МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Вятский государственный университет»**

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

ЧИСЛЕННЫЕ МЕТОДЫ РЕШЕНИЯ НЕЛИНЕЙНЫХ УРАВНЕНИЙ

Отчет по лабораторной работе №1

по дисциплине «Вычислительная математика»

Выполнил студент группы ИВТ-23 \_\_\_\_\_\_\_\_\_\_/Кудяшев Я.Ю./

Проверил преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Исупов К.С./

Киров 2020

1. Постановка задачи

Построить график функции f(x) и отделить один из корней уравнения: f(x). Сузить интервал изоляции корня, если необходимо, проверив условие: M<=2m. Уточнить корень с погрешностью e<=0,00001 двумя численными методами: комбинированным методом и методом итераций. Проверить полученное значение корня, используя систему Mathcad.

Вариант 11

Уравнение: 2\*x-ln(x)-7=0

Интервал: [4,0;5,0]

2 Краткие теоретические сведения

2.1 Комбинированный метод хорд и касательных

Методы хорд и касательных дают приближения корня с разных сторон. Поэтому их часто применяют в сочетании друг с другом, тогда уточнение корня происходит быстрее.

Пусть дано уравнение *f(x) = 0*, корень отделен на отрезке *[a, b]*.

Рассмотрим случай, когда *f ‘(x) f ’’(x)>0* (рис. 1).

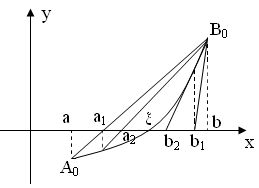


Рис. 1

В этом случае метод хорд дает приближенное значение корня с недостатком (конец *b* неподвижен), а метод касательных – с избытком (за начальное приближение берем точку *b*).

Тогда вычисления следует проводить по формулам:





Теперь корень *ξ* заключен в интервале *[a1, b1]*. Применяя к этому отрезку комбинированный метод, получим:





и т.д.

|  |  |
| --- | --- |
|  | (2.6) |

Если же *f ‘(x) f ’’(x)<0* (рис. 2), то, рассуждая аналогично, получим следующие формулы для уточнения корня уравнения:

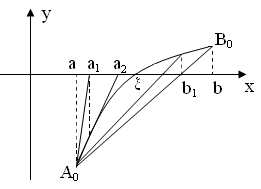


Рис. 2





Вычислительный процесс прекращается, как только выполнится условие:



2.2 Метод итераций

Этот метод требует приведения исходного уравнения к каноническому виду: . Тогда одношаговый итерационный процесс строится по формуле:  при выборе любого нулевого приближения из интервала изоляции.

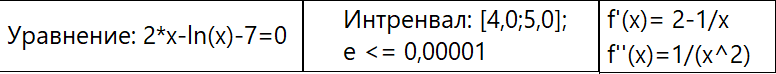
Главное - проверка соблюдения условий сходимости для канонического уравнения:  - это обеспечит сходящийся итерационный процесс и получение значения корня уравнения с требуемой точностью за конечное число шагов.

Существует ***стандартный приём*** преобразования уравнения к каноническому виду, обеспечивающий сходимость итерационного процесса. . Знак  совпадает со знаком первой производной исходной функции. Итерации продолжают до тех пор, пока не будет выполнено условие:

, где  - погрешность.

Если функция  возрастает, приближённые значения сходятся к точному значению корня ***монотонно,*** если же функция  убывает, то приближённые значения  ***колеблются*** вокруг точного значения.

3 Выполнение задания



4 Экранные формы работы программы

Экранные формы работы программы представлены на рисунках 3-6.

