

**МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
имени М.В. ЛОМОНОСОВА**

Механико-математический факультет

Кафедра вычислительной математики

К.Ю. Богачев

Практикум на ЭВМ.

**Методы решения линейных систем и
нахождения собственных значений**

Москва 1998 год

Содержание

ПРЕДИСЛОВИЕ	6
Глава I. ТОЧНЫЕ МЕТОДЫ РЕШЕНИЯ ЛИНЕЙНЫХ СИСТЕМ	7
§1. МАТРИЧНЫЕ НОРМЫ	7
§2. ОБРАТИМОСТЬ МАТРИЦЫ, БЛИЗКОЙ К ОБРАТИМОЙ МАТРИЦЕ	12
§3. ОШИБКИ В РЕШЕНИЯХ ЛИНЕЙНЫХ СИСТЕМ	12
§4. МЕТОД ГАУССА	15
§4.1. Алгоритм метода Гаусса	15
§4.2. Оценка количества арифметических операций в методе Гаусса	17
§4.3. Представление метода Гаусса в виде последовательности элементарных преобразований	18
§4.4. Алгоритм построения LU -разложения	19
§4.5. Оценка количества арифметических операций в алгоритме построения LU -разложения	21
§4.6. Осуществимость метода Гаусса	21
§5. МЕТОДЫ ПОСЛЕДОВАТЕЛЬНОГО ИСКЛЮЧЕНИЯ НЕИЗВЕСТНЫХ ДЛЯ ЛЕНТОЧНЫХ МАТРИЦ	22
§5.1. Метод Гаусса для ленточных матриц	22
§5.2. Алгоритм LU -разложения для трехдиагональных матриц .	23
§5.3. Метод прогонки для трехдиагональных матриц	24
§6. ЗАДАЧА ОБРАЩЕНИЯ МАТРИЦЫ	26
§7. МЕТОД ГАУССА С ВЫБОРОМ ГЛАВНОГО ЭЛЕМЕНТА	27
§8. МЕТОД ЖОРДАНА (ГАУССА-ЖОРДАНА)	31
§9. ПОЛОЖИТЕЛЬНО ОПРЕДЕЛЕННЫЕ МАТРИЦЫ	33
§10. МЕТОД ХОЛЕЦКОГО (КВАДРАТНОГО КОРНЯ)	35
§10.1. Разложение Холецкого	35
§10.2. Алгоритм построения разложения Холецкого	36
§10.3. Оценка количества арифметических операций в алгоритме построения разложения Холецкого	39

§11. МЕТОД ОРТОГОНАЛИЗАЦИИ	39
§12. МЕТОД ВРАЩЕНИЙ	43
§12.1. Матрица элементарного вращения и ее свойства	43
§12.2. Алгоритм метода вращений	46
§12.3. Оценка количества арифметических операций в методе вращений	48
§12.4. Построение QR -разложения методом вращений	50
§12.5. Оценка количества арифметических операций в алгоритме построения QR -разложения методом вращений	51
§13. МЕТОД ОТРАЖЕНИЙ	52
§13.1. Матрица отражения и ее свойства	53
§13.2. Алгоритм метода отражений	55
§13.3. Оценка количества арифметических операций в методе отражений	58
§13.4. Построение QR -разложения методом отражений	59
§13.5. Оценка количества арифметических операций в алгоритме построения QR -разложения методом отражений	61
§14. ПРИВЕДЕНИЕ МАТРИЦЫ К ПОЧТИ ТРЕУГОЛЬНОМУ ВИДУ УНИТАРНЫМ ПОДОБИЕМ МЕТОДОМ ВРАЩЕНИЙ	62
§14.1. Случай произвольной матрицы	62
§14.2. Случай симметричной матрицы	67
§15. ПРИВЕДЕНИЕ МАТРИЦЫ К ПОЧТИ ТРЕУГОЛЬНОМУ ВИДУ УНИТАРНЫМ ПОДОБИЕМ МЕТОДОМ ОТРАЖЕНИЙ	71
§15.1. Случай произвольной матрицы	71
§15.2. Случай самосопряженной матрицы	75
Глава II. МЕТОДЫ НАХОЖДЕНИЯ СОБСТВЕННЫХ ЗНАЧЕНИЙ	79
§1. ТОЧНЫЕ И ИТЕРАЦИОННЫЕ МЕТОДЫ	79
§2. ЛОКАЛИЗАЦИЯ СОБСТВЕННЫХ ЗНАЧЕНИЙ	80
§3. ОШИБКИ ПРИ НАХОЖДЕНИИ СОБСТВЕННЫХ ЗНАЧЕНИЙ	82
§4. СТЕПЕННОЙ МЕТОД	83
§4.1. Описание алгоритма	83
§4.2. Оценка количества арифметических операций на один шаг алгоритма	85
§5. МЕТОД ВРАЩЕНИЙ ЯКОБИ	85
§5.1. Описание алгоритма	85
§5.2. Выбор угла вращения	87
§5.3. Стратегии выбора обнуляемого элемента	89

§5.3.1.	Метод вращений с выбором максимального элемента	90
§5.3.2.	Метод вращений с циклическим выбором обнуляемого элемента	91
§5.3.3.	Метод вращений с выбором оптимального элемента	92
§6.	МЕТОД БИСЕКЦИИ	93
§6.1.	Алгоритм вычисления k -го по величине собственного значения методом бисекции	94
§6.2.	Алгоритм вычисления всех собственных значений на заданном интервале методом бисекции	94
§6.2.1.	Рекурсивный алгоритм	94
§6.2.2.	Алгоритм последовательного поиска собственных значений	95
§6.3.	Алгоритм вычисления всех собственных значений методом бисекции	95
§6.4.	Вычисление числа перемен знака в последовательности главных миноров	95
§6.4.1.	Вычисление числа перемен знака в последовательности главных миноров с помощью LU -разложения	96
§6.4.2.	Вычисление числа перемен знака в последовательности главных миноров с помощью реккурентных формул	96
§7.	<i>LR</i> АЛГОРИТМ	98
§7.1.	<i>LR</i> -разложение, используемое в <i>LR</i> алгоритме	98
§7.1.1.	Алгоритм построения <i>LR</i> -разложения для произвольной матрицы	98
§7.1.2.	Алгоритм построения <i>LR</i> -разложения для почти треугольной матрицы	100
§7.1.3.	Алгоритм построения <i>LR</i> -разложения для трехдиагональной матрицы	101
§7.2.	<i>LR</i> алгоритм нахождения собственных значений	101
§7.2.1.	<i>LR</i> алгоритм нахождения собственных значений для почти треугольной матрицы	102
§7.2.2.	<i>LR</i> алгоритм нахождения собственных значений для трехдиагональной матрицы	103
§7.3.	Ускорение сходимости алгоритма	104
§7.3.1.	Исчерпывание матрицы	105
§7.3.2.	Сдвиги	106
§7.3.3.	Практическая организация вычислений в <i>LR</i> алгоритме	106

§8. МЕТОД ХОЛЕЦКОГО	107
§8.1. Разложение Холецкого, используемое в методе Холецкого	107
§8.1.1. Алгоритм построения разложения Холецкого для произвольной самосопряженной матрицы	107
§8.1.2. Алгоритм построения разложения Холецкого для трехдиагональной матрицы	109
§8.2. Метод Холецкого нахождения собственных значений	110
§8.2.1. Метод Холецкого нахождения собственных значений для трехдиагональной матрицы	111
§8.3. Ускорение сходимости алгоритма	112
§8.3.1. Исчерпывание матрицы	112
§8.3.2. Сдвиги	113
§8.3.3. Практическая организация вычислений в методе Холецкого	113
§9. QR АЛГОРИТМ	114
§9.1. QR-разложение, используемое в QR алгоритме	114
§9.1.1. Алгоритм построения QR-разложения для произвольной матрицы	114
§9.1.2. Алгоритм построения QR-разложения для почти треугольной матрицы	114
§9.1.3. Алгоритм построения QR-разложения для трехдиагональной матрицы	120
§9.2. QR алгоритм нахождения собственных значений	124
§9.2.1. QR алгоритм нахождения собственных значений для почти треугольной матрицы	125
§9.2.2. QR алгоритм нахождения собственных значений для самосопряженной трехдиагональной матрицы	126
§9.3. Ускорение сходимости алгоритма	129
§9.3.1. Исчерпывание матрицы	129
§9.3.2. Сдвиги	129
§9.3.3. Практическая организация вычислений в QR алгоритме	130
§10. МЕТОД ОБРАТНОЙ ИТЕРАЦИИ НАХОЖДЕНИЯ СОБСТВЕННЫХ ВЕКТОРОВ	130
ПРОГРАММА КУРСА	133
ЛИТЕРАТУРА	137

ПРЕДИСЛОВИЕ

Настоящее пособие содержит описания алгоритмов, предлагаемых к реализации на ЭВМ студентам механико-математического факультета МГУ на занятиях по "Практикуму на ЭВМ". Для всех алгоритмов приводится необходимое теоретическое обоснование, соответствующие расчетные соотношения и рекомендации по их практическому осуществлению на ЭВМ (организация процесса вычислений, хранения данных и результатов в памяти ЭВМ и т.п.).

Многообразие алгоритмов объясняется, с одной стороны, необходимостью обеспечить преподавателей достаточным набором задач для проведения занятий, а с другой стороны, желанием продемонстрировать различные подходы к решению задачи решения линейных систем и нахождения собственных значений матриц. Алгоритмы требуют разных вычислительных затрат, имеют разную чувствительность к погрешностям во входных данных, их свойства по-разному зависят от числа обусловленности матрицы. "Самого лучшего" метода решения поставленной задачи не существует, и выбор алгоритма зависит от конкретной задачи. Этот выбор будет различным, например, для симметричных и несимметричных матриц, для трехдиагональных и заполненных матриц, и т.д.

Подбор алгоритмов для Практикума диктовался, в основном, возможностью реализации их студентами при существующих ресурсах времени на ЭВМ, что привело к отказу от рассмотрения усложненных подходов. Часть описанных алгоритмов вытеснена из широкой вычислительной практики более эффективными (и более сложными) алгоритмами, рассмотреть которые в курсе "Практикум на ЭВМ" не представляется возможным. Тем не менее, эти алгоритмы представляют интерес для решения определенного круга задач и включены в пособие.

Форма отчетности студентов по Практикуму призвана стимулировать как развитие практических навыков решения математических задач с помощью компьютера, так и создание определенного кругозора в области существующих методов решения поставленной задачи. Поэтому в рамках Практикума студентам предлагается как разработать программу на ЭВМ, реализующую заданный алгоритм, так и письменно ответить хотя бы на половину вопросов из предложенного варианта, составленного из вопросов, приведенных в конце пособия.

В основе настоящего пособия лежат материалы лекций, читавшихся автором в течении 4-х лет в рамках факультативного курса "Практикум на ЭВМ". В электронном варианте оно уже более 5-ти лет используется при проведении занятий со студентами в дисплейном классе.

Предложения, замечания и отмеченные опечатки прошу сообщать автору на кафедру вычислительной математики.

Глава I.

ТОЧНЫЕ МЕТОДЫ РЕШЕНИЯ ЛИНЕЙНЫХ СИСТЕМ

§ 1. МАТРИЧНЫЕ НОРМЫ

Обозначим через \mathbf{M}_n пространство (кольцо) матриц размера $n \times n$ над полем \mathbf{C} .

Определение. Нормой на кольце \mathbf{M}_n называется неотрицательный функционал $\|\cdot\| : \mathbf{M}_n \rightarrow \mathbf{R}_+^1$, удовлетворяющий следующим условиям для всех $A, B \in \mathbf{M}_n$:

- 1) $\|A\| = 0 \Leftrightarrow A = 0$,
- 2) $\|\lambda A\| = |\lambda| \|A\|$ для всех $\lambda \in \mathbf{C}$,
- 3) $\|A + B\| \leq \|A\| + \|B\|$,
- 4) $\|AB\| \leq \|A\| \|B\|$.

Простейшие свойства нормы:

- 1) $\|I\| \geq 1$ (где I означает единичную матрицу).

Действительно, $\|I\| = \|I \cdot I\| \leq \|I\| \|I\| = \|I\|^2$.

- 2) $\|A^{-1}\| \geq \frac{\|I\|}{\|A\|}$ и, в частности, $\|A^{-1}\| \|A\| \geq 1$.

Доказательство: $\|I\| = \|AA^{-1}\| \leq \|A\| \|A^{-1}\|$.

Поскольку \mathbf{M}_n можно рассматривать как векторное пространство размерности n^2 , то на нем можно вести векторные нормы, некоторые из которых оказываются матричными. Для таких норм в проверке нуждается только свойство 4) (поскольку первые три свойства составляют определение векторной нормы).

Пример. $\|A\|_E = \|(a_{ij})_{i,j=1,\dots,n}\|_E = \left(\sum_{i,j=1}^n |a_{ij}|^2 \right)^{1/2}$ является матричной нормой. Проверим свойство 4):

$$\begin{aligned} \|AB\|_E^2 &= \sum_{i,j=1}^n \left| \sum_{k=1}^n a_{ik} b_{kj} \right|^2 \leq \sum_{i,j=1}^n \left(\sum_{k=1}^n |a_{ik}|^2 \right) \left(\sum_{m=1}^n |b_{mj}|^2 \right) = \\ &= \left(\sum_{i,k=1}^n |a_{ik}|^2 \right) \left(\sum_{j,m=1}^n |b_{mj}|^2 \right) = \|A\|_E \|B\|_E. \end{aligned}$$

Упражнение. Проверить, что выражение $\|A\|_{\ell_\infty} = \max_{1 \leq i, j \leq n} |a_{ij}|$ не является матричной нормой. Указание: рассмотреть матрицу

$$A = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}.$$

Упражнение. Проверить, что выражение $\|A\| = n \|A\|_{\ell_\infty}$ является матричной нормой.

Определение. Пусть $\|\cdot\|$ есть векторная норма на пространстве \mathbf{C}^n . Определим матричную норму $\|\cdot\|$ на \mathbf{M}_n формулой

$$\|A\| = \max_{\|x\|=1} \|Ax\|.$$

Такая норма называется *подчиненной*, или *операторной*, или *индуцированной* по отношению к векторной норме $\|\cdot\|$.

