Лабораторная работа № 2

«Решение систем линейных алгебраических уравнений»

Работу выполнили:

Воложанин Владик

Клементьев Алексей

Лотуга Данила

Сафин Рамазик

Цель

Разработать программу по решению СЛУ методом Гаусса, модифицировать реализацию алгоритма оптимального исключения неизвестных и реализацию метода Гаусса-Жордана.

Оборудование

1. ПК.
2. ОС Windows.
3. IDE WebStorm.
4. Языки программирования: HTML5. CSS3. JavaScript 3.1.
5. Node.js.
6. Библиотеки для Node.js: Math.js, MathJax, Bootstrap 5.

В качестве основного фреймворка для JavaScript используется Node.js, позволяющий подключать к программе дополнительные библиотеки и расширять функционал языка.

Библиотека Math.js является расширением стандартной библиотеки Math в JavaScript, в данной лабораторной работе она используется для синтаксического анализа и интерпретирования математических выражений.

Библиотека MathJax является кроссбраузерной библиотекой, предназначенной для отображения математических обозначений с использованием языков разметки MathML, LaTeX и ASCIIMathML. В данной лабораторной работе MathJax используется для отображения на страницах сайта математических выражений с помощью LaTeX.

Библиотека Bootstrap 5 является набором шаблонов для HTML разметки и таблиц стилей CSS, используемых для упрощения процесса разработки дизайна функциональных элементов сайта и его оформления.

Постановка задачи

Реализовать алгоритмы Метода Гаусса (оптимального исключения неизвестных, исключения неизвестных по столбцам), Гаусса-Жордана.

Разработать интерфейс взаимодействия с модулями программы, реализующих метод оптимального исключения неизвестных, алгоритм исключения неизвестных по столбцам, Гаусса-Жордана. В качестве интерфейса использовать веб-сайт.

Математическая модель

Исходная система линейных алгебраических уравнений представлена в виде:

$$\begin{cases} a_{11}x_1 + \dots + a_{1n}x_n = b_1 \\ a_{21}x_1 + \dots + a_{2n}x_n = b_2 \\ \vdots \\ a_{m1}x_1 + \dots + a_{mn}x_n = b_m \end{cases}$$

Тогда можно записать эту систему в матричном виде, получив основную матрицу системы A, столбец свободных членов B и столбец X, являющийся корнями этой системы:

$$A = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix}, \qquad X = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}, \qquad B = \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix}$$

Основную матрицу системы следует объединить с столбцом свободных членов, получив расширенную матрицу системы $\tilde{A} = (A|B)$:

$$\tilde{A} = \left(\begin{array}{ccc|c} a_{11} & \cdots & a_{1n} & b_1 \\ \vdots & \ddots & \vdots & \vdots \\ a_{m1} & \cdots & a_{mn} & b_m \end{array}\right)$$

Элементарными преобразованиями над матрицей являются:

- ✓ Перестановка строк (столбцов).
- ✓ Умножение строк (столбцов) на ненулевую константу.
- ✓ Сложение одной строки (столбца) с другой строкой (столбцом).

Верхняя треугольная матрица:

$$\left(\begin{array}{ccc|c} a_{11} & \cdots & a_{1n} & b_1 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & a_{mn} & b_m \end{array}\right)$$

Нижняя треугольная матрица:

$$\left(\begin{array}{ccc|c} a_{11} & \cdots & 0 & b_1 \\ \vdots & \ddots & \vdots & \vdots \\ a_{m1} & \cdots & a_{mn} & b_m \end{array}\right)$$

Диагональная матрица:

$$\left(\begin{array}{ccc|c} a_{11} & \cdots & 0 & b_1 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & a_{mn} & b_m \end{array}\right)$$

1. Алгоритм последовательного исключения неизвестных по столбцам, выполненный по схеме единственного деления.

Расширенная матрица:

$$\tilde{A} = \left(\begin{array}{ccc|c} a_{11} & \cdots & a_{1n} & b_1 \\ \vdots & \ddots & \vdots & \vdots \\ a_{m1} & \cdots & a_{mn} & b_m \end{array}\right)$$

Первый ненулевой элемент каждой i-ой строкой будем называть ведущим элементом.

Для получения верхней треугольной матрицы, воспользуемся элементарными преобразованиями:

- ✓ Нормализовать ведущий элемент первой строки ($j = 1$), используя умножение строки на ненулевой коэффициент $\left(\frac{1}{a_{11}}\right)$.

$$\tilde{a}_{1j} = \frac{a_{1j}}{a_{11}}, \qquad \tilde{b}_1 = \frac{b_1}{a_{11}}$$

- ✓ Сложить каждую следующую строку ($j = 2 \dots m$) с первой, умноженной на -1. Таким образом, получить нули после ведущего элемента в первом столбце.

$$\tilde{a}_{kj} = a_{kj} - \tilde{a}_{1j} \cdot a_{k1}$$
$$\tilde{b}_k = b_k - \tilde{b}_1 \cdot a_{k1}$$

- ✓ Повторить эти два пункта для каждой следующей строки ($j = \overline{2 \dots m}$)

$$\tilde{A} = \left(\begin{array}{ccc|c} \tilde{a}_{11} & \cdots & \tilde{a}_{1n} & \tilde{b}_1 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & \tilde{a}_{mn} & \tilde{b}_m \end{array}\right)$$

Элементы главной диагонали при этом равны единице.

