Отчет по практической работе №5

Вайкус Яков Михайлович

ИC-32-1

*Решить систему линейных уравнений с тремя неизвестными с точностью ε =* 0,0001:

* *методом Гаусса, написав программу в С++;*
* *методом простой итерации с евклидовой метрикой, написав программу в С++; отследить сходимость итерационного процесса;*
* *при помощи программы MathCad.*
* *Вычислить невязку решения при учете четырех значащих цифр после запятой.*

Код программы:

internal class Program

{

    static void Main(string[] args)

    {

        double[,] matrix = {

        {0.21, -0.45, -0.20, 1.97},

        {0.30, 0.25, 0.43, 0.32},

        {0.60, -0.35, -0.25, 1.83}

         };

        Gauss(matrix);

        double e = 0.0001;

        double[] res = { };

        for (int i = 0; i < matrix.GetLength(0); i++)

        {

            bool ok = false;

            for (int j = 0; j < matrix.GetLength(0); j++)

            {

                if (i == j) { continue; }

                for (int k = 0; k < matrix.GetLength(1); k++)

                {

                    double t = matrix[i, k];

                    matrix[i, k] = matrix[j, k];

                    matrix[j, k] = t;

                }

                res = Iternal(matrix, e);

                if (!Double.IsNaN(res[0]) && !Double.IsNaN(res[1]) && !Double.IsNaN(res[2]))

                {

                    ok = true;

                    break;

                }

            }

            if (ok == true) { break; }

        }

        Console.WriteLine("Методом простой итерации:");

        for (int i = 0; i < 3; i++)

        {

            Console.WriteLine($"x{i+1} = {res[i]}") ;

        }

        double nev = ComputeResidual(matrix, res);

        Console.WriteLine($"Невязка: {nev:F10}");

    }

    public static void Gauss(double[,] matrix)

    {

        double[,] mtrx = new double[3,4];

        Array.Copy(matrix, mtrx, 12);

        int n = mtrx.GetLength(0);

        double[] solution = new double[n];

        for (int i = 0; i < n; i++)

        {

            int maxRow = i;

            for (int k = i + 1; k < n; k++)

            {

                if (Math.Abs(mtrx[k, i]) > Math.Abs(mtrx[maxRow, i]))

                {

                    maxRow = k;

                }

            }

            for (int k = i; k < n + 1; k++)

            {

                double temp = mtrx[maxRow, k];

                mtrx[maxRow, k] = mtrx[i, k];

                mtrx[i, k] = temp;

            }

            for (int k = i + 1; k < n; k++)

            {

                double factor = mtrx[k, i] / mtrx[i, i];

                for (int j = i; j < n + 1; j++)

                {

                    mtrx[k, j] -= factor \* mtrx[i, j];

                }

            }

        }

        for (int i = n - 1; i >= 0; i--)

        {

            solution[i] = mtrx[i, n] / mtrx[i, i];

            for (int k = i - 1; k >= 0; k--)

            {

                mtrx[k, n] -= mtrx[k, i] \* solution[i];

            }

        }

        Console.WriteLine("методом Гаусса:");

        for (int i = 0; i < n; i++)

        {

            Console.WriteLine($"x{i + 1} = {solution[i]}");

        }

    }

public static double[] Iternal(double[,] matrix, double epsilon)

{

    double[] guess = { 0, 0, 0 }; // Начальное приближение

    int n = guess.Length;

    double[] newGuess = new double[n];

    double[] oldGuess = new double[n];

    Array.Copy(guess, oldGuess, n);

    int iteration = 0; // Счетчик итераций

    bool convergence = false;

    // Сбор данных для лога

    List<string> convergenceLog = new List<string>();

    while (!convergence)

    {

        iteration++; // Увеличиваем номер итерации

        for (int i = 0; i < n; i++)

        {

            double sum = 0;

            for (int j = 0; j < n; j++)

            {

                if (i != j)

                {

                    sum += matrix[i, j] \* oldGuess[j];

                }

            }

            newGuess[i] = (matrix[i, n] - sum) / matrix[i, i];

        }

        // Проверяем сходимость

        convergence = true;

        for (int i = 0; i < n; i++)

        {

            if (Math.Abs(newGuess[i] - oldGuess[i]) > epsilon)

            {

                convergence = false;

                break;

            }

        }

        // Проверяем на корректность значений

        if (!double.IsNaN(newGuess[0]) && !double.IsInfinity(newGuess[0]) &&

            iteration <= 1000)

        {

            convergenceLog.Add(

                $"{iteration,-10} {newGuess[0],-10:F6} {newGuess[1],-10:F6} {newGuess[2],-10:F6}");

        }

        // Если итераций слишком много, считаем, что сходимости нет

        if (iteration > 1000)

        {

            //Console.WriteLine("Сходимость не достигнута за 1000 итераций.");

            return new double[] { double.NaN, double.NaN, double.NaN };

        }

        // Обновляем старые значения

        Array.Copy(newGuess, oldGuess, n);

    }

    // Выводим процесс сходимости только для успешной попытки

    if (convergence)

    {

        Console.WriteLine("Процесс сходимости метода простой итерации:");

        Console.WriteLine("Итерация    x1          x2          x3");

        foreach (var log in convergenceLog)

        {

            Console.WriteLine(log);

        }

    }

    return newGuess;

}

    static double ComputeResidual(double[,] matrix, double[] result)

    {

        double residual = 0;

        int n = 3;

        for (int i = 0; i < n; i++)

        {

            double sum = 0.0;

            for (int j = 0; j < n; j++)

            {

                sum += matrix[i, j] \* result[j];

            }

            residual += Math.Abs(sum - matrix[i, n]);

        }

        return residual;

    }

}

Вывод:  