Проверим вначале, что

$$\|A\| = \max_{\|x\|=1} \|Ax\| = \max_{x \neq 0} \frac{\|Ax\|}{\|x\|}.$$

Действительно,

$$\begin{aligned} \max_{x \neq 0} \frac{\|Ax\|}{\|x\|} &= \max_{x \neq 0} \|A \frac{x}{\|x\|}\| \leq \max_{\|y\|=1} \|Ay\| = \|A\|, \\ \|A\| &= \max_{\|x\|=1} \|Ax\| = \max_{\|x\|=1} \frac{\|Ax\|}{\|x\|} \leq \max_{x \neq 0} \frac{\|Ax\|}{\|x\|} \end{aligned}$$

Теорема 1. Так определенный функционал на \mathbf{M}_n действительно является матричной нормой, которая обладает следующими свойствами:

$$\begin{aligned} \|Ax\| &\leq \|A\| \|x\| \quad \text{для всех } A \in \mathbf{M}_n, x \in \mathbf{C}^n, \\ \|I\| &= 1. \end{aligned}$$

Доказательство. Проверим выполнение свойств 1)-4) из определения матричной нормы.

- 1) $\|A\| = \max_{\|x\|=1} \|Ax\| \geq 0$;
- 2) $\|\lambda A\| = \max_{\|x\|=1} \|\lambda Ax\| = \max_{\|x\|=1} |\lambda| \|Ax\| = |\lambda| \max_{\|x\|=1} \|Ax\| = |\lambda| \|A\|$.
- 3) $\|A + B\| = \max_{\|x\|=1} \|(A + B)x\| = \max_{\|x\|=1} \|Ax + Bx\| \leq \max_{\|x\|=1} (\|Ax\| + \|Bx\|) \leq \max_{\|x\|=1} \|Ax\| + \max_{\|x\|=1} \|Bx\| = \|A\| + \|B\|$.
- 4) $\|AB\| = \max_{x \neq 0} \frac{\|ABx\|}{\|x\|} = \max_{x \neq 0} \frac{\|ABx\|}{\|Bx\|} \frac{\|Bx\|}{\|x\|} \leq \max_{y \neq 0} \frac{\|Ay\|}{\|y\|} \max_{x \neq 0} \frac{\|Bx\|}{\|x\|} \leq \|A\| \|B\|$.

Проверим выполнение дополнительных свойств этой нормы.

$$\|A\| = \max_{x \neq 0} \frac{\|Ax\|}{\|x\|}. \text{ Следовательно, для всех } x \in \mathbf{C}^n \quad \frac{\|Ax\|}{\|x\|} \leq \|A\|, \text{ т.е.}$$

$$\|Ax\| \leq \|A\| \|x\|.$$

$$\|I\| = \max_{\|x\|=1} \|Ix\| = \max_{\|x\|=1} \|x\| = 1.$$

Определение. *Максимальной столбцовой нормой* называется

$$\|A\|_1 = \max_{1 \leq j \leq n} \sum_{i=1}^n |a_{ij}|.$$

Лемма 1. *Норма $\|A\|_1$ подчинена векторной норме $\|x\|_1 = \sum_{i=1}^n |x_i|$.*

Доказательство. Надо проверить, что $\max_{1 \leq j \leq n} \sum_{i=1}^n |a_{ij}| = \max_{\|x\|_1=1} \|Ax\|_1$. Запишем матрицу через ее столбцы: $A = [a_1, \dots, a_n]$, где $a_j = (a_{1j}, \dots, a_{nj})^t$. Тогда $\|A\|_1 = \max_{1 \leq j \leq n} \|a_j\|_1$. С другой стороны, для $x = (x_1, \dots, x_n)^t \in \mathbf{C}^n$

$$\|Ax\|_1 = \left\| \sum_{j=1}^n x_j a_j \right\|_1 \leq \sum_{i=1}^n |x_i| \|a_j\|_1 \leq \sum_{j=1}^n |x_j| \max_{k=1, \dots, n} \|a_k\|_1 = \|x\|_1 \|A\|_1.$$

Следовательно, $\max_{\|x\|_1=1} \|Ax\|_1 = \max_{x \neq 0} \frac{\|Ax\|_1}{\|x\|_1} \leq \|A\|_1$.

Если максимум $\max_{1 \leq i \leq n} \|a_i\|_1 = \|A\|_1$ достигается при $i = i_0$, то, выбрав x равным e_{i_0} - стандартному координатному орту, получаем

$$\max_{\|x\|_1=1} \|Ax\|_1 \geq \|Ae_{i_0}\|_1 = \|a_{i_0}\|_1 = \|A\|_1.$$

Отсюда вытекает требуемое равенство $\max_{1 \leq j \leq n} \sum_{i=1}^n |a_{ij}| = \max_{\|x\|_1=1} \|Ax\|_1$.

Определение. *Максимальной строчкой нормой* называется

$$\|A\|_\infty = \max_{1 \leq i \leq n} \sum_{j=1}^n |a_{ij}|.$$

Лемма 2. *Норма $\|A\|_\infty$ подчинена векторной норме $\|x\|_\infty = \max_{i=1, \dots, n} |x_i|$.*

Задача. Доказать самостоятельно.

Доказательство. Имеем для всех $x \in \mathbf{C}^n$:

$$\|Ax\|_\infty = \max_{1 \leq i \leq n} \left| \sum_{j=1}^n a_{ij} x_j \right| \leq \max_{1 \leq i \leq n} \sum_{j=1}^n |a_{ij}| |x_j| \leq \max_{1 \leq i \leq n} \sum_{j=1}^n |a_{ij}| \|x\|_\infty = \|A\|_\infty \|x\|_\infty.$$

Следовательно, $\max_{\|x\|_\infty=1} \|Ax\|_\infty \leq \|A\|_\infty$.

Пусть максимум $\max_{1 \leq i \leq n} \sum_{j=1}^n |a_{ij}| = \|A\|_\infty$ достигается при $i = i_0$: $\|A\|_\infty = \sum_{j=1}^n |a_{i_0 j}|$. Рассмотрим вектор $y = (y_1, \dots, y_n)^t$ с компонентами

$$y_j = \begin{cases} \frac{\bar{a}_{i_0 j}}{|a_{i_0 j}|}, & \text{если } a_{i_0 j} \neq 0, \\ 1, & \text{если } a_{i_0 j} = 0. \end{cases}$$

Тогда $\|y\|_\infty = 1$, $y_j a_{i_0 j} = |a_{i_0 j}|$, $j = 1, \dots, n$. Поэтому

$$\begin{aligned} \max_{\|x\|_\infty=1} \|Ax\|_\infty &\geq \|Ay\|_\infty = \max_{1 \leq i \leq n} \left| \sum_{j=1}^n a_{ij} y_j \right| \geq \\ &\left| \sum_{j=1}^n a_{i_0 j} y_j \right| = \left| \sum_{j=1}^n |a_{i_0 j}| \right| = \sum_{j=1}^n |a_{i_0 j}| = \|A\|_\infty, \end{aligned}$$

т.е. $\max_{\|x\|_\infty=1} \|Ax\|_\infty \geq \|A\|_\infty$. С учетом доказанного выше это означает требуемое равенство $\max_{\|x\|_\infty=1} \|Ax\|_\infty = \|A\|_\infty$.

Определение. Спектральной нормой называется

$$\|A\|_2 = \max\{\sqrt{\lambda} : \lambda \text{ собственное значение матрицы } A^*A\}.$$

(здесь A^* означает матрицу, сопряженную к матрице A : $A^* = (\bar{A})^t$ в комплексном случае и транспонированную матрицу A^t в вещественном случае). Отметим, что это определение корректно, т.е. указанные квадратные корни всегда существуют. Действительно, для всякого собственного значения λ матрицы A^*A и соответствующего ему собственного вектора x из равенства $A^*Ax = \lambda x$ вытекает $(A^*Ax, x) = \lambda(x, x)$, откуда $\lambda = \frac{\|Ax\|_2}{\|x\|_2}$, и потому $\lambda \in \mathbf{R}$, $\lambda \geq 0$ (здесь (\cdot, \cdot) означает евклидово скалярное произведение в \mathbf{C}^n).

Лемма 3. Норма $\|A\|_2$ подчинена евклидовой векторной норме $\|x\|_2 = (\sum_{i=1}^n |x_i|^2)^{1/2}$.

Доказательство. Рассмотрим

$$\max_{\|x\|_2=1} \|Ax\|_2 = \max_{\|x\|_2=1} (Ax, Ax)^{1/2} = \max_{\|x\|_2=1} (A^*Ax, x)^{1/2}.$$

Обозначим $B = A^*A$. Матрица B самосопряжена (симметрична в вещественном случае): $B^* = (A^*A)^* = A^*(A^*)^* = A^*A = B$. Как показывается в курсе линейной алгебры для матрицы B существует такая унитарная (ортогональная в вещественном случае) матрица U , что $B = U^*\Lambda U$, где $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_n)$, λ_j - собственные значения матрицы $B = A^*A$. Поэтому

$$\max_{\|x\|_2=1} \|Ax\|_2 = \max_{\|x\|_2=1} (U^*\Lambda U x, x)^{1/2} = \max_{\|x\|_2=1} (\Lambda U x, U x)^{1/2}.$$

Поскольку матрица $U^{-1} = U^*$ является унитарной (ортогональной в вещественном случае) и не изменяет евклидовой длины векторов : $\|U^{-1}x\|_2 = \|x\|_2$ для всех x , то (заменив в последнем неравенстве $y = Ux$)

$$\max_{\|x\|_2=1} \|Ax\|_2 = \max_{\|U^{-1}y\|_2=1} (\Lambda y, y)^{1/2} = \max_{\|y\|_2=1} (\Lambda y, y)^{1/2} = \max_{\|y\|_2=1} \left(\sum_{j=1}^n \lambda_j |y_j|^2 \right)^{1/2}.$$

Пусть λ_{j_0} - максимальное из λ_j . Тогда

$$\max_{\|x\|_2=1} \|Ax\|_2 \leq \sqrt{\lambda_{j_0}} \max_{\|y\|_2=1} \left(\sum_{j=1}^n |y_j|^2 \right)^{1/2} = \sqrt{\lambda_{j_0}} = \|A\|_2.$$

С другой стороны, если e_{j_0} есть j_0 -й координатный орт, то

$$\max_{\|x\|_2=1} \|Ax\|_2 = \max_{\|y\|_2=1} \left(\sum_{j=1}^n \lambda_j |y_j|^2 \right)^{1/2} \geq \left(\sum_{j=1}^n \lambda_j |e_{j_0,j}|^2 \right)^{1/2} = \sqrt{\lambda_{j_0}} = \|A\|_2.$$

Из последних двух соотношений вытекает требуемое равенство

$$\max_{\|x\|_2=1} \|Ax\|_2 = \|A\|_2.$$

Определение. Спектральным радиусом матрицы A называется

$$\rho(A) = \max\{|\lambda| : \lambda \text{ - собственное значение матрицы } A\}.$$

Лемма 4. Для всякой матрицы $A \in M_n$ и всякой матричной нормы на M_n справедливо неравенство $\rho(A) \leq \|A\|$.

Доказательство. Пусть λ - максимальное по модулю собственное значение матрицы A , т.е. $|\lambda| = \rho(A)$, x - соответствующий собственный вектор, т.е. $Ax = \lambda x$. Рассмотрим матрицу $X \in M_n$, все столбцы которой равны собственному вектору x . Тогда $AX = \lambda X$. Для всякой матричной нормы $\|\cdot\|$ имеем $|\lambda| \|X\| = \|\lambda X\| = \|AX\| \leq \|A\| \|X\|$, следовательно, $|\lambda| = \rho(A) \leq \|A\|$.

Замечание 1. Если $A = A^*$, то $\rho(A) = \|A\|_2$. Это следует непосредственно из определения спектральной нормы и того, что собственные значения матрицы $A^*A = A^2$ равны квадратам собственных значений матрицы A .

Определение. Унитарно инвариантной матричной нормой называется матричная норма $\|\cdot\|$, удовлетворяющая равенству $\|A\| = \|UAV\|$ для всех матриц $A \in M_n$ и всех унитарных матриц $U, V \in M_n$.

Лемма 5. Спектральная норма является унитарно инвариантной.

Доказательство. Пусть A - произвольная матрица из M_n , U, V - произвольные унитарные матрицы. Собственные значения матрицы

$$(UAV)^*(UAV) = V^*A^*U^*UAV = V^{-1}A^*U^{-1}UAV = V^{-1}A^*AV$$

те же, что и у матрицы A^*A . Следовательно, спектральные нормы матриц UAV и A совпадают.

§ 2. ОБРАТИМОСТЬ МАТРИЦЫ, БЛИЗКОЙ К ОБРАТИМОЙ МАТРИЦЕ

Теорема 1. Пусть $\|\cdot\|$ - матричная норма на \mathbf{M}_n . Если $\|A\| < 1$, то матрица $I - A$ обратима, причем $(I - A)^{-1} = \sum_{k=0}^{\infty} A^k$.

Доказательство. Рассмотрим ряд $\sum_{k=0}^{\infty} A^k$. Поскольку для всякого $p > 0$ $\left\| \sum_{k=m}^{m+p} A^k \right\| \leq \sum_{k=m}^{m+p} \|A\|^k \rightarrow 0$ при $m \rightarrow \infty$, то последовательность частичных сумм $s_m = \sum_{k=0}^m A^k$ является последовательностью Коши. Так как в силу полноты \mathbf{C}^{n^2} пространство \mathbf{M}_n полно по норме $\|\cdot\|$, то определен предел $B = \lim_{m \rightarrow \infty} s_m = \sum_{k=0}^{\infty} A^k$, прием в силу непрерывности нормы $\|B\| \leq \sum_{k=0}^{\infty} \|A\|^k = \frac{1}{1 - \|A\|}$. Имеем: $B(I - A) = \sum_{k=0}^{\infty} A^k(I - A) = \sum_{k=0}^{\infty} A^k - \sum_{k=1}^{\infty} A^k = I$. Аналогично проверяем $(I - A)B = I$. Следовательно, $B = (I - A)^{-1}$.

Теорема 2. Пусть $\|\cdot\|$ - матричная норма на \mathbf{M}_n , A - обратимая матрица. Если $B \in \mathbf{M}_n$, $\|B\| < \frac{1}{\|A^{-1}\|}$, то матрица $C = A + B$ обратима, причем $C^{-1} = (A + B)^{-1} = \sum_{k=0}^{\infty} (-1)^k (A^{-1}B)^k A^{-1}$.

