**Министерство науки и высшего образования РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра ИС**

отчет

**по лабораторной работе №1**

**по дисциплине «Конструирование программ»**

Тема: Определение абсолютной и относительной погрешностей приближенных чисел. Оценка погрешностей результата

|  |  |  |
| --- | --- | --- |
| Студент гр. 8363 |  | Нерсисян А.С. |
| Преподаватель |  | Копыльцов А.В. |

Санкт-Петербург

2020

**Цель работы.**

Заработать навыки практически оценивать погрешности результата. Разумная оценка погрешности позволяет удерживать оптимальное число знаков при вычислениях, оптимизируя трудоемкость расчетов.

**Основные теоретические положения.**

**1.1 Источники и классификация погрешностей результата численного эксперимента**

Точный результат решения получить численно невозможно, он всегда содержит погрешность, то есть является приближенным. Наличие погрешности решения обусловлено следующими причинами.

1. Математическая модель всегда приближенна. Характеристики истинного процесса всегда отличаются от модельных характеристик.

2. Исходные данные всегда содержат погрешность, ибо результат эксперимента неизбежно получается с ошибкой.

3. Теоретические и численные методы решения задач являются приближенными. Лишь решение тривиальной задачи можно получить в виде конечной формулы.

4. При вводе и выводе с ЭВМ производятся округления. Такие же округления производятся и при вычислениях.

Если - точное значение вычисляемой величины, то это значение содержит следующие погрешности: - неустранимая погрешность (пункты 1 и 2), - погрешность метода (пункт 3),  - вычислительная погрешность (пункт 4). Таким образом,   Не следует думать, что  совершенно неизвестна. Конечно, она не известна точно, но ее можно оценить приближенно, адаптируя модель к изучаемому процессу  должна быть примерно на порядок меньше , а  на порядок меньше . В этом случае достигается разумный компромисс и повышается достоверность конечного результата.

## 1.2. Погрешности чисел

Пусть - точное и неизвестное значение некоторой величины, а - ее известное приближенное значение.

Ошибкой (или погрешностью) приближенного значения числа  называется разность . Количественной мерой ошибки является абсолютная погрешность

.

По ней не всегда можно сделать правильное заключение о качестве приближения. Для этого вводится понятие относительной погрешности.

Относительной погрешностью приближенного значения числа называется



Эта погрешность не зависит от масштаба величины единицы измерения. Непосредственное вычисление по формулам (1.2.1) и (1.2.2) невозможно, так как  неизвестно. Часто задают величины  верхние границы погрешностей и полагают



При записи приближенных чисел руководствуются следующими правилами. Пусть  задано в виде конечной десятичной дроби .

Значащими цифрами числа  называются все цифры в его записи, начиная с первой ненулевой слева. Например,

все значащие цифры подчеркнуты.

Значащую цифру числа  называют верной, если абсолютная погрешность числа не превосходит единицы разряда, соответствующего этой цифре.

Пусть  Тогда  имеет три верные значащие цифры (они подчеркнуты). Если бы было бы четыре верные значащие цифры. Число верных значащих цифр тесно связано с величиной относительной погрешности числа. Имеют место следующие результаты.

*Теорема 1.1.* Если  содержит  верных значащих цифр, то 

*Теорема 1.2.* Для того чтобы число  содержало  верных значащих цифр, достаточно, чтобы 

*Теорема 1.3.* Если  имеет ровно значащих цифр, то , то есть 

Эти теоремы позволяют оценивать  по числу значащих цифр и наоборот. Например, если дано , то есть  имеет шесть значащих цифр, то 

При округлении возникает погрешность, называемая погрешностью округления. Существуют два способа округления.

1. Усечение – отбрасывание всех цифр, расположенных правее - ой значащей цифры. При этом погрешность  не превышает (достигает) единицы того же разряда.

2. Округление по дополнению. Это следующее правило: если первая цифра слева от отбрасываемых меньше пяти, то лишнее просто отбрасывается, как при усечении; если же первая цифра слева от отбрасываемых больше или равна пяти, то в младший сохраняемый разряд добавляется единица. Абсолютная величина погрешности по дополнению не превышает половины единицы последней оставляемой значащей цифры.

## 1.3. Погрешности арифметических операций

*Теорема 1.4.* Абсолютная погрешность алгебраической суммы не превосходит суммы абсолютных погрешностей слагаемых, то есть 

Доказательство

Запишем формулу для абсолютной погрешности алгебраической суммы двух величин по определению и воспользуемся свойствами модуля. Получим сразу необходимый результат



*Теорема 1.5.* Пусть  - ненулевые числа одного знака. Тогда



*Доказательство*

Поступим так же, как и в предыдущей теореме. Выразим абсолютную погрешность через относительную:





Из последних двух равенств видно, что при вычитании двух близких чисел может произойти катастрофическая потеря точности, так как при  в последней формуле 

*Теорема 1.6*. Для относительных погрешностей произведения и частного приближенных чисел верны следующие оценки:





Доказательство

Первая требуемая формула получается традиционным путем:



Для вывода второй оценки предварительно получим, используя свойства модуля, следующую формулу:  Тогда



*Следствие.*Если , то для оценки границ относительных погрешностей можно использовать следующие приближенные равенства  Чаще всего на практике делают именно так.

