Московский авиационный институт
(национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №2 по курсу «Численные методы»

Студент:	А. А. Литвина
Преподаватель:	И. Э. Иванов
Группа:	М8О-306Б
Дата:	
Оценка:	
Подпись:	

Москва, 2020

Вариант №13

1 Решение нелинейных уравнений

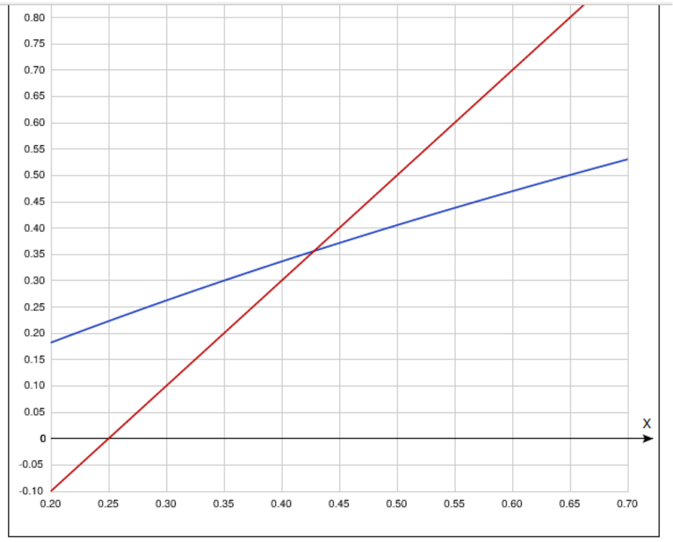
1 Постановка задачи

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.

Функция: $ln(x+1) - 2x + 0,5 = 0$.

2 Описание

Для локализации корней применим графический способ. Преобразуем исходное уравнение к виду $ln(x+1) = 2x - 0,5$ и построим графики функций $ln(x+1)$ и $2x - 0,5$.



Из графика видно, что у уравнения имеется только один корень, который находится в интервале $0,35 < x < 0,5$.

2.1 Метод простой итерации

Представим уравнение в виде $x = \varphi(x)$.

Для данной задачи: $x = \frac{\ln(x+1)+0.5}{2}$.

Сначала проверим условия сходимости:
1) $\varphi(x) \in [a,b] \ \ \forall x \in [a,b]$,
2) $\exists q : |\varphi'(x)| \leq q < 1 \ \ \forall x \in (a,b)$.

За $x^{(0)}$ примем $\frac{a+b}{2}$.
Итерационный процесс опередляется формулой:

$$x^{(k+1)} = \varphi(x^{(k)}).$$

Условия окончания итерационного процесса:

$$\frac{q}{1-q}|x^{(k+1)} - x^{(k)}| \leq \varepsilon.$$

2.2 Метод Ньютона

За $x^{(0)}$ примем b.
Условия сходимости метода: $f(x^{(0)})f''(x^{(0)}) > 0$.
Итерационный процесс опередляется формулой:

$$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})}.$$

Для данной задачи $f'(x) = \frac{1}{x+1} - 2$.

Условия окончания итерационного процесса:

$$|x^{(k+1)} - x^{(k)}| < \varepsilon.$$

3 Исходный код

3.1 Метод простой итерации

```
#include <iostream>
#include <cmath>

using namespace std;

const double eps=0.0001;

int main() {
  double t1, t2, phi1, phi2, q, x, x1, e;
  int it=0;
  t1=0.35;
  t2=0.5;
  phi1=(log(t1+1)+0.5)/2;
  phi2=(log(t2+1)+0.5)/2;
  q=1/(2*(t1+1));
  if ((phi1>=t1) and (phi2<=t2) and (q<1)) {
    x=(t1+t2)/2;
    q=1/(2*(x+1));
    e=q/(1-q)*abs(x);
    while (e>eps) {
      it++;
      x1=(log(x+1)+0.5)/2;
      q=1/(2*(x1+1));
      e=q/(1-q)*abs(x1-x);
      x=x1;
    }
    cout << "Iterations " << it << endl;
    cout << "x=" << x1 << endl;
    cout << "eps=" << e << endl;
  }
}
```

3.2 Метод Ньютона

```
#include <iostream>
#include <cmath>

using namespace std;

const double eps=0.0001;

int main() {
  double t2, f, f_pp, x, x1, e;
  int it=0;
  t2=0.5;
  f=log(t2+1)-2*t2+0.5;
  f_pp=-1/pow((t2+1),2);
  if (f*f_pp>0) {
    e=t2;
    x=t2;
    while (e>eps) {
      it++;
      x1=x-(log(x+1)-2*x+0.5)/(1/(x+1)-2);
      e=abs(x1-x);
      x=x1;
    }
    cout << "Iterations " << it << endl;
    cout << "x=" << x1 << endl;
  }
}
```

