Дисциплина: Численные методы

Лабораторное задание №3

Отчет

Тема: Численные методы решения спектральных задач линейной алгебры

|  |  |
| --- | --- |
|  | Выполнили:  студенты 3 курса 8 группы  Крутько А.С.  Сикарев Р.О.  Проверила:  старший преподаватель  Фролова О.А. |

Оглавление

[Постановка задачи 3](#_Toc91112033)

[Теоретическая часть 4](#_Toc91112034)

[Алгоритм 6](#_Toc91112035)

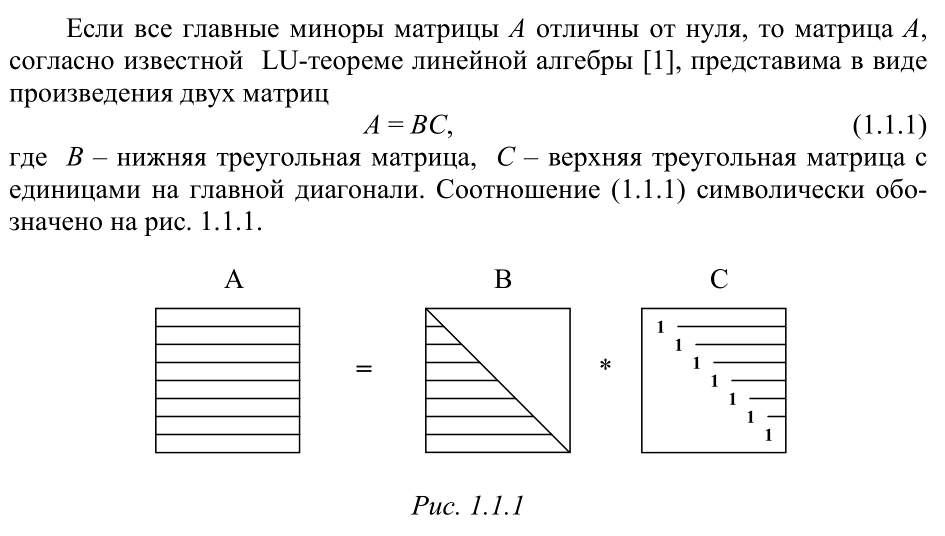
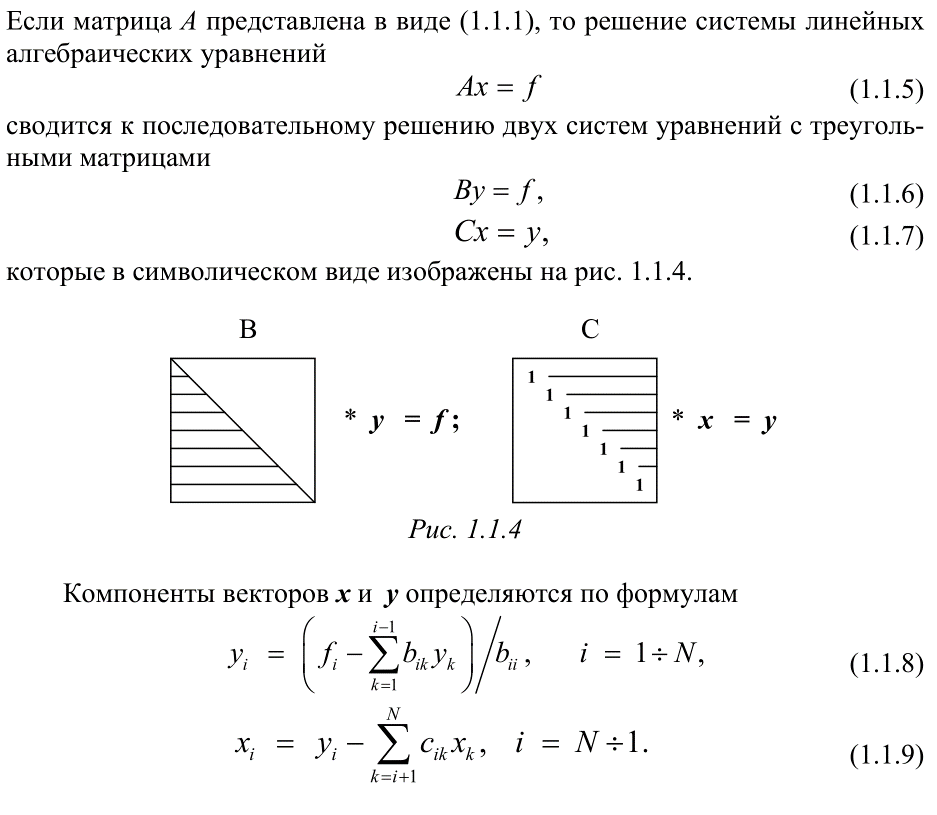
[Тестирование 9](#_Toc91112036)