## 1.4. Погрешности функций

Пусть  - функция - переменных, дифференцируемая в рассматриваемой области (например, на отрезке ).

*Теорема 1.7.* Для абсолютной погрешности значения  справедлива следующая формула: . (1.4.1)

Доказательство

Вспомним сначала формулу Тейлора для функции нескольких переменных. Для функции одного переменного разложение в окрестности точки  будет иметь вид



Для функции  переменных форма записи формулы Тейлора остается точно такой же, если вместо производных записать дифференциалы соответствующих порядков:



где, . Например, для функции двух переменных





Отбрасывая все члены второго порядка и выше, получим



Таким образом, искомая формула сразу вытекает из формулы Лагранжа. Если  достаточно мало, то для предельных значений погрешностей можно положить



Для относительных погрешностей тогда имеем следующие формулы:



**I частный случай.** Функция  - функция одного переменного. Здесь следует положить , тогда **.** Для относительных погрешностей все аналогично:

.

**II частный случай.** Функция  - неявная. Этот случай отличается от исходного только формулой для нахождения частных производных:



## 1.5. Особенности машинной арифметики

Причиной появления вычислительных погрешностей является способ представления чисел на ЭВМ. В привычной нам десятичной системе счисления вещественное число  представляется последовательностью символов, начиная со знака , и продолжается цепочкой десятичных цифр , то есть



Так как десятичная система счисления позиционная, то значение числа вычисляется так:

 Все вычислительные машины работают в двоичной системе счисления. В ней то же число выглядит следующим образом:

.

По форме представления, способу хранения и реализации вычислительных операций на ЭВМ целые и вещественные числа существенно различаются.

**Целые** представляются так: , где  - фиксированное число. Всего для хранения числа  отводят  разряда, в том числе один разряд на знак числа. Таким образом,   обычно не слишком велико, например, для ЭВМ РС  Тогда 

Арифметические операции сложения, вычитания и умножения производятся точно, если результат меньше или равен . Если же это не так, то ситуация не доводится до сведения пользователя, а результату присваивается некоторое значение меньшее абсолютной величины .

**Вещественные числа** представляются в форме с плавающей точкой, то есть в виде , где обычно - размерность мантиссы, - двоичный порядок. Порядоктакже записывается как двоичное целое число , на его хранение отводятся  двоичных разрядов . Следует помнить, что на ЭВМ представимы не все числа, а лишь конечный набор рациональных чисел. Это – представимое множество данной вычислительной машины. Для всех остальных чисел возможно лишь их приближенное представление с ошибкой представления (ошибкой округления). Если округление производится усечением, то , если имеет место округление по дополнению, то , называется машинной точностью или машинным эпсилон.

Диапазон изменения чисел в ЭВМ ограничен. Если , то  Если же , то 

  0  

-∞ 0 ∞

Для ЭВМ РС диапазон представления вещественных чисел примерно равен: при этом 

**Экспериментальные результаты.**

**Задание № 1**

Величина подъемной силы крыла самолета оценивается по формуле где - площадь проекции крыла на горизонтальную плоскость,  - скорость натекания воздуха на крыло, - плотность атмосферы на заданной высоте, - угол атаки, отсчитываемый от направления нулевой подъемной силы, - коэффициент, зависящий от формы крыла. Требуется вычислить  при заданных значениях и заданных абсолютных или относительных значениях этих величин. (В таблице вариантов задано ).

**Дано:**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Вариант** |  |  |  |  |  |  |  |  |  |  |
| 11 | 0.005 | 0.001 | 15 | 0.01 | 0.95 | 0.01 | 200 | 3 | 20 | 0.001 |

**Задание № 2.**

Найти абсолютную погрешность вычисления функции при заданных значениях аргументов.

