,0639835434068187 МПТ 0,0516219450252707 МСТ 0,0579650760195568

Численные методы решения нелинейных уравнений

Метод половинного деления (метод дихотомии)

Метод половинного деления известен также как **метод дихотомии**. В данном методе интервал делится ровно пополам.

Такой подход обеспечивает гарантированную сходимость метода независимо от сложности функции - и это весьма важное свойство. Недостатком метода является то же самое - метод никогда не сойдется быстрее, т.е. сходимость метода всегда равна сходимости в наихудшем случае.

Метод половинного деления:

1. Один из простых способов поиска корней функции одного аргумента.

2. Применяется для нахождения значений действительно-значной функции, определяемому по какому-либо критерию (это может быть сравнение на минимум, максимум или конкретное число).

Метод половинного деления как метод поиска корней функции

Изложение метода

Перед применением метода для поиска корней функции необходимо отделить корни одним из известных способов, например, графическим методом. Отделение корней необходимо в случае, если неизвестно на каком отрезке нужно искать корень.

Будем считать, что корень t функции $f(x)=0$ отделён на отрезке $[a,b]$. Задача заключается в том, чтобы найти и уточнить этот корень методом половинного деления. Другими словами, требуется найти приближённое значение корня с заданной точностью ϵ.

Пусть функция f непрерывна на отрезке $[a,b]$, $f(a)\cdot f(b)<0$, $\epsilon=0{,}01$ и $t\in[a,b]$ - единственный корень уравнения $f(x)=0$, $a\le t\le b$.

(Мы не рассматриваем случай, когда корней на отрезке $[a,b]$ несколько, то есть более одного. В качестве ϵ можно взять и другое достаточно малое положительное число, например, $0{,}001$.)

Поделим отрезок $[a,b]$ пополам. Получим точку $c=\frac{a+b}{2}$, $a<c<b$ и два отрезка $[a,c]$, $[c,b]$.

Если $f(c)=0$, то корень t найден ($t=c$).

Если нет, то из двух полученных отрезков $[a,c]$ и $[c,b]$ надо выбрать один $[a_1;b_1]$ такой, что $f(a_1)\cdot f(b_1)<0$, то есть

- $[a_1;b_1]=[a,c]$, если $f(a)\cdot f(c)<0$ или
- $[a_1;b_1]=[c,b]$, если $f(c)\cdot f(b)<0$.

Новый отрезок $[a_1;b_1]$ делим пополам. Получаем середину этого отрезка $c_1=\frac{a_1+b_1}{2}$ и так далее. Для того, чтобы найти приближённое значение корня с точностью до $\epsilon>0$, необходимо остановить процесс половинного деления на таком шаге n, на котором $|b_n-c_n|<\epsilon$ и вычислить $x=\frac{a_n+b_n}{2}$. Тогда можно взять $t\approx x$.

Алгоритм

Пусть корень уравнения

$f(x)=0$

отделен на отрезке *[a, b]*, т.е. *f(a)f(b)< 0* и *f '(x)* сохраняет знак (рис. 2.6).

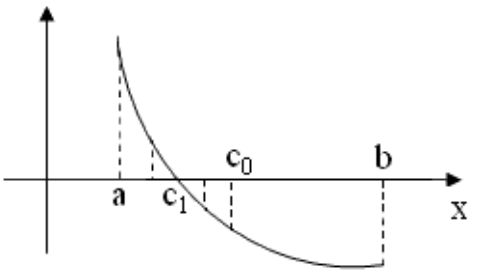


Рис. 2.6

В качестве начального приближения корня возьмем точку c_0 – середину отрезка.

$$c_0 = \frac{a+b}{2}.$$

Если $f(c_0) = 0$, то c_0 – искомый корень уравнения, если

$$f(c_0) \neq 0,$$

то из двух отрезков $[a, c_0]$ и $[c_0, b]$ выбираем тот, на концах которого функция принимает значение разных знаков.

Новый отрезок опять делим пополам и далее поступаем аналогично вышеизложенному. Длина каждого нового отрезка вдвое меньше длины предыдущего отрезка, т.е. за n шагов сократится в 2^n раз.

