МИНИСТЕРСТВО НАУКИ и высшего образования РОССИЙСКОЙ ФЕДЕРАЦИИ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**

**ОТЧЁТ**

**по лабораторной работе №2**

**по дисциплине: «** *Вычислительная математика* **»**

**Тема: «** *Решение систем линейных алгебраических уравнений* **»**

**Вариант 2**

Выполнили:Проверил:

Студенты гр. АП-227 *Ландовский В.В.*

*Федотов И.В.  
 Бузмаков А.И.*

*Шестаков К.Д.*

«\_\_\_» \_\_\_\_\_\_ 20\_\_г.«\_\_\_» \_\_\_\_\_\_ 20\_\_ г.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (подпись)

Новосибирск

2024

**Задание**

1. Подготовить программную реализацию метода Гаусса без выбора главного элемента, использовать 32 битное представление вещественных чисел. Реализацию выполнить так, чтобы ей было возможно воспользоваться при разработке программ для последующих лабораторных работ. С помощью полученной программы решить соответствующую варианту систему СЛАУ 1 из таблицы 2.1. Добавить в программу выбор главного элемента заданным способом. Решить ту же систему с выбором главного элемента. Объяснить полученные результаты.

2. Подготовить программную реализацию приведения системы уравнений к виду (2.3) и решения системы заданным итерационным методом. Выполнение 28 условия сходимости перед началом итерационного процесса не контролировать, в качестве критерия остановки использовать (2.9). Попытаться решить соответствующую варианту СЛАУ 2 из таблицы 2.1. Вручную преобразовать систему уравнений так, чтобы гарантировать сходимость итерационного процесса. Добавить в программу опциональную проверку условия (2.5) и проверку окончания в соответствии с (2.8) или (2.10) в зависимости от метода. Решить преобразованную систему с точностью 10-3. Экспериментально подобрать минимальное значение заданной погрешности, при которой вычисления сходятся к конечному значению (не возникает переполнения) и завершаются корректно (программа не входит в бесконечный цикл). Объяснить все полученные результаты.

3. Подготовить программный модуль позволяющий решать систему с трехдиагональной матрицей произвольной размерности. Реализацию выполнить так, чтобы ей было удобно воспользоваться при разработке программ для последующих лабораторных работ. Провести тестирование на собственных примерах.

