|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ Информатики и систем управления

КАФЕДРА Теоретической информатики и компьютерных технологий

**Лабораторная работа № 6**

«Решение нелинейных уравнений и систем нелинейных

уравнений методом Ньютона»

по курсу «Численные методы»

Выполнил:

студент группы ИУ9-61Б

Митрошкин Алексей

Проверила:

Домрачева А. Б.

Москва, 2024

1. **Цель**

Целью данной работы является изучение метода Ньютона для решения нелинейных уравнений и систем нелинейных уравнений и сравнение различных методов решение уравнения.

1. **Постановка задачи**

**Дано:** система нелинейных уравнений

**Задание:**

* Найти решение аналитически;
* Найти решение системы с точностью , начиная итерации из точки, заданной собственноручно;
* Сравнить полученные результаты.

**Индивидуальный вариант:**

1. **Основные теоретические сведения и этапы работы**

**Описание метода:**

Пусть задана система нелинейных уравнений:

или в векторном виде f(x) = 0, где Х = вектор неизвестных; f = вектор-функция.

Выбрав начальное приближение к решению системы, следующие приближения в методе Ньютона строим по рекуррентной зависимости:

**Здесь:**

,

столбцы (k+1)-го и k-го приближения к решению;

значение столбца левой части системы в точке ;

При матрица Якоби системы, являющаяся производной вектор-функции в точке ;

матрица, обратная матрице Якоби.

Мы предполагаем, что матрица Якоби обратима в достаточно большой окрестности точного решения системы.

Приближения метода Ньютона удобно искать в два этапа. Вначале решаем систему линейных уравнений с матрицей - матрицей Якоби вектор-функции f:

любым из способов решения СЛАУ, например методом Гаусса. Теперь (k+1)-е приближение еcть сумма K-то приближения и решения СЛАУ Y = ,

Для решения системы нелинейных уравнений с заданной точностью ε необходимо сравнить ε с погрешностью к-го приближения

Метод Ньютона cходится, если две функции

дважды непрерывно дифференцируемы по всем переменный и начальное приближение находится достаточно близко к точному решению системы. Рецепта выбора начального приближения при n > 1 нет.

Поэтому желательно оценить, хотя бы грубо, значение точного решения, например, решив систему графичекси.

1. **Реализация**

Листинг 1. Сравнение приближённых методов решения нелинейных уравнений:

namespace lab6.\_1

{

public class Program

{

static decimal eps = 0.001m;

static decimal[] coeffs = { 2.0m, 0.0m, -9.0m, 1.0m };

static decimal f(decimal x)

{

return coeffs[0] \* (decimal)Math.Pow((double)x, 3.0) +

coeffs[1] \* (decimal)Math.Pow((double)x, 2.0) +

coeffs[2] \* x +

coeffs[3];

}

static decimal derivative\_f(decimal x)

{

return 3 \* coeffs[0] \* (decimal)Math.Pow((double)x, 2.0) +

2 \* coeffs[1] \* x +

coeffs[2];

}

static decimal second\_derivative\_f(decimal x)

{

return 6 \* coeffs[0] \* x +

2 \* coeffs[1];

}

static int sgn(decimal x)

{

if (x > 0)

return 1;

else if (x < 0)

return -1;

return 0;

}

static (decimal, int) BisectionMethod(Func<decimal, decimal> f, (decimal, decimal) segment)

{

decimal left = segment.Item1;

decimal right = segment.Item2;

decimal mid = (left + right) / 2.0m;

int i = 0;

while (Math.Abs(f(mid)) > eps)

{

if (f(left) \* f(mid) < 0)

right = mid;

else if (f(right) \* f(mid) < 0)

left = mid;

else

return (mid, i);

i++;

mid = (left + right) / 2.0m;

}

return (mid, i);

}

static (decimal, int) NewtonMethod(Func<decimal, decimal> f, (decimal, decimal) segment)

{

decimal start = segment.Item1;

decimal end = segment.Item2;

if (f(end) \* second\_derivative\_f(end) > 0)

start = end;

decimal prev = start;

decimal cur = start;

int i = 0;

while (f(cur) \* f(cur + sgn(cur - prev) \* eps) >= 0)

{

prev = cur;

cur = cur - f(cur) / derivative\_f(cur);

i++;

}

return (cur, i);

}

public static void Main(string[] args)