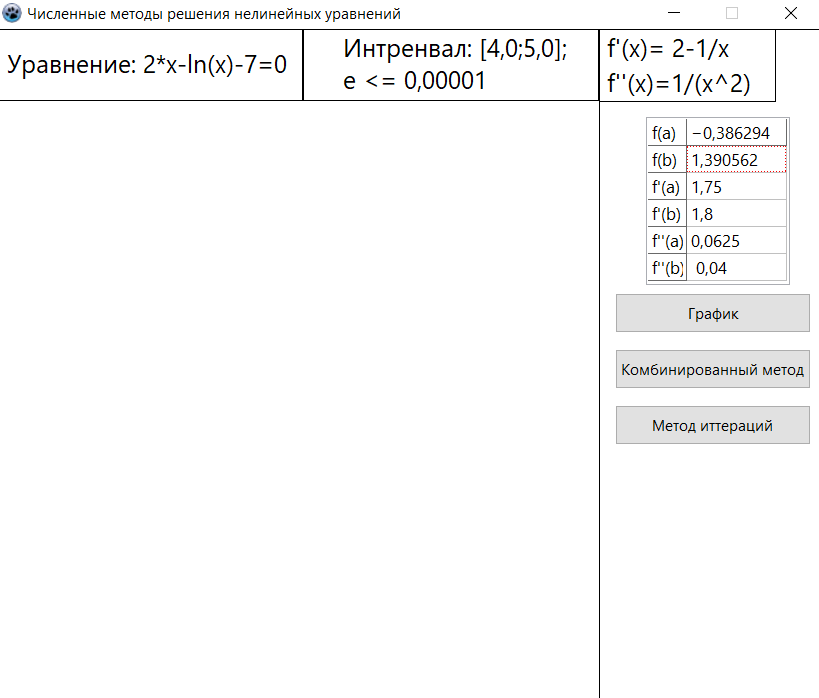


Рисунок 3 – Начальная форма

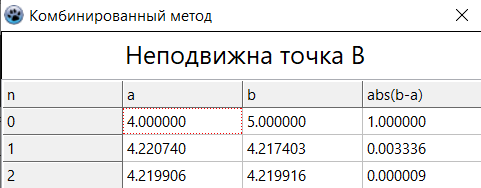


Рисунок 4 – Решение уравнения комбинированным методом

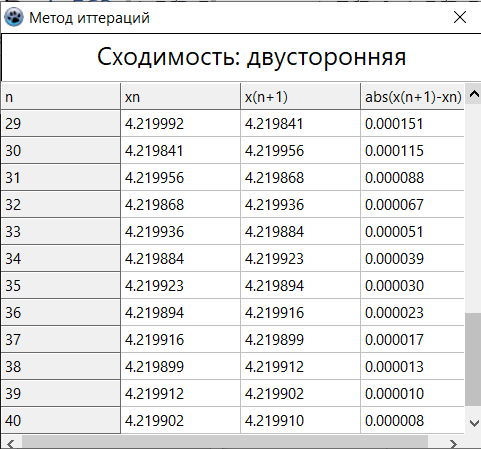


Рисунок 5 – Решение уравнения методом итерации

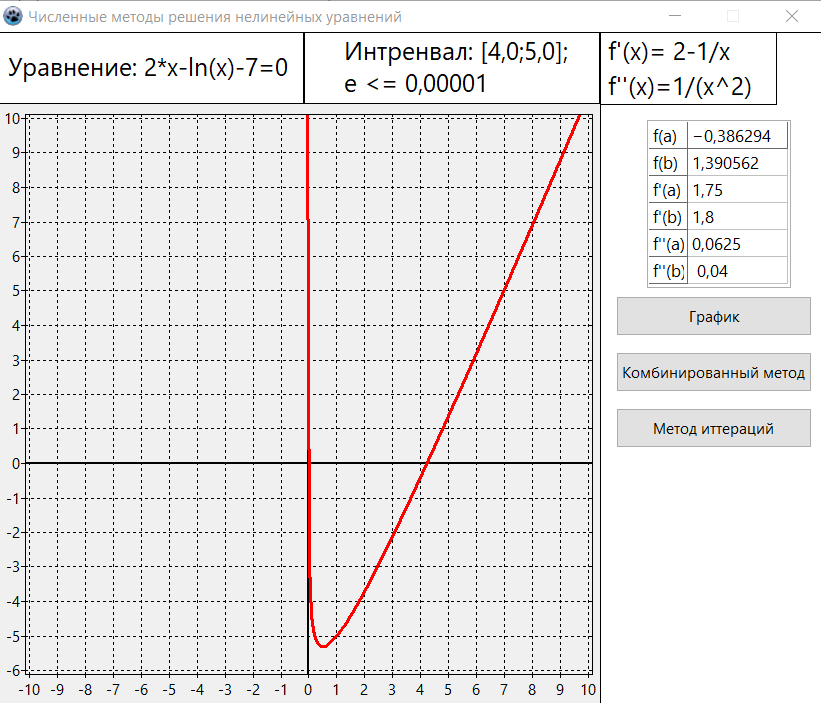


Рисунок 6 – График функции

5 Проверка вычислений корня в системе wolframalphа:



Рисунок 7 – результат проверки корня в системе wolframalphа

6 Листинг программы

unit Unit1;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils, Forms, Controls, Graphics, Dialogs, ExtCtrls, StdCtrls,

Grids, TAGraph, TASeries, TARadialSeries, TAFuncSeries,unit2,unit3;

type

{ TForm1 }

TForm1 = class(TForm)

Button1: TButton;

Button2: TButton;

Button3: TButton;

Chart1: TChart;

Chart1ConstantLine1: TConstantLine;

Chart1ConstantLine2: TConstantLine;

Chart1FuncSeries1: TFuncSeries;

Image1: TImage;

Label1: TLabel;

Label2: TLabel;

Label3: TLabel;

Label4: TLabel;

PaintBox1: TPaintBox;

Shape1: TShape;

Shape2: TShape;

Shape3: TShape;

Shape4: TShape;

StringGrid1: TStringGrid;

procedure Button1Click(Sender: TObject);

procedure Button2Click(Sender: TObject);

procedure Button3Click(Sender: TObject);

procedure Chart1FuncSeries1Calculate(const AX: Double; out AY: Double);

private

public

end;

var

Form1: TForm1;

xc,yc:integer;

ms:real;

s:string;

implementation

{$R \*.lfm}

{ TForm1 }

procedure TForm1.Button1Click(Sender: TObject);

var i:integer;

begin

Form1.Chart1.Visible:=true;