Доказательство. Рассмотрим матрицу $D = A^{-1}C = A^{-1}(A+B) = I + A^{-1}B$. Так как по условию $\|A^{-1}B\| \leq \|A^{-1}\| \|B\| < 1$, то по предыдущей теореме существует матрица $D^{-1} = \sum_{k=0}^{\infty} (-1)^k (A^{-1}B)^k$, обратная к D . Рассмотрим матрицу $\hat{C} = D^{-1}A^{-1}$ и вычислим $\hat{C}C = D^{-1}A^{-1}(A+B) = D^{-1}(I + A^{-1}B) = D^{-1}D = I$, $C\hat{C} = (A+B)D^{-1}A^{-1} = AA^{-1}(A+B)D^{-1}A^{-1} = A(I + A^{-1}B)D^{-1}A^{-1} = AA^{-1} = I$. Следовательно, существует $C^{-1} = \hat{C} = D^{-1}A^{-1} = \sum_{k=0}^{\infty} (-1)^k (A^{-1}B)^k A^{-1}$.

§ 3. ОШИБКИ В РЕШЕНИЯХ ЛИНЕЙНЫХ СИСТЕМ

Пусть решается линейная система $Ax = b$, $A \in \mathbf{M}_n$, $b \in \mathbf{C}^n$. Пусть β - алгоритм решения этой системы на идеальной вычислительной системе, так, что $x = \beta(A, b)$ – точное решение системы и потому $x = \beta(A, b)$ – отображение на классе невырожденных матриц ($x = A^{-1}b$). Пусть $\hat{\beta}$ – тот же алгоритм, реализованный на реальной вычислительной системе. В результате его проведения получено приближенное решение $\hat{x} = \hat{\beta}(A, b)$. Определим матрицу \hat{A} и вектор \hat{b} из условия $\hat{x} = \beta(\hat{A}, \hat{b})$ ($\hat{b} = \hat{A}^{-1}\hat{x}$), т.е. \hat{x} является точным решением системы

$\hat{A}\hat{x} = \hat{b}$. Обозначим $\hat{A} = A + E$, $\hat{b} = b + e$. Таким образом, точное решение x удовлетворяет системе $Ax = b$, а приближенное решение \hat{x} удовлетворяет системе $(A + E)\hat{x} = b + e$.

Пусть $\|\cdot\|$ - произвольная матричная норма, согласованная с векторной нормой $\|\cdot\|$. Будем считать, что ошибка, вносимая в матрицу при проведении алгоритма, не очень велика: $\|E\| < \frac{1}{\|A^{-1}\|}$. По теореме 2.2 отсюда следует, что матрица $A + E$ обратима и $(A + E)^{-1} = \sum_{k=0}^{\infty} (-1)^k (A^{-1}E)^k A^{-1}$. Вычислим погрешность $x - \hat{x}$:

$$\begin{aligned} x - \hat{x} &= A^{-1}b - (A + E)^{-1}(b + e) = (A^{-1} - (A + E)^{-1})b - (A + E)^{-1}e \\ &= -\sum_{k=1}^{\infty} (-1)^k (A^{-1}E)^k A^{-1}b - \sum_{k=0}^{\infty} (-1)^k (A^{-1}E)^k A^{-1}e \\ &= -\sum_{k=1}^{\infty} (-1)^k (A^{-1}E)^k x - \sum_{k=0}^{\infty} (-1)^k (A^{-1}E)^k A^{-1}e. \end{aligned}$$

Следовательно,

$$\begin{aligned} \|x - \hat{x}\| &\leq \left(\sum_{k=1}^{\infty} \|A^{-1}E\|^k\right)\|x\| + \left(\sum_{k=0}^{\infty} \|A^{-1}E\|^k\right)\|A^{-1}e\|, \\ \frac{\|x - \hat{x}\|}{\|x\|} &\leq \frac{\|A^{-1}E\|}{1 - \|A^{-1}E\|} + \frac{1}{1 - \|A^{-1}E\|} \frac{\|A^{-1}e\|}{\|x\|}. \end{aligned}$$

Так как $Ax = b$, то $\|b\| = \|Ax\| \leq \|A\| \|x\|$ и $\frac{1}{\|x\|} \leq \frac{\|A\|}{\|b\|}$. Поэтому

$$\begin{aligned} \frac{\|x - \hat{x}\|}{\|x\|} &\leq \frac{\|A^{-1}\| \|E\|}{1 - \|A^{-1}\| \|E\|} + \frac{1}{1 - \|A^{-1}\| \|E\|} \|A\| \|A^{-1}\| \frac{\|e\|}{\|b\|}, \\ \frac{\|x - \hat{x}\|}{\|x\|} &\leq \frac{\|A\| \|A^{-1}\| \frac{\|E\|}{\|A\|}}{1 - \|A\| \|A^{-1}\| \frac{\|E\|}{\|A\|}} + \frac{1}{1 - \|A\| \|A^{-1}\| \frac{\|E\|}{\|A\|}} \|A\| \|A^{-1}\| \frac{\|e\|}{\|b\|} \end{aligned}$$

Определение. Числом обусловленности матрицы A по отношению к матричной норме $\|\cdot\|$ называется

$$\kappa(A) = \begin{cases} \|A\| \|A^{-1}\|, & \text{если } A \text{ невырождена,} \\ \infty, & \text{если } A \text{ вырождена.} \end{cases}$$

С использованием этого обозначения последнее неравенство можно переписать в виде

$$\frac{\|x - \hat{x}\|}{\|x\|} \leq \frac{\kappa(A)}{1 - \kappa(A)} \frac{\|E\|}{\|A\|} \left(\frac{\|E\|}{\|A\|} + \frac{\|e\|}{\|b\|} \right).$$

Здесь $\frac{\|x - \hat{x}\|}{\|x\|}$ называется *относительной погрешностью в решении* x , $\frac{\|E\|}{\|A\|}$ называется *относительной погрешностью в матрице* A , $\frac{\|e\|}{\|b\|}$ называется *относительной погрешностью в правой части* b .

Часто, чтобы оценить точность полученного приближенного решения \hat{x} , вычисляют *вектор невязки* $r = b - A\hat{x}$. Оценим относительную погрешность решения через невязку r .

$$x - \hat{x} = A^{-1}b - \hat{x} = A^{-1}(b - A\hat{x}) = A^{-1}r,$$

$$\frac{\|x - \hat{x}\|}{\|x\|} = \frac{\|A^{-1}r\|}{\|x\|} \leq \frac{1}{\|x\|} \|A^{-1}\| \|r\| \leq \frac{\|A\|}{\|b\|} \|A^{-1}\| \|r\| = \kappa(A) \frac{\|r\|}{\|b\|}.$$

Пример. Рассмотрим линейную систему

$$Ax = \begin{pmatrix} 1 & -1 \\ 1 & -1 + \varepsilon \end{pmatrix} x = b = \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

Ее точное решение $x = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$. Рассмотрим приближенное решение $\hat{x} = \begin{pmatrix} 1 + \varepsilon^{-1/2} \\ \varepsilon^{-1/2} \end{pmatrix}$. Тогда вектор невязки $r = \begin{pmatrix} 0 \\ -\varepsilon^{1/2} \end{pmatrix}$, вектор ошибки $x - \hat{x} = \begin{pmatrix} -\varepsilon^{-1/2} \\ -\varepsilon^{-1/2} \end{pmatrix}$. Поэтому

- a) относительная величина невязки $\frac{\|r\|}{\|b\|} = O(\varepsilon^{1/2}) \rightarrow 0 (\varepsilon \rightarrow 0)$;
- б) относительная величина ошибки $\frac{\|\hat{x} - x\|}{\|x\|} = O(\varepsilon^{-1/2}) \rightarrow \infty (\varepsilon \rightarrow 0)$;

Свойства числа обусловленности

1. $\kappa(A) \geq 1$.
2. $\kappa(A) = \kappa(A^{-1})$.
3. $\kappa(AB) \leq \kappa(A)\kappa(B)$.

Первые три свойства следуют непосредственно из определения числа обусловленности и основных свойств матричных норм.

4. Если $A = A^*$, то по отношению к спектральной норме $\kappa(A) = \left| \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)} \right|$, где $\lambda_{\max}(A)$ и $\lambda_{\min}(A)$ соответственно максимальное и минимальное по модулю собственные значения матрицы A .

Доказательство. В силу замечания 1.1 (см. лемму 1.4) для самосопряженных матриц $\|A\|_2 = \rho(A) = |\lambda_{\max}(A)|$, $\|A^{-1}\|_2 = \rho(A^{-1}) = |\lambda_{\max}(A^{-1})| = |\lambda_{\min}(A)|$. Поэтому $\kappa(A) = \|A\| \|A^{-1}\| = \left| \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)} \right|$.

5. Для всякой матрицы $A \in \mathbf{M}_n$ число обусловленности относительно любой матричной нормы $\kappa(A) \geq \left| \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)} \right|$.

Доказательство вытекает из леммы 1.4: $\|A\| \geq \rho(A) = |\lambda_{\max}(A)|$, $\|A^{-1}\| \geq \rho(A^{-1}) = |\lambda_{\max}(A^{-1})| = |\lambda_{\min}(A)|$; поэтому $\kappa(A) = \|A\| \|A^{-1}\| \geq \left| \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)} \right|$.

6. Для всякой матрицы $A \in \mathbf{M}_n$ и любых унитарных (ортогональных) матриц $U, V \in \mathbf{M}_n$ число обусловленности относительно спектральной нормы $\kappa(A) = \kappa(UAV)$.

Это свойство непосредственно следует из леммы 1.5.

7. Для всякой невырожденной матрицы $A \in \mathbf{M}_n$

$$\kappa(A) \geq \sup_{\substack{B \in \mathbf{M}_n, \\ B - \text{вырожденная}}} \frac{\|A\|}{\|A - B\|}$$

Доказательство. Покажем, что для всякой вырожденной матрицы $B \in \mathbf{M}_n$ справедливо неравенство $\|A - B\| \geq 1/\|A^{-1}\|$. Предположим, что это не так, т.е. существует вырожденная матрица B , такая, что для матрицы $C = A - B$ выполнено $\|C\| < 1/\|A^{-1}\|$. Тогда по теореме 2.2 матрица $A - C = A - (A - B) = B$ обратима, что противоречит вырожденности B . Таким образом, установлено, что для всякой вырожденной матрицы B $\|A^{-1}\| \geq 1/\|A - B\|$. Следовательно, $\kappa(A) = \|A\| \|A^{-1}\| \geq \|A\|/\|A - B\|$ для всякой вырожденной $B \in \mathbf{M}_n$. Поскольку левая часть этого неравенства не зависит от B , то из него вытекает требуемое соотношение.

§ 4. МЕТОД ГАУССА

§ 4.1. Алгоритм метода Гаусса

Пусть требуется решить линейную систему $Ax = b$, $A \in \mathbf{M}_n$:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n &= b_2 \\ \vdots &\quad \vdots \quad \vdots \quad \ddots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nn}x_n &= b_n \end{aligned} \tag{1}$$

Метод Гаусса состоит в том, что элементарными преобразованиями над строками матрицы она приводится к треугольному виду с главной диагональю, состоящей из единичных элементов (прямой ход метода Гаусса); полученная система с треугольной матрицей решается в явном виде (обратный ход метода Гаусса).

Предположим, что $a_{11} \neq 0$. Поделив первое уравнение системы (1) на a_{11} , перепишем его в виде

$$x_1 + c_{12}x_2 + \dots + c_{1n}x_n = y_1, \tag{2}$$

где $c_{1j} = a_{1j}/a_{11}$, $j = 2, \dots, n$, $y_1 = b_1/a_{11}$. Умножим уравнение (2) на a_{i1} и вычтем его из i -го уравнения системы (1) ($i = 2, \dots, n$). В результате система (1) примет вид

$$\begin{array}{lclllll} x_1 & + & c_{12}x_2 & + & \dots & + & c_{1n}x_n & = & y_1 \\ & a_{22}^{(1)}x_2 & + & \dots & + & a_{2n}^{(1)}x_n & = & b_2^{(1)} \\ & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ & a_{n2}^{(1)}x_2 & + & \dots & + & a_{nn}^{(1)}x_n & = & b_n^{(1)} \end{array} \quad (3)$$

где $a_{ij}^{(1)} = a_{ij} - c_{1j}a_{11}$, $b_i^{(1)} = b_i - y_1a_{i1}$, $i, j = 2, \dots, n$. Далее этот процесс применяется к подматрице $A^{(1)} = (a_{ij}^{(1)})_{i,j=2,\dots,n} \in \mathbf{M}_{n-1}$.

Пусть сделаны $k-1$, $k = 1, \dots, n$ шагов этого процесса, т.е. система (1) преобразована к виду

$$\begin{array}{lclllll} x_1 + c_{12}x_2 + \dots + c_{1,k-1}x_{k-1} + c_{1,k}x_k & + \dots & + c_{1n}x_n & = & y_1 \\ x_2 + \dots + c_{2,k-1}x_{k-1} + c_{2,k}x_k & + \dots & + c_{2n}x_n & = & y_2 \\ \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ x_{k-1} & + c_{k-1,k}x_k & + \dots & + c_{k-1,n}x_n & = & y_{k-1} \\ a_{k,k}^{(k-1)}x_k + \dots + a_{k,n}^{(k-1)}x_n & = & b_k^{(k-1)} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n,k}^{(k-1)}x_k + \dots + a_{n,n}^{(k-1)}x_n & = & b_n^{(k-1)} \end{array} \quad (4)$$

Предположим, что $a_{kk}^{(k-1)} \neq 0$. Поделив k -е уравнение системы (4) на $a_{kk}^{(k-1)}$, перепишем его в виде

$$x_k + c_{k,k+1}x_{k+1} + \dots + c_{kn}x_n = y_k, \quad (5)$$

где

$$c_{kj} = \frac{a_{kj}^{(k-1)}}{a_{kk}^{(k-1)}}, \quad j = k+1, \dots, n, \quad y_k = \frac{b_k^{(k-1)}}{a_{kk}^{(k-1)}}. \quad (6)$$

Умножим уравнение (5) на $a_{ik}^{(k-1)}$ и вычтем его из i -го уравнения системы (4) ($i = k+1, \dots, n$). В результате система (4) примет вид

$$\begin{array}{lclllll} x_1 + c_{12}x_2 + \dots + c_{1,k-1}x_{k-1} + c_{1,k}x_k + c_{1,k+1}x_{k+1} + \dots + c_{1n}x_n & = & y_1 \\ x_2 + \dots + c_{2,k-1}x_{k-1} + c_{2,k}x_k + c_{2,k+1}x_{k+1} + \dots + c_{2n}x_n & = & y_2 \\ \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ x_{k-1} + c_{k-1,k}x_k + c_{k-1,k+1}x_{k+1} + \dots + c_{k-1,n}x_n & = & y_{k-1} \\ x_k + c_{k,k+1}x_{k+1} + \dots + c_{kn}x_n & = & y_k \\ a_{k+1,k+1}^{(k)}x_{k+1} + \dots + a_{k+1,n}^{(k)}x_n & = & b_{k+1}^{(k)} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n,k+1}^{(k)}x_{k+1} + \dots + a_{n,n}^{(k)}x_n & = & b_n^{(k)} \end{array} \quad (7)$$

где

$$a_{ij}^{(k)} = a_{ij}^{(k-1)} - a_{ik}^{(k-1)} c_{kj}, \quad b_i^{(k)} = b_i^{(k-1)} - a_{ik}^{(k-1)} y_k, \quad i, j = k+1, \dots, n. \quad (8)$$

Выражения (6), (8) являются формулами перехода от системы (4) к системе (7). Если обозначить $a_{ij}^{(0)} = a_{ij}$, $b_i^{(0)} = b_i$, $i, j = 1, \dots, n$, то переход от системы (1) к системе (3) будет осуществляться по тем же формулам при $k = 1$.

