Цель лабораторной работы: изучение эффекта потери точности компьютерных вычислений при выполнении арифметических операций с числами, сильно отличающихся друг от друга.

Задание №1

Демонстрация вычислительного дефекта арифметики с плавающей точкой - потери точности при работе с разномасштабными величинами.

Для демонстрации вычислительного дефекта арифметики с плавающей точкой были проведены операции скалярного сложения с числами разномасштабных порядков.

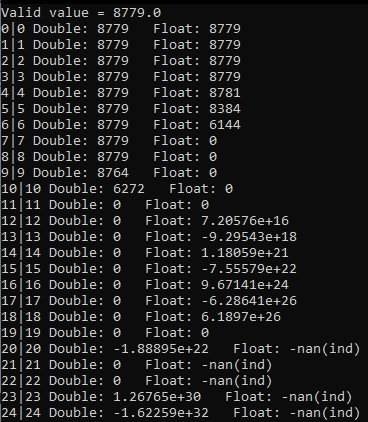


Рисунок 1 – Результаты тестирования

Вектора для тестирования заданы следующим образом:

,

,

где α – первое число до разделительного знака , представленное на рисунке 1

β – второе число после разделительного знака, представленное на рисунке 2

По результатам тестирования был построен график, отображающий процент ошибки при увеличении порядка расчетов. График представлен на рисунке 2.

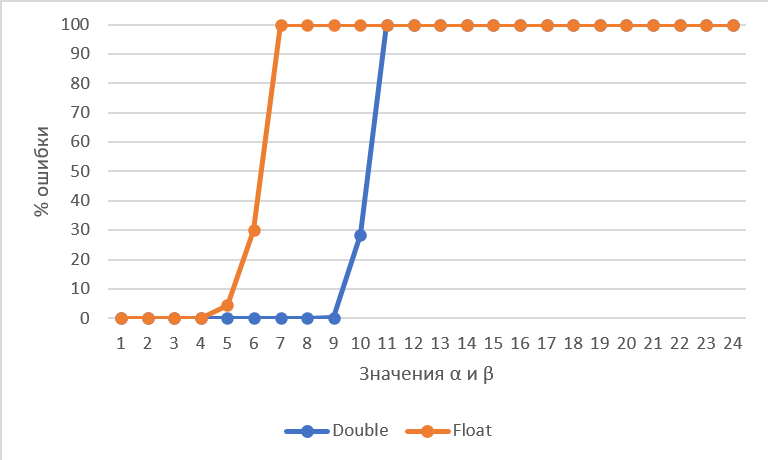


Рисунок 2 – График увеличения процента ошибки расчета

Исходя из графика, можно сделать выводы, что при выборе системы с одинарной точностью погрешность результат резко увеличивается при α и β равным и большим 5, а для системы с двойной точностью погрешность увеличивается при α и β равным и большим 9.

Задание №2

Решение квадратного уравнения



Подобрать коэффициенты квадратного уравнения сильно отличающимися друг от друга, чтобы вычисленное значение корней существенно отличалось от точного. Заполнить таблицу при каких коэффициентах это достигается.

Коэффициентов a, b, c были подобраны следующим образом: при каждой итерации коэффициенты a и c увеличивалось на 1, b увеличивалось на 1000. Начальные значения a, b, c равны 1, 2000 и -25 соответственно. Результаты подбора представлены на рисунке 3.

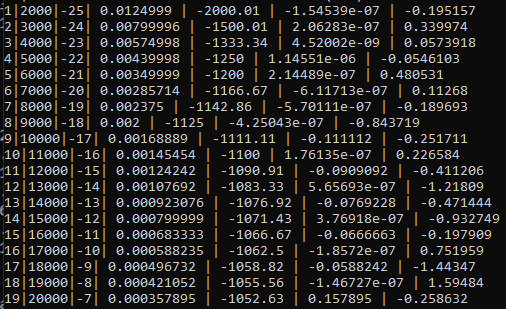


Рисунок 3 – Подбор коэффициентов

На рисунке 3 представлены результаты подбора коэффициентов, исходя из которых, можно сделать вывод о том, что точность вычисления результата квадратного уравнения понижается при различии в порядке коэффициентов.

Задание №3

Вычислить машинное эпсилон для формата с плавающей точкой одинарной точности и двойной точности.

На рисунке 4 представлены результаты вычисления машинного эпсилон.

