

Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное

образовательное учреждение высшего образования

Санкт-Петербургский университет аэрокосмического приборостроения

КАФЕДРА № 2

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| доц., канд. техн. наук |  |  |  | С.Л. Козенко |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ № 1 |
| Нелинейные уравнения Вариант 5 |
| по дисциплине: ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4128 |  | 25.02.2023 |  | В.А. Воробьёв |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт – Петербург, 2023

|  |
| --- |
| Нелинейные уравнения Вариант 5 |
| по дисциплине: ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4128 |  | 25.02.2023 |  | В.А. Воробьёв |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт – Петербург, 2023

**СОДЕРЖАНИЕ**

[1 Цели и постановка задачи 3](#_Toc127557461)

[1.1 Цели работы 3](#_Toc127557462)

[1.2 Задание 3](#_Toc127557463)

[2 Описание метода решения 4](#_Toc127557464)

[3 Аналитические расчеты 5](#_Toc127557465)

[4 Схема алгоритма решения задачи 9](#_Toc127557466)

[5 Текст программы на C++ 13](#_Toc127557467)

[6 Результаты программных расчетов 14](#_Toc127557468)

[7 Сравнение программных и аналитических расчётов 16](#_Toc127557469)

[8 Выводы 18](#_Toc127557470)

1. **Цели и постановка задачи**

**1.1 Цели работы**

а) освоение основных методов решения систем линейных алгебраических уравнений (СЛАУ); б) Совершенствование навыков по алгоритмизации и программированию вычислительных задач.

б) совершенствование навыков по алгоритмизации и программированию вычислительных задач.

**1.2 Задание**

Составить схему алгоритма и программу на языке C/C++ решения задачи по теме «Решение СЛАУ» в соответствии с индивидуальным заданием.

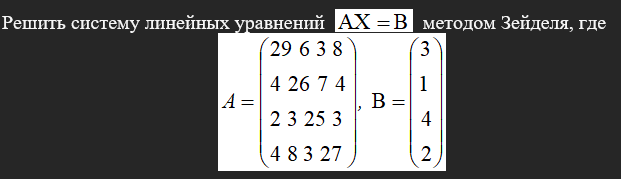


Рисунок 1 – Вариант задания

**2 Описание метода решения**

Итерационные методы позволяют получить значения корней системы с заданной точностью в виде предела последовательности некоторых векторов C(0), C(1), …, C(k) . Процесс получения элементов такой последовательности носит итерационный (повторяющийся) характер [4]. Эффективность применения таких методов зависит от удачного выбора начального вектора C(0) и быстроты сходимости процесса.

Метод Зейделя относится к итерационным методам решения систем линейных уравнений, обеспечивающим хорошую сходимость итерационного процесса поиска корней системы. 34 Пусть задана система (3.2). Прежде всего необходимо задать значения начальных (нулевых) приближений корней С1 (0), С2 (0),..., Сm(0). Если отсутствует какая–нибудь информация об этих значениях, то их можно принять равными свободным членам системы уравнений (3.2) или даже принять равными нулю. Выбранные таким образом нулевые приближения корней подставим в первое уравнение системы (3.2) и получим первое приближение корня С1.



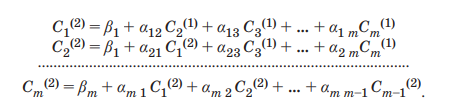
Используя во втором уравнении системы (3.2) найденное первое приближение корня С1 и нулевые приближения остальных корней, получим первое приближение корня С2