После проведения вычислений по формулам (6), (8) при $k = 1, \dots, n$ (которые составляют *прямой ход метода Гаусса*) система (1) примет вид

$$\begin{aligned} x_1 + c_{12}x_2 + \dots + c_{1n}x_{n-1} + c_{1n}x_n &= y_1 \\ x_2 + \dots + c_{2n}x_{n-1} + c_{2n}x_n &= y_2 \\ \ddots \vdots \vdots \vdots \vdots \vdots \vdots &= \vdots \vdots \vdots \vdots \vdots \vdots \\ x_{n-1} + c_{n-1,n}x_n &= y_{n-1} \\ x_n &= y_n \end{aligned} \quad (9)$$

Решение системы (9) с треугольной матрицей может быть найдено непосредственно (методом последовательного исключения неизвестных в порядке x_n, x_{n-1}, \dots, x_1):

$$x_n = y_n, \quad x_i = y_i - \sum_{j=i+1}^n c_{ij}x_j, \quad i = n-1, \dots, 1. \quad (10)$$

Вычисления по формулам (10) составляют *обратный ход метода Гаусса*.

§ 4.2. Оценка количества арифметических операций в методе Гаусса

Здесь и далее при оценке количества арифметических операций мы будем отдельно находить количество аддитивных операций (сложений и вычитаний) и количество мультипликативных операций (умножений и делений). Для упрощения выкладок мы будем находить только главный член асимптотики количества операций при $n \rightarrow \infty$.

1. На вычисление c_{kj} при $j = k+1, \dots, n$, $k = 1, \dots, n$ по формулам (6) требуется $\sum_{k=1}^n (n-k) = \sum_{i=0}^{n-1} i = n(n-1)/2 = O(n^2)$ ($n \rightarrow \infty$) операций деления.

2. На вычисление $a_{ij}^{(k)}$ при $i, j = k+1, \dots, n$, $k = 1, \dots, n$ по формулам (8) требуется $\sum_{k=1}^n (n-k)^2 = \sum_{i=0}^{n-1} i^2 = (n-1)n(2n-1)/6 = n^3/3 + O(n^2)$ ($n \rightarrow \infty$) операций умножения и столько же операций вычитания.

Итак, на вычисление коэффициентов c_{kj} , $j = k+1, \dots, n$, $k = 1, \dots, n$ системы (9) требуется $O(n^2) + n^3/3 + O(n^2) = n^3/3 + O(n^2)$ ($n \rightarrow \infty$) мультипликативных операций и столько же аддитивных операций.

3. На вычисление y_k при $k = 1, \dots, n$ по формулам (6) требуется n операций деления.

4. На вычисление $b_i^{(k)}$ при $i = k+1, \dots, n$, $k = 1, \dots, n$ по формулам (8) требуется $\sum_{k=1}^n (n-k) = \sum_{i=0}^{n-1} i = n(n-1)/2 = O(n^2)$ ($n \rightarrow \infty$) операций умножения и столько же операций вычитания.

Итак, на вычисление правых частей y_k , $k = 1, \dots, n$ системы (9) требуется $O(n^2)$ ($n \rightarrow \infty$) мультипликативных операций и столько же аддитивных операций.

Таким образом, прямой ход метода Гаусса требует $n^3/3 + O(n^2) + O(n^2) = n^3/3 + O(n^2)$ ($n \rightarrow \infty$) мультипликативных операций и столько же аддитивных операций.

5. На вычисление решения по формулам (10) (т.е. на проведение обратного хода метода Гаусса) требуется $\sum_{i=1}^{n-1} (n-i) = \sum_{i=1}^{n-1} i = n(n-1)/2 = O(n^2)$ ($n \rightarrow \infty$) операций умножения и столько же операций вычитания.

Следовательно, метод Гаусса требует $n^3/3 + O(n^2) + O(n^2) = n^3/3 + O(n^2)$ ($n \rightarrow \infty$) мультипликативных операций и столько же аддитивных операций. Всего: $(2/3)n^3 + O(n^2)$ ($n \rightarrow \infty$) арифметических операций.

§ 4.3. Представление метода Гаусса в виде последовательности элементарных преобразований

Преобразование системы, задаваемое формулами (6), эквивалентно умножению матрицы системы слева на матрицу

$$D_k = \text{diag} [\underbrace{1, \dots, 1}_{k-1}, \left(a_{kk}^{(k-1)} \right)^{-1}, 1, \dots, 1],$$

(где $\text{diag} [d_1, \dots, d_n]$ означает диагональную матрицу с элементами d_1, \dots, d_n на главной диагонали).

Преобразование системы, задаваемое формулами (8), эквивалентно умножению матрицы системы слева на матрицу

$$L_k = \begin{pmatrix} 1 & & & \\ & \ddots & & \\ & & 1 & \\ & & -a_{k+1,k}^{(k-1)} & 1 \\ & & \vdots & \ddots \\ & & -a_{n,k}^{(k-1)} & & 1 \end{pmatrix}$$

(не обозначенные элементы матрицы L_k равны нулю).

Следовательно, прямой ход метода Гаусса эквивалентен умножению матрицы системы (1) последовательно на матрицы $L_k D_k$, $k = 1, \dots, n$:

$$L_n D_n \cdot \dots \cdot L_1 D_1 \cdot A x = L_n D_n \cdot \dots \cdot L_1 D_1 \cdot b. \quad (11)$$

причем матрица $U = L_n D_n \cdot \dots \cdot L_1 D_1 \cdot A$ есть матрица системы (9), т.е. является верхней треугольной с единицами на главной диагонали. Обозначим через $\text{LT}(n)$ группу невырожденных верхних треугольных матриц. Тогда для всех $k = 1, \dots, n$ $D_k, L_k \in \text{LT}(n)$ и потому матрица $\hat{L} = L_n D_n \cdot \dots \cdot L_1 D_1 \in \text{LT}(n)$.

Из (11) получаем: $U = \hat{L}A$, откуда $A = LU$, где $L = \hat{L}^{-1} \in \text{LT}(n)$, U верхняя треугольная с единицами на главной диагонали. Полученное представление матрицы A называется *LU-разложением*. Итак, показано, что метод Гаусса эквивалентен построению *LU-разложения*.

Знание *LU-разложения* матрицы A может быть полезно, например, в следующей ситуации. Пусть требуется решить ряд систем вида $Ax_m = b_m$, $m = 1, \dots, M$ с одной и той же матрицей A и разными правыми частями b_m . Применяя M раз метод Гаусса это можно сделать за $(2/3)Mn^3 + O(Mn^2)$ ($n \rightarrow \infty$) арифметических операций. Предположим, что нам известно *LU-разложение* матрицы A : $A = LU$. Тогда решение каждой из систем $Ax_m = b_m$ может быть сведено к решению двух систем $Ly_m = b_m$, $Ux_m = y_m$ с треугольными матрицами. Решение системы с треугольной матрицей осуществляется обратным ходом метода Гаусса за $O(n^2)$ ($n \rightarrow \infty$) арифметических операций. Следовательно, если *LU-разложение* матрицы A уже построено и на это построение потребовалось $d(n)$ арифметических операций, то решение M систем с той же матрицей A потребует $d(n) + O(Mn^2)$ ($n \rightarrow \infty$) арифметических операций. Подчеркнем, что число $d(n)$ не зависит от M . Сейчас мы построим алгоритм для получения *LU-разложения* и покажем, что $d(n) = (2/3)n^3 + O(n^2)$ ($n \rightarrow \infty$) (т.е. равно числу операций, необходимых для проведения метода Гаусса).

§ 4.4. Алгоритм построения *LU-разложения*

Пусть требуется найти нижнюю треугольную матрицу $L = (l_{ij})$ и верхнюю треугольную матрицу $U = (u_{ij})$ с единицами на главной диагонали такую, что $A = LU$, т.е.

$$\sum_{j=1}^n l_{ij} u_{jk} = a_{ik}, \quad i, k = 1, \dots, n. \quad (12)$$

Поскольку $l_{ij} = 0$ при $i < j$, $u_{jk} = 0$ при $j > k$, $u_{jj} = 1$, то (12) есть система из n^2 уравнений относительно $n(n+1)/2$ неизвестных l_{ij} , $i \geq j$ и $n(n-1)/2$ неизвестных u_{jk} , $j < k$, всего $n(n+1)/2 + n(n-1)/2 = n^2$ неизвестных. Получим формулы для решения системы (12), которые и составляют алгоритм нахождения *LU-разложения*.

В силу $l_{ij} = 0$ при $i < j$, $u_{jk} = 0$ при $j > k$ сумма в (12) имеет вид

$$\sum_{j=1}^{\min\{i,k\}} l_{ij} u_{jk} = a_{ik}, \quad i, k = 1, \dots, n,$$

или

$$\begin{cases} \sum_{j=1}^k l_{ij} u_{jk} = a_{ik}, & k \leq i, \quad i, k = 1, \dots, n, \\ \sum_{j=1}^i l_{ij} u_{jk} = a_{ik}, & k > i, \quad i, k = 1, \dots, n. \end{cases}$$

Выделим в первой из этих сумм отдельно случай $k = 1$, а во второй - случай $i = 1$, и учтем, что $u_{kk} = 1$ для всех $k = 1, \dots, n$,

$$\left[\begin{array}{ll} l_{i1} = a_{i1}, & i = 1, \dots, n, \\ \sum_{j=1}^{k-1} l_{ij} u_{jk} + l_{ik} = a_{ik}, & 1 < k \leq i, \quad i, k = 2, \dots, n, \\ l_{11} u_{1k} = a_{1k}, & k = 2, \dots, n, \\ \sum_{j=1}^{i-1} l_{ij} u_{jk} + l_{ii} u_{ik} = a_{ik}, & k > i > 1, \quad i, k = 2, \dots, n. \end{array} \right]$$

Перегруппируем эти формулы:

$$\left[\begin{array}{ll} l_{i1} = a_{i1}, & i = 1, \dots, n, \\ u_{1k} = a_{1k}/l_{11}, & k = 2, \dots, n, \\ l_{ik} = a_{ik} - \sum_{j=1}^{k-1} l_{ij} u_{jk}, & 1 < k \leq i, \quad i, k = 2, \dots, n, \\ u_{ik} = (a_{ik} - \sum_{j=1}^{i-1} l_{ij} u_{jk})/l_{ii}, & k > i > 1, \quad i, k = 2, \dots, n. \end{array} \right] \quad (13)$$

Процесс вычислений по этим формулам строится следующим образом: вначале по первой из формул (13) вычисляются неизвестные элементы первого столбца матрицы L : l_{i1} , $i = 1, \dots, n$, затем по второй из формул (13) вычисляются неизвестные элементы первой строки матрицы U : u_{1k} , $k = 2, \dots, n$ (напомним, элемент u_{11} известен, он равен 1). Далее в вычислениях участвуют только третья и четвертая из формул (13). По третьей формуле (13) вычисляются неизвестные элементы второго столбца матрицы L : l_{i2} , $i = 2, \dots, n$ (напомним, $l_{12} = 0$, так как L -нижняя треугольная)

$$l_{i2} = a_{i2} - l_{i1} u_{12}, \quad i = 2, \dots, n.$$

По четвертой формуле (13) вычисляются неизвестные элементы второй строки матрицы U : u_{2k} , $k = 3, \dots, n$ (напомним, $u_{21} = 0$, так как U -верхняя треугольная, $u_{22} = 1$, так как U имеет единичную главную диагональ)

$$u_{2k} = (a_{2k} - l_{21} u_{1k})/l_{22}, \quad k = 3, \dots, n.$$

Затем по третьей формуле (13) вычисляются неизвестные элементы третьего столбца матрицы L : l_{i3} , $i = 3, \dots, n$, а по четвертой формуле (13) вычисляются неизвестные элементы третьей строки матрицы U : u_{3k} , $k = 4, \dots, n$ и так далее.

Замечание 1. Организация хранения матриц L и U в памяти. Формулы (13) таковы, что при вычислении элемента l_{ij} или u_{ij} используются значения элемента a_{ij} и вычисленных ранее элементов l_{km} , $m < j$ и u_{km} , $k < i$. Это позволяет хранить нижнюю треугольную матрицу L на месте нижнего треугольника матрицы A : $l_{ij} \equiv a_{ij}$, $i \geq j$, $i, j = 1, \dots, n$, а верхнюю треугольную матрицу U (без единичной главной диагонали) - на месте верхнего треугольника матрицы A : $u_{ij} \equiv a_{ij}$, $i < j$, $i, j = 1, \dots, n$.

§ 4.5. Оценка количества арифметических операций в алгоритме построения LU-разложения

1. При фиксированном $k = 1, \dots, n$ вычисление элементов l_{ik} для всех $i = k, \dots, n$ по третьей формуле (13) требует $\sum_{i=k}^n (k-1) = (n-k+1)(k-1)$ мультипликативных и столько же аддитивных операций. Следовательно, вычисление всех элементов матрицы L требует $\sum_{k=1}^n (n-k+1)(k-1) = n \sum_{k=1}^n (k-1) - \sum_{k=1}^n (k-1)^2 = n \sum_{i=0}^{n-1} i - \sum_{i=0}^{n-1} i^2 = n^2(n-1)/2 - (n-1)n(2n-1)/6 = n^3/2 - n^3/3 + O(n^2) = n^3/6 + O(n^2)$ ($n \rightarrow \infty$) мультипликативных и столько же аддитивных операций.

