Московский авиационный институт
(национальный исследовательский университет)

Институт №8 «Информационные технологии и прикладная математика»

Кафедра 806 «Вычислительная математика и программирование»

Лабораторные работы по курсу «Численные методы»

Студент:	Г. С. Будайчиев
Преподаватель:	Д. Е. Пивоваров
Группа:	М8О-303Б-21
Дата:	
Оценка:	
Подпись:	

Москва, 2024

2.1 Методы простой итерации и Ньютона

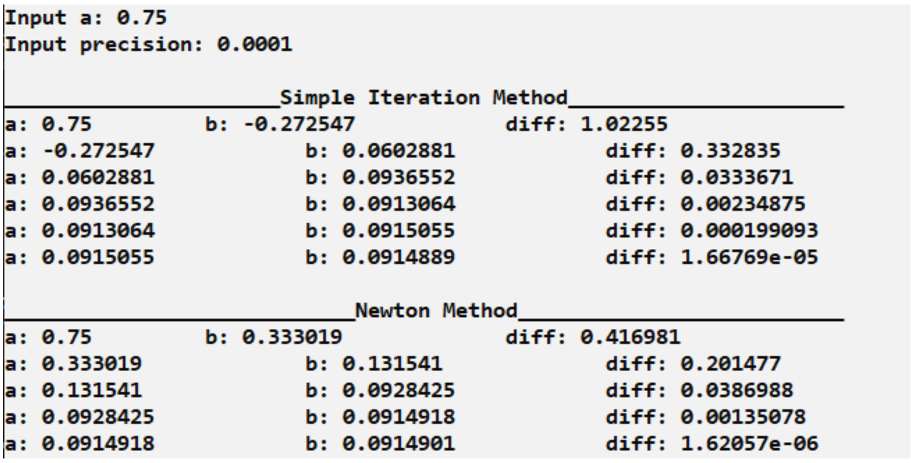
1 Постановка задачи

Реализовать методы простой итерации и Ньютона решения нелинейных уравнений в виде программ, задавая в качестве входных данных точность вычислений. С использованием разработанного программного обеспечения найти положительный корень нелинейного уравнения (начальное приближение определить графически). Проанализировать зависимость погрешности вычислений от количества итераций.

Вариант: 3

$\sqrt{1-x^2} - e^x + 0.1 = 0$

2 Результаты работы



```
Input a: 0.75
Input precision: 0.0001

______________________Simple Iteration Method______________________
a: 0.75         b: -0.272547            diff: 1.02255
a: -0.272547            b: 0.0602881            diff: 0.332835
a: 0.0602881            b: 0.0936552            diff: 0.0333671
a: 0.0936552            b: 0.0913064            diff: 0.00234875
a: 0.0913064            b: 0.0915055            diff: 0.000199093
a: 0.0915055            b: 0.0914889            diff: 1.66769e-05

____________________________Newton Method___________________________
a: 0.75         b: 0.333019             diff: 0.416981
a: 0.333019             b: 0.131541             diff: 0.201477
a: 0.131541             b: 0.0928425            diff: 0.0386988
a: 0.0928425            b: 0.0914918            diff: 0.00135078
a: 0.0914918            b: 0.0914901            diff: 1.62057e-06
```

Рис. 1: Вывод программы в консоли

3 Исходный код

```
#include <iostream>
#include <cmath>
using namespace std;

double iterationalF(double x)
{
  return log(sqrt(1 - x*x) + 0.1);
}

double simpleIteration(double a, const double precision)
{
  cout << "\n______________________Simple Iteration Method______________________";
  double b = iterationalF(a);
  double diff = abs(b - a);
  cout << "\na: " << a << "\t\tb: " << b << "\t\tdiff: " << diff;
  a = b;

  while (precision < diff)
  {
    b = iterationalF(a);
    diff = abs(b - a);
    cout << "\na: " << a << "\t\tb: " << b << "\t\tdiff: " << diff;
    a = b;
  }

  return b;
}

double F(double x)
{
  return (sqrt(1 - x * x) - pow(M_E, x) + 0.1);
}

double dF(double x)
{
  return ((-x / sqrt(1 - x * x)) - pow(M_E, x));
}

double Newton(double a, const double precision)
{
  cout << "\n____________________________Newton Method__________________________";
  double b = a - F(a)/dF(a);
  double diff = abs(b - a);
  cout << "\na: " << a << "\t\tb: " << b << "\t\tdiff: " << diff;
  a = b;

  while (precision < diff)
  {
    b = a - F(a) / dF(a);
    diff = abs(b - a);
    cout << "\na: " << a << "\t\tb: " << b << "\t\tdiff: " << diff;
    a = b;
  }

  return b;
}

int main()
{
  double a;
  cout << "Input a: ";
  cin >> a;

  double precision;
  cout << "Input precision: ";
  cin >> precision;

  simpleIteration(a, precision);

  cout << ’\n’;

  Newton(a, precision);
}
```