Для нахождения корней СЛАУ, следует воспользоваться рекуррентными формулами:

$$x_n = \tilde{b}_m$$

$$x_i = \tilde{b}_i - \sum_{j=i+1}^{n} \tilde{a}_{ij} x_i, \qquad (i = \overline{n+1 \dots 1})$$

2. СЛУ метод Гаусса (метод оптимального исключения неизвестных).

Для получения верхней треугольной матрицы воспользуемся рекуррентными формулами:

$$\mathrm{c}_{ki} = \frac{a_{ki}}{a_{ii}}$$
$$\tilde{a}_{ki} = a_{kj} - c_{ki} a_{ij}$$

$$i = \overline{1 \dots n-1}, \qquad k = \overline{j+1 \dots n}, \qquad j = \overline{i \dots n+1}$$

Для нахождения корней СЛАУ, следует воспользоваться рекуррентными формулами:

$$x_n = \frac{\tilde{b}_m}{\tilde{a}_{nn}}$$

$$x_i = \left(\tilde{b}_i - \sum_{j=i+1}^{n} \tilde{a}_{ij} x_i \right) / \tilde{a}_{ii}, \qquad \left(i = \overline{n+1 \dots 1}\right)$$

3. СЛУ метода Гаусса-Жордана.

Метод Жордана-Гаусса аналогичен классическому методу. Отличие заключается в том, что матрица приводится не к треугольному, а к диагональному виду. Для этого:

- ✓ Нормализовать ведущий элемент первой строки ($j = 1$), используя умножение строки на ненулевой коэффициент $\left(\frac{1}{a_{11}}\right)$.

$$\tilde{a}_{1j} = \frac{a_{1j}}{a_{11}}, \qquad \tilde{b}_1 = \frac{b_1}{a_{11}}$$

- ✓ Сложить каждую строку, кроме выбранной (т. е. кроме первой $j = 2 \dots m$) с первой, умноженной на -1. Таким образом, получить нули до и после ведущего элемента в первом столбце.

$$\tilde{a}_{kj} = a_{kj} - \tilde{a}_{kj} \cdot a_{ij}$$
$$\tilde{b}_k = b_k - \tilde{b}_i \cdot a_{ki}$$

$$i = \overline{1 \dots n-1}, \qquad k = \overline{1 \dots n} \neq i, \qquad j = \overline{i \dots n+1}$$

- ✓ Повторить для остальных строк.

Корни СЛАУ при этом будут соответствовать столбцу свободных членов.

Программа

Метод Гаусса (алгоритм исключения неизвестных по столбцам).

Код программы:

```
1 function gauss(matrix) {
2   for (let row in matrix) {
3     row = parseInt(row);
4
5     const normalMultiplier = 1 / matrix[row][row];
6     matrix[row] = matrix[row].map(cell => {
7       return math.round(cell * normalMultiplier, 10);
8     });
9
10    for (let i = row + 1; i < matrix.length; i++) {
11      matrix[i] = matrix[i].map((cell, index) => {
12        return cell - (matrix[row][index] * matrix[i][row]);
13      });
14    }
15  }
16  return matrix.map(cell => {return math.round(cell, 10);});
17}
```

Функция принимает на вход расширенную матрицу $\tilde{A}$ и приводит её к верхнему треугольному виду.

На строке 5 инициализируется нормирующий коэффициент, который используется для умножения текущей строки row для получения единицы в ведущем элементе на строках 6-7.

В цикле на строках 10-13 из всех строк после рассматриваемой вычитается строка row, умноженная на ведущий коэффициент этих строк.

Для получения корней СЛАУ, расширенную матрицу необходимо представить в виде основной преобразованной матрицы и преобразованного столбца свободных членов. Следующая функция принимает треугольную расширенную матрицу и возвращает Основную преобразованную матрицу и преобразованный столбец свободных членов:

```
function matrixToLinearEquations(matrix) {
  matrix = math.matrix(matrix);

  const size = matrix.size();

  let A = math.column(matrix, 0);
  for (let i = 1; i < size[1] - 1; i++) {
    A = math.concat(A, math.column(matrix, i));
  }

  let B = math.column(matrix, size[1] - 1);

  return {A: A.toArray(), B: B.toArray()};
}
```

Следующая функция выполняет алгоритм обратного хода, принимая основную матрицу и столбец свободных членов, и возвращая столбец корней СЛАУ:

```
1 function solveLinearEquations(coefficients, results) {
2   for (let i = results.length - 1; i >= 0; i--) {
3     const accumulatedSum = coefficients[i].reduce(
4         function(currentSum, currentNumber, currentIndex) {
5           if (currentIndex === i) { return 0; }
6           return currentSum + (currentNumber * results[currentIndex][0]);
7         }, 0);
8
9     results[i][0] = math.round(
10        (results[i][0] - accumulatedSum) / coefficients[i][i], 10);
11  }
12  return results;
13}
```

