Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Национальный исследовательский университет ИТМО»
Факультет программной инженерии и компьютерной техники

Вариант №11
Лабораторная работа №1
по дисциплине
Вычислительная математика

Выполнил Студент группы P3112
Пархоменко Кирилл Александрович
Преподаватель:
Наумова Надежда Александровна

г. Санкт-Петербург
2024г.

Содержание

1 Задание

В программе реализуемый численный метод решения системы линейных алгебраических уравнений (СЛАУ) должен быть реализован в виде отдельного класса /метода/функции, в который исходные/выходные данные передаются в качестве параметров.
Задавать размерность матрицы ($n < 20$) из файла или с клавиатуры – по выбору конечного пользователя.
Должна быть реализована возможность ввода коэффициентов матрицы, как с клавиатуры, так и из файла (по выбору конечного пользователя).
Сформировать не менее 3 файлов (тестов) с различным набором данных.
Программа должна быть протестирована на различных наборах данных, в том числе и некорректных.

1.1 Вариант

Для прямых методов должно быть реализовано:

- Вычисление определителя
- Вывод треугольной матрицы (включая преобразованный столбец B)
- Вывод вектора неизвестных: $x_1, x_2, \ldots, x_n$
- Вывод вектора невязок: $r_1, r_2, \ldots, x_n$

Метод	№ варианта
Метод Гаусса	1, 3, 5, 8, 21, 24, 26, 28, 31, 39
Метод Гаусса с выбором главного элемента по столбцам	11, 17, 19, 22, 25, 27, 30, 34, 40
Метод простых итераций	2, 4, 6, 7, 10, 13, 15, 23, 32, 36, 38
Метод Гаусса-Зейделя	9, 12, 14, 16, 18, 20, 29, 33, 35, 37

1.2 Цель работы

Реализовать программу для численного метода решения систем линейных уравнений, включая возможность ввода коэффициентов матрицы, вычисления определителя, вывода треугольной матрицы, вектора неизвестных и вектора невязок.

1.3 Описание метода, расчетные формулы

Описание метода
Схема с выбором главного элемента является одной из модификаций метода Гаусса.

Среди ведущих элементов могут оказаться очень маленькие по абсолютной величине. При делении на такие ведущие элементы получается большая погрешность округления (вычислительная погрешность). Идеей метода Гаусса с выбором главного элемента является такая перестановка уравнений, чтобы на k-ом шаге исключения ведущим элементом aii оказывался наибольший по модулю элемент k-го столбца. Т.е. на очередном шаге k в уравнениях, начиная от k до последнего ($i = k, k+1, \ldots, n$) в столбце k выбирают максимальный по модулю элемент и строки i и k меняются местами. Это выбор главного элемента «по столбцу». Выбор главного элемента «по строке» - на очередном шаге k в строке k, начиная со столбца k ($j = k, k+1, \ldots, n$) справа выбирается максимальный по модулю элемент. Столбцы j и k меняются местами.

Алгоритм метода Гаусса с выбором главного элемента:
Инициализация: Получаем матрицу расширенной системы линейных уравнений (с учетом правой части системы) и определяем количество неизвестных (n).
Прямой ход: Проходим по каждой строке матрицы (i от 1 до n-1) и для каждой строки выбираем главный элемент. Главный элемент выбирается как элемент с наибольшим по модулю значением в текущем столбце (начиная с i-го элемента). Если главный элемент не находится в i-й строке, производится перестановка строк так, чтобы главный элемент оказался на i-й позиции. Затем применяется операция преобразования строк (вычитание соответствующего кратного i-й строки из остальных строк), чтобы обнулить все элементы под главным элементом в текущем столбце.
Обратный ход: Начиная с последнего уравнения и двигаясь к первому, решаем систему уравнений обратным ходом, находя значения неизвестных последовательно.