Постановка задачи

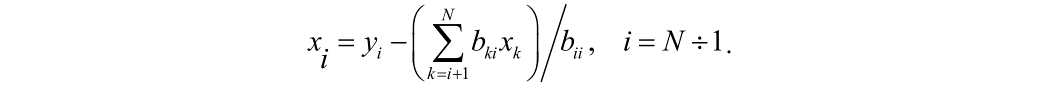
Составить программу, которая, используя нижеописанный метод решения задачи, определяет пару с третьим минимальным по модулю собственным значением симметричной матрицы простой структуры. При выполнении данной задачи нами был использован метод обратной итерации с исчерпыванием определения пары, причем для решения линейной системы был использован метод Халецкого.

Теоретическая часть

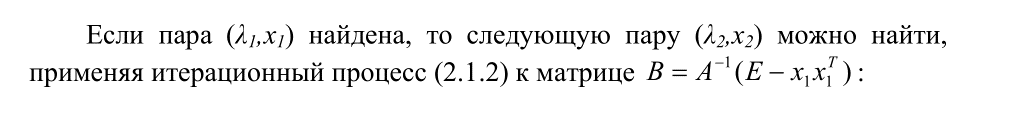
Метод решения СЛАУ Халецкого:

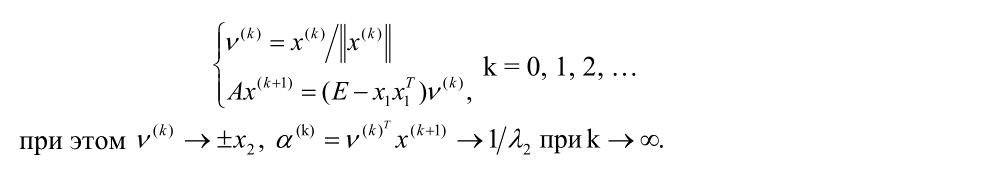
 

В случае же симметричной матрицы элемент ищется по формуле:



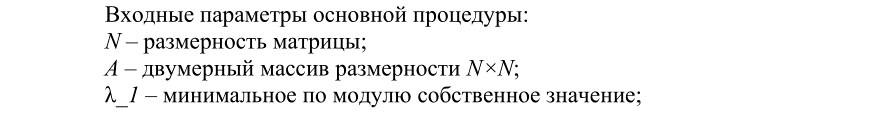
Метод обратных итераций с исчерпыванием:

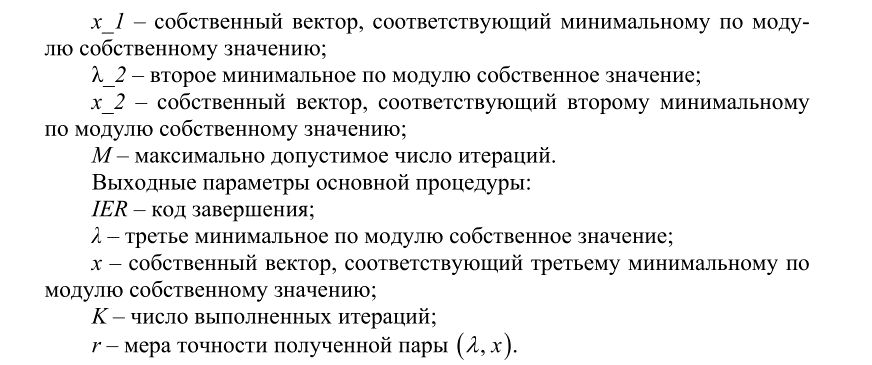




При этом, если для решения системы уравнений с матрицей применяется один из методов матрицы , то один раз найденное используется и для определения пары и для определения пары ().

Алгоритм





Будем считать, что собственные значения пронумерованны в порядке возрастания их модулей, т.е.