4 Консоль

4.1 Метод простой итерации

```
Iterations 4
x=0.428163
eps=4.8322e-05
```

4.2 Метод Ньютона

```
Iterations 3
x=0.428211
```

2 Решение систем нелинейных уравнений

1 Постановка задачи

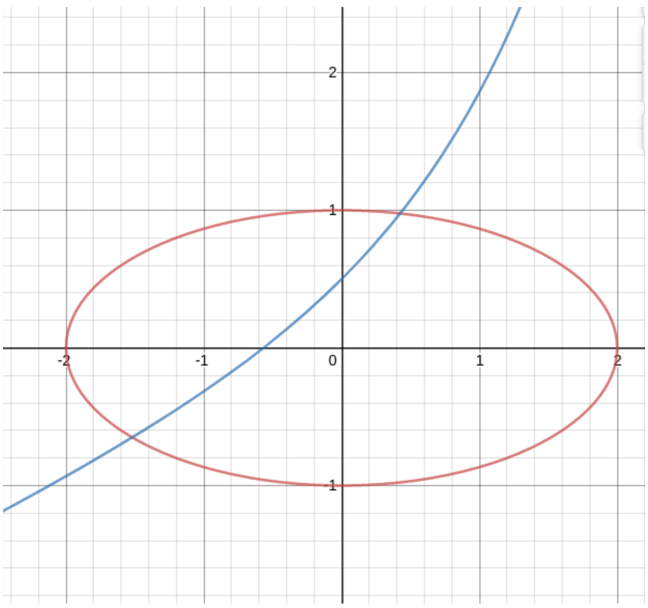
Реализовать методы простой итерации и Ньютона решения систем нелинейных уравнений в виде программного кода, задавая в качестве входных данных точность вычислений. С использованием разработанного программного обеспечения решить систему нелинейных уравнений (при наличии нескольких решений найти то из них,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

Система:

$$\begin{cases} \frac{x_1^2}{4} + x_2^2 - 1 = 0 \\ 2x_2 - e^{x_1} - x_1 = 0 \end{cases}$$

2 Описание

Для выбора начального приближения применим графический способ. Построим графики функций $\frac{x_1^2}{4} + x_2^2 - 1 = 0$ и $2x_2 - e^{x_1} - x_1 = 0$ и определим, что положительное решение системы уравнений находится в квадрате $0,2 < x_1 < 0,6$, $0,8 < x_2 < 1,2$.



2.1 Метод простой итерации

Сначала система уравнений приводится к эквивалентной системе специального вида $x = \varphi(x)$.

Для данной задачи:

$x_1 = \ln(\sqrt{4 - x_1^2} - x_1)$

$x_2 = \sqrt{1 - \frac{x_1^2}{4}}$.

Проверим достаточное условие сходимости:

$$\max \|\varphi'(x)\| \le q < 1,$$

где

$$\max \|\varphi'(x)\| = \max \left\{ \max_{(i)} \sum_{j=1}^{n} \left| \frac{\partial \varphi_i(x_1, x_2)}{\partial x_j} \right| \right\}.$$

За начальное приближение примем $x_1^{(0)} = \frac{a_1+b_1}{2}$, $x_2^{(0)} = \frac{a_2+b_2}{2}$.

Итерационный процесс оперделяется формулой:

$$x^{(k+1)} = \varphi(x^{(k)}).$$

Условия окончания итерационного процесса:

$$\frac{q}{1-q} \|x^{(k+1)} - x^{(k)}\| \le \varepsilon,$$

где

$$\|x^{(k+1)} - x^{(k)}\| = \max_i |x_i^{(k+1)} - x_i^{(k)}|.$$

2.2 Метод Ньютона

За начальное приближение примем $x_1^{(0)} = \frac{a_1+b_1}{2}$, $x_2^{(0)} = \frac{a_2+b_2}{2}$.

Итерационный процесс опередляется формулой:

$$\begin{cases} x_1^{(k+1)} = x_1^{(k)} - \frac{detA_1^{(k)}}{detJ^{(k)}} \\ x_2^{(k+1)} = x_2^{(k)} - \frac{detA_2^{(k)}}{detJ^{(k)}} \end{cases}$$

где

$$\mathbf{A}_1^{(k)} = \begin{bmatrix} f_1(x_1^{(k)}, x_2^{(k)}) & \dfrac{\partial f_1(x_1^{(k)}, x_2^{(k)})}{\partial x_2} \\ f_2(x_1^{(k)}, x_2^{(k)}) & \dfrac{\partial f_2(x_1^{(k)}, x_2^{(k)})}{\partial x_2} \end{bmatrix},$$

$$\mathbf{A}_2^{(k)} = \begin{bmatrix} \dfrac{\partial f_1(x_1^{(k)}, x_2^{(k)})}{\partial x_1} & f_1(x_1^{(k)}, x_2^{(k)}) \\ \dfrac{\partial f_2(x_1^{(k)}, x_2^{(k)})}{\partial x_1} & f_2(x_1^{(k)}, x_2^{(k)}) \end{bmatrix}.$$

$$\mathbf{J}^{(k)} = \begin{bmatrix} \dfrac{\partial f_1(x_1^{(k)}, x_2^{(k)})}{\partial x_1} & \dfrac{\partial f_1(x_1^{(k)}, x_2^{(k)})}{\partial x_2} \\ \dfrac{\partial f_2(x_1^{(k)}, x_2^{(k)})}{\partial x_1} & \dfrac{\partial f_2(x_1^{(k)}, x_2^{(k)})}{\partial x_2} \end{bmatrix},$$