Вычисления прекращаем, если длина отрезка $[c_к, c_{к+1}]$, станет меньше заданной погрешности ε, т.е.:

$$\left|c_k - c_{k+1}\right| < \varepsilon$$

Блок-схема алгоритма для решения уравнений методом Ньютона дана на рис. 2.7

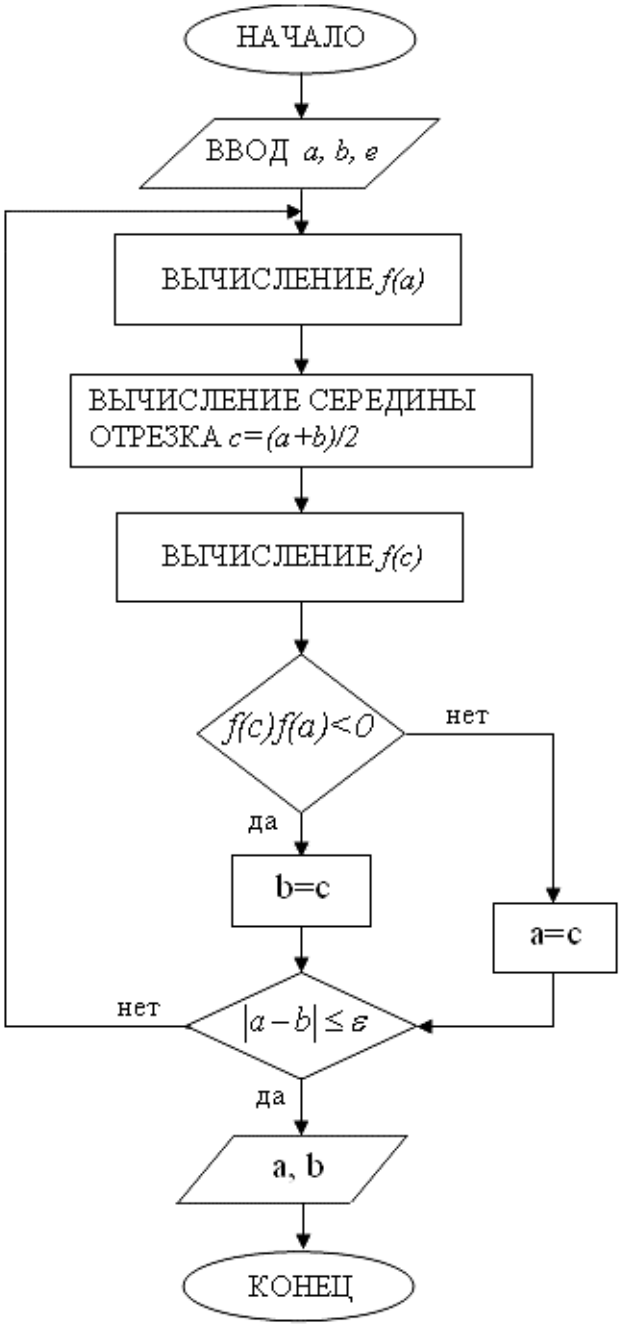


Рисунок 2.7 Блок-схема метода половинного деления

Индивидуальный вариант №22

(рисунок 1)

(рисунок 2)

Текст программы

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication9
{
class Program
{
static double f(double x)
{
return 2*x + Math.Cos(x) - 0.5;
}

static void Main(string[] args)
{

double a, b, c, epsilon = 0.001;
a = -1;
b = 0;
int count = 0;
while (b - a > epsilon)
{
c = (a + b) / 2;
if (f(b) * f(c) < 0)
a = c;
else
b = c;

count++;
}
Console.WriteLine("x = "+(a + b) / 2);
Console.WriteLine("n = "+count);

