**Санкт-Петербургский национальный исследовательский университет**

**информационных технологий, механики и оптики**

**Кафедра информатики и прикладной математики**

Вычислительная математика

Лабораторная работа №1

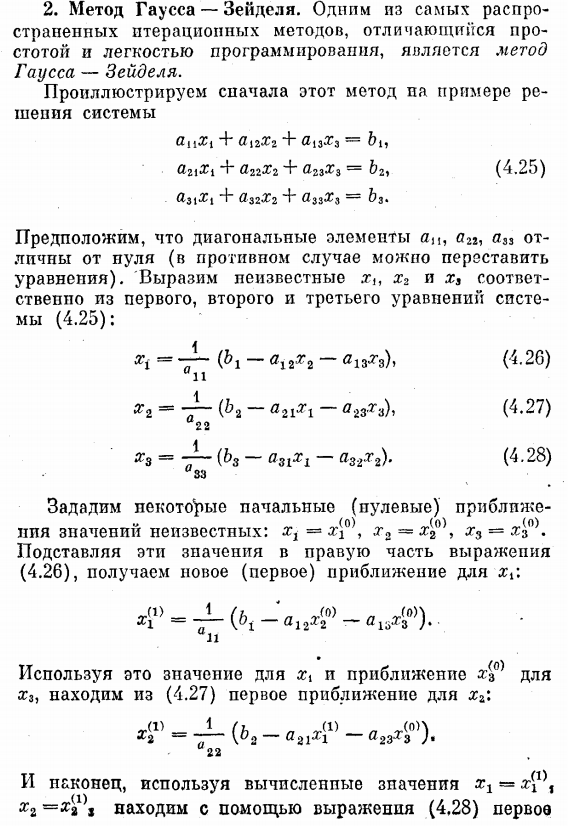
Выполнил: Гхази Даниэль

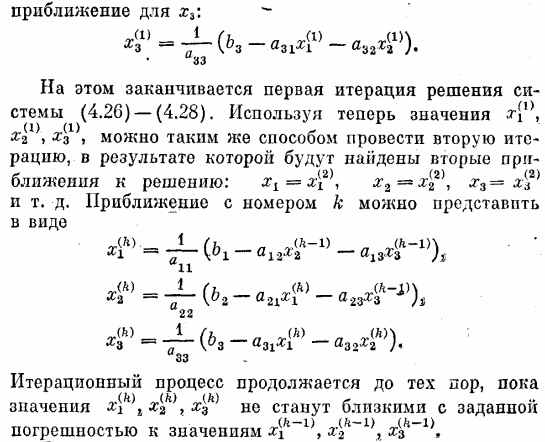
Группа P3218

Преподаватель: Кучер Алексей Владимирович

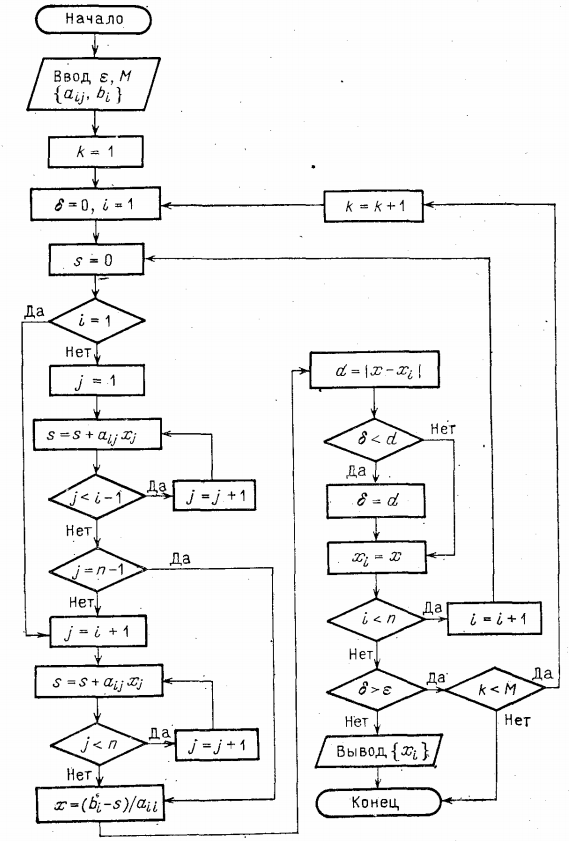
2016 г.

**1. Описание метода**





**2. Блок-схема**

****

**3. Реализация метода**

private static void GaussSeidelMethodRealisation(decimal error, decimal[,] valuesOfLinearSystemLeftSide, int maxIterations, decimal[] valuesOfLinearSystemRightSide, int totalNumberOfRows)

{

int iteration = 1;

decimal[] approximationsOfUnknowns = new decimal[totalNumberOfRows];

decimal delta = 0;

bool overflow = false;

for (; iteration < maxIterations; iteration++)

{

delta = 0;

overflow = computingNewUnknownsApproximations(ref delta, ref approximationsOfUnknowns, iteration, valuesOfLinearSystemLeftSide, overflow, valuesOfLinearSystemRightSide, totalNumberOfRows);

if (overflow) return;

TableForOutput.Deltas.Add(delta);

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

{

TableForOutput.CalculatedUnknowns[iteration - 1, j] = approximationsOfUnknowns[j];

}

if (delta <= error)

{

TableForOutput.ActualNumberOfIterations = iteration;

Converges = true;

return;

}

}

Converges = false;

TableForOutput.ActualNumberOfIterations = iteration - 1;

}

private static bool computingNewUnknownsApproximations(ref decimal delta, ref decimal[] approximationsOfUnknowns, int iteration, decimal[,] valuesOfLinearSystemLeftSide, bool overflow, decimal[] valuesOfLinearSystemRightSide, int totalNumberOfRows)