**Исходные системы уравнений**

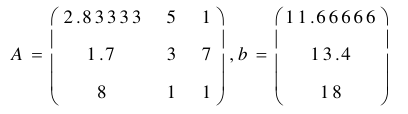


Рисунок 1 - СЛАУ 1

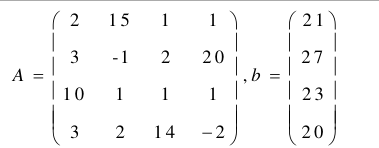


Рисунок 2 – СЛАУ 2

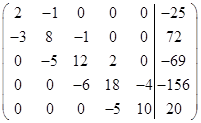


Рисунок 2 – Система с трехдиагональной матрицей

**Тексты разработанных программ**

Таблица 1 – Решение системы методом Гаусса

|  |
| --- |
| GaussMethod.cs |
| namespace вычмат2  {  class GaussMethod  {  public static float[] SolveGaussMethod(float[,] matrix, float[] b)  {  int n = b.Length;  // Прямой ход для преобразования системы к верхнетреугольному виду  for (int i = 0; i < n - 1; ++i)  {  // Проверка на ноль текущего элемента диагонали  if (Math.Abs(matrix[i, i]) < 1e-7f)  {  throw new Exception("Деление на ноль в методе Гаусса.");  }  for (int row = i + 1; row < n; ++row)  {  float factor = matrix[row, i] / matrix[i, i];  for (int col = i; col < n; ++col)  {  matrix[row, col] -= factor \* matrix[i, col];  }  b[row] -= factor \* b[i];  }  }  // Обратный ход для вычисления решений  float[] solution = new float[n];  for (int i = n - 1; i >= 0; --i)  {  float sum = b[i];  for (int j = i + 1; j < n; ++j)  {  sum -= matrix[i, j] \* solution[j];  }  solution[i] = sum / matrix[i, i]; // Здесь происходит деление для нахождения неизвестных  }  return solution;  }  public static float[] SolveGaussMethodColumn(float[,] originalMatrix, float[] originalB)  {  int n = originalB.Length;  float[,] matrix = new float[n, n];  float[] b = new float[n];  Array.Copy(originalB, b, n);  Array.Copy(originalMatrix, matrix, n \* n);  int[] columnIndices = Enumerable.Range(0, n).ToArray();  for (int i = 0; i < n; i++)  {  // Search for the column with the maximum absolute value in the row  int maxColumn = i;  float maxVal = Math.Abs(matrix[i, i]);  for (int k = i + 1; k < n; k++)  {  float absVal = Math.Abs(matrix[i, k]);  if (absVal > maxVal)  {  maxColumn = k;  maxVal = absVal;  }  }  // Swap columns  SwapColumns(matrix, columnIndices, i, maxColumn);  if (Math.Abs(matrix[i, i]) < 1e-7f)  {  throw new Exception("Division by zero in Gauss method with main element selection by row.");  }  // Transform the matrix to upper triangular form  for (int k = i + 1; k < n; k++)  {  float coeff = matrix[k, i] / matrix[i, i];  for (int j = i; j < n; j++)  {  matrix[k, j] -= matrix[i, j] \* coeff;  }  b[k] -= b[i] \* coeff;  }  }  // Back substitution  float[] solution = new float[n];  for (int i = n - 1; i >= 0; i--)  {  double sum = 0;  for (int j = i + 1; j < n; j++)  {  sum += matrix[i, j] \* solution[columnIndices[j]];  }  solution[columnIndices[i]] = (float)(b[i] - sum) / matrix[i, i];  }  return solution;  }  private static void SwapColumns(float[,] matrix, int[] columnIndices, int column1, int column2)  {  int n = matrix.GetLength(0);  for (int row = 0; row < n; row++)  {  float temp = matrix[row, column1];  matrix[row, column1] = matrix[row, column2];  matrix[row, column2] = temp;  }  int tempIndex = columnIndices[column1];  columnIndices[column1] = columnIndices[column2];  columnIndices[column2] = tempIndex;  }  }  } |

Таблица 2 – Решение системы итерационным методом

|  |
| --- |
| IterativeMethod.cs |
| namespace вычмат2  {  class IterativeMethod  {  public static float[] SolveIterativeMethod(float[,] a, float[] b, float epsilon, int iterations)  {  int n = b.Length;  // Проверка условия диагонального преобладания  if (!CheckDiagonalDominance(a))  {  Console.WriteLine("Матрица не удовлетворяет условию диагонального преобладания.");  EnsureDiagonallyDominant(a, b);  }  // Преобразование системы уравнений к виду x = alpha \* x + beta  float[,] alpha = new float[n, n];  float[] beta = new float[n];  for (int i = 0; i < n; i++)  {  beta[i] = b[i] / a[i, i];  for (int j = 0; j < n; j++)  {  if (i != j)  {  alpha[i, j] = -a[i, j] / a[i, i];  }  }  }  // Проверка условия нормы матрицы alpha < 1  float alphaNorm = CalculateMatrixNorm(alpha, n);  if (alphaNorm >= 1)  {  Console.WriteLine("Условие нормы матрицы alpha < 1 не выполняется. Норма alpha: " + alphaNorm);  return null;  }  // Начальное приближение  float[] x = new float[n];  float[] newX = new float[n];  for (int k = 0; k < iterations; k++)  {  for (int i = 0; i < n; i++)  {  newX[i] = beta[i];  for (int j = 0; j < n; j++)  {  newX[i] += alpha[i, j] \* x[j];  }  }  // Проверка условия сходимости  float norm = CalculateVectorNorm(SubtractVectors(newX, x));  if (norm <= epsilon)  {  Console.WriteLine($"Сходимость достигнута на итерации {k + 1}");  return newX;  }  Array.Copy(newX, x, n); // Подготовка к следующей итерации  }  Console.WriteLine("Метод не сошелся после заданного числа итераций.");  return null; // Метод не сходится  }  static bool CheckDiagonalDominance(float[,] matrix)  {  int n = matrix.GetLength(0);  for (int i = 0; i < n; i++)  {  float sum = 0;  for (int j = 0; j < n; j++)  {  if (i != j)  {  sum += Math.Abs(matrix[i, j]);  }  }  if (Math.Abs(matrix[i, i]) <= sum)  {  return false;  }  }  return true;  }  // Вычисление нормы матрицы  static float CalculateMatrixNorm(float[,] matrix, int n)  {  float maxSum = float.MinValue;  for (int i = 0; i < n; i++)  {  float rowSum = 0;  for (int j = 0; j < n; j++)  {  rowSum += Math.Abs(matrix[i, j]);  }  maxSum = Math.Max(maxSum, rowSum);  }  return maxSum;  }  // Вычисление нормы вектора  static float CalculateVectorNorm(float[] vector)  {  float sumOfSquares = 0;  foreach (var element in vector)  {  sumOfSquares += (float)Math.Pow(element, 2);  }  return (float)Math.Sqrt(sumOfSquares);  }  // Вычитание векторов  static float[] SubtractVectors(float[] vector1, float[] vector2)  {  int n = vector1.Length;  float[] result = new float[n];  for (int i = 0; i < n; i++)  {  result[i] = vector1[i] - vector2[i];  }  return result;  }  static float[,] EnsureDiagonallyDominant(float[,] coefficients, float[] constants)  {  int n = coefficients.GetLength(0);  // Перестановка строк матрицы  for (int i = 0; i < n; i++)  {  float diagonalElement = coefficients[i, i];  float rowSum = 0;  for (int j = 0; j < n; j++)  {  rowSum += Math.Abs(coefficients[i, j]);  }  if (Math.Abs(diagonalElement) <= rowSum - Math.Abs(diagonalElement))  {  // Ищем строку с максимальным элементом в текущем столбце  int maxRow = i;  float maxElement = Math.Abs(coefficients[i, i]);  for (int k = i + 1; k < n; k++)  {  float currentElement = Math.Abs(coefficients[k, i]);  if (currentElement > maxElement)  {  maxElement = currentElement;  maxRow = k;  }  }  // Перестановка строк  if (maxRow != i)  {  for (int j = 0; j < n; j++)  {  float temp = coefficients[i, j];  coefficients[i, j] = coefficients[maxRow, j];  coefficients[maxRow, j] = temp;  }  float tempConstant = constants[i];  constants[i] = constants[maxRow];  constants[maxRow] = tempConstant;  }  }  }  return coefficients;  }  }  } |