}
}
}
```

Результаты работы программы

	x = -0,23583984375 n = 10

	x = 1,03369140625 n = 12

Метод последовательных приближении (метод итерации)

Метод итерации — численный метод решения математических задач, приближённый метод решения системы линейных алгебраических уравнений. Суть такого метода заключается в нахождении по приближённому значению величины следующего приближения (являющегося более точным). Метод позволяет получить значения корней системы с заданной точностью в виде предела последовательности некоторых векторов (итерационный процесс). Характер сходимости и сам факт сходимости метода зависит от выбора начального приближения корня x_0.

Метод простых итераций применяется для решения уравнений частного вида, когда уравнение может быть записано в виде

$$F(x) = x - f(x) = 0.$$

На рисунке 3.1 приведена геометрическая интерпретация метода.

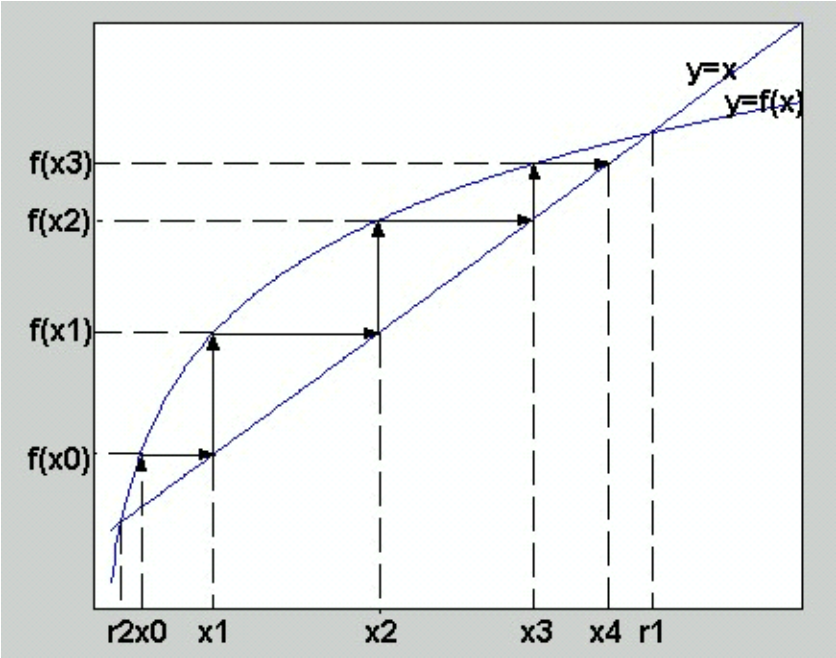


Рис 3.1 Геометрическая интерпретация метода простых итераций

