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

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:

Если все главные миноры матрицы A отличны от нуля, то матрица A, согласно известной LU-теореме линейной алгебры [1], представима в виде произведения двух матриц

$$A = BC, \tag{1.1.1}$$

где B – нижняя треугольная матрица, C – верхняя треугольная матрица с единицами на главной диагонали. Соотношение (1.1.1) символически обозначено на рис. 1.1.1.

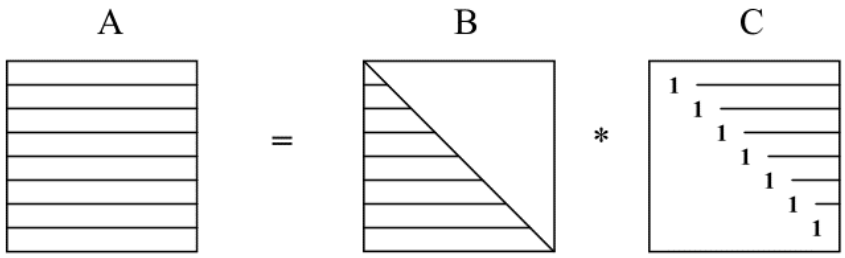

Рис. 1.1.1

Если матрица A представлена в виде (1.1.1), то решение системы линейных алгебраических уравнений

$$Ax = f \tag{1.1.5}$$

сводится к последовательному решению двух систем уравнений с треугольными матрицами

$$By = f, \tag{1.1.6}$$

$$Cx = y, \tag{1.1.7}$$

которые в символическом виде изображены на рис. 1.1.4.

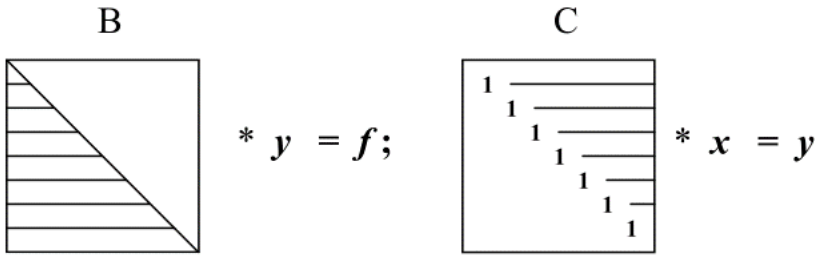

Рис. 1.1.4

Компоненты векторов $\boldsymbol{x}$ и $\boldsymbol{y}$ определяются по формулам

$$y_i = \left(f_i - \sum_{k=1}^{i-1} b_{ik} y_k \right) \Big/ b_{ii}, \quad i = 1 \div N, \tag{1.1.8}$$

$$x_i = y_i - \sum_{k=i+1}^{N} c_{ik} x_k, \quad i = N \div 1. \tag{1.1.9}$$

В случае же симметричной матрицы элемент x_i ищется по формуле:

$$x_i = y_i - \left(\sum_{k=i+1}^{N} b_{ki} x_k \right) \Big/ b_{ii}, \quad i = N \div 1.$$

Метод обратных итераций с исчерпыванием:

Если пара (λ_1, x_1) найдена, то следующую пару (λ_2, x_2) можно найти, применяя итерационный процесс (2.1.2) к матрице $B = A^{-1}(E - x_1 x_1^T)$:

$$\begin{cases} \nu^{(k)} = x^{(k)} \Big/ \left\| x^{(k)} \right\| \\ A x^{(k+1)} = (E - x_1 x_1^T) \nu^{(k)}, \end{cases} \quad \mathrm{k} = 0, 1, 2, \ldots$$

при этом $\nu^{(k)} \to \pm x_2$, $\alpha^{(\mathrm{k})} = \nu^{(k)^T} x^{(k+1)} \to 1/\lambda_2$ при $\mathrm{k} \to \infty$.

При этом, если для решения системы уравнений с матрицей A применяется один из методов $LU-$разложения матрицы A, то один раз найденное $LU-$разложение используется и для определения пары (λ_1, x_1) и для определения пары (λ_2, x_2).

Алгоритм

Входные параметры основной процедуры:

N – размерность матрицы;

A – двумерный массив размерности $N \times N$;

λ_1 – минимальное по модулю собственное значение;

x_1 – собственный вектор, соответствующий минимальному по модулю собственному значению;

λ_2 – второе минимальное по модулю собственное значение;

x_2 – собственный вектор, соответствующий второму минимальному по модулю собственному значению;

M – максимально допустимое число итераций.

Выходные параметры основной процедуры:

IER – код завершения;

λ – третье минимальное по модулю собственное значение;

x – собственный вектор, соответствующий третьему минимальному по модулю собственному значению;

K – число выполненных итераций;

r – мера точности полученной пары (λ, x).

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

$$|\lambda_1| < |\lambda_2| < \cdots < |\lambda_n|$$

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

$$Ax^{(k+1)} = (E - x_1 x_1^T)$$

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 ($\varepsilon_\lambda = \varepsilon_g$)	Ср. оценка точности собственных значений	Ср. оценка точности собственных векторов	Средняя мера точности r	Среднее число итераций
1	10	-3÷3	10^{-5}	7.4e-9	5.3e-4	3.21e-6	87
2	10	-3÷3	10^{-8}	7.8e-12	2.3e-6	3.11e-6	92
3	10	-60÷60	10^{-5}	1.25e-8	2.02e-3	6.87e-5	101
4	10	-6÷60	10^{-8}	3.31e-11	4.98e-5	6.68e-5	112
5	30	-3÷3	10^{-5}	7.98e-7	2.23e-4	1.26e-5	306
6	30	-3÷3	10^{-8}	5.67e-9	2.13e-4	7.23e-3	381
7	30	-60÷60	10^{-5}	1.24e-6	3.73e-3	4.31e-4	791
8	30	-60÷60	10^{-8}	4.23e-10	4.84e-3	7.98e-4	1243
9	60	-3÷3	10^{-5}	5.01e-7	3.87e-2	6.98e-3	341
10	60	-3÷3	10^{-8}	3.41e-8	3.82e-3	7.11e-3	542
11	60	-60÷60	10^{-5}	3.73e-6	9.28e-2	4.21e-2	1534
12	60	-60÷60	10^{-8}	3.21e-6	6.87e-2	8.98e-2	3214