**Дано:**

#### 11.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **1** | **2** | **3** |
|  | 52.34÷0.01 | 0.0545÷0.00005 | 0.65÷0.02 |
|  | 2.0435÷0.0001 | 0.82÷0.01 | 6.3÷0.02 |

**Обработка результатов эксперимента.**

**Задание № 1. решение:**

Относительная погрешность функции равна

, абсолютная

Так как относительная погрешность велика, то значение функции следует вычислять не более чем с двумя – тремя знаками (с запасом), то есть

Отсюда

Поскольку то результат целесообразно округлить до двух знаков. Окончательно,

**Задание № 2. решение:**

Абсолютные погрешности аргументов (1) Относительные погрешности будут равны:



По формуле



получим

где

|  |  |  |  |
| --- | --- | --- | --- |
|  | **1** | **2** | **3** |
|  | 52.34÷0.01 | 0.0545÷0.00005 | 0.65÷0.02 |
|  | 2.0435÷0.0001 | 0.82÷0.01 | 6.3÷0.02 |
|  | 1.99392 | 0.00440765 | 0.0105886 |
|  | 0.00607883 | 1.99559 | 1.98941 |
|  | 4316.28 | 2.11707 | 125.353 |
|  | 0.000381253% | 9.84977e-05% | 9.93762e-05% |
|  | 1.64559 | 0.000208527 | 0.0124572 |

Само значение функции указана в таблице. Следует округлить значение функции до трех значащих цифр. Итак в результате получаем следующее:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 4316.28 | 2.117 | 125.353 |
|  | 0.038% | 0.01% | 0.01% |
|  | 1.6456 | 0.0002 | 0.0125 |

**Выводы.**

В результате выполнения данной лабораторной работы был оценен степень соответствия полученных результатов расчетов и экспериментов с теоретическими данными. Научились практически оценивать погрешности результата. Разумная оценка погрешности позволяет удерживать оптимальное число знаков при вычислениях, оптимизируя трудоемкость расчетов.

ПРИЛОЖЕНИЕ А

**Исходный код программы по заданию 1:**

#include <iostream>

#include <math.h>

double force(double\* mass) {

double ans = 1;

for (size\_t i = 0; i < 5; i++) {

i == 3 ? ans\*= pow(mass[i],2) : ans \*= mass[i];

}

return ans;

}

// функция для получения целой части и степени десятки

//передаем число и требуемый степень десятки, т.е. пов = 1 - значит будет 0,1-0,9 \*10 в какой то степени

//будет pow = 10 -- значит будет 1-9 \*10 в какой то степени

//будет pow = 100 -- значит будет 10-99 \*10 в какой то степени

//будет pow = 1000 -- значит будет 100-999 \*10 в какой то степени и.т.д

int GetPow(double y, int pow)

{

double x = fabs(y);

int n = 0;

if (x >= pow)

while (x >= pow)

{

n++;

x /= 10;

}

else if (x < 1)

while (x < 1)

{

n--;

x \*= 10;

}

return n;

}

int main() {

setlocale(LC\_ALL, "RU");

std::cout << "Программа подчета величины подъемной силы крыла самолета." << std::endl;

double TableDataForce[] = { 0.005, 15, 0.95, 200, 20 }; // coeff; alpha; ro; v; S;

double relativeErrTable[] = { 0.001, 0.01, 0.01, 3, 0.001 }; // from table

//рассчитаем относительную погрешность.

double relativeErrors = 0;

for (size\_t i = 0; i < 5; i++) {

i == 3 ? relativeErrors += 2 \* relativeErrTable[i]/TableDataForce[i] : relativeErrors += relativeErrTable[i];

}

double Force = force(TableDataForce);

int degree = GetPow(Force, 1000);

Force = Force / pow(10, degree);

std::cout << "Величина подъемной силы = " << Force << " \* 10^" << degree << std::endl;

double absoluteErrors = fabs(Force\*pow(10, degree)) \* relativeErrors;

int absoluteErrDeg = GetPow(absoluteErrors, 100);

// info from math.h libriary :)

// round() -- математически правильное округление

// ceil() -- округляет в сторону большего

// floor() -- в сторону меньшего

// trunc() -- простое отбрасывание дробной части

absoluteErrors = round(absoluteErrors / (pow(10, absoluteErrDeg)));

std::cout << "Абсолютная погрешность = " << absoluteErrors << " \* 10^" << absoluteErrDeg << std::endl;

double relativeErrPercent = relativeErrors \* 100;

std::cout << "Относительная погрешность = " << relativeErrPercent << "%" << std::endl;

std::cout << std::endl << "Оканчательный результат:" << std::endl;

int degree2 = GetPow(Force \* (pow(10, degree)), 10);

Force = Force \* pow(10, degree) / pow(10, degree2);

std::cout << "Величина подъемной силы = " << Force << " \* 10^" << degree2 << std::endl;

int absoluteErrDeg2 = GetPow(absoluteErrors\*(pow(10, absoluteErrDeg)), 1);

absoluteErrors = absoluteErrors \* pow(10, absoluteErrDeg) / (pow(10, absoluteErrDeg2));

std::cout << "Абсолютная погрешность = " << absoluteErrors << " \* 10^" << absoluteErrDeg2 << std::endl;

std::cout << "Относительная погрешность = " << relativeErrPercent << "%" << std::endl;

std::cin.get();

return 0;