На рисунке показаны два графика для функций y=x и y=f(x). Точки пересечения этих кривых определяют корни уравнения x - f(x) = 0. Эти корни обозначены r1 и r2. Для нахождения корня методом простых итераций предлагается следующая процедура. Выберем произвольную точку x0 на оси x. Проведем перпендикуляр из точки x0 до пересечения с кривой y=f(x). В качестве первого приближения к корню возьмем точку x1=y(x0). Из точки x1 проведем перпендикуляр до пересечения с кривой y=f(x). В качестве второго приближения к корню выберем точку x2=f(x1).

Продолжая этот процесс, можно видеть, что каждое последующее приближение к корню определяется через предыдущее по формуле

$$x_{n+1} = f(x_n) \qquad (3.1)$$

Справедливо следующее утверждение (без доказательства). Если f(C0) в точке пересечения графиков y=x и y=f(x), тогда итерационный процесс (3.1) сходится к этой точке.

В нашем примере такой точкой является точка x=r1. Такие корни называются притягивающими для метода простых итераций

Корень r2 в нашем примере является отталкивающим и не может быть найден методом простых итераций (3.1). Таким образом очевидным недостатком метода простых итераций является то, что не все корни уравнения x-f(x)=0 могут быть с его помощью найдены.

Блок-схема алгоритма для решения уравнений методом простых итераций дана на рис. 3.2.

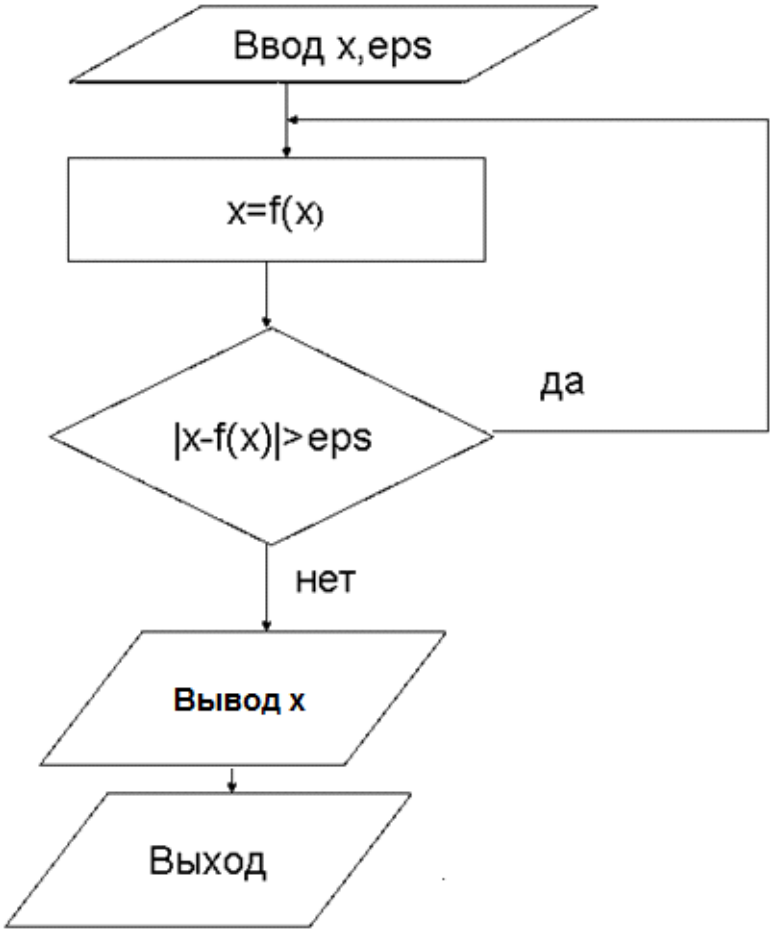


Рисунок 3.2

Пример

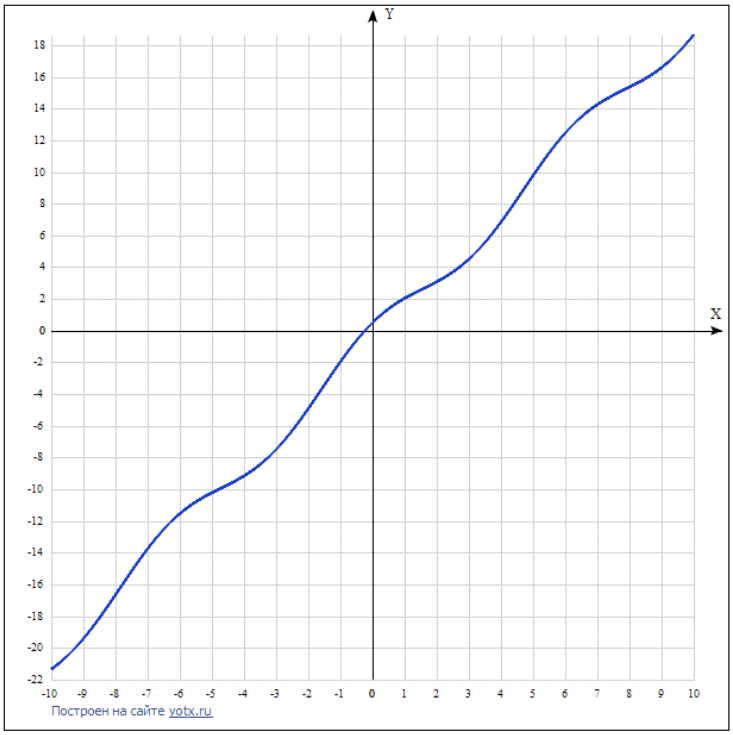
– условие сходимости выполняется

Ответ:

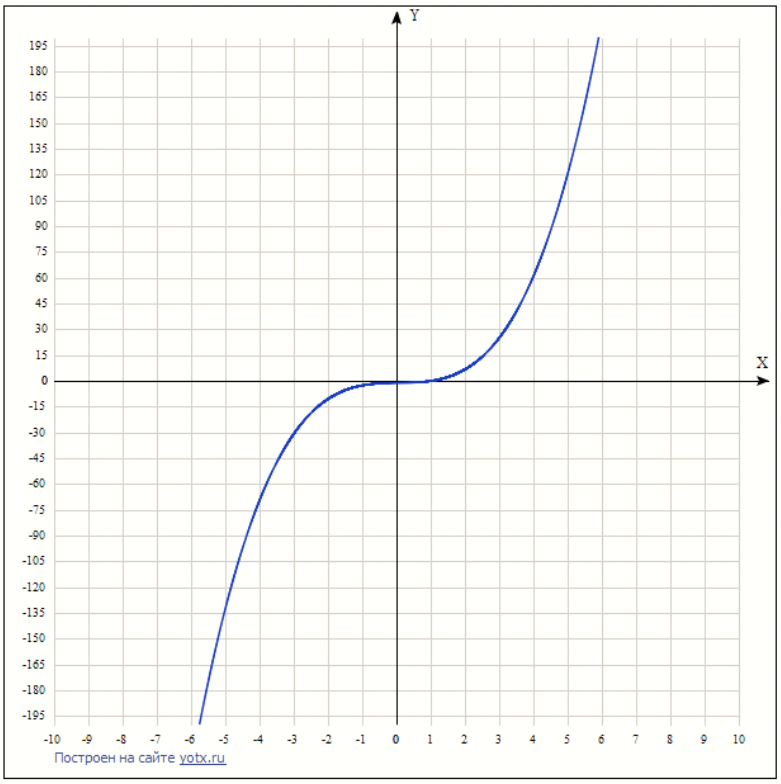
Индивидуальный вариант №22

график (рисунок 1)

– условие сходимости выполняется


Y
X
18
16
14
12
10
8
6
4
2
0
-2
-4
-6
-8
-10
-12
-14
-16
-18
-20
-22
-10 -9 -8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 8 9 10
Построен на сайте yotx.ru

(рисунок 2)



Текст программы

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace math
{
class Program
{
static double f(double x)
{
return (0.5 - Math.Cos(x))/2;
}

static void Main(string[] args)
{
double c = -0.5;
double eps = 0.001;
```

```
int count = 0;
double x;
while (true)
{
count++;
x = f(c);
if (Math.Abs(x - c) < eps)
break;
else
c = x;

}
Console.WriteLine("x = " + x);
Console.WriteLine("n = " + count);


}
}
}
```