Расчетные формулы

$$\det(\mathbf{A}) = (-1)^k \prod_{i=1}^{n} a_{ii}$$

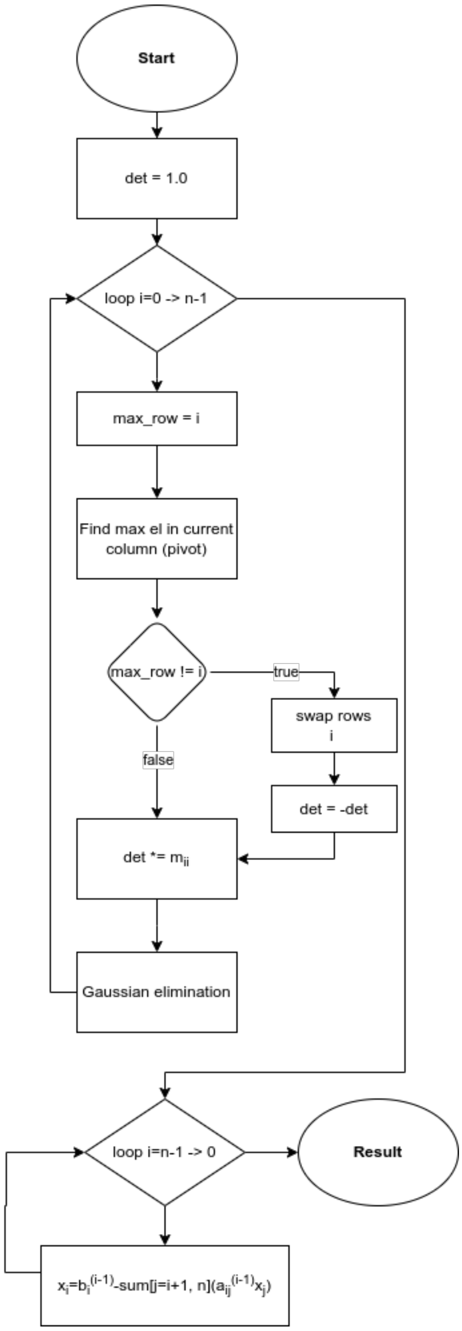
$$x_n = \frac{b_n^{n-1}}{a_{nn}^{n-1}}$$

1.4 Реализация численного метода

```cpp
const double EPSILON = 1e-10;

void print_residual_vector(Matrix &m, std::vector<double> &result_vec) {
  std::cout << "Residual vector:" << std::endl;
  if (result_vec.size() != m.get_size()) {
    std::cout << "Error: size of residual vector is not equal to size of matrix"
          << std::endl;
    return;
  }
  for (int i = 0; i < m.get_size(); i++) {
    double res = 0;
    for (int j = 0; j < m.get_size(); j++) {
      res += m[i][j] * result_vec[j];
    }
    double r = m[i][m.get_size()] - res;
    std::cout << "r" << i + 1 << " = " << r
          << std::endl;
  }
}

std::vector<double> gauss_elimination(Matrix &m) {
  const int n = m.get_size();
  std::cout << "Matrix before Gaussian elimination:" << std::endl;
  m.print();

  Matrix old_m = m.copy();

  double det = 1.0; // Initialize determinant

  for (int i = 0; i < n; i++) {
    double max_el = std::fabs(m[i][i]);
    int max_row = i;

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

```cpp
      if (std::fabs(m[k][i]) > max_el) {
        max_el = std::fabs(m[k][i]);
        max_row = k;
      }
    }

    // Swap current row with the row containing maximum element
    if (max_row != i) {
      m.swap_rows(i, max_row);
      det *= -1; // Change the sign of determinant for row swaps
    }

    // Check if the pivot element is too close to zero
    if (std::fabs(m[i][i]) < EPSILON) {
      det = 0;
      // Skip this column, it's effectively zero
      continue;
    }

    // Update determinant
    det *= m[i][i];

    // Perform Gaussian elimination to make elements below the pivot zero
    for (int k = i + 1; k < n; k++) {
      double factor = m[k][i] / m[i][i];
      for (int j = i; j <= n; j++) {
        m[k][j] -= factor * m[i][j];
      }
    }
  }
  std::cout << "Matrix determinant: " << pretty_double(det, 3) << std::endl;
  if (det == 0) {
    std::cout << "Determinant is zero, no solution exists." << std::endl;
    return {};
  }
  std::cout << "------------------------------------------------------------\n";

  std::cout << "Matrix after Gaussian elimination:" << std::endl;
  m.print();