Для данной задачи:

$\frac{\partial f_1}{\partial x_1} = \frac{x_1}{2}$

$\frac{\partial f_1}{\partial x_2} = 2x_2$

$\frac{\partial f_2}{\partial x_1} = -e^{x_1} - 1$

$\frac{\partial f_2}{\partial x_2} = 2.$

Условия окончания итерационного процесса:

$$\|x^{(k+1)} - x^{(k)}\| \leq \varepsilon,$$

где

$$\|x^{(k+1)} - x^{(k)}\| = \max_i |x_i^{(k+1)} - x_i^{(k)}|.$$

3 Исходный код

3.1 Метод простой итерации

```
#include <iostream>
#include <cmath>

using namespace std;

const double eps=0.0001;

int main() {
  double x1, x2, q, t1, t2, e, g1, g2, x1_n, x2_n;
  double d_phi1_x1, d_phi1_x2, d_phi2_x1, d_phi2_x2;
  int it=0;
  x1=0.4;
  x2=1;
  g1=0.6;
  g2=1.2;
  d_phi1_x1=(g1/pow(4-pow(g1,2),0.5)-1)/(pow(4-pow(g1,2),0.5)-g1);
  d_phi1_x2=0;
  d_phi2_x1=1/4*g1/pow(1-pow(g1,2)/4,0.5);
  d_phi2_x2=0;
  t1=abs(d_phi1_x1)+abs(d_phi1_x2);
  t2=abs(d_phi2_x1)+abs(d_phi2_x2);
  q=max(t1,t2);
  e=q/(1-q)*max(x1,x2);
  while (e>eps) {
    it++;
    x1_n=log(pow(4-pow(x1,2),0.5)-x1);
    x2_n=pow(1-pow(x1,2)/4,0.5);
    e=q/(1-q)*max(abs(x1_n-x1), abs(x2_n-x2));
    x1=x1_n;
    x2=x2_n;
  }
  cout << "Iterations " << it << endl;
  cout << "x1=" << x1 << endl;
  cout << "x2=" << x2 << endl;
  cout << "eps=" << e << endl;
}
```

3.2 Метод Ньютона

```
#include <iostream>
#include <cmath>
#include <vector>

using namespace std;

const double eps=0.0001;

double det (vector <vector <double>> A) {
  return A[0][0]*A[1][1] - A[0][1]*A[1][0];
}

int main() {
  double f1, f2, x1, x2, e, x1_n, x2_n;
  double d_f1_x1, d_f1_x2, d_f2_x1, d_f2_x2;
  vector <vector <double>> A1 (2, vector <double> (2,0));
  vector <vector <double>> A2 (2, vector <double> (2,0));
  vector <vector <double>> J (2, vector <double> (2,0));
  int it=0;
  x1=0.4;
  x2=1;
  e=max(x1,x2);
  while (e>eps) {
    it++;
    f1=pow(x1,2)/4+pow(x2,2)-1;
    f2=2*x2-exp(x1)-x1;
    d_f1_x1=x1/2;
    d_f1_x2=2*x2;
    d_f2_x1=-exp(x1)-1;
    d_f2_x2=2;

    A1[0][0]=f1;
    A1[1][0]=f2;
    A1[0][1]=d_f1_x2;
    A1[1][1]=d_f2_x2;

    A2[0][0]=d_f1_x1;
    A2[1][0]=d_f2_x1;
    A2[0][1]=f1;
    A2[1][1]=f2;

    J[0][0]=d_f1_x1;
    J[0][1]=d_f1_x2;
    J[1][0]=d_f2_x1;
    J[1][1]=d_f2_x2;

    x1_n=x1-det(A1)/det(J);
```

```
    x2_n=x2-det(A2)/det(J);
    e=max(abs(x1_n-x1), abs(x2_n-x2));
    x1=x1_n;
    x2=x2_n;
  }
  cout << "Iterations " << it << endl;
  cout << "x1=" << x1 << endl;
  cout << "x2=" << x2 << endl;
}
```

4 Консоль

4.1 Метод простой итерации

```
Iterations 29
x1=0.424935
x2=0.977176
eps=8.20711e-05
```

4.2 Метод Ньютона

```
Iterations 3
x1=0.424902
x2=0.977172
```

Выводы

В этой лабораторной работе я познакомилась с методами решения нелинейных уравнений и систем нелинейных уравнений. Я изучила и реализовала метод простой итерации и метод Ньютона для каждой из этих задач.
Как можно увидеть из результатов работы программ, для решения одиночных уравнений и метод простой итерации, и метод Ньютона оказались одинаково эффективны. Для решения систем уравнений метод Ньютона оказался намного быстрее. Поэтому, на практике, прежде чем приступить к реализации того или иного метода, следует выбрать наиболее оптимальный метод решения.