

Университет ИТМО
Мегафакультет компьютерных технологий и управления
Факультет программной инженерии и компьютерной техники



ЛАБОРАТОРНАЯ РАБОТА №1
РЕШЕНИЕ СИСТЕМЫ ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ
УРАВНЕНИЙ
Вариант №10

Группа: Р3211
Студент: Орчиков Даниил Валерьевич
Преподаватель: Малышева Татьяна Алексеевна

г. Санкт-Петербург
2022

Оглавление

Цель работы	2
Описание используемого метода	2
Расчетные формулы	2
Листинг программы.....	2
Примеры и результаты работы программы	2
Пример 1	5
Пример 2	5
Пример 3	6

Цель работы

Изучить прямые и итерационные методы решения систем линейных алгебраических уравнений, выполнить программную реализацию методов.

Описание используемого метода

1. Рассмотрим систему линейных уравнений с невырожденной матрицей
2. Выразим неизвестные x_1, x_2, \dots, x_n соответственно из первого, второго и т. д. уравнений системы
3. Обозначим $c_{ij} = \begin{cases} 0, & \text{при } i = j \\ -\frac{a_{ij}}{a_{ii}}, & \text{при } i \neq j \end{cases} \quad d_i = \frac{b_i}{a_{ii}}, \quad i = 1, 2, \dots, n$
4. Получим в векторно-матричном виде $x = Cx + D$, где x – вектор неизвестных, C – матрица коэффициентов c_{ij} , D – вектор коэффициентов d_i
Или в сокращенном виде: $x_i = \sum_{j=1}^n c_{ij}x_j + d_i$

Достаточным условием сходимости итерационного процесса к решению системы при любом начальном векторе $x_i^{(0)}$ является выполнение условия преобладания диагональных элементов:

$$|a_{ii}| \geq \sum_{j \neq i} |a_{ij}|, \quad i = 1, 2, \dots, n$$

Расчетные формулы

$$x_i^{k+1} = \frac{b_i}{a_{ii}} - \sum_{\substack{j=1 \\ j \neq i}}^n \frac{a_{ij}}{a_{ii}} x_j^k, \quad i = 1, 2, \dots, n \quad k - \text{номер итерации}$$

Листинг программы

Представлен только код, непосредственно выполняющий вычисления

Весь код можно посмотреть [здесь \(GitHub\)](#)

```
function checkDiagonalDominance(matrix) {  
    // Смотрим в каждой ли строке есть элемент модуль которого больше суммы модулей остальных элементов в строке  
    // Также смотрим возможно ли будет переставить строки так что эти элементы встанут на диагональ  
    let f = new Array(n).fill(0);
```

```

for (let i = 0; i < n; i++) {
  let max_value = 0
  let ind = -1
  for (let j = 0; j < n; j++) {
    if (Math.abs(matrix[i][j]) > max_value) {
      max_value = Math.abs(matrix[i][j])
      ind = j
    }
  }
  if (ind !== -1) f[ind] = true
  if (matrix[i].reduce((partialSum, a) => partialSum + Math.abs(a), 0) - Math.abs(max_value) > Math.abs(max_value)) {
    return false; // выполнение достаточного условия невозможно
  }
}
return f.reduce((flag, a) => flag && a) // имеет ли смысл пытаться переставлять строки?
}

function calc() {
  readMatrix()
  let variable_matrix = []
  let variable_right_parts = []
  for (let i = 0; i < n; i++) {
    variable_matrix.push(...matrix[i])
  }
  variable_right_parts.push(...right_parts)
  // ----
  // Смотрим возможно ли диагональное преобладание и если возможно - добиваемся этого с помощью переставления строк
  // Иначе пытаемся убрать с диагонали все нули так как в противном случае на следующем шаге будет деление на ноль
  if (!checkDiagonalDominance(variable_matrix)) {
    document.getElementById("message").innerHTML = "Достигнуть диагонального преобладания невозможно<br>"
    // ----
    // Пробуем убрать нули с диагонали
    for (let i = 0; i < n; i++) {
      let ind = 0
      for (let j = 0; j < n; j++) {
        if (variable_matrix[i][j] !== 0 && variable_matrix[j][i] !== 0) {
          ind = j
          break
        }
      }
      variable_matrix[i] = [variable_matrix[ind], variable_matrix[ind] = variable_matrix[i]][0]
      variable_right_parts[i] = [variable_right_parts[ind], variable_right_parts[ind] = variable_right_parts[i]][0]
    }
    let f = true
    for (let i = 0; i < n; i++)
      if (!variable_matrix[i][i]) f = false
    if (!f) { // Не получилось убрать с диагонали все нули :(
      clear()
      document.getElementById("message").innerHTML += "Система несовместна"
      return
    }
    // ----
  } else {
    // Добиваемся диагонального преобладания переставляя строки
    for (let i = 0; i < n; i++) {
      let max_value = 0
      let ind = -1
      for (let j = 0; j < n; j++) { // Находим индекс максимального элемента в строке
        if (Math.abs(variable_matrix[i][j]) >= max_value) {
          max_value = Math.abs(variable_matrix[i][j])
          ind = j
        }
      }
      // переставляем строку так, чтобы максимальный элемент в ней встал на диагональ
      if (ind !== -1) {
        variable_matrix[i] = [variable_matrix[ind], variable_matrix[ind] = variable_matrix[i]][0]
        variable_right_parts[i] = [variable_right_parts[ind], variable_right_parts[ind] = variable_right_parts[i]][0]
      }
      if (ind !== i)