****

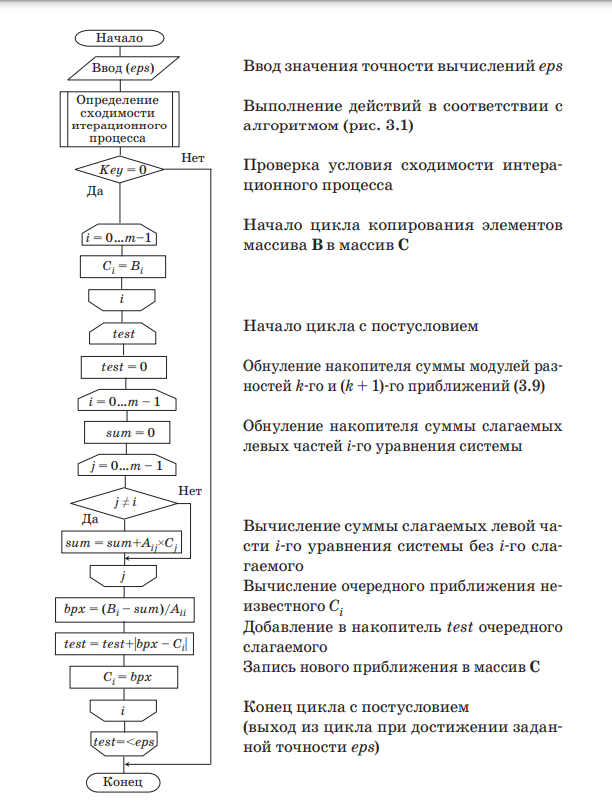
Повторяя эту процедуру последовательно для всех уравнений системы (3.2), получим в итоге первое приближение корня.

****

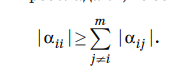
Используя первые приближение корня системы, можно аналогичным образом найти вторые приближения.



Затем, используя вторые приближения, можно вычислить третьи и т.д. Итерационный процесс решения системы линейных уравнений методом Зейделя сходится к единственному решению при любом выборе начальных приближений искомых корней, если выполняется одно из условий (3.5).

****

Условия сходимости также выполняются, если в матрице A диагональные элементы преобладают, то есть



Другими словами, модули диагональных коэффициентов в каждом уравнении системы больше суммы модулей недиагональных коэффициентов (свободные члены не рассматриваются). Замечание. Для итерационного метода Зейделя достаточно, чтобы хотя бы для одного i неравенство было строгим.

**3 Аналитические расчеты**

**4 Схема алгоритма решения задачи**

**5 Текст программы на C++**

Исходный код доступен на GitHub: https://github.com/vladcto/SUAI\_homework/blob/a635fedd450ca83705278b949d5cc37749177647/4\_semester/ComputationalMathematics/program.cpp

#include <iostream>

#include <cmath>

#include <functional>

using namespace std;

double dichotomy(std::function<double(double)> f ,double a, double b, double epsilon) {

double c;

int step = 0;

cout << "Начало нахождения решения методом половинного деления." << endl;

while (abs(a - b) > epsilon) {

c = (a + b) / 2;

cout << "Шаг " << step++ << ": f(" << c << ") = " << f(c) << endl;

if (f(c) == 0) return c;

f(a)\* f(c) < 0 ? b = c : a = c;

}

return (a + b) / 2;

}

int main() {

setlocale(LC\_ALL, "RUS");

double start, end, epsilon, a , b;

cout << "Для уравнения f(x) = sqrt(a - x \* x) + b \* x \* x введите," << endl;

cout << "a = ";

cin >> a;

cout << "b = ";

cin >> b;

auto fx = [&](double x) {

return sqrt(a - x \* x) + b \* x \* x;

};

cout << "Введите начало промежутка: ";

cin >> start;

cout << "Введите конец промежутка: ";

cin >> end;

cout << "Введите точность: ";

cin >> epsilon;

cout << "Решением является: " << dichotomy(fx, start, end, epsilon) << endl;

return 0;