В цикле на строках 2-7 вычисляется линейная комбинация коэффициентов из матрицы основных элементов последней строки с подстановкой известных корней, вычисляемых на строках 9-11. Для первой итерации эта сумма равна нулю, так как цикл идет по строкам снизу-вверх. Для всех последующих строк массив `results` уже содержит необходимые корни и цикл корректно считает накопленную сумму, вычитаемую из столбца свободных членов (соответствующего строке элемента).
Алгоритм универсален как для случаев с ненормированной диагональю (оптимизированного алгоритма Гаусса), так и для случаев с нормированной диагональю и метода Жордана-Гаусса. В случае, если диагональ нормированная, в `results` добавляется свободных член, деленный на единицу.

Метод Гаусса (метод оптимального исключения неизвестных).

Код программы:

```
1 function gaussOptimalExclusion(matrix) {
2   for (let row in matrix) {
3     row = parseInt(row);
4
5     for (let i = row + 1; i < matrix.length; i++) {
6       const multiplier = -matrix[i][row] / matrix[row][row];
7
8       matrix[i] = matrix[i].map((cell, index) => {
9         return cell + (matrix[row][index] * multiplier);
10      });
11    }
12  }
13  return matrix.map(cell => {return math.round(cell, 10);});
14}
```

Функция прямого хода программы практически аналогична классическому методу. В цикле на строках 5-12 вычисляется коэффициент (для каждого элемента рассматриваемого столбца), на который умножается текущая строка `row` для сложения с каждой последующей.

Получение корней СЛАУ аналогично выполняется с помощью функций `matrixToLinearEquations` и `gaussOptimalExclusion`.

Метод Гаусса-Жордана.

Код программы:

```
1 function gaussJordan(matrix) {
2   matrix = gauss(matrix);
3   for (let row = matrix.length - 1; row >= 0; row--) {
4     for (let i = row - 1; i >= 0; i--) {
5       matrix[i] = matrix[i].map((cell, index) => {
6         return cell - (matrix[row][index] * matrix[i][row]);
7       });
8     }
9   }
10  return matrix.map(cell => {return math.round(cell, 10);});
11}
```

Функция использует вызов классического метода Гаусса для приведения матрицы к верхнему треугольному виду и единичной главной диагональю на строке 2. А затем использует этот же алгоритм в обратном порядке (снизу-вверх) для обнуления всех элементов основной матрицы выше главной диагонали в цикле на строках 3-9.

Получение корней СЛАУ аналогично выполняется с помощью функций `matrixToLinearEquations` и `gaussOptimalExclusion`.

Результаты работы программы

КАЛЬКУЛЯТОР-АПРОКСИМАТОР

Метод Гаусса

$$\tilde{A} = \left(\begin{array}{cccc|c} 5 & 7 & 6 & 5 & 23 \\ 7 & 10 & 8 & 7 & 32 \\ 6 & 8 & 10 & 9 & 33 \\ 5 & 7 & 9 & 10 & 31 \end{array}\right) \sim \left(\begin{array}{cccc|c} 1 & 1.4 & 1.2 & 1 & 4.6 \\ 0 & 1 & -2 & 0 & -1 \\ 0 & 0 & 1 & 1.5 & 2.5 \\ 0 & 0 & 0 & 1 & 1 \end{array}\right)$$

$$X = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

Метод Гаусса (метод оптимального исключения)

$$\tilde{A} = \left(\begin{array}{cccc|c} 5 & 7 & 6 & 5 & 23 \\ 7 & 10 & 8 & 7 & 32 \\ 6 & 8 & 10 & 9 & 33 \\ 5 & 7 & 9 & 10 & 31 \end{array}\right) \sim \left(\begin{array}{cccc|c} 5 & 7 & 6 & 5 & 23 \\ 0 & 0.2 & -0.4 & 0 & -0.2 \\ 0 & 0 & 2 & 3 & 5 \\ 0 & 0 & 0 & 0.5 & 0.5 \end{array}\right)$$

$$X = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

Метод Гаусса — Жордана

$$\tilde{A} = \left(\begin{array}{cccc|c} 5 & 7 & 6 & 5 & 23 \\ 7 & 10 & 8 & 7 & 32 \\ 6 & 8 & 10 & 9 & 33 \\ 5 & 7 & 9 & 10 & 31 \end{array}\right) \sim \left(\begin{array}{cccc|c} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{array}\right)$$

$$X = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

Метод Гаусса — Жордана

$$\tilde{A} = \left(\begin{array}{cccc|c} 5 & 7 & 6 & 5 & 23.01 \\ 7 & 10 & 8 & 7 & 32 \\ 6 & 8 & 10 & 9 & 33.001 \\ 5 & 7 & 9 & 10 & 31 \end{array}\right) \sim \left(\begin{array}{cccc|c} 1 & 0 & 0 & 0 & 1.663 \\ 0 & 1 & 0 & 0 & 0.6 \\ 0 & 0 & 1 & 0 & 0.835 \\ 0 & 0 & 0 & 1 & 1.097 \end{array}\right)$$