Задание №1. Вычисление значений элементарных функций

Вариант №10

***Цель задания***: *практическое освоение методов анализа погрешностей в задаче вычисления значений элементарных функций.*

1. По указанной точности ( ) решить обратную задачу теории погрешностей для заданной функции z(x).

*Примечание:* Номер варианта задания (см. Приложение 2) должен совпадать с Вашим номером в списке группы. Решение задачи необходимо представить набранным в любом текстовом редакторе / или же в виде скана(фото) рукописного текста при условии разборчивого почерка и хорошего качества снимка.

1. Программно реализовать вычисление значений функции z(x) в заданном интервале *[а, b]* ( *х = а* (h) b*,* где *xi+1 =* *xi*  + *h* ).
2. Для функции (и всех составляющих ее функций построить таблицу *«Итоговые результаты*» значений для узлов с требуемой точностью.
3. Оценить полученные результаты: сравнить значения функций, вычисленные с помощью Вашей программной реализации, со значением встроенных функций.

Решение

Примем

; ;

Чтобы вычислить с точностью, необходимо чтобы

Согласно принципу равных влияний примем ()

Значит получаем

Далее

Итак, чтобы вычислить с точностью необходимо вычислить с точностью , с точностью и с точностью .

По формуле Герона, погрешность определяется по формуле для метода Ньютона решения уравнения .

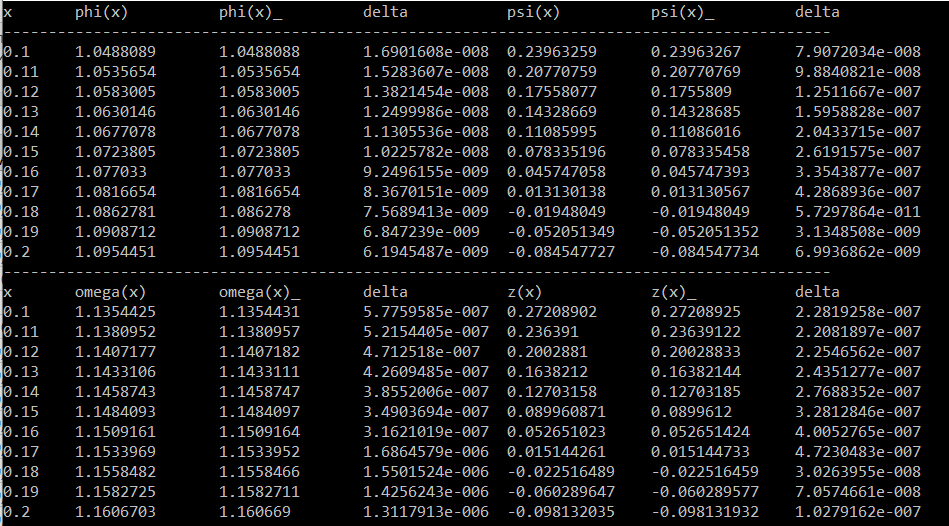


где  -- наименьшее значение  при 

 -- наибольшее значение  при 



Таблица «Итоговые результаты».



Здесь это phi(x), это phi(x)\_, это delta, далее аналогично.

В соответствии с расчётами

Таким образом, полученные теоретические погрешности не превышают заданные на всех этапах расчёта z(x).

Приложение

#include <iostream>

#include<math.h>

#include<assert.h>

#include<stdlib.h>

#include <iomanip>

using namespace std;

const double PI = 3.141592653589793;

double Mcos(double, double);

double Marctg(double, double);

double Msqrt(double, double);

double sgn(double);

double phi(double);

double psi(double);

double omega(double);

double z(double);

int main()

{

double x,eps,epsphi,epspsi,epsomega,fz[11],dfz[11];

double f1[11],f2[11],f3[11],df1[11],df2[11],df3[11];

eps=0.000001;//указанная точность

epsphi=0.432\*eps;

epspsi=0.431\*eps;

epsomega=2.083\*eps;

double h=0.01, a=0.1, b=0.2;

for(int i=0;i<=10;i++)

{

x=a+i\*h;

f1[i]=Msqrt(1+x,epsphi);//вычисление phi

df1[i]=fabs(f1[i]-phi(x));//полученная погрешность phi

f2[i]=Mcos(2.8\*x+f1[i],epspsi);

df2[i]=fabs(f2[i]-psi(x));

f3[i]=Marctg(1.5\*x+2,epsomega);

df3[i]=fabs(f3[i]-omega(x));

fz[i]=f2[i]\*f3[i];

dfz[i]=fabs(fz[i]-z(x));

}

cout << "x\t" << "phi(x)\t\t"<< "phi(x)\_\t\t" <<"delta\t\t" <<"psi(x)\t\t"<< "psi(x)\_\t\t"<< "delta\t"<<endl;

cout<<"--------------------------------------------------------------------------------------------" <<endl;

for(int i=0;i<=10;i++)

{

x=a+i\*h;

cout << x <<setprecision(8)<< "\t" << f1[i]<< "\t" <<phi(x)<<"\t"<<df1[i] << "\t"<< f2[i]<< "\t" << psi(x)<< "\t" <<df2[i] <<endl;

}

cout<<"--------------------------------------------------------------------------------------------" <<endl;

cout << "x\t" << "omega(x)\t"<< "omega(x)\_\t"<< "delta\t\t" <<"z(x)\t\t"<< "z(x)\_\t\t"<< "delta\t"<<endl;

for(int i=0;i<=10;i++)

{

x=a+i\*h;

cout << x<<setprecision(8) << "\t" << f3[i]<< "\t" <<omega(x)<<"\t"<<df3[i] << "\t"<< fz[i]<< "\t" <<z(x)<<"\t"<<dfz[i] <<endl;

}

return 0;

}

//описание функций

double Mcos(double x, double e)

{

double S=0;

double a=1;

int n=0;

while (fabs(a)>e)

{

S+=a;

n++;

a=-a\*(x\*x)/(2\*n\*(2\*n-1));

}

return S;

}

double Msqrt(double x, double e)

{

double x1,d;

double c=x;

if (x==0) return 0;

else

{

int n=0;x1=2\*x;

d=fabs(x1-x);

x=2;

while (d>sqrt(1.1\*e))

{

x1 = 0.5\*(x+c/x);

n++;

d=fabs(x1-x);

x=x1;

}

return x;

}

}

double Marctg(double x, double e)

{

double S=0;

int n=0;

if(fabs(x)<1)

{

double a=x;

while (fabs(a)>e)

{

S+=a;

n++;

a=-a\*x\*x\*(2\*n-1)/(2\*n+1);

}

}

else

{

if (x==1) S=0.25\*PI; else

{

double a=1/x;

while (fabs(a)>e)

{

S+=a;

n++;

a=-(a\*(2\*n-1)/(2\*n+1))/(x\*x);

}

S = 0.5\*PI\*sgn(x)-S;

}

}

return S;

}

double sgn(double x)

{

if (x<0) return -1; else return 1;

}

double phi(double x)

{

return sqrt(1+x);

}

double psi(double x)

{

return cos(2.8\*x+phi(x));

}

double omega(double x)

{

return atan(1.5\*x+2);

}

double z(double x)

{

return psi(x)\*omega(x);

}