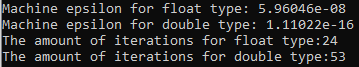


Рисунок 4 – Результаты вычисления машинного эпсилон

Выводы

В ходе лабораторной работы был изучен эффект потери точности компьютерных вычислений при выполнения арифметических операций с плавающей точкой с одинарной и двойной точностью. Вычисления в ходе лабораторной работы были выполнения при помощи средств программного языка C++.

Исходя из первого задания, можно сделать выводы:

* при выборе системы с одинарной точностью погрешность результат резко увеличивается при α и β равным и большим 5
* при выборе системы с двойной точностью погрешность резко увеличивается при α и β равным и большим 9.

Исходя из второго задания можно сделать вывод о том, что точность вычисления результата квадратного уравнения понижается при различии в порядке коэффициентов.

Результатом третьего задания является вычисленное программным методом эпсилон для систем с одинарной и двойной точностью.

ПРИЛОЖЕНИЕ А. ЛИСТИНГ КОДА

|  |
| --- |
| #include <iostream>  #include <vector>  #include <cmath>  #include <string>  using namespace std;  class Expression {  int a, b, c;  float x1, x2, D;  public:  Expression(int \_a, int \_b, int \_c) {  a = \_a, b = \_b, c = \_c;  D = pow(b, 2) - 4 \* a \* c;  }  void calculate() {  if (D < 0) {  return;  }  x1 = (-b + pow(D, 0.5)) / (2 \* a);  x2 = (-b - pow(D, 0.5)) / (2 \* a);  }  void getResult() {  if (D < 0) {  cout << "D less then zero" << endl;  return;  }  cout << a << "|" << b << "|" << c << "| " << x1 << " | " << x2 << " | ";  cout << a \* pow(x1, 2) + b \* x1 + c << " | " << a \* pow(x2, 2) + b \* x2 + c << endl;  //string resultString = to\_string(a) + "|" + to\_string(b) + "|" + to\_string(c) + "| " + to\_string(x1) + " | " + to\_string(x2) + " | ";  //resultString += to\_string(a \* pow(x1, 2) + b \* x1 + c) + " | ";  //resultString += to\_string(a \* pow(x2, 2) + b \* x2 + c);  }  };  template<typename T>  T calculatePoints(vector<T>& vec1, vector<T>& vec2) {  vector<T> resultVector;  T result = 0;  for (int i = 0; i < vec1.size(); i++) {  resultVector.push\_back(vec1[i] \* vec2[i]);  }  for (auto& item : resultVector) {  result += item;  }  return result;  }  void printTask1(int alpha, int beta) {  //double  vector<double> vec1 = {  pow(10, alpha),  1223,  pow(10, alpha - 1),  pow(10, alpha - 2),  3,  -pow(10, alpha - 5)  };  vector<double> vec2 = {  pow(10, beta),  2,  -pow(10, beta + 1),  pow(10, beta),  2111,  pow(10, beta + 3)  };  //float  vector<float> vec3 = {  (float)pow(10, alpha),  1223,  (float)pow(10, alpha - 1),  (float)pow(10, alpha - 2),  3,  -(float)pow(10, alpha - 5)  };  vector<float> vec4 = {  (float)pow(10, beta),  2,  -(float)pow(10, beta + 1),  (float)pow(10, beta),  2111,  (float)pow(10, beta + 3)  };  cout << "Double: " << calculatePoints(vec1, vec2);  cout << " Float: " << calculatePoints(vec3, vec4) << endl;  }  void printTask2(int a, int b, int c) {  Expression myExpr(a, b, c);  myExpr.calculate();  myExpr.getResult();  }  void printTask3() {  int i = 0;  float epsilon\_f = 1.0;  while (1.0f + epsilon\_f > 1.0f)  {  epsilon\_f = epsilon\_f / 2.0f;  i++;  }  int j = 0;  double epsilon\_d = 1.0;  while (1.0 + epsilon\_d > 1.0)  {  epsilon\_d = epsilon\_d / 2.0;  j++;  }  cout << "Machine epsilon for float type: " << epsilon\_f << "\nMachine epsilon for double type: " << epsilon\_d << endl;  cout << "The amount of iterations for float type:" << i << "\nThe amount of iterations for double type:" << j;  }  int main()  {  cout << "Valid value = 8779.0" << endl;  for (int i = 0; i < 25; i++) {  cout << to\_string(i) + "|" + to\_string(i) + " ";  printTask1(i, i);  }  for (int i = 1, j = 2000, k = -25; i < 20; i+=1, j+=1000, k+=1) {  printTask2(i, j, k);  }  printTask3();  } |