2. При фиксированном $i = 1, \dots, n$ вычисление элементов u_{ik} для всех $k = i+1, \dots, n$ по четвертой формуле (13) требует $\sum_{k=i+1}^n i = (n-i)i$ мультипликативных и $\sum_{k=i+1}^n (i-1) = (n-i)(i-1)$ аддитивных операций. Следовательно, вычисление всех элементов матрицы U требует $\sum_{i=1}^n (n-i)i = n^2(n+1)/2 - n(n+1)(2n+1)/6 = n^3/6 + O(n^2)$ ($n \rightarrow \infty$) мультипликативных и $\sum_{i=1}^n (n-i)(i-1) = n^2(n-1)/2 - n(n+1)(2n+1)/6 + n(n+1)/2 = n^3/6 + O(n^2)$ ($n \rightarrow \infty$) аддитивных операций.

Таким образом, алгоритм построения LU-разложения требует для своего проведения выполнения $n^3/3 + O(n^2)$ ($n \rightarrow \infty$) мультипликативных и столько же аддитивных операций, а в сумме — $(2/3)n^3 + O(n^2)$ ($n \rightarrow \infty$) арифметических операций.

§ 4.6. Осуществимость метода Гаусса

Теорема 1. *Метод Гаусса осуществим (т.е. возможно LU-разложение) тогда и только тогда, когда для всех $k = 1, \dots, n$ ее главные угловые миноры $\det A_k \neq 0$, где*

$$A_k = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1k} \\ a_{21} & a_{22} & \dots & a_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ a_{k1} & a_{k2} & \dots & a_{kk} \end{pmatrix}.$$

— главные угловые подматрицы матрицы A .

Доказательство. Можно показать (мы этого делать не будем), что если $\det A_k \neq 0$ для всех $k = 1, \dots, n$, то существует и единственное LU-разложение матрицы A .

Обозначим через L_k и U_k главные угловые подматрицы матриц L и U . Так как L — нижняя треугольная, а U — верхняя треугольная матрицы, и $A = LU$, то $A_k = L_k U_k$. Следовательно, $\det A_k = \det L_k \det U_k$. Но матрица U — верхняя треугольная с 1 на главной диагонали, поэтому $\det U_k = 1$ для всех $k = 1, \dots, n$. Значит, $\det A_k = \det L_k$. Матрица L_k — нижняя треугольная, поэтому $\det L_k = l_{11} \dots l_{kk}$. Следовательно, $\det A_k = l_{11} \dots l_{kk}$.

Пусть для всех $k = 1, \dots, n$ $\det A_k \neq 0$. Тогда $l_{11} = \det A_1 \neq 0$, $l_{kk} = \det A_k / \det A_{k-1} \neq 0$ и в формулах (13) возможно осуществить деление на l_{ii} , $i =$

$1, \dots, n$. Следовательно, осуществим алгоритм построения LU -разложения, а значит, и метод Гаусса.

Пусть осуществим алгоритм построения LU -разложения, т.е. в формулах (13) возможно осуществить деление на l_{ii} , $i = 1, \dots, n$. Следовательно, для всех $i = 1, \dots, n$ $l_{ii} \neq 0$ и потому $\det A_k = l_{11} \dots l_{kk} \neq 0$ для всех $k = 1, \dots, n$.

§ 5. МЕТОДЫ ПОСЛЕДОВАТЕЛЬНОГО ИСКЛЮЧЕНИЯ НЕИЗВЕСТНЫХ ДЛЯ ЛЕНТОЧНЫХ МАТРИЦ

Определение. Матрица $A \in M_n$ называется *ленточной*, если $a_{ij} = 0$ при $i - j > k_1$ либо $j - i > k_2$. Величина $k_1 + k_2 + 1$ называется *шириной ленты*. Если $k_1 = k_2 = 1$, то матрица называется *трехдиагональной*.

§ 5.1. Метод Гаусса для ленточных матриц

Расчетные формулы метода Гаусса (4.6), (4.8), (4.10) остаются справедливыми и для ленточных матриц, но объем вычислений по ним может быть сокращен при учете структуры матрицы A . В формулах прямого хода (4.6), (4.8) вычисления надо вести для $j = k + 1, \dots, k + k_2$, $i = k + 1, \dots, k + k_1$. Ширина ленты системы (4.9), получающейся после прямого хода, равна $k_2 + 1$, поэтому в формуле (4.10) обратного хода метода Гаусса надо учитывать только $k_2 + 1$ слагаемых.

Аналогично тому, как это было сделано выше, можно подсчитать, что

1. На вычисление c_{kj} при $j = k + 1, \dots, k + k_2$, $k = 1, \dots, n$ по формулам (4.6) требуется $n k_2 + O(1)$ операций деления.

2. На вычисление $a_{ij}^{(k)}$ при $i = k + 1, \dots, k + k_1$, $j = k + 1, \dots, k + k_2$, $k = 1, \dots, n$ по формулам (4.8) требуется $n k_1 k_2 + O(1)$ операций умножения и столько же операций вычитания.

Итак, на вычисление коэффициентов c_{kj} , $j = k + 1, \dots, k + k_2$, $k = 1, \dots, n$ системы (4.9) требуется $n(k_1 k_2 + k_2) + O(1)$ мультипликативных и столько же аддитивных операций.

3. На вычисление y_k при $k = 1, \dots, n$ по формулам (4.6) требуется n операций деления.

4. На вычисление $b_i^{(k)}$ при $i = k + 1, \dots, k + k_1$, $k = 1, \dots, n$ по формулам (4.8) требуется $n k_1 + O(1)$ операций умножения и столько же операций вычитания.

Итак, на вычисление правых частей y_k , $k = 1, \dots, n$ системы (4.9) требуется $n(k_1 + 1)$ мультипликативных и столько же аддитивных операций.

Таким образом, прямой ход метода Гаусса требует $n(k_1 k_2 + k_1 + k_2 + 1)$ мультипликативных операций и столько же аддитивных операций.

5. На вычисление решения по формулам (4.10) (т.е. на проведение обратного хода метода Гаусса) требуется $n k_2 + O(1)$ операций умножения и столько же операций вычитания.

Следовательно, метод Гаусса требует $n(k_1 k_2 + k_1 + 2 k_2 + 1)$ мультипликативных операций и столько же аддитивных операций. Всего: $2n(k_1 k_2 + k_1 + 2 k_2 + 1)$ арифметических операций.

Аналогичное сокращение вычислительной работы может быть произведено для алгоритма LU -разложения.

§ 5.2. Алгоритм LU -разложения для трехдиагональных матриц

Пусть требуется трехдиагональную матрицу вида

$$A = \begin{pmatrix} a_1 & c_1 & & & \\ d_1 & a_2 & c_2 & & \\ & d_2 & a_3 & \ddots & \\ & & \ddots & \ddots & c_{n-2} \\ & & & d_{n-2} & a_{n-1} & c_{n-1} \\ & & & & d_{n-1} & a_n \end{pmatrix} \quad (1)$$

представить в виде $A = LU$, где

$$L = \begin{pmatrix} l_1 & & & & \\ \lambda_1 & l_2 & & & \\ & \lambda_2 & l_3 & & \\ & & \ddots & \ddots & \\ & & & \lambda_{n-1} & l_n \end{pmatrix}, \quad U = \begin{pmatrix} 1 & u_1 & & & \\ & 1 & u_2 & & \\ & & 1 & \ddots & \\ & & & \ddots & u_{n-1} \\ & & & & 1 \end{pmatrix}. \quad (2)$$

В этом случае формулы (4.13) алгоритма построения LU -разложения могут быть получены путем непосредственного перемножения матриц L и U и решения получающихся уравнений:

$$\begin{aligned} l_1 &= a_1, & u_1 &= c_1/l_1 \\ \lambda_1 &= d_1, & l_2 &= a_2 - \lambda_1 u_1, & u_2 &= c_2/l_2 \\ &\vdots & &\vdots & &\vdots \\ \lambda_i &= d_i, & l_{i+1} &= a_{i+1} - \lambda_i u_i, & u_{i+1} &= c_{i+1}/l_{i+1} \\ &\vdots & &\vdots & &\vdots \\ \lambda_{n-2} &= d_{n-2}, & l_{n-1} &= a_{n-1} - \lambda_{n-2} u_{n-2}, & u_{n-1} &= c_{n-1}/l_{n-1} \\ \lambda_{n-1} &= d_{n-1}, & l_n &= a_n - \lambda_{n-1} u_{n-1} & & \end{aligned} \quad (3)$$

Построение LU -разложения по этим формулам требует $n - 1$ аддитивных и $2(n - 1)$ мультипликативных операций.

Решение линейной системы $Ax = b$ может быть осуществлено следующим образом:

1. Вначале строится LU -разложение матрицы A по формулам (3), на это требуется $n - 1$ аддитивных и $2(n - 1)$ мультипликативных операций.

2. Затем находится решение y линейной системы $Ly = b$ путем последовательного исключения неизвестных, начиная с первого уравнения:

$$y_1 = b_1/l_1, \quad y_i = (b_i - \lambda_{i-1}y_{i-1})/l_i, \quad i = 2, \dots, n. \quad (4)$$

На этом этапе требуется произвести $n - 1$ аддитивных и $2(n - 1) + 1$ мультипликативных операций.

3. Искомый вектор x находится как решение линейной системы $Ux = y$ путем последовательного исключения неизвестных, начиная с последнего уравнения:

$$x_n = y_n, \quad x_i = y_i - u_i x_{i+1}, \quad i = n - 1, \dots, 1. \quad (5)$$

На этом этапе требуется произвести $n - 1$ аддитивных и $n - 1$ мультипликативных операций.

Складывая оценки трудоемкости на каждом шаге, находим, что для осуществления алгоритма требуется произвести $3(n - 1)$ аддитивных и $5(n - 1) + 1$ мультипликативных операций.

Замечание 1. Хранение в памяти ЭВМ матриц A , L и U . Матрицу вида (1) в памяти ЭВМ обычно не хранят. Вместо этого запоминают вектора $a = (a_1, \dots, a_n)^t$, $c = (c_1, \dots, c_{n-1})^t$, $d = (d_1, \dots, d_{n-1})^t$. Аналогично, вместо матрицы L хранятся вектора $l = (l_1, \dots, l_n)^t$, $\lambda = (\lambda_1, \dots, \lambda_{n-1})^t$, а вместо матрицы U – вектор $u = (u_1, \dots, u_{n-1})^t$. Как и в обычном алгоритме LU -разложения, матрицы L и U можно хранить на месте матрицы A : вектор l – на месте вектора a , вектор λ – на месте d , u – на месте c .

Замечание 2. Вспомогательный вектор y в приведенном выше алгоритме решения линейной системы может храниться на месте вектора x . Это следует из формул (4), (5), которые можно записать в виде

$$x_1 = b_1/l_1, \quad x_i = (b_i - \lambda_{i-1}x_{i-1})/l_i, \quad i = 2, \dots, n, \quad x_i := x_i - u_i x_{i+1}, \quad i = n - 1, \dots, 1.$$

§ 5.3. Метод прогонки для трехдиагональных матриц

Пусть требуется трехдиагональную матрицу A вида (1) представить в виде $A = LU$, где

$$L = \begin{pmatrix} l_1 & & & \\ 1 & l_2 & & \\ & 1 & l_3 & \\ & & \ddots & \ddots & \\ & & & 1 & l_n \end{pmatrix}, \quad U = \begin{pmatrix} v_1 & u_1 & & & \\ v_2 & u_2 & & & \\ v_3 & & \ddots & & \\ & \ddots & & u_{n-1} & \\ & & & & v_n \end{pmatrix}.$$

В отличие от LU -разложения, это представление не единственно. Перемножая матрицы L и U , находим уравнения для определения коэффициентов l_i, v_i, u_i :

$$\begin{aligned} l_1 v_1 &= a_1, & l_1 u_1 &= c_1 \\ v_{i-1} = d_{i-1}, \quad u_{i-1} + l_i v_i &= a_i, & l_i u_i &= c_i, \quad i = 2, \dots, n-1 \\ v_{n-1} = d_{n-1}, \quad u_{n-1} + l_n v_n &= a_n. \end{aligned} \quad (6)$$

Число уравнений в этой системе, равное $2 + 3(n-2) + 2 = 3n - 2$, на 1 меньше числа неизвестных l_i, v_i, u_i . Перепишем (6) в виде

$$\begin{aligned} v_1 &= d_1, & l_1 v_1 &= a_1, & l_1 u_1 &= c_1 \\ v_i &= d_i, & u_{i-1} + l_i v_i &= a_i, & l_i u_i &= c_i, \quad i = 2, \dots, n-1 \\ u_{n-1} + l_n v_n &= a_n. \end{aligned}$$

Отсюда получаем расчетные формулы:

$$\begin{aligned} v_1 &= d_1, & l_1 &= a_1/v_1, & u_1 &= c_1/l_1 \\ v_i &= d_i, & l_i &= (a_i - u_{i-1})/v_i, & u_i &= c_i/l_i, \quad i = 2, \dots, n-1 \\ l_n v_n &= a_n - u_{n-1}. \end{aligned} \quad (7)$$

Из последнего уравнения в (7) мы можем определить только произведение $l_n v_n$. Зафиксировав один из параметров l_n или v_n , мы определим второй. Мы будем считать, что $v_n = 1$. В этом случае, количество арифметических операций, необходимое для осуществления разложения по формулам (7) равно количеству арифметических операций, необходимому для осуществления разложения по формулам (3).

Решение линейной системы $Ax = b$ может быть осуществлено следующим образом:

1. Вначале строится LU -разложение матрицы A по формулам (7), на это требуется $n-1$ аддитивных и $2(n-1)$ мультипликативных операций.
2. Затем находится решение y линейной системы $Ly = b$ путем последовательного исключения неизвестных, начиная с первого уравнения:

$$y_1 = b_1/l_1, \quad y_i = (b_i - y_{i-1})/l_i, \quad i = 2, \dots, n. \quad (8)$$

На этом этапе требуется произвести $n-1$ аддитивных и $(n-1)+1$ мультипликативных операций.