2.2 Методы простой итерации и Ньютона

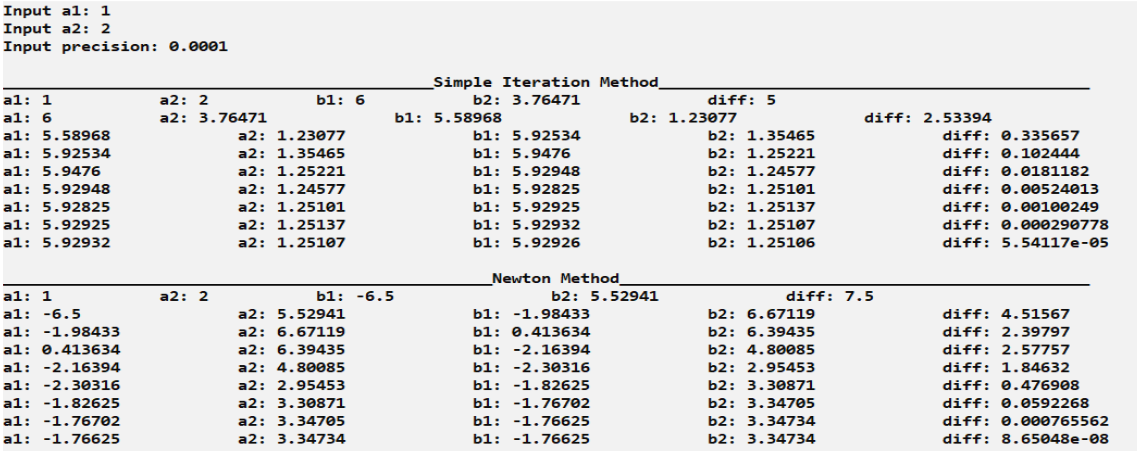
4 Постановка задачи

Реализовать методы простой итерации и Ньютона решения систем нелинейных уравнений в виде программного кода, задавая в качестве входных данных точность вычислений. С использованием разработанного программного обеспечения решить систему нелинейных уравнений (при наличии нескольких решений найти то из них, в котором значения неизвестных являются положительными); начальное приближение определить графически. Проанализировать зависимость погрешности вычислений от количества итераций.

Вариант: 3

$$\begin{cases} (x_1^2+16)x_2-64=0 \\ (x_1-2)^2+(x_2-2)^2-16=0 \end{cases}$$

5 Результаты работы



```
Input a1: 1
Input a2: 2
Input precision: 0.0001

____________________________________________Simple Iteration Method____________________________________________
a1: 1           a2: 2           b1: 6           b2: 3.76471             diff: 5
a1: 6           a2: 3.76471             b1: 5.58968             b2: 1.23077             diff: 2.53394
a1: 5.58968             a2: 1.23077             b1: 5.92534             b2: 1.35465             diff: 0.335657
a1: 5.92534             a2: 1.35465             b1: 5.9476              b2: 1.25221             diff: 0.102444
a1: 5.9476              a2: 1.25221             b1: 5.92948             b2: 1.24577             diff: 0.0181182
a1: 5.92948             a2: 1.24577             b1: 5.92825             b2: 1.25101             diff: 0.00524013
a1: 5.92825             a2: 1.25101             b1: 5.92925             b2: 1.25137             diff: 0.00100249
a1: 5.92925             a2: 1.25137             b1: 5.92932             b2: 1.25107             diff: 0.000290778
a1: 5.92932             a2: 1.25107             b1: 5.92926             b2: 1.25106             diff: 5.54117e-05

__________________________________________________Newton Method________________________________________________
a1: 1           a2: 2           b1: -6.5                b2: 5.52941             diff: 7.5
a1: -6.5                a2: 5.52941             b1: -1.98433            b2: 6.67119             diff: 4.51567
a1: -1.98433            a2: 6.67119             b1: 0.413634            b2: 6.39435             diff: 2.39797
a1: 0.413634            a2: 6.39435             b1: -2.16394            b2: 4.80085             diff: 2.57757
a1: -2.16394            a2: 4.80085             b1: -2.30316            b2: 2.95453             diff: 1.84632
a1: -2.30316            a2: 2.95453             b1: -1.82625            b2: 3.30871             diff: 0.476908
a1: -1.82625            a2: 3.30871             b1: -1.76702            b2: 3.34705             diff: 0.0592268
a1: -1.76702            a2: 3.34705             b1: -1.76625            b2: 3.34734             diff: 0.000765562
a1: -1.76625            a2: 3.34734             b1: -1.76625            b2: 3.34734             diff: 8.65048e-08
```