Таблица 3 – Решение трехдиагональной матрицы

|  |
| --- |
| TridiagonalMatrixSolver.cs |
| namespace вычмат2  {  class TridiagonalMatrixSolver  {  public static float[] Solve(float[] a, float[] b, float[] c, float[] d) // метод прогонки  {  int n = d.Length;  float[] P = new float[n]; // Массив коэффициентов P  float[] Q = new float[n]; // Массив коэффициентов Q  float[] x = new float[n]; // Решение  // Прямой проход (вычисление коэффициентов P и Q)  P[0] = -a[0] / b[0];  Q[0] = d[0] / b[0];  for (int i = 1; i < n; i++)  {  float denom = b[i] + c[i] \* P[i - 1];  P[i] = -a[i] / denom;  Q[i] = (d[i] - c[i] \* Q[i - 1]) / denom;  }  // Обратный проход (вычисление решения)  x[n - 1] = Q[n - 1];  for (int i = n - 2; i >= 0; i--)  {  x[i] = P[i] \* x[i + 1] + Q[i];  }  return x;  }  }  } |

Таблица 4 – Основной класс

|  |
| --- |
| Program.cs |
| namespace вычмат2  {  internal class Program  {  static void part1()  {  Console.WriteLine("|Задание 1|");  float[,] A = {  { 2.83333f, 5.0f, 1.0f },  { 1.7f, 3.0f, 7.0f },  { 8.0f, 1.0f, 1.0f }};  float[] b = { 11.66666f, 13.4f, 18.0f };  float[] result = GaussMethod.SolveGaussMethod(A, b);  Console.WriteLine("Метод Гаусса");  Console.WriteLine("Решение системы:");  for (int i = 0; i < result.Length; i++)  {  Console.WriteLine($"x{i + 1} = {result[i]:F9}");  }  Console.WriteLine();  result = GaussMethod.SolveGaussMethodRow(A, b);  Console.WriteLine("Метод Гаусса с выбором главного элемента по строке(вариант 2)");  Console.WriteLine("Решение системы:");  for (int i = 0; i < result.Length; i++)  {  Console.WriteLine($"x{i + 1} = {result[i]:F9}");  }  Console.WriteLine();  }  static void part2()  {  Console.WriteLine("|Задание 2|");  // Введите коэффициенты системы уравнений в виде матрицы a \* x = b  float[,] A = {  { 2.0f, 15.0f, 1.0f, 1.0f },  { 3.0f, -1.0f, 2.0f, 20.0f },  { 10.0f, 1.0f, 1.0f, 1.0f },  { 3.0f, 2.0f, 14.0f, -2.0f }};  float[] b = { 21.0f, 27.0f, 23.0f, 20.0f };  // Количество итераций  int iterations = 1000;  // Точность вычислений  float epsilon = 1e-3f;  float[] result = IterativeMethod.SolveIterativeMethod(A, b, epsilon, iterations);  for (int i = 0; i < result.Length; i++)  {  Console.WriteLine($"x{i + 1} = {result[i]:F9}");  }  Console.WriteLine();  }  static void part3()  {  Console.WriteLine("|Задание 3|");  float[] a = { 1, 1, 1, 1, 1, 1, 0 }; // коэф. верхней диагонали  float[] b = { 4, 4, 4, 4, 4, 4, 4 }; // коэф. главной диагонали  float[] c = { 0, 1, 1, 1, 1, 1, 1 }; // коэф. нижней диагонали  float[] d = { 1, 2, 3, 4, 5, 6, 7 }; //вектор свободных членов системы уравнений  float[] result = TridiagonalMatrixSolver.Solve(a, b, c, d);  Console.WriteLine("Решение системы:");  for (int i = 0; i < result.Length; i++)  {  Console.WriteLine($"x[{i}] = {result[i]:F4}");  }  }  static void Main(string[] args)  {  part1();  part2();  part3();  }  }  } |