3. Искомый вектор x находится как решение линейной системы $Ux = y$ путем последовательного исключения неизвестных, начиная с последнего уравнения:

$$x_n = y_n/v_n, \quad x_i = (y_i - u_i x_{i+1})/v_i, \quad i = n-1, \dots, 1. \quad (9)$$

На этом этапе требуется произвести $n-1$ аддитивных и $2(n-1)$ мультипликативных операций (с учетом $v_n = 1$).

Складывая оценки трудоемкости на каждом шаге, находим, что для осуществления алгоритма требуется произвести $3(n-1)$ аддитивных и $5(n-1)+1$ мультипликативных операций.

Замечание 3. Хранение в памяти ЭВМ матриц A , L и U осуществляется так, как описано в Замечании 1. Вместо матрицы A запоминают вектора $a = (a_1, \dots, a_n)^t$, $c = (c_1, \dots, c_{n-1})^t$, $d = (d_1, \dots, d_{n-1})^t$, вместо матрицы L – вектор $l = (l_1, \dots, l_n)^t$, а вместо матрицы U – вектора $v = (v_1, \dots, v_n)^t$, $u = (u_1, \dots, u_{n-1})^t$. Аналогично алгоритму LU -разложения, матрицы L и U можно хранить на месте матрицы A : вектор l – на месте вектора a , вектор v – на месте d (последняя, не определяемая однозначно компонента v_n вектора v не хранится; ее можно считать равной 1), u – на месте c .

Замечание 4. Вспомогательный вектор y в приведенном выше алгоритме решения линейной системы может храниться на месте вектора x . Это следует из формул (8), (9), которые можно записать в виде

$$x_1 = b_1/l_1, \quad x_i = (b_i - x_{i-1})/l_i, \quad i = 2, \dots, n, \quad x_i := (x_i - u_i x_{i+1})/v_i, \quad i = n-1, \dots, 1.$$

(здесь считается $v_n = 1$.)

§ 6. ЗАДАЧА ОБРАЩЕНИЯ МАТРИЦЫ

Рассмотрим задачу нахождения матрицы, обратной к данной.

Случай произвольного алгоритма. Пусть \mathfrak{B} – некоторый алгоритм решения линейных систем вида $Ax = b$, так, что $x = \mathfrak{B}(A, b)$. Тогда j -й столбец x_j матрицы A^{-1} равен $x_j = \mathfrak{B}(A, e_j)$, где $e_j = (\underbrace{0, \dots, 0}_{j-1}, 1, 0, \dots, 0)^t$ есть j -й

орт стандартного базиса. Если алгоритм \mathfrak{B} требует для своего проведения $d(n)$ арифметических операций, то этот способ нахождения обратной матрицы потребует $n d(n)$ арифметических операций. Например, если \mathfrak{B} – это метод Гаусса, то потребуется $2/3 n^4 + O(n^3)$ арифметических операций.

Случай специального алгоритма. Многие алгоритмы решения линейных систем (в частности, все алгоритмы, рассматриваемые нами) обладают следующим свойством: алгоритм (по крайней мере его самая трудоемкая с вычислительной точки зрения часть) состоит в проведении над системой преобразований, которые выполняются над матрицей системы и правой частью независимо. Эта особенность позволяет вместо правой части – вектора b – рассматривать набор правых частей, т.е. матрицу B . Преобразования алгоритма выполняются над матрицей системы и набором правых частей. Таким образом, такой алгоритм решения системы $Ax = b$ может быть преобразован в алгоритм решения матричного уравнения $AX = B$, где X, B – $n \times n$ матрицы. Обычно алгоритм решения системы $Ax = b$ требует $O(n^3)$ арифметических операций для проведения преобразований над матрицей и $O(n^2)$ арифметических операций для проведения преобразований над правой частью. Поэтому алгоритм решения матричной системы $AX = B$ требует $O(n^3) + n \cdot O(n^2) = O(n^3)$ арифметических операций.

Использование LU-разложения для обращения матрицы. Пусть для матрицы A возможно осуществить LU-разложение. Действительно, если $A = LU$, то $A^{-1} = U^{-1}L^{-1}$. Матрицы, обратные к L и U , строятся, например, описанным выше способом. Поскольку системы с матрицами L и U решаются методом последовательного исключения неизвестных за соответственно $n(n - 1) + n = n^2 + O(n)$ и $n(n - 1) = n^2 + O(n)$ действий (см. формулы (4.10)), то матрицы U^{-1} и L^{-1} могут быть вычислены с затратой $n^3 + O(n^2)$ арифметических операций. Следовательно, для вычисления обратной матрицы требуется: $2/3 n^3 + O(n^2)$ арифметических операций для построения LU-разложения, $n^3 + O(n^2)$ арифметических операций для вычисления U^{-1} и L^{-1} , $n^3 + O(n^2)$ арифметических операций для вычисления $A^{-1} = U^{-1}L^{-1}$; всего: $8/3 n^3 + O(n^2)$ арифметических операций.

§ 7. МЕТОД ГАУССА С ВЫБОРОМ ГЛАВНОГО ЭЛЕМЕНТА

Теорема 4.1 показывает, что метод Гаусса в изложенном выше виде применим не ко всем невырожденным матрицам. Например, если в системе (4.1) $a_{11} = 0$, то нельзя осуществить первый же шаг алгоритма. Модернизируем алгоритм следующим образом.

Уравнения в системе (4.1) равноправны, мы можем их занумеровать в произвольном порядке. Присвоим номер 1 тому уравнению, в котором коэффициент при x_1 отличен от 0. Если такого уравнения не нашлось, то матрица A имеет нулевой первый столбец, т.е. вырождена. После этой перенумерации уравнений мы сделаем первый шаг метода Гаусса, т.е. перейдем от системы (4.1) к системе (4.3). Далее в подматрице $A^{(1)} = (a_{ij}^{(1)})_{i,j=2,\dots,n} \in M_{n-1}$ присвоим номер 2 тому уравнению, в котором коэффициент при x_2 отличен от 0, и сделаем следующий шаг метода Гаусса. Затем этот процесс применяется к подматрице $A^{(2)} \in M_{n-2}$ и так далее.

Если уравнений, в которых коэффициент при x_1 отличен от 0, несколько, то с вычислительной точки зрения не безразлично, какое из этих уравнений получит номер 1. Пусть погрешность в элементе a_{ij} матрицы A равна ε_{ij} , т.е. вместо точной матрицы A рассматривается матрица \hat{A} , элементы которой содержат вычислительные погрешности: $\hat{a}_{ij} = a_{ij} + \varepsilon_{ij}$. Для простоты будем считать, что элементы первого столбца известны точно: $\varepsilon_{i1} = 0$, $i = 1, \dots, n$. Из формул для элементов матрицы $A^{(1)}$ (см. (4.8), (4.6)):

$$\begin{aligned}\hat{a}_{ij}^{(1)} &= \hat{a}_{ij} - \hat{c}_{1j}\hat{a}_{i1} = \hat{a}_{ij} - \frac{\hat{a}_{1j}}{\hat{a}_{11}}\hat{a}_{i1} = \hat{a}_{ij} - \hat{a}_{1j}\frac{\hat{a}_{i1}}{\hat{a}_{11}} = a_{ij} + \varepsilon_{ij} - (a_{1j} + \varepsilon_{1j})\frac{a_{i1}}{a_{11}} \\ &= a_{ij} - a_{1j}\frac{a_{i1}}{a_{11}} + \varepsilon_{ij} - \varepsilon_{1j}\frac{a_{i1}}{a_{11}} = a_{ij}^{(1)} + \varepsilon_{ij}^{(1)}, \quad i, j = 2, \dots, n,\end{aligned}$$

где

$$\varepsilon_{ij}^{(1)} = \varepsilon_{ij} - \varepsilon_{1j}\frac{a_{i1}}{a_{11}}, \quad i, j = 2, \dots, n, \tag{1}$$

Эти равенства показывают, как преобразуется погрешность после шага алгоритма. Из соотношений (1) вытекает, что если отношение $\frac{a_{i1}}{a_{11}}$ очень велико, то вычислительная погрешность, вносимая на шаге алгоритма, может быть недопустимо большой.

Из (1) следует, что погрешность будет наименьшей, если модуль отношения $\frac{a_{i1}}{a_{11}}$ наименьший из возможных. Это будет в том случае, если a_{11} – наибольший по модулю элемент в первом столбце. Поэтому описанный выше алгоритм преобразуем к следующему виду.

Присвоим номер 1 тому уравнению, в котором коэффициент при x_1 наибольший по модулю. Этот коэффициент отличен от нуля, так как противное означало бы, что матрица A имеет нулевой первый столбец, т.е. вырождена. После этой перенумерации уравнений мы сделаем первый шаг метода Гаусса, т.е. перейдем от системы (4.1) к системе (4.3). Далее в подматрице $A^{(1)} = (a_{ij}^{(1)})_{i,j=2,\dots,n} \in M_{n-1}$ присвоим номер 2 тому уравнению, в котором коэффициент при x_2 наибольший по модулю, и сделаем следующий шаг метода Гаусса. Затем этот процесс применяется к подматрице $A^{(2)} \in M_{n-2}$ и так далее. Этот алгоритм называется *методом Гаусса с выбором главного элемента по столбцу*.

Можно преобразовать метод Гаусса и по-другому. Неизвестные в системе (4.1) равноправны, мы можем их занумеровать в произвольном порядке. Присвоим номер 1 той неизвестной, при которой коэффициент в первой строке отличен от 0. Если такой неизвестной не нашлось, то матрица A имеет нулевую первую строку, т.е. вырождена. После этой перенумерации неизвестных мы сделаем первый шаг метода Гаусса, т.е. перейдем от системы (4.1) к системе (4.3). Далее в подматрице $A^{(1)} = (a_{ij}^{(1)})_{i,j=2,\dots,n} \in M_{n-1}$ присвоим номер 2 той неизвестной, при которой коэффициент в первой строке матрицы $A^{(1)}$ (т.е. во второй строке матрицы A) отличен от 0, и сделаем следующий шаг метода Гаусса. Затем этот процесс применяется к подматрице $A^{(2)} \in M_{n-2}$ и так далее.

Если неизвестных, при которых коэффициент отличен от 0, несколько, то с вычислительной точки зрения не безразлично, какая из них получит номер 1. Рассуждениями, аналогичными вышеприведенным, можно установить, что погрешность, вносимая на шаге алгоритма, будет минимальной, если если a_{11} – наибольший по модулю элемент в первой строке. Поэтому описанный выше алгоритм преобразуем к следующему виду.

Присвоим номер 1 той неизвестной, при которой коэффициент в первой строке наибольший по модулю. Этот коэффициент отличен от нуля, так как противное означало бы, что матрица A имеет нулевую первую строку, т.е. вырождена. После этой перенумерации неизвестных мы сделаем первый шаг метода Гаусса, т.е. перейдем от системы (4.1) к системе (4.3). Далее в подматрице $A^{(1)} = (a_{ij}^{(1)})_{i,j=2,\dots,n} \in M_{n-1}$ присвоим номер 2 той неизвестной, при которой коэффициент в первой строке матрицы $A^{(1)}$ (т.е. во второй строке матрицы A) наибольший по модулю, и сделаем следующий шаг метода Гаусса. Затем этот процесс применяется к подматрице $A^{(2)} \in M_{n-2}$ и так далее. Этот алгоритм

называется *методом Гаусса с выбором главного элемента по строке*.

Для уменьшения вычислительной погрешности используют следующую комбинацию приведенных выше методов. В качестве a_{11} выбирается элемент, имеющий наибольший модуль среди всех элементов матрицы. Если этот элемент есть a_{ij} , то меняются номера у 1-й и i -й строк и у 1-го и j -го столбцов. После этой перенумерации уравнений и неизвестных делается первый шаг метода Гаусса, т.е. осуществляется переход от системы (4.1) к системе (4.3). Далее в подматрице $A^{(1)} = (a_{ij}^{(1)})_{i,j=2,\dots,n} \in M_{n-1}$ выбирается элемент a_{ij} с наибольшим модулем среди всех элементов матрицы $A^{(1)}$ и меняются номера у 1-й и i -й строк и у 1-го и j -го столбцов матрицы $A^{(1)}$ (т.е. у 2-й и i -й строк и у 2-го и j -го столбцов матрицы A). Затем этот процесс применяется к подматрице $A^{(2)} \in M_{n-2}$ и так далее. Этот алгоритм называется *методом Гаусса с выбором главного элемента по всей матрице*.

Вычислим дополнительные (по сравнению с обычным методом Гаусса) затраты вычислительной работы на решение системы по этим алгоритмам. На k -ом шаге ($k = 1, \dots, n$) метода Гаусса с выбором главного элемента по столбцу или строке требуется $n - k$ операций сравнения элементов матрицы A для нахождения максимального по модулю элемента. В методе Гаусса с выбором главного элемента по всей матрице это число равно $(n - k)^2$. Следовательно, в первых двух методах дополнительно требуется $\sum_{k=1}^n (n - k) = n(n - 1)/2 = O(n^2)$ ($n \rightarrow \infty$) операций сравнения, а в последнем методе — $\sum_{k=1}^n (n - k)^2 = (n - 1)n(2n - 1)/6 = n^3/3 + O(n^2)$ ($n \rightarrow \infty$) операций сравнения.

На большинстве ЭВМ операция сравнения двух чисел с плавающей точкой выполняется за время, по порядку равное времени вычитания этих чисел. (Это связано с тем, что вместо сравнения двух чисел выполняется операция вычитания одного числа из другого и сравнения результата с нулем. Поскольку сам результат нигде не запоминается и от него используется лишь его знак, то операция сравнения обычно осуществляется быстрее операции вычитания, однако следующая за операцией сравнения команда условного перехода с лихвой компенсирует эту разницу.) Поэтому в методе Гаусса с выбором главного элемента по столбцу или строке количество операций асимптотически то же, что в обычном методе Гаусса: $2/3 n^3 + O(n^2)$. В методе Гаусса с выбором главного элемента по всей матрице количество операций асимптотически в полтора раза больше, чем в обычном методе Гаусса: $n^3 + O(n^2)$. По этой причине этот метод обычно применяется тогда, когда с помощью других методов не удалось получить приемлемого по точности результата из-за сильного роста вычислительной погрешности (такая ситуация возникает, если матрица A имеет большое число обусловленности).