}

**6 Результаты программных расчетов**

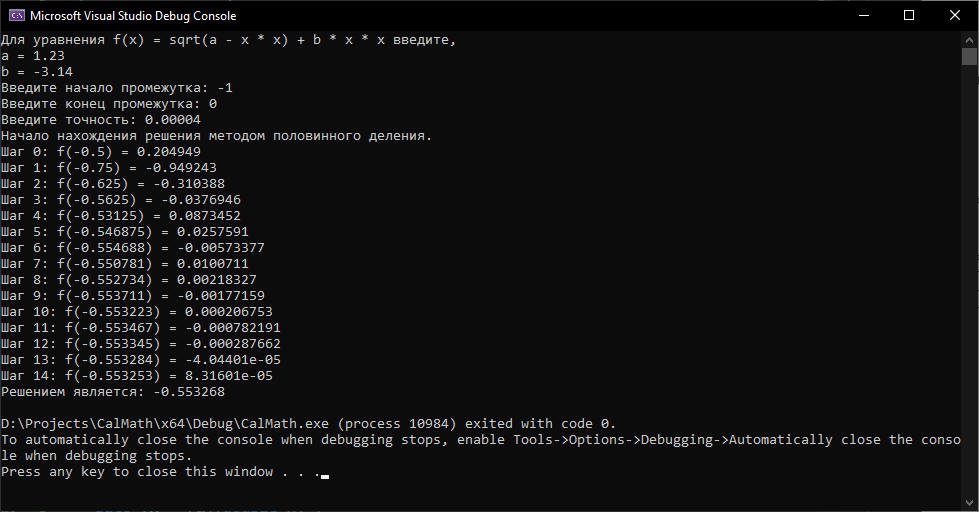


Рисунок 10 – Результат работы программы с предоставленными исходными данными

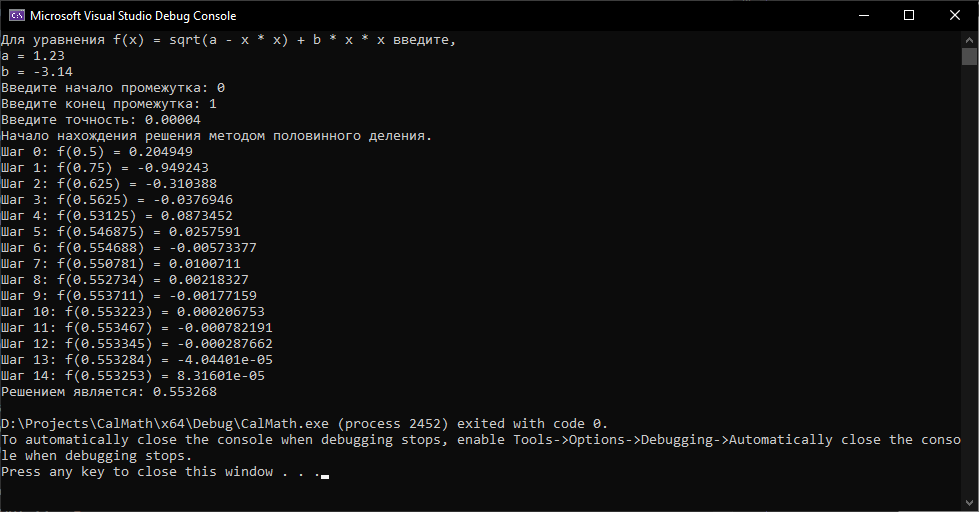


Рисунок 11 – Результат работы программы с произвольными исходными данными

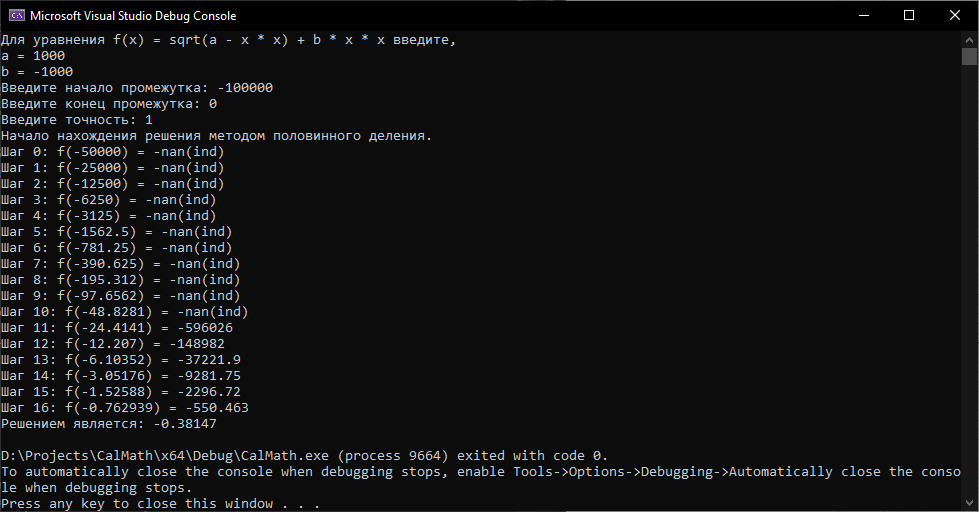


Рисунок 12 – Результат работы программы с произвольными исходными данными



Рисунок 13 – Пример работы алгоритма для корня -0.5533

**7 Сравнение программных и аналитических расчётов**

**8 Выводы**

В ходе выполнения лабораторной работы были составлены схема алгоритма и программа на языке C++, решающая поставленную задачу в соответствии с вариантом. Результаты при аналитических и программных расчётах оказались одинаковы – итог и все промежуточные значения оказались равны.

В результате вычислений были получены два корня, что соответствует графику:

х1 = -0.5533;

х2 = 0.5533.

В результате выполнения работы был освоен метод дихотомии для решения нелинейных уравнений. Этот метод основан на последовательном разделение отрезка на два одинаковых, что схоже с алгоритмом бинарного поиска. Также в результате выполнения работы были усовершенствованы навыки по алгоритмизации и программированию вычислительных задач.

**Приложение**

**Листинг программы**

#include <iostream>

#include <cmath>

#include <vector>

#include <string>

using namespace std;

bool convergence(vector<vector<double>> a) {

if (a.size() != a[0].size()) return false;

double sum = 0;

for (int i = 0; i < a.size(); i++) {

for (int j = 0; j < a.size(); j++) {

if (i == j) continue;

sum += abs(a[i][j]);

}

if (abs(a[i][i]) > sum) return true;