Результаты работы программы

	x = -0,236194691993718 n = 4

	x = 1,0313170464277 n = 9

Метод Ньютона (метод касательных)

Метод Ньютона, алгоритм Ньютона (также известный как метод касательных) — это итерационный численный метод нахождения корня (нуля) заданной функции. Метод был впервые предложен английским физиком, математиком и астрономом Исааком Ньютоном (1643—1727).

Поиск решения осуществляется путём построения последовательных приближений и основан на принципах простой итерации. Метод обладает квадратичной сходимостью. Модификацией метода является метод хорд и касательных. Также метод Ньютона может быть использован для решения задач оптимизации, в которых требуется определить нуль первой производной либо градиента в случае многомерного пространства.

Обоснование

Чтобы численно решить уравнение $f(x)=0$ методом простой HYPERLINK "https://dic.academic.ru/dic.nsf/ruwiki/1034707"итерации, его необходимо привести к следующей форме: $x=\varphi(x)$, где φ — сжимающее HYPERLINK "https://dic.academic.ru/dic.nsf/ruwiki/682178"отображение. Для наилучшей сходимости метода в точке очередного приближения x^* должно выполняться условие $\varphi'(x^*)=0$. Решение данного уравнения ищут в виде $\varphi(x)=x+\alpha(x)f(x)$, тогда:

$$\varphi'(x^*)=1+\alpha'(x^*)f(x^*)+\alpha(x^*)f'(x^*)=0$$

В предположении, что точка приближения «достаточно близка» к корню $\tilde{x}$, и что заданная функция непрерывна $(f(x^*)\approx f(\tilde{x})=0)$, окончательная формула для $\alpha(x)$ такова:

$$\alpha(x)=-\frac{1}{f'(x)}$$

С учётом этого функция $\varphi(x)$ определяется выражением:

$$\varphi(x)=x-\frac{f(x)}{f'(x)}$$

Эта функция в окрестности корня осуществляет сжимающее отображение[1], и алгоритм нахождения численного решения уравнения $f(x)=0$ сводится к итерационной процедуре вычисления:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Геометрическая интерпретация

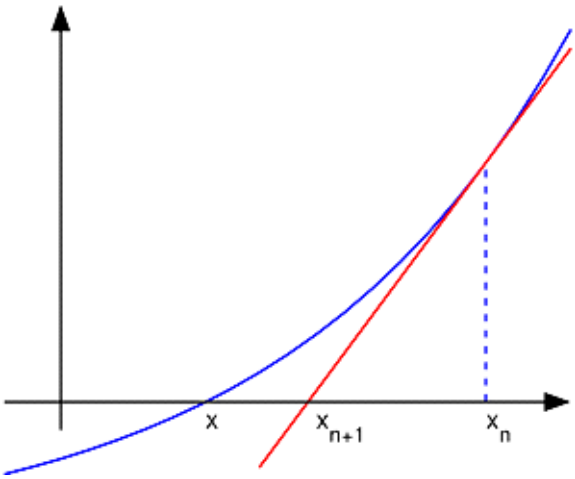


Иллюстрация метода Ньютона изображена функция $f(x)$, нуль которой необходимо найти, касательная в точке очередного приближения x_n Здесь мы можем увидеть, что последующее приближение x_{n+1} лучше предыдущего x_n.

Основная идея метода заключается в следующем: задаётся начальное п риближение вблизипредположительного корня, после чего строится касатель ная к исследуемой функции в точке приближения, для которой находится пер есечение с осью абсцисс. Эта точка и берётся в качестве следующегоприближ ения. И так далее, пока не будет достигнута необходимая точность.

Пусть $f(x) : [a, b] \to \mathbb{R}$ — определённая на отрезке [*a*, *b*] и дифференцируемая на нёмдействительнозна чная функция. Тогда формула исчисления приближений может быть выведе наследующим образом:

$$f'(x_n) = \operatorname{tg}\alpha = \frac{\Delta y}{\Delta x} = \frac{f(x_n) - 0}{x_n - x_{n+1}} = \frac{0 - f(x_n)}{x_{n+1} - x_n},$$

где α — угол наклона касательной в точке x_n.

Следовательно искомое выражение для x_{n+1} имеет вид:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}.$$

Итерационный процесс начинается с некого начального приближения x_0 (чем ближе к нулю, тем лучше, но если предположения о нахождении решения отсутствуют, методом проб и ошибок можно сузить область возможных значений, применив теорему о промежуточных HYPERLINK "https://dic.academic.ru/dic.nsf/ruwiki/1253994"значениях).

Алгоритм

Задаются начальным приближением x_0.