На каждом шаге итерации, используя метод Халецкого, находится решение системы линейных уравнений:

Итерационный процесс прекращается в одном из двух случаев:

* Достигнуты, требуемые по условию, точности определения собственного значения и собственного вектора
* Число итераций превысило максимально допустимое значение

const int SIZE = A.colsCount();

Matrix<> f\_1(1, SIZE, 1);

Matrix<> f\_2(1, SIZE);

Matrix<> B(SIZE); // Нижнетреугольная матрица LU-разложения

Matrix<> C(SIZE); // Верхнетреугольная матрица LU-разложения

Matrix<> E(SIZE, SIZE, 1); // Единичная матрица

Matrix<> e(1, SIZE, 1); // Единичный вектор

// Производим LU-разложение, в случае успеха запускаем процесс итерации

bool continueToWork = LU(A, B, C);

int count = 0;

double a1 = 1, a2;

while (continueToWork) {

Matrix<> y(1, SIZE), x(1, SIZE);

// Если удаось найти Y, ищем X

if (findY(B, f\_1, y)) {

findX(C, y, x);

}

// Запоминаем предыдущее собственное значение, начиная со второй итерации

if (count >= 2) {

a2 = a1;

}

a1 = vectProd(f\_1, x);

// Запоминаем предыдущее значение собственного вектора, начиная со второй итерации

if (count >= 2) {

f\_2 = f\_1;

}

f\_1 = x;

// Нормируем вектор

normalize(f\_1);

count++;

// Начиная с третьей итерации

if (count >= 3) {

// Если мера собственный достигла требуемой точности, запоминаем потребовавшееся количество итераций

if (abs(a1 - a2) < eps\_l) {

k\_l = count;

}

// Если угол между векторами достиг требуемой точности, запоминаем потребовавшееся количество итераций

if (abs(acos(cosBetweenVectors(f\_1, f\_2))) < eps\_v) {

k\_v = count;

if (k\_l == -1) {

k\_l = count;

}

// Завершаем процесс итерации

continueToWork = false;

}

}

// Завершаем процесс итераций, если превышено максимальное их количество

if (count >= M) {

continueToWork = false;

}

}

// Если удалось достичь требуемой точности, записываем полученные

if (k\_v != -1) {

lambda = 1 / a2;

xn = f\_1;

}

Исходя из полученных данных, формируем значения, говорящие о точности решения:

const int SIZE = A.colsCount();

Matrix<> b(1, SIZE);

b = A \* x;

// Записываем отклонени я векторов Ax − λx от нуля

for (int i = 0; i < SIZE; i++) {

b(0, i) = b(0, i) - l \* x(0, i);

}

r = b(0, 0);

// Находим максимальное отклонение среди полученных

for (int i = 1; i < SIZE; i++) {

if (b(0, i) > r) {

r = b(0, i);

}

}

Тестирование

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № Теста | Размерность системы N | Диапазон значений λ | Точность  () | Ср. оценка точности собственных значений | Ср. оценка точности собственных векторов | Средняя мера точности r | Среднее число итерации |
| 1 | 10 | -3÷3 |  | 7.4e-9 | 5.3e-4 | 3.21e-6 | 87 |
| 2 | 10 | -3÷3 |  | 7.8e-12 | 2.3e-6 | 3.11e-6 | 92 |
| 3 | 10 | -60÷60 |  | 1.25e-8 | 2.02e-3 | 6.87e-5 | 101 |
| 4 | 10 | -6÷60 |  | 3.31e-11 | 4.98e-5 | 6.68e-5 | 112 |
| 5 | 30 | -3÷3 |  | 7.98e-7 | 2.23e-4 | 1.26e-5 | 306 |
| 6 | 30 | -3÷3 |  | 5.67e-9 | 2.13e-4 | 7.23e-3 | 381 |
| 7 | 30 | -60÷60 |  | 1.24e-6 | 3.73e-3 | 4.31e-4 | 791 |
| 8 | 30 | -60÷60 |  | 4.23e-10 | 4.84e-3 | 7.98e-4 | 1243 |
| 9 | 60 | -3÷3 |  | 5.01e-7 | 3.87e-2 | 6.98e-3 | 341 |
| 10 | 60 | -3÷3 |  | 3.41e-8 | 3.82e-3 | 7.11e-3 | 542 |
| 11 | 60 | -60÷60 |  | 3.73e-6 | 9.28e-2 | 4.21e-2 | 1534 |
| 12 | 60 | -60÷60 |  | 3.21e-6 | 6.87e-2 | 8.98e-2 | 3214 |