Рис. 2: Вывод программы в консоли

6 Исходный код

```
#include <iostream>
#include <cmath>
#include <utility>
#include <vector>
using namespace std;

double F1(double x1, double x2)
{
  return (pow(x1, 2) + 16)*x2 - 64;
}

double F2(double x1, double x2)
{
  return pow((x1 - 2), 2) + pow((x2 - 2), 2) - 16;
}

double dF1_x1(double x1, double x2)
{
  return x2*x1*2;
}

double dF1_x2(double x1)
{
  return pow(x1, 2) + 16;
}

double dF2_x1(double x1)
{
  return 2*x1-4;
}

double dF2_x2(double x2)
{
  return 2*x2-4;
}

double max_diff(double a1, double b1, double a2, double b2)
{
  double max = abs(b1-a1);

  if (max > abs(b2 - a2))
    return max;
  else
    return abs(b2 - a2);
}

double determ2x(vector<vector<double>>& A)
```

```
{
  return A[0][0] * A[1][1] - A[0][1] * A[1][0];
}

pair<double, double> Newton(double a1, double a2, const double precision)
{
  cout << "\n__________________________________________________Newton
      Method________________________________________________";

  vector<vector<double>> A1 = { {F1(a1,a2), dF1_x2(a1)}, {F2(a1,a2),dF2_x2(a2)} };
  vector<vector<double>> A2 = { {dF1_x1(a1,a2), F1(a1,a2)}, {dF2_x1(a1), F2(a1,a2)} };
  vector<vector<double>> J = { {dF1_x1(a1,a2), dF1_x2(a1)}, {dF2_x1(a1),dF2_x2(a2)} };

  double b1 = a1 - determ2x(A1) / determ2x(J);
  double b2 = a2 - determ2x(A2) / determ2x(J);

  double diff = max_diff(a1, b1, a2, b2);

  cout << "\na1: " << a1 << "\t\ta2: " << a2 << "\t\tb1: " << b1 << "\t\tb2: " << b2
      << "\t\tdiff: " << diff;
  a1 = b1; a2 = b2;

  while (precision < diff)
  {
    A1 = { {F1(a1,a2), dF1_x2(a1)}, {F2(a1,a2),dF2_x2(a2)} };
    A2 = { {dF1_x1(a1,a2), F1(a1,a2)}, {dF2_x1(a1), F2(a1,a2)} };
    J = { {dF1_x1(a1,a2), dF1_x2(a1)}, {dF2_x1(a1),dF2_x2(a2)} };

    b1 = a1 - determ2x(A1) / determ2x(J);
    b2 = a2 - determ2x(A2) / determ2x(J);


    diff = max_diff(a1, b1, a2, b2);

    cout << "\na1: " << a1 << "\t\ta2: " << a2 << "\t\tb1: " << b1 << "\t\tb2: " << b2
        << "\t\tdiff: " << diff;
    a1 = b1; a2 = b2;
  }

  return make_pair(b1, b2);
}

double phi1(double a2)
{
  return sqrt(16 - pow((a2 - 2), 2)) + 2;
}

double phi2(double a1)
{
  return 64 / (pow(a1, 2) + 16);
}

pair<double, double> simpleIteration(double a1, double a2, const double precision)
{
  cout << "\n____________________________________________Simple Iteration Method____________________________________________";
  double b1 = phi1(a2);
  double b2 = phi2(a1);

  double diff = max_diff(a1, b1, a2, b2);

  cout << "\na1: " << a1 << "\t\ta2: " << a2 << "\t\tb1: " << b1 << "\t\tb2: " << b2 << "\t\tdiff: " << diff;
  a1 = b1; a2 = b2;

  while (precision < diff)
  {
    b1 = phi1(a2);
    b2 = phi2(a1);

    diff = max_diff(a1, b1, a2, b2);

    cout << "\na1: " << a1 << "\t\ta2: " << a2 << "\t\tb1: " << b1 << "\t\tb2: " << b2 << "\t\tdiff: " << diff;
    a1 = b1; a2 = b2;
  }

  return make_pair(b1, b2);
}

int main()
{
  double a1;
  cout << "Input a1: ";
  cin >> a1;

  double a2;
  cout << "Input a2: ";
  cin >> a2;

  double precision;
  cout << "Input precision: ";
  cin >> precision;

  simpleIteration(a1, a2, precision);

  cout << '\n';

  Newton(a1, a2, precision);
}
```