**Результаты вычислений, полученные с помощью разработанных программ**

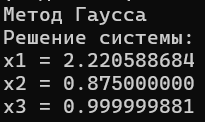


Рисунок 4 – Решение системы методом Гаусса

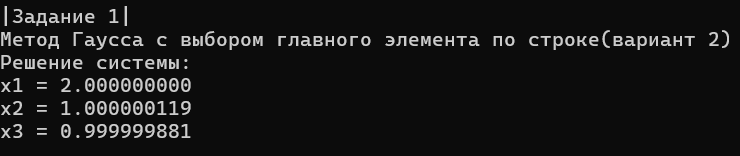


Рисунок 5 – Решение системы методом Гаусса с выбором главного элемента по строке

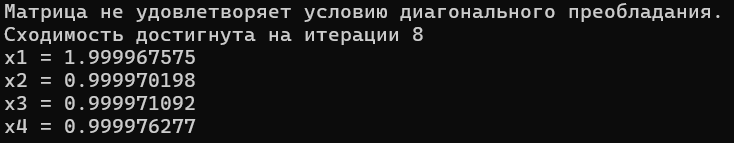
****

Рисунок 6 – Решение системы итерационным методом

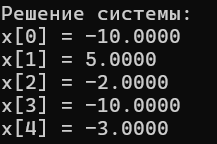


Рисунок 7 – Решение системы с трехдиагональной матрицей

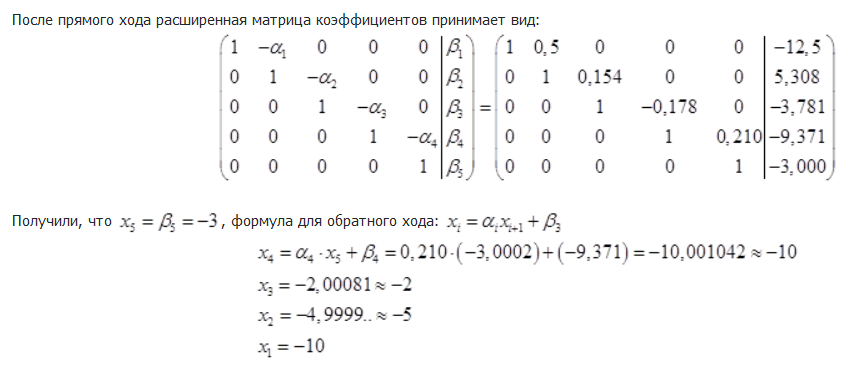
****

Рисунок 8 – Решение системы с трехдиагональной матрицей вручную

**Объяснение полученных результатов**

Решение системы методом Гаусса и методом Гаусса с выбором главного элемента в строке

1. **Метод SolveGaussMethod**:
   * Этот метод применяет классический метод Гаусса без выбора главного элемента. Он выполняет прямой и обратный ход.
   * Прямой ход преобразует систему уравнений к верхнетреугольному виду путем вычитания из одних уравнений других.
   * Обратный ход вычисляет значения неизвестных, начиная с последнего уравнения и последовательно находя значения предыдущих неизвестных.
2. **Метод SolveGaussMethodRow**:
   * Этот метод реализует метод Гаусса с выбором главного элемента в строке.
   * Перед началом преобразований в каждом столбце выбирается строка с максимальным по модулю элементом и перемещается на верхнюю позицию.
   * Затем применяется аналогичный прямой и обратный ход.