{

(decimal, decimal)[] segments = { (-3.0m, -2.0m), (0.05m, 0.5m), (1.0m, 3.0m) };

foreach (var segment in segments)

{

var bisectionResult = BisectionMethod(f, segment);

var newtonResult = NewtonMethod(f, segment);

Console.WriteLine("bisection: " + bisectionResult.Item1 + " iters: " + bisectionResult.Item2);

Console.WriteLine("newton: " + newtonResult.Item1 + " iters: " + newtonResult.Item2);

Console.WriteLine("diff: " + Math.Abs(bisectionResult.Item1 - newtonResult.Item1));

}

}

}

}

Листинг 2. Решение системы нелинейных уравнений методом Ньютона:

using System;

class NewtonMethod

{

static double f1(double x, double y)

{

return Math.Sin(x+1) - y - 1; // sin(x+1) - y - 1 =0

}

static double f2(double x, double y)

{

return 2\*x + Math.Cos(y) - 2; // 2x + cos(y) - 2 =0

}

static double df1\_dx(double x, double y)

{

return Math.Cos(x+1);

}

static double df2\_dx(double x, double y)

{

return 2;

}

static double df1\_dy(double x, double y)

{

return -1;

}

static double df2\_dy(double x, double y)

{

return -Math.Sin(y);

}

static void NewtonMethodSolver(double x0, double y0, double eps)

{

double x = x0;

double y = y0;

double dx, dy;

for (int i = 0; i < 100; i++) // Ограничим количество итераций

{

double J11 = df1\_dx(x, y);

double J12 = df1\_dy(x, y);

double J21 = df2\_dx(x, y);

double J22 = df2\_dy(x, y);

double detJ = J11 \* J22 - J12 \* J21;

double invJ11 = J22 / detJ;

double invJ12 = -J12 / detJ;

double invJ21 = -J21 / detJ;

double invJ22 = J11 / detJ;

dx = -(invJ11 \* f1(x, y) + invJ12 \* f2(x, y));

dy = -(invJ21 \* f1(x, y) + invJ22 \* f2(x, y));

x += dx;

y += dy;

// Проверяем на невалидные значения

if (double.IsNaN(x) || double.IsNaN(y) || double.IsInfinity(x) || double.IsInfinity(y))

{

Console.WriteLine("Метод расходится. Выберите другие начальные значения.");

return;

}

if (Math.Abs(dx) < eps && Math.Abs(dy) < eps)

{

Console.WriteLine($"x = {x}, y = {y}");

Console.WriteLine($"Кол-во итераций: {i}");

return;

}

}

Console.WriteLine("Не удалось найти корни. Попробуйте другие начальные значения или увеличить количество итераций.");

}

static void Main(string[] args)

{

double x0 = 0; // Начальное приближение для x

double y0 = 0; // Начальное приближение для y

double eps = 1e-5; // Точность

NewtonMethodSolver(x0, y0, eps);

}

}

1. **Результаты**

Было написано программное обеспечение для метода Ньютона для решения нелинейных уравнений и систем нелинейных уравнений. Получены результаты.

Рисунок 1. Сравнение методов:

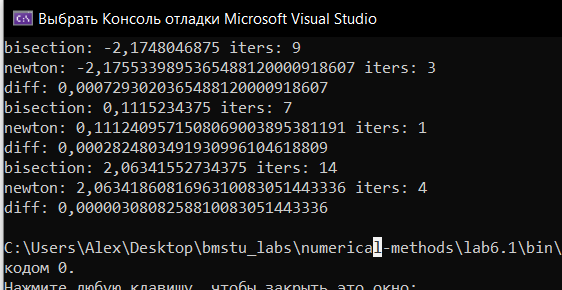
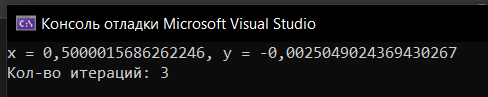


Рисунок 2 – Пример вывода программы для системы нелинейных уравнений:



1. **Вывод**

В ходе лабораторной работы мы исследовали и применили метод Ньютона для решения нелинейных уравнений и систем таких уравнений. Мы успешно получили приближенные решения системы и корни уравнений с необходимой точностью уже после двух итераций. В процессе тестирования мы сравнили метод Ньютона с методом деления отрезка пополам для решения нелинейных уравнений и пришли к выводу, что метод Ньютона сходится к решению быстрее.