}

ПРИЛОЖЕНИЕ Б

**Исходный код программы по заданию 2:**

#include <iostream>

#define \_USE\_MATH\_DEFINES

#include <math.h>

#include <cmath>

int main()

{

setlocale(LC\_ALL, "RU");

std::cout << "Программа подчета значения функции." << std::endl;

double data[3][2][2] = {

{ {52.34, 2.0435},

{0.01, 0.0001} },

{ {0.0545, 0.82},

{0.00005, 0.01} },

{ {0.65, 6.3},

{0.02, 0.02} },

};

std::cout << "Абсолютные погрешности аргументов:" << std::endl;

for (size\_t i = 0; i < 3; i++) {

for (size\_t j = 0; j < 2; j++) {

std::cout << data[i][1][j] << "\t";

}

std::cout << std::endl;

};

std::cout << std::endl << "Относительные погрешности равны:" << std::endl;

double relative[6];

for (size\_t i = 0, k = 0; i < 3; i++) {

for (size\_t j = 0; j < 2; j++) {

relative[k] = data[i][1][j] / data[i][0][j];

k++;

}

}

double relativePercent[6];

for (size\_t i = 0; i < 6; i++) {

//relativePercent[i] = relative[i]\* 100;

relativePercent[i] = round(relative[i] \*100\*1000)/1000;

}

for (size\_t i = 0; i < 6; i++) {

std::cout << relativePercent[i] << "%" << (i % 2 == 1 ? "\n" : "\t");

}

std::cout << std::endl << "Значение функции:" << std::endl;

double SFuncResult[3];

for (size\_t i = 0; i < 3; i++) {

SFuncResult[i] = (M\_PI\*pow(data[i][0][0], 2))/2+ (M\_PI \* pow(data[i][0][1], 2));

}

for (size\_t i = 0; i < 3; i++) {

std::cout << SFuncResult[i] << std::endl;

}

std::cout << std::endl << "Коеффиценты:" << std::endl;

double diff[2][3];

for (size\_t i = 0; i < 3; i++) {

diff[0][i] = fabs(data[i][0][0]) / fabs(SFuncResult[i]) \* M\_PI \* data[i][0][0];

}

for (size\_t i = 0; i < 3; i++) {

diff[1][i] = fabs(data[i][0][1]) / fabs(SFuncResult[i]) \* 2 \* M\_PI \* data[i][0][1];

}

for (size\_t i = 0; i < 2; i++) {

for (size\_t j = 0; j < 3; j++) {

std::cout << diff[i][j] << "\t\t\t";

}

std::cout << std::endl;

};

//std::cout << std::endl << "delta(S):" << std::endl;

double deltaS[3], deltaSPercent[3];

for (size\_t i = 0, k = 0; i < 3; i++) {

deltaS[i] = diff[0][i] \* relative[k] + diff[1][i] \* relative[k + 1];

}

for (size\_t i = 0; i < 3; i++) {

deltaSPercent[i]= round(deltaS[i] \* 100 \* 1000) / 1000;

}

std::cout << std::endl << "DELTA S:" << std::endl;

double DELTA\_S[3];

for (size\_t i = 0; i < 3; i++) {

DELTA\_S[i] = fabs(SFuncResult[i]) \* deltaS[i];

}

for (size\_t i = 0; i < 3; i++) {

std::cout << DELTA\_S[i] << "\t";

}

std::cout << std::endl << std::endl << "Значение функции округленное до трех значащих цифр:" << std::endl;

for (size\_t i = 0; i < 3; i++) {

SFuncResult[i] = round(SFuncResult[i] \* 1000) / 1000;

}

for (size\_t i = 0; i < 3; i++) {

std::cout << SFuncResult[i] << std::endl;

}

std::cout << std::endl << "DELTA S округленное до трех значащих цифр::" << std::endl;

for (size\_t i = 0; i < 3; i++) {

DELTA\_S[i] = round(DELTA\_S[i] \* 10000) / 10000;

}

for (size\_t i = 0; i < 3; i++) {

std::cout << DELTA\_S[i] << "\t";

}

std::cout << std::endl << std::endl << "delta(S):" << std::endl;

for (size\_t i = 0; i < 3; i++) {

std::cout << deltaSPercent[i] << "%" << "\t";

}

std::cin.get();

//std::cout << "" << std::endl;

return 0;

};

Протокол

1. Задали формулу для подсчёта относительной погрешности функции и вычислили.
2. Рассчитали значение самой функции, а также её абсолютную погрешность.
3. Произвели округление результатов.
4. Рассчитали относительные погрешности аргументов исходной функции.
5. Вычислили относительные погрешности функции по формуле, приведённой в основных теоретических положениях.
6. Рассчитали значения функции и её абсолютные погрешности.