Теорема 1. *Метод Гаусса с выбором главного элемента по столбцу осуществим тогда и только тогда, когда $\det A \neq 0$.*

Доказательство. Шаг метода Гаусса переводит невырожденную матрицу

в невырожденную. Действительно, после перехода от матрицы (4.4) к матрице (4.7) по формулам (4.6), (4.8) определитель матрицы (4.4) равен определителю матрицы (4.7), умноженному на $a_{kk}^{(k-1)}$ (множитель возникает при вычислении по формулам (4.6), при преобразовании матрицы по формулам (4.8) определитель не изменяется, так как они задают элементарные преобразования матрицы). Согласно правилам вычисления определителей, определитель матрицы (4.4), получающейся после $k - 1$ шагов метода Гаусса, равен определителю матрицы $A^{(k-1)} = (a_{ij}^{(k-1)})_{i,j=k,\dots,n}$. Следовательно, невырожденность матрицы A эквивалентна невырожденности матриц $A^{(k)}$ для всех $k = 1, \dots, n$.

Очередной, k -й шаг метода Гаусса с выбором главного элемента по столбцу возможен тогда и только тогда, когда первый столбец матрицы $A^{(k-1)}$ ненулевой, т.е. эта матрица невырождена (см. подробное обоснование этого при построении метода).

Таким образом, осуществимость всех n шагов метода Гаусса эквивалентна невырожденности матриц $A^{(k)}$ для всех $k = 1, \dots, n$, что эквивалентно невырожденности матрицы A .

Теорема 2. *Метод Гаусса с выбором главного элемента по строке осуществим тогда и только тогда, когда $\det A \neq 0$.*

Доказательство повторяет доказательство предыдущей теоремы. Изменения только в том, что очередной, k -й шаг метода Гаусса с выбором главного элемента по строке возможен тогда и только тогда, когда первая строка матрицы $A^{(k-1)}$ ненулевая, т.е. эта матрица невырождена (см. подробное обоснование этого при построении метода).

Теорема 3. *Метод Гаусса с выбором главного элемента по всей матрице осуществим тогда и только тогда, когда $\det A \neq 0$.*

Доказательство повторяет доказательство теоремы 1. Изменения только в том, что очередной, k -й шаг метода Гаусса с выбором главного элемента по всей матрице возможен тогда и только тогда, матрица $A^{(k-1)}$ ненулевая (см. подробное обоснование этого при построении метода).

Замечание 1. Программная реализация методов Гаусса с выбором главного элемента. При реализации этих методов можно переставлять не строки или столбцы матрицы, а их номера. Сделать это можно, например, следующим способом.

Рассмотрим метод Гаусса с выбором главного элемента по всей матрице. Пусть массив $indi$ длиной n содержит номер строки матрицы A , массив $indj$ длиной n содержит номер столбца матрицы A . Вначале $indi(i)=i$, $indj(j)=j$, $i, j = 1, \dots, n$. Обращение к элементам матрицы A происходит следующим образом: элемент a_{ij} есть $a(indi(i), indj(j))$. Для того, чтобы переставить местами

i_1 -ю и i_2 -ю строки матрицы A , достаточно переставить местами i_1 -й и i_2 -й элементы массива `indi`; для того, чтобы переставить местами j_1 -й и j_2 -й столбцы матрицы A , достаточно переставить местами j_1 -й и j_2 -й элементы массива `indj`.

Существенным недостатком такого способа реализации перестановок строк и столбцов является замедление доступа к элементам массива.

Замечание 2. Для матриц A произвольного вида методы Гаусса с выбором главного элемента практически вытеснили обычный метод Гаусса из вычислительной практики. Совершенно иная ситуация для случая ленточных матриц A . Дело здесь в том, что перестановка строк или столбцов в таких матрицах приводит к увеличению ширины ленты, что часто недопустимо (поскольку вместо матрицы хранится только ее лента).

§ 8. МЕТОД ЖОРДАНА (ГАУССА-ЖОРДАНА)

Пусть требуется решить линейную систему $Ax = b$, $A \in M_n$ вида (4.1). Первый шаг метода Жордана сопадает с первым шагом метода Гаусса: система (4.1) преобразуется к виду

$$\begin{aligned} x_1 + a_{12}^{(1)}x_2 + \dots + a_{1n}^{(1)}x_n &= b_1^{(1)} \\ a_{22}^{(1)}x_2 + \dots + a_{2n}^{(1)}x_n &= b_2^{(1)} \\ \vdots &\quad \vdots \quad \ddots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \\ a_{n2}^{(1)}x_2 + \dots + a_{nn}^{(1)}x_n &= b_n^{(1)} \end{aligned} \tag{1}$$

по тем же формулам

$$\begin{aligned} a_{ij}^{(1)} &= a_{1j}/a_{11}, & b_1^{(1)} &= b_1/a_{11}, & j &= 2, \dots, n, \\ a_{ij}^{(1)} &= a_{ij} - a_{1j}^{(1)}a_{11}, & b_i^{(1)} &= b_i - b_1^{(1)}a_{11}, & i, j &= 2, \dots, n. \end{aligned}$$

После $k-1$, $k = 1, \dots, n$ шагов метода Жордана система (4.1) преобразована к виду

$$\begin{aligned} x_1 &+ a_{1,k}^{(k-1)}x_k + \dots + a_{1n}^{(k-1)}x_n = b_1^{(k-1)} \\ x_2 &+ a_{2,k}^{(k-1)}x_k + \dots + a_{2n}^{(k-1)}x_n = b_2^{(k-1)} \\ \vdots &\quad \vdots \quad \vdots \quad \ddots \quad \vdots \quad \vdots \quad \vdots \\ x_{k-1} &+ a_{k-1,k}^{(k-1)}x_k + \dots + a_{k-1,n}^{(k-1)}x_n = b_{k-1}^{(k-1)} \\ a_{k,k}^{(k-1)}x_k + \dots + a_{k,n}^{(k-1)}x_n &= b_k^{(k-1)} \\ \vdots &\quad \vdots \quad \ddots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \\ a_{n,k}^{(k-1)}x_k + \dots + a_{n,n}^{(k-1)}x_n &= b_n^{(k-1)} \end{aligned} \tag{2}$$

Предположим, что $a_{kk}^{(k-1)} \neq 0$. Поделив k -е уравнение системы (2) на $a_{kk}^{(k-1)}$, перепишем его в виде

$$x_k + a_{k,k+1}^{(k)}x_{k+1} + \dots + a_{kn}^{(k)}x_n = b_k^{(k)}, \tag{3}$$

где

$$a_{kj}^{(k)} = \frac{a_{kj}^{(k-1)}}{a_{kk}^{(k-1)}}, \quad j = k+1, \dots, n, \quad b_k^{(k)} = \frac{b_k^{(k-1)}}{a_{kk}^{(k-1)}}. \quad (4)$$

Умножим уравнение (3) на $a_{ik}^{(k-1)}$ и вычтем его из i -го уравнения системы (2), $i = 1, \dots, n$. В результате система (2) примет вид

$$\begin{array}{lcl} x_1 & + a_{1,k+1}^{(k)} x_{k+1} + \dots + a_{1n}^{(k)} x_n = b_1^{(k)} \\ x_2 & + a_{2,k+1}^{(k)} x_{k+1} + \dots + a_{2n}^{(k)} x_n = b_2^{(k)} \\ \vdots & \vdots & \vdots \ddots \vdots \vdots \vdots \\ x_{k-1} & + a_{k-1,k+1}^{(k)} x_{k+1} + \dots + a_{k-1,n}^{(k)} x_n = b_{k-1}^{(k)} \\ x_k & + a_{k,k+1}^{(k)} x_{k+1} + \dots + a_{kn}^{(k)} x_n = b_k^{(k)} \\ a_{k+1,k+1}^{(k)} x_{k+1} + \dots + a_{k+1,n}^{(k)} x_n = b_{k+1}^{(k)} \\ \vdots & \vdots \ddots \vdots \vdots \vdots \vdots \\ a_{n,k+1}^{(k)} x_{k+1} + \dots + a_{nn}^{(k)} x_n = b_n^{(k)} \end{array} \quad (5)$$

где

$$\begin{aligned} a_{ij}^{(k)} &= a_{ij}^{(k-1)} - a_{ik}^{(k-1)} a_{kj}^{(k)}, & i = 1, \dots, n, \quad i \neq k, \quad j = k+1, \dots, n. \\ b_i^{(k)} &= b_i^{(k-1)} - a_{ik}^{(k-1)} b_k^{(k)}, & i = 1, \dots, n, \quad i \neq k. \end{aligned} \quad (6)$$

Выражения (4), (6) являются формулами перехода от системы (2) к системе (5). Если обозначить $a_{ij}^{(0)} = a_{ij}$, $b_i^{(0)} = b_i$, $i, j = 1, \dots, n$, то переход от системы (4.1) к системе (1) будет осуществляться по тем же формулам при $k = 1$.

После проведения вычислений по формулам (4), (6) при $k = 1, \dots, n$ матрица системы (4.1) станет единичной матрицей. Следовательно, правая часть системы содержит искомое решение: $x_i = b_i^{(n)}$, $i = 1, \dots, n$.

Метод Жордана удобно применять для нахождения обратной матрицы. При этом вместо правой части b используется набор правых частей, состоящий из n столбцов единичной матрицы, над которыми одновременно производятся преобразования, задаваемые соотношениями (4), (6). После проведения n шагов метода Жордана этот набор будет состоять из столбцов обратной матрицы A^{-1} .

Поскольку на каждом шаге метода Жордана подматрица $A^{(k-1)} = (a_{ij}^{(k-1)})_{i,j=k,\dots,n}$ — та же, что на соответствующем шаге метода Гаусса, то метод Жордана осуществим тогда и только тогда, когда осуществим метод Гаусса, т.е. когда все главные угловые миноры матрицы A отличны от нуля.

Оценка количества арифметических операций в методе Жордана

- На вычисление $a_{kj}^{(k)}$ при $j = k+1, \dots, n$, $k = 1, \dots, n$ по формулам (4) требуется $\sum_{k=1}^n (n-k) = n(n-1)/2 = O(n^2)$ ($n \rightarrow \infty$) операций деления.

2. На вычисление $a_{ij}^{(k)}$ при $i = 1, \dots, n$, $i \neq k$, $j = k+1, \dots, n$, $k = 1, \dots, n$ по формулам (6) требуется $\sum_{k=1}^n (n-k)(n-1) = (n-1)^2 n/2 = n^3/2 + O(n^2)$ ($n \rightarrow \infty$) операций умножения и столько же операций вычитания.

3. На вычисление $b_k^{(k)}$ при $k = 1, \dots, n$ по формулам (4) требуется n операций деления.

4. На вычисление $b_i^{(k)}$ при $i = 1, \dots, n$, $i \neq k$, $k = 1, \dots, n$ по формулам (6) требуется $\sum_{k=1}^n (n-1) = n(n-1) = O(n^2)$ ($n \rightarrow \infty$) операций умножения и столько же операций вычитания.

Таким образом, метод Жордана требует $O(n^2) + n^3/2 + n + O(n^2) = n^3/2 + O(n^2)$ ($n \rightarrow \infty$) мультиплексивных операций и столько же аддитивных операций. Всего: $n^3 + O(n^2)$ ($n \rightarrow \infty$) арифметических операций.

По аналогии с методом Гаусса можно строить *метод Жордана с выбором главного элемента*. Именно, в подматрице $A^{(k-1)} = (a_{ij}^{(k-1)})_{i,j=k,\dots,n}$ (совпадающей с подматрицей из метода Гаусса) той же процедурой, что и в методе Гаусса, выбирается главный элемент.

§ 9. ПОЛОЖИТЕЛЬНО ОПРЕДЕЛЕННЫЕ МАТРИЦЫ

Определение. Матрица $A \in M_n$ называется *положительно определенной* (обозначается $A > 0$), если для всех $x \in \mathbf{C}^n$ выражение (Ax, x) вещественно и $(Ax, x) > 0$ для всех $x \in \mathbf{C}^n$, $x \neq 0$ (здесь (\cdot, \cdot) означает обычное скалярное произведение в \mathbf{C}^n , $(x, y) = y^*x$, где $y^* = (\bar{y}_1, \dots, \bar{y}_n)$, черта над символом обозначает, как обычно, знак комплексного сопряжения). Если рассматриваемая матрица A вещественна, то часто положительно определенной называется матрица A , для которой $(Ax, x) > 0$ для всех $x \in \mathbf{R}^n$, $x \neq 0$.

Замечание 1. Если матрица $A \in M_n$ – самосопряженная (т.е. $A^* = A$), то выражение (Ax, x) вещественно для всех $x \in \mathbf{C}^n$.

Действительно, $(Ax, x) = (x, A^*x) = (x, Ax) = \overline{(Ax, x)}$ и потому (Ax, x) вещественно.

Лемма 1. *Если матрица A положительно определена, то она невырождена.*

Доказательство. Предположим противное, $\det A = 0$. Тогда линейная система $Ax = 0$ имеет решение $x \in \mathbf{C}^n$, $x \neq 0$. Для этого x выражение $(Ax, x) = (0, x) = 0$, что противоречит положительной определенности матрицы A .

Лемма 2. *Если матрица A положительно определена, то для нее существует LU-разложение.*

Доказательство. В соответствии с теоремой 4.1 нам надо проверить, что главные угловые миноры положительно определенной матрицы A отличны от нуля.

Определим отображение $x_{(k)}$ пространства $\mathbf{C}^n \rightarrow \mathbf{C}^k$, $k \leq n$ действующее по правилу: для всякого $x = (x_1, \dots, x_n)^t \in \mathbf{C}^n$ $x_{(k)} = (x_1, \dots, x_k)^t \in \mathbf{C}^k$. Это отображение есть отображение "на", т.е. для каждого элемента $x = (x_1, \dots, x_k)^t \in \mathbf{C}^k$ найдется элемент $\hat{x} \in \mathbf{C}^n$, являющийся прообразом x при этом отображении (например, $\hat{x} = (x_1, \dots, x_k, 0, \dots, 0)^t \in \mathbf{C}^n$).