//Form1.Image1.Visible:=true;

{ xc:=Form1.PaintBox1.Width div 2;

yc:=Form1.PaintBox1.Height div 2;

ms:=(yc-30)/10;

Form1.PaintBox1.Canvas.Pen.Width:=1;

Form1.PaintBox1.Canvas.Pen.Color:=clBlack;

Form1.PaintBox1.Canvas.MoveTo(0,yc);

Form1.PaintBox1.Canvas.LineTo(Form1.Width,yc);

Form1.PaintBox1.Canvas.MoveTo(xc,0);

Form1.PaintBox1.Canvas.LineTo(xc,Form1.Height);

for i:=1 to 12 do

if i mod 2=0 then

begin

Form1.PaintBox1.Canvas.MoveTo(xc-3,yc-round(i\*ms));

Form1.PaintBox1.Canvas.LineTo(xc+3,yc-round(i\*ms));

Form1.PaintBox1.Canvas.MoveTo(xc-3,yc+round(i\*ms));

Form1.PaintBox1.Canvas.LineTo(xc+3,yc+round(i\*ms));

Form1.PaintBox1.Canvas.MoveTo(xc+round(i\*ms),yc-3);

Form1.PaintBox1.Canvas.LineTo(xc+round(i\*ms),yc+3);

Form1.PaintBox1.Canvas.MoveTo(xc-round(i\*ms),yc-3);

Form1.PaintBox1.Canvas.LineTo(xc-round(i\*ms),yc+3);

str(i,s);

Form1.PaintBox1.Canvas.textout(xc-20,yc-round(i\*ms),s);

Form1.PaintBox1.Canvas.textout(xc-25,yc+round(i\*ms),'-'+s);

Form1.PaintBox1.Canvas.textout(xc+round(i\*ms),yc+10,s);

Form1.PaintBox1.Canvas.textout(xc-round(i\*ms),yc+10,'-'+s);

end;

Form1.PaintBox1.Canvas.textout(xc+5,yc+10,'0');

Form1.PaintBox1.Canvas.Pixels[xc,yc]:=clRed;

for i:=300 to 600 do

Form1.PaintBox1.Canvas.Pixels[i,round(2\*i-ln(i)-7)]:=clRed; }

end;

procedure TForm1.Button2Click(Sender: TObject);

var a,b,e,delta:real;

colvo,n,aaa,bbb:integer;

aa,bb,cc:string;

begin

Form3.Show;

colvo:=0;

n:=1;

e:=0.00001;

a:=4;

aaa:=4;

b:=5;

bbb:=5;

repeat

Form3.StringGrid1.Cells[0,n] :=inttostr(colvo);

str(a:0:6,aa);

Form3.StringGrid1.Cells[1,n] :=aa;

str(b:0:6,bb);

Form3.StringGrid1.Cells[2,n] :=bb;

delta:=abs(b-a);//abs((b+a)/2);

str(delta:0:6,cc);

Form3.StringGrid1.Cells[3,n] :=cc;

n:=n+1;

a:=a-(2\*a-ln(a)-7)/(2-1/a);

b:=b-((2\*b-ln(b)-7)\*(aaa-b))/((2\*aaa-ln(aaa)-7)-(2\*b-ln(b)-7));

Form3.StringGrid1.RowCount:=Form3.StringGrid1.RowCount+1;

colvo:=colvo+1;

until delta<=e;//(2\*delta-ln(delta)-7)<=e;

end;

procedure TForm1.Button3Click(Sender: TObject);

var xn,fi,delta,e:real;

colvo,n:integer;

a,b,c:string;

begin

Form2.Show;

xn:=4;

e:=0.00001;

fi:=xn-(2\*xn-ln(xn)-7);

colvo:=0;

n:=1;

repeat

Form2.StringGrid1.Cells[0,n] :=inttostr(colvo);

str(xn:0:6,a);

Form2.StringGrid1.Cells[1,n] :=a;

fi:=xn-(2\*xn-ln(xn)-7);

str(fi:0:6,b);

Form2.StringGrid1.Cells[2,n] :=b;

delta:=abs(fi-xn);

str(delta:0:6,c);

Form2.StringGrid1.Cells[3,n] :=c;

n:=n+1;

xn:=fi;

Form2.StringGrid1.RowCount:=Form2.StringGrid1.RowCount+1;

colvo:=colvo+1;

until delta<=e;

end;

procedure TForm1.Chart1FuncSeries1Calculate(const AX: Double; out AY: Double);

begin

AY:=2\*(AX)-ln(AX)-7;

end;

end.

7 Вывод

В ходе лабораторной работы были рассмотрены комбинированный метод и метод итераций для решения нелинейных уравнений.

В комбинированном методе уточнение корня происходит быстрее и количество вычислений значительно меньше, чем в методе итерации.

В методе итераций эти вычисления значительно проще. Была разработана программа, позволяющая использовать эти методы для решения конкретного уравнения.