sum = 0;

}

return false;

}

vector<double> seindel(vector<vector<double>> a,

vector<double> b,

double epsilon) {

if (!convergence(a)) {

cout << "SLAE does not converge.";

return vector<double>();

}

vector<double> x(a.size(), 0), prev\_x = x;

int iteration = 1;

double error = epsilon + 1;

while (error >= epsilon)

{

// Seindel method.

for (int i = 0; i < a.size(); i++)

{

prev\_x[i] = x[i];

double sum = 0;

for (int j = 0; j < a.size(); j++)

{

if (j != i)

{

sum += a[i][j] \* x[j];

}

}

x[i] = (b[i] - sum) / a[i][i];

}

// Calculate error.

error = 0;

for (int i = 0; i < a.size(); i++)

{

error += abs(x[i] - prev\_x[i]);

}

cout << iteration++ << ": " << error << endl;

}

return x;

}

vector<double> split\_s(string inp) {

vector<double> res;

string token = "";

inp += " ";

for (int i = 0; i < inp.size(); i++) {

if (inp[i] == ' ') {

res.push\_back(stod(token));

token.clear();

}

else {

token += inp[i];

}

}

return res;

}

int main()

{

vector<vector<double>> a;

vector<double> b;

double epsilon;

//User data input.

{

double size;

string inp;

cout << "Enter size: ";

cin >> size;

cout << "Enter matrix: " << endl;

getline(cin, inp);

for (int i = 0; i < size; i++) {

cout << i << " : ";

getline(cin, inp);

a.push\_back(split\_s(inp));

}

cout << "Enter b:" << endl;

getline(cin, inp);

b = split\_s(inp);

cout << "Enter epsilon: ";

cin >> epsilon;

}

vector<double> x = seindel(a, b, epsilon);

if (x.empty()) {

cout << "SLAE solutions is empty.";

return 0;

}

cout << "Solution: ";

for (int i = 0; i < 4; i++)

{

cout << x[i] << " ";

}

cout << endl;

return 0;

}