Пока не выполнено условие остановки, в качестве которого можно взять $|x_{n+1} - x_n| < \varepsilon$ или $|f(x_{n+1})| < \varepsilon$

(то есть погрешность в нужных пределах), вычисляют новое приближение:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Блок-схема алгоритма для решения уравнений методом Ньютона дана на рис.4

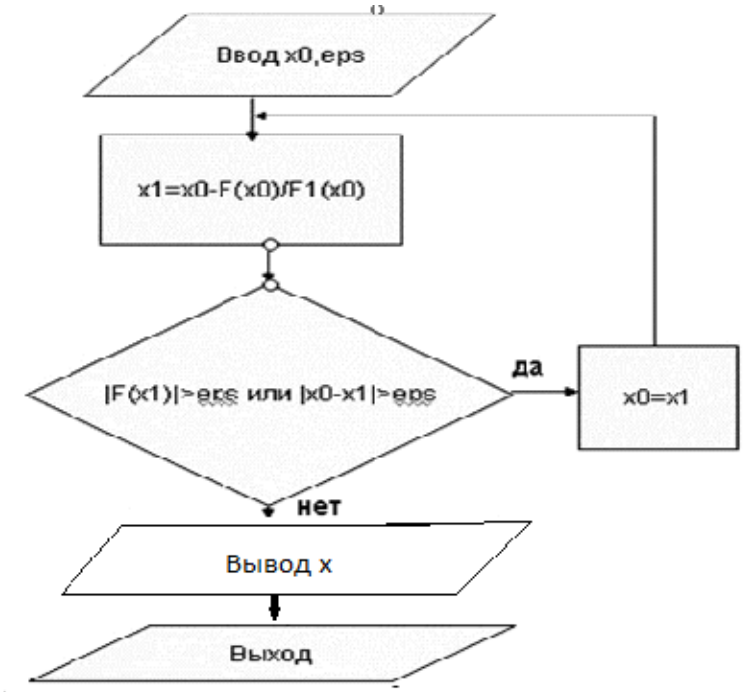


Рисунок 4

Пример

Индивидуальный вариант №22

Текст программы

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace math
{
class Program
{
static double f(double x)
{
return 2 * x + Math.Cos(x) - 0.5;
}
static double p(double x)
{
return 2 - Math.Sin(x);
}
static void Main(string[] args)
{
double eps = 0.001;
double c = -0.5;
double x;
int count = 0;
while (true)
{
count++;
```

```
x = c - f(c) / p(c);

if (Math.Abs(x-c) < eps)
break;
else
c=x;
}
Console.WriteLine("x = "+x);
Console.WriteLine("n = " + count);

}
}
}
```

Результаты работы программы

	x = -0,24896707789014 n = 1

	x = 1,03339326044463 n = 3

Заключение

В ходе вычисления интегралов методами численного интегрирования, метод Симпсона оказался наиболее точным методом вычисления интеграла.

В ходе решения уравнении численными методами наиболее эффективным оказался метод Ньютона.

Метод Дихотомии	10	12

Метод Итерации	4	9
Метод Ньютона	1	3

Из таблицы видно, что методом Ньютона ответ вычислялся за меньшое количество итерации, чем методом итерации и методом дихотомии.

Список литературы

- Солодовников, А.С. Численные методы линейной алгебры: Учебное пособие / А.С. Солодовников. - М.: Финансы и статистика, 2012. - 480 с.

- Вабищевич, П.Н. Численные методы: Вычислительный практикум / П.Н. Вабищевич. - М.: Ленанд, 2016. - 320 с.
- Протасов И.Д. Лекции по вычислительной математике: учеб. пособ. – М.: Гелиос АРВ, 2009.
- Ерохин, Б.Т. Численные методы: Учебное пособие / Б.Т. Ерохин. - СПб.: Лань КПТ, 2016. - 256 с.
- Пирумов, У.Г. Численные методы: теория и практика: Учебное пособие для бакалавров / У.Г. Пирумов, В.Ю. Гидаспов, И.Э. Иванов. - М.: Юрайт, 2012. - 421 с.
- Ширяев, В.И. Исследование операций и численные методы оптимизации: Учебное пособие / В.И. Ширяев. - М.: Ленанд, 2015. - 216 с.
- Киреев, В.И. Численные методы в примерах и задачах: Учебное пособие / В.И. Киреев, А.В. Пантелеев. - СПб.: Лань, 2015. - 448 с.
- Калиткин, Н.Н. Численные методы / Н.Н. Калиткин. - СПб.: BHV, 2014. - 592 с.