После выполнения метода Гаусса для решения системы линейных уравнений, получаем массив solution, который содержит значения, которые являются решениями исходной системы уравнений.

Решение методом Гаусса с выбором главного элемента является более точным по следующим причинам:

* **Предотвращение деления на ноль**: Метод Гаусса с выбором главного элемента предварительно проверяет наличие нулевых или очень малых значений на диагонали матрицы перед делением. Это помогает избежать деления на ноль или очень малое число, что может привести к нестабильности численных вычислений.
* **Повышение устойчивости численных методов**: Метод Гаусса с выбором главного элемента обеспечивает более стабильные и устойчивые вычисления, что приводит к более точным результатам, особенно в случае систем уравнений с малыми отличиями в коэффициентах или большими значениями условных чисел.

Решение системы заданным итерационным методом

1. **Метод SolveIterativeMethod**:
   * Этот метод принимает матрицу a, вектор b, значение точности epsilon и максимальное количество итераций iterations.
   * Сначала он проверяет условие диагонального преобладания матрицы a, а если это условие не выполняется, производит перестановку строк для обеспечения диагонального преобладания.
   * Затем матрица a преобразуется к виду, пригодному для итераций, где x = alpha \* x + beta, где alpha и beta соответствующие матрица и вектор.
   * Проверяется условие сходимости, что норма матрицы alpha должна быть меньше 1. Если это условие не выполняется, метод завершается.
   * Начальное приближение устанавливается в нулевой вектор.
   * Затем осуществляется итерационный процесс: на каждой итерации вычисляется новое приближение newX и проверяется условие сходимости.
   * Если условие сходимости выполнено, возвращается результат. В противном случае, происходит продолжение итераций. Если достигнуто максимальное число итераций, метод завершает работу и возвращает null.
2. **Остальные методы**:
   * CheckDiagonalDominance: Проверяет диагональное преобладание матрицы.
   * CalculateMatrixNorm: Вычисляет норму матрицы.
   * CalculateVectorNorm: Вычисляет норму вектора.
   * SubtractVectors: Выполняет вычитание векторов.
   * EnsureDiagonallyDominant: Обеспечивает диагональное преобладание матрицы.

Результатом работы метода является массив, содержащий приближенное решение системы уравнений. Если метод успешно сходится к решению, результат возвращается с указанием номера итерации, на которой была достигнута сходимость. Если метод не сходится после заданного числа итераций или не выполняются условия диагонального преобладания и нормы матрицы alpha, возвращается null.

Преимущества решения системы итерационным методом:

* **Потенциальная более быстрая сходимость**: В некоторых случаях итерационные методы могут сходиться быстрее, чем прямые методы, особенно если система линейных уравнений имеет определенные свойства, такие как диагональное преобладание или низкую численную обусловленность.
* **Удобство параллелизации**: Некоторые итерационные методы могут легко параллелизоваться, что может привести к ускорению решения задачи.

Решение системы с трехдиагональной матрицей

1. **Метод Solve**:
   * Этот метод принимает четыре массива a, b, c и d, которые представляют диагонали трехдиагональной матрицы.
   * Первый массив a содержит элементы нижней диагонали (ниже главной диагонали).
   * Массив b содержит элементы главной диагонали.
   * Массив c содержит элементы верхней диагонали (выше главной диагонали).
   * Массив d содержит правую часть системы уравнений.
   * Вначале вычисляются коэффициенты P и Q в прямом проходе по матрице.
   * Затем эти коэффициенты используются для вычисления решения x в обратном проходе.
   * Результатом является массив x, содержащий решение системы уравнений.

Результатом работы метода является массив x, который представляет собой решение исходной системы уравнений.

Представленное решение системы с трехдиагональной матрицей было проведено методом прогонки, который имеет следующие преимущества:

* **Эффективность вычислений**: Метод прогонки требует всего O(n) операций для решения трехдиагональной системы уравнений. Это делает его очень эффективным с вычислительной точки зрения, особенно для больших систем.
* **Удобство распараллеливания**: Поскольку операции в методе прогонки выполняются последовательно, это позволяет эффективно распараллеливать алгоритм, что может привести к значительному ускорению вычислений.