```

```

        i = Math.min(i, ind) - 1
    }
}
// ----
// Рисуем матрицу с переставленными строками
let m = ""
for (let i = 0; i < n; i++) {
    m += "<div class='row'>"
    for (let j = 0; j < n; j++) {
        m += '<input type='number' class='cell' value='${variable_matrix[i][j]}' readonly>'
    }
    m += '<input type='number' class='cell right_part' value='${variable_right_parts[i]}' readonly></div>'
}
document.getElementById("new_matrix").innerHTML = m
// ----
// Выражаем неизвестные x1, x2, ..., xn (делим строку на элемент на диагонали)
for (let i = 0; i < n; i++) {
    let mid = variable_matrix[i][i]
    for (let j = 0; j < n; j++) {
        variable_matrix[i][j] /= -mid
    }
    variable_matrix[i][i] = 0
    variable_right_parts[i] /= mid
}
// ----
// Итерационно вычисляем значения
let x_n = []
document.getElementById("init_approx").querySelectorAll("*").forEach(input => {
    x_n.push(parseInt(input.value, 10)) // считываем начальные аппроксимации
})
let x_n1 = new Array(n)

let v = 0
while (true) {
    v += 1
    for (let i = 0; i < n; i++) {
        let a = 0
        for (let j = 0; j < n; j++) {
            a += variable_matrix[i][j] * x_n[j]
        }
        a += variable_right_parts[i]
        x_n1[i] = a
    }
    let max_deviation = 0
    for (let i = 0; i < n; i++) { // вычисляем максимальную погрешность
        if (Math.abs(x_n[i] - x_n1[i]) > max_deviation)
            max_deviation = Math.abs(x_n[i] - x_n1[i])
    }
    if (max_deviation < document.getElementById("accuracy").value) // если достигнута нужная точность - выходим из цикла
        break
    document.getElementById("answer").innerHTML = ""
    if (v >= document.getElementById("maximum_number_of_iterations").value) { // если достигнуто максимально количество
итераций - выходим из цикла
        document.getElementById("answer").innerHTML = "Не удалось достигнуть требуемой точности за отведенное количество
итераций<br>"
        break
    }
    x_n = [...x_n1]
    x_n1 = new Array(n)
}
// ----
printAnswer(x_n, x_n1, v)
}

```

Примеры и результаты работы программы

Пример 1

Размер матрицы - 4

Точность - 0.00001

Максимальное число итераций - 200

Начальные приближения - 1 2 3 4

1 0 1 4 5

6 1 1 3 0

0 9 4 4 6

1 1 3 0 1

1	0	1	4	5
6	1	1	3	0
0	9	4	4	6
1	1	3	0	1

6	1	1	3	0
0	9	4	4	6
1	1	3	0	1
1	0	1	4	5

x1 = -0.7104214878574268

x2 = -0.17760269907834514

x3 = 0.6293468195831238

x4 = 1.2702725600141807

Число итераций: 38

Вектор погрешностей: [-0.000007825140916795092, -0.00000844667268262711, -0.000007746641503159069, -0.000005560029524787069]

Пример 2

Размер матрицы - 4

Точность - 0.00001

Максимальное число итераций - 200

Максимальное число итераций - 1 1 1 1

-3 1 2 2 -5

3 0 -3 3 -1

-4 2 -5 3 -3

-3 0 3 2 1

-3	1	2	2	-5
3	0	-3	3	-1
-4	2	-5	3	-3
-3	0	3	2	1

Достигнуть диагонального преобладания невозможно

-3	0	3	2	1
-3	1	2	2	-5
3	0	-3	3	-1
-4	2	-5	3	-3

Не удалось достигнуть требуемой точности за отведенное количество итераций

$x_1 = 6.282402354692587e+73$

$x_2 = -7.650656382743055e+73$

$x_3 = 7.379839751996119e+73$

$x_4 = 1.1004296752862616e+74$

Число итераций: 200

Вектор погрешностей: $[-3.600391037972714e+73, 4.3845257148266015e+73, -4.2293230208210294e+73, -6.306468317586369e+73]$

Пример 3

Размер матрицы - 3

Точность - 0.001

Максимальное число итераций - 200

Максимальное число итераций - 0 0 0

1 2 0 1

3 4 0 2

5 6 0 1

1	2	0	1
3	4	0	2
5	6	0	1

Матрица несовместна