  // Back substitution to solve for unknowns
  std::vector<double> result(n);
  for (int i = n - 1; i >= 0; i--) {
    result[i] = m[i][m.get_size()];
    for (int j = i + 1; j < n; j++) {
      result[i] -= m[i][j] * result[j];
    }
    result[i] /= m[i][i]; // divide by diagonal element
  }

  print_residual_vector(old_m, result);

  return result;
}
```

1.5 Блок-схема реализованного алгоритма



1.6 Ссылка на GitHub с кодом

Github

1.7 Примеры и результаты работы программы

```
..[$] <( (git)-[master]-)> ./lab1_cpp -f ./inputs/input1.txt
Matrix before Gaussian elimination:
        10        -7         0 |         7
        -3         2         6 |         4
         5        -1         5 |         6
Matrix determinant: -155
----------------------------------------------------------
Matrix after Gaussian elimination:
        10        -7         0 |         7
         0     2.500         5 |     2.500
         0         0     6.200 |     6.200
----------------------------------------------------------
Residual vector:
r1 = 0
r2 = 8.88178e-16
r3 = 0
----------------------------------------------------------
Solution vector:
x1 = 0
x2 = -1
x3 = 1
```

```
..[$] <( (git)-[master]-)> ./lab1_cpp -g 5
Matrix before Gaussian elimination:
-59.545197 -81.601106 -13.354673 -10.119931 -77.589964 |  73.033585
 -8.799412 -43.968320 -60.328926 -56.091867  24.982067 | -36.624516
 40.370736 -53.646620  39.973830  96.782456  42.139063 | -59.141431
 51.126955 -40.426541  33.577755  69.566332  15.368800 | -68.673279
-72.201013 -28.571951  84.128820 -34.447234 -27.900062 | -55.955262
Matrix determinant: -3277596396.756
----------------------------------------------------------
Matrix after Gaussian elimination:
-72.201013 -28.571951  84.128820 -34.447234 -27.900062 | -55.955262
        -0 -69.622444  87.013924  77.521504  26.538922 | -90.428459
         0          0 -155.271888 -46.332897 -76.703289 | 194.562032
         0          0          0 -60.812807  72.812564 | -129.065353
         0          0          0          0 -69.051804 |  50.669845
----------------------------------------------------------
Residual vector:
r1 = 4.26326e-14
r2 = -7.10543e-15
r3 = -7.10543e-15
r4 = 1.42109e-14
r5 = -7.10543e-15
----------------------------------------------------------
Solution vector:
x1 = -1.33229
x2 = 0.82714
x3 = -1.26168
x4 = 1.24375
x5 = -0.73379
```

```
● ..[$] <( (git)-[master]-)> ./lab1_cpp -i
Enter precision (ex: `5`): 4
Enter matrix size (ex: `3`): 3
Enter matrix elements row by row (ex: `1 2 3`): 1 2 3 4
2 3 4 5
1 1 34 5
Matrix before Gaussian elimination:
         1          2          3 |          4
         2          3          4 |          5
         1          1         34 |          5
Matrix determinant: -33
-----------------------------------------------------------
Matrix after Gaussian elimination:
         2          3          4 |          5
         0     0.5000          1 |     1.5000
         0          0         33 |          4
-----------------------------------------------------------
Residual vector:
r1 = 0
r2 = 0
r3 = 0
-----------------------------------------------------------
Solution vector:
x1 = -1.87879
x2 = 2.75758
x3 = 0.12121
```

2 Заключение

Я познакомился с новым методом для решения СЛАУ и создал его программную реализацию.

Список литературы

[1] Слайды с лекций (2023). // Кафедра информатики и вычислительной техники – Малышева Татьяна Алексеевна, к.т.н., доцент.