Поскольку матрица A положительно определена, то для всякого $k = 1, \dots, n$ и всякого $x \in \mathbf{C}^n$, такого, что $x_{(k)} \neq 0$ выражение $(Ax_{(k)}, x_{(k)})$ вещественно и положительно. По правилу перемножения матриц $(Ax_{(k)}, x_{(k)}) = (A_k x_{(k)}, x_{(k)})_k$, где

$$A_k = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1k} \\ a_{21} & a_{22} & \dots & a_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ a_{k1} & a_{k2} & \dots & a_{kk} \end{pmatrix}$$

-главный угловая подматрица A , $(\cdot, \cdot)_k$ – обычное скалярное произведение в пространстве \mathbf{C}^k , $(x, y)_k = y^* x$, $x, y \in \mathbf{C}^k$. Следовательно, выражение $(A_k x, x)_k$ вещественно и положительно для всех $x \in \mathbf{C}^k$, т.е. матрицы $A_k \in \mathbf{M}_k$, $k = 1, \dots, n$ положительно определены. Пользуясь леммой 1, получаем, что матрицы A_k , $k = 1, \dots, n$ невырождены. Из теоремы 4.1 теперь вытекает требуемый результат.

Лемма 3. Самосопряженная матрица $A \in \mathbf{M}_n$ положительно определена тогда и только тогда, когда все ее собственные значения вещественны и положительны.

Доказательство. Пусть $A \in \mathbf{M}_n$ самосопряженная положительно определенная матрица, λ – ее собственное значение, $x \neq 0$ – соответствующий собственный вектор: $Ax = \lambda x$. Умножим это равенство скалярно на x , получим $(Ax, x) = \lambda(x, x)$ и $\lambda = \frac{(Ax, x)}{\|x\|^2}$. Поскольку (Ax, x) вещественно и положительно, то $\lambda > 0$.

Пусть $A \in \mathbf{M}_n$ самосопряженная матрица и $\lambda_i > 0$, $i = 1, 2, \dots, n$ – ее собственные значения. В курсе линейной алгебры было доказано, что всякая самосопряженная (симметричная в вещественном случае) матрица диагонализуема в евклидовом базисе, т.е. существует ортонормированный базис x_1, x_2, \dots, x_n , $(x_i, x_j) = \delta_{ij}$, состоящий из собственных векторов матрицы A : $Ax_i = \lambda_i x_i$. Пусть $x \neq 0$ – произвольный вектор, $x = \sum_{i=1}^n c_i x_i$ – его разложение по базису $\{x_i\}$, причем $\|x\|^2 = \sum_{i=1}^n |c_i|^2 \neq 0$. Рассмотрим выражение $(Ax, x) = (A \sum_{i=1}^n c_i x_i, \sum_{i=1}^n c_i x_i) = (\sum_{i=1}^n c_i \lambda_i x_i, \sum_{i=1}^n c_i x_i) = \sum_{i=1}^n \lambda_i |c_i|^2$. Следовательно, (Ax, x) вещественно. Поскольку $\lambda_i > 0$ и не все c_i равны 0, то (Ax, x) положительно. Итак, для всякого вектора $x \neq 0$ выражение (Ax, x) вещественно и положительно, что и означает положительную определенность матрицы A .

§ 10. МЕТОД ХОЛЕЦКОГО (КВАДРАТНОГО КОРНЯ)

Пусть требуется решить линейную систему $Ax = b$ с самосопряженной (симметричной в вещественном случае) матрицей $A \in \mathbf{M}_n$, $A^* = A$.

§ 10.1. Разложение Холецкого

Обозначим через $\text{RT}(n)$ подгруппу невырожденных верхних треугольных матриц в \mathbf{M}_n , а через $\text{UT}(n)$ – подгруппу в $\text{RT}(n)$ матриц с единицами на главной диагонали.

Теорема 1. *Пусть матрица A – самосопряженная ($A^* = A$) и все ее главные угловые миноры отличны от нуля. Тогда существуют матрица $R = (r_{ij}) \in \text{RT}(n)$ с вещественными положительными элементами на главной диагонали ($r_{ii} > 0$ для всех $i = 1, \dots, n$) и диагональная матрица D с вещественными равными по модулю единице диагональными элементами ($d_{ii} \in \{-1, 1\}$ для всех $i = 1, \dots, n$) такие, что $A = R^*DR$.*

Доказательство. По теореме 4.1 для матрицы A осуществим LU -разложение, т.е. существуют $L \in \text{LT}(n)$ и $U \in \text{UT}(n)$ такие, что $A = LU$. Поскольку матрица $L = (l_{ij})$ невырождена, то $l_{ii} \neq 0$, $i = 1, \dots, n$ и матрица

$$\hat{D} = \text{diag}(l_{11}, \dots, l_{nn}) \quad (1)$$

обратима, $\hat{D}^{-1} = \text{diag}(l_{11}^{-1}, \dots, l_{nn}^{-1})$. Положим $\hat{L} = L\hat{D}^{-1} \in \text{LT}(n)$. Тогда по правилам перемножения матриц $\hat{l}_{ii} = 1$, $i = 1, \dots, n$.

Подставим это представление матрицы $L = \hat{L}\hat{D}$ в LU -разложение матрицы A : $A = \hat{L}\hat{D}U$. Так как $A = A^*$, то $A = \hat{L}\hat{D}U = A^* = U^*\hat{D}^*\hat{L}^*$. Поэтому $U = \hat{D}^{-1}\hat{L}^{-1}U^*\hat{D}^*\hat{L}^*$,

$$U(\hat{L}^*)^{-1} = \hat{D}^{-1}\hat{L}^{-1}U^*\hat{D}^*. \quad (2)$$

Заметим, что $\hat{L}^* \in \text{RT}(n)$, причем главная диагональ этой матрицы состоит из единиц. Следовательно $\hat{L}^* \in \text{UT}(n)$. Поэтому в левой части равенства (2) стоит матрица $U(\hat{L}^*)^{-1} \in \text{UT}(n)$. В право же части равенства (2) стоит произведение нижних треугольных матриц, которое является опять нижней треугольной матрицей, т.е. принадлежит $\text{LT}(n)$. Поэтому из (2) вытекает

$$U(\hat{L}^*)^{-1} \in \text{UT}(n) \cap \text{LT}(n).$$

Единственной матрицей, которая принадлежит одновременно подгруппам $\text{UT}(n)$ и $\text{LT}(n)$ является I – единичная матрица. Следовательно,

$$U(\hat{L}^*)^{-1} = \hat{D}^{-1}\hat{L}^{-1}U^*\hat{D}^* = I. \quad (3)$$

Таким образом, $U = \hat{L}^*$ и $A = U^*\hat{D}U$. Далее, из (3)

$$I = \hat{D}^{-1}\hat{L}^{-1}U^*\hat{D}^* = \hat{D}^{-1}\hat{L}^{-1}(\hat{L}^*)^*\hat{D}^* = \hat{D}^{-1}\hat{L}^{-1}\hat{L}\hat{D}^* = \hat{D}^{-1}\hat{D}^*$$

т.е. $\hat{D} = \hat{D}^*$. В силу (1) получаем $l_{kk} = \overline{l_{kk}}$, т.е. l_{kk} – вещественные для всех $k = 1, \dots, n$. Представив матрицу \hat{D} в виде $\hat{D} = |\hat{D}|^{1/2} D |\hat{D}|^{1/2}$, где

$$D = \text{diag}(\text{sign } l_{11}, \dots, \text{sign } l_{nn}), \quad |\hat{D}|^{1/2} = \text{diag}(\sqrt{|l_{11}|}, \dots, \sqrt{|l_{nn}|}),$$

получаем $A = U^* |\hat{D}|^{1/2} D |\hat{D}|^{1/2} U$. Обозначим $R = |\hat{D}|^{1/2} U \in RT(n)$. Тогда $A = R^* D R$, причем диагональные элементы r_{ii} матрицы R равны $\sqrt{|l_{ii}|} > 0$. Следовательно, полученное разложение является требуемым.

Замечание 1. Если матрица A – вещественная, то все участвующие в теореме 1 матрицы вещественные.

Замечание 2. Если в условиях теоремы 1 матрица A положительно определена, то матрица D в теореме 1 – единичная, т.е. разложение, даваемое этой теоремой, имеет вид $A = R^* R$.

Действительно, если матрица A положительно определена и для нее справедливо разложение $A = R^* D R$, то для всякого $x \in \mathbf{C}^n$, $x \neq 0$ выражение $(Ax, x) = (R^* D Rx, x) = (D Rx, Rx)$ вещественно и положительно. Для всякого $y \in \mathbf{C}^n$ положим $x = R^{-1}y$. Тогда выражение $(Dy, y) = (Ax, x)$ вещественно и положительно для всех $y \in \mathbf{C}^n$, $y \neq 0$, т.е. матрица D положительно определена.

Поскольку $D = \text{diag}(d_{11}, \dots, d_{nn})$, то $(Dy, y) = \sum_{j=1}^n d_{jj} y_j \overline{y_j} = \sum_{j=1}^n d_{jj} |y_j|^2$. По доказанному, эта сумма вещественна и положительна. Выбирая здесь $y = e_k$, $k = 1, \dots, n$, где e_k – орты стандартного базиса, находим, что все d_{kk} , $k = 1, \dots, n$ должны быть вещественными и положительными. Поскольку $|d_{kk}| = 1$, то это означает, что $d_{kk} = 1$, $k = 1, \dots, n$.

Замечание 3. Если матрица A положительно определена, то для нее выполнены условия теоремы 1. Это вытекает из леммы 9.2.

§ 10.2. Алгоритм построения разложения Холецкого

Построим алгоритм нахождения разложения из теоремы 1.

Пусть для самосопряженной матрицы A ($A^* = A$) требуется найти верхнюю треугольную матрицу $R = (r_{ij})$ с вещественными положительными элементами на главной диагонали ($r_{ii} > 0$ для всех $i = 1, \dots, n$) и диагональную матрицу D с равными по модулю единице диагональными элементами ($|d_{ii}| = 1$ для всех $i = 1, \dots, n$) такую, что $A = R^* D R$.

Элемент (k, j) матрицы DR равен $(DR)_{kj} = \sum_{i=1}^n d_{ki} r_{ij} = d_{kk} r_{kj}$, так как матрица D – диагональная; элемент (i, k) матрицы R^* равен $(R^*)_{ik} = \overline{r_{ki}}$; элемент (i, j) матрицы $R^* DR$ равен $(R^* DR)_{ij} = \sum_{k=1}^n (R^*)_{ik} (DR)_{kj} = \sum_{k=1}^n \overline{r_{ki}} d_{kk} r_{kj}$. Следовательно, равенство $A = R^* DR$ дает нам уравнения

$$\sum_{k=1}^n \overline{r_{ki}} d_{kk} r_{kj} = a_{ij}, \quad i, j = 1, \dots, n. \quad (4)$$

Поскольку матрицы A и R^*DR – самосопряженные, то уравнение с номером (j, i) получается из уравнения с номером (i, j) путем комплексного сопряжения и не дает ничего нового. Поэтому система (4) эквивалентна системе

$$\sum_{k=1}^n \overline{r_{ki}} d_{kk} r_{kj} = a_{ij}, \quad i \leq j, \quad i, j = 1, \dots, n. \quad (5)$$

Таким образом, (5) представляет собой систему из $n(n+1)/2$ уравнений с $n(n+1)/2$ неизвестными r_{ij} , $i \leq j$ (напомним, $R \in \text{RT}(n)$ и $r_{ij} = 0$ при $i > j$) и n неизвестными d_{kk} , $k = 1, \dots, n$ (при этом $r_{kk} > 0$ и $d_{kk} \in \{-1, 1\}$).

Получим формулы для решения системы (5), которые и составляют алгоритм метода Холецкого.

Перепишем (5) в виде

$$\sum_{k=1}^i \overline{r_{ki}} d_{kk} r_{kj} + \sum_{k=i+1}^n \overline{r_{ki}} d_{kk} r_{kj} = a_{ij}, \quad i \leq j, \quad i, j = 1, \dots, n. \quad (6)$$

Поскольку матрица R – верхняя треугольная, то $r_{ki} = 0$ при $k > i$ и вторая из сумм в (6) равна нулю. Следовательно, система (5) эквивалентна следующей

$$\sum_{k=1}^{i-1} \overline{r_{ki}} d_{kk} r_{kj} + r_{ii} d_{ii} r_{ij} = a_{ij}, \quad i \leq j, \quad i, j = 1, \dots, n, \quad (7)$$

(здесь считается, что сумма вида $\sum_{k=1}^{i-1}$ равна нулю, если верхний предел суммирования меньше нижнего; это позволяет не рассматривать отдельно случай $i = 1$; также в (7) мы учли, что r_{ii} – вещественный элемент). Выделим в сумме (7) отдельно случай $i = j$

$$\begin{cases} r_{ii}^2 d_{ii} = a_{ii} - \sum_{k=1}^{i-1} |r_{ki}|^2 d_{kk}, & i = 1, \dots, n, \\ r_{ii} d_{ii} r_{ij} = a_{ij} - \sum_{k=1}^{i-1} \overline{r_{ki}} d_{kk} r_{kj}, & i < j, \quad i, j = 1, \dots, n, \end{cases}$$

Отсюда получаем расчетные формулы:

$$\begin{cases} d_{ii} = \text{sign}(a_{ii} - \sum_{k=1}^{i-1} |r_{ki}|^2 d_{kk}), & i = 1, \dots, n, \\ r_{ii} = \sqrt{|a_{ii} - \sum_{k=1}^{i-1} |r_{ki}|^2 d_{kk}|}, & i = 1, \dots, n, \\ r_{ij} = (a_{ij} - \sum_{k=1}^{i-1} \overline{r_{ki}} d_{kk} r_{kj}) / (r_{ii} d_{ii}), & i < j, \quad i, j = 1, \dots, n. \end{cases} \quad (8)$$

Процесс вычислений по этим формулам строится следующим образом: вначале вычисляются неизвестные элементы первых строк матриц D и R :

$$d_{11} = \text{sign } a_{11}, \quad r_{11} = \sqrt{|a_{11}|}, \quad r_{1j} = a_{1j} / (r_{11} d_{11}), \quad j = 2, \dots, n;$$

потом по формулам (8) при $i = 2$ вычисляются неизвестные элементы вторых строк матриц D и R :

$$\begin{aligned} d_{22} &= \text{sign}(a_{22} - |r_{12}|^2 d_{11}), \quad r_{22} = \sqrt{|a_{22} - |r_{12}|^2 d_{11}|}, \\ r_{2j} &= (a_{2j} - \bar{r}_{12} d_{11} r_{1j}) / (r_{22} d_{22}), \quad j = 3, \dots, n; \end{aligned}$$

затем по формулам (8) при $i = 3$ вычисляются неизвестные элементы третьих строк матриц D и R и так далее.

Замечание 4. Организация хранения матриц A , R и D в памяти.

Поскольку для самосопряженной матрицы $a_{ji