{

int i = 0;

int j = 0;

decimal sum = 0;

decimal newX = 0;

decimal alpha = 0;

for (; i < totalNumberOfRows; i++)

{

sum = 0;

if (i != 0)

{

overflow = CountFullSum(ref sum, ref j, i, valuesOfLinearSystemLeftSide, approximationsOfUnknowns, iteration, totalNumberOfRows, overflow);

if (overflow) return true;

}

else

{

j = i + 1;

overflow = CountRightSideSum(ref sum, ref j, i, valuesOfLinearSystemLeftSide, approximationsOfUnknowns, iteration, totalNumberOfRows);

if (overflow) return true;

}

overflow = ComputeUnknownApproximation(ref sum, ref j, ref newX, i, valuesOfLinearSystemLeftSide, valuesOfLinearSystemRightSide, iteration);

if (overflow) return true;

alpha = Math.Abs(newX - approximationsOfUnknowns[i]);

if (delta < alpha) delta = alpha;

approximationsOfUnknowns[i] = newX;

}

return false;

}

private static bool CountFullSum(ref decimal sum, ref int j, int i, decimal[,] valuesOfLinearSystemLeftSide, decimal[] approximationsOfUnknowns, int iteration, int totalNumberOfRows,

bool overflow)

{

j = 0;

overflow = CountLeftSideSum(ref sum, ref j, i, valuesOfLinearSystemLeftSide,

approximationsOfUnknowns, iteration);

if (overflow) return true;

if (j != totalNumberOfRows - 1)

{

j = i + 1;

overflow = CountRightSideSum(ref sum, ref j, i, valuesOfLinearSystemLeftSide,

approximationsOfUnknowns, iteration, totalNumberOfRows);

if (overflow) return true;

}

return false;

}

private static bool CountLeftSideSum(ref decimal sum, ref int j, int i, decimal[,] valuesOfLinearSystemLeftSide, decimal[] approximationsOfUnknowns, int iteration)

{

for (; j < i; j++)

{

try

{

sum += valuesOfLinearSystemLeftSide[i,j] \* approximationsOfUnknowns[j];

}

catch (OverflowException)

{

Converges = false;

TableForOutput.ActualNumberOfIterations = iteration - 1;

return true;

}

j++;

}

return false;

}

private static bool CountRightSideSum(ref decimal sum, ref int j, int i, decimal[,] valuesOfLinearSystemLeftSide, decimal[] approximationsOfUnknowns, int iteration, int totalNumberOfRows)

{

for (; j < totalNumberOfRows; j++)

{

try

{

sum += valuesOfLinearSystemLeftSide[i,j] \* approximationsOfUnknowns[j];

}

catch (OverflowException)

{

Converges = false;

TableForOutput.ActualNumberOfIterations = iteration - 1;

return true;

}

j++;

}

return false;

}

private static bool ComputeUnknownApproximation(ref decimal sum, ref int j, ref decimal newX, int i, decimal[,] valuesOfLinearSystemLeftSide, decimal[] valuesOfLinearSystemRightSide, int iteration)

{

try

{

newX = (valuesOfLinearSystemRightSide[i] - sum) / valuesOfLinearSystemLeftSide[i,i];

}

catch (OverflowException)

{

Converges = false;

TableForOutput.ActualNumberOfIterations = iteration - 1;

return true;

}

return false;

}

**4. Примеры работы программы**

**4.1 Сходится**

Матрица 5х5

74.3321 **x**₁ - 31.1199 **x**₂ + 41.4904 **x**₃ - 95.6328 **x**₄ - 48.5143 **x**₅ = - 91.1236

- 3.0085 **x**₁ + 73.6034 **x**₂ - 29.0464 **x**₃ - 94.5516 **x**₄ - 82.3243 **x**₅ = 81.3541

20.4164 **x**₁ - 46.405 **x**₂ - 73.7569 **x**₃ - 7.7658 **x**₄ - 1.434 **x**₅ = - 61.393

- 39.5847 **x**₁ + 47.0947 **x**₂ - 65.577 **x**₃ - 87.8862 **x**₄ - 17.9937 **x**₅ = - 51.5308

- 55.9788 **x**₁ + 41.2718 **x**₂ + 66.6471 **x**₃ - 50.24 **x**₄ - 88.9993 **x**₅ = - 7.4365

Погрешность вычислений: 0.0001

Результат:

№ итерации | погрешность | X1 | X2 | X3



X4 | X5



**4.2 Расходится**

Матрица 5х5

**x**₁ + 2 **x**₂ + 3 **x**₃ + 4 **x**₄ + 234 **x**₅ = 234

**x**₁ + 123 **x**₂ + 3 **x**₃ + 4 **x**₄ + 345 **x**₅ = 62312

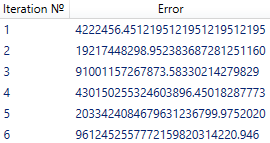
**x**₁ + 2 **x**₂ + 3 **x**₃ + 465 **x**₄ + 123 **x**₅ = 423

**x**₁ + 33532 **x**₂ + 123 **x**₃ + 4 **x**₄ + 4234 **x**₅ = 345

**x**₁ + 234 **x**₂ + 123 **x**₃ + 4 **x**₄ + 123 **x**₅ = 123

Погрешность вычислений: 0.0001

Результат: погрешность увеличивается с каждой итерацией



**5. Вывод**

В ходе работы мною был изучен и реализован метод Гаусса-Зейделя для решения СЛАУ. Также мною были дополнительно изучены метод Гаусса, метод Гаусса с выбором главного элемента и метод простых итераций.

Полученные знания позволили мне узнать о проблемах, возникающих при реализации методов решения СЛАУ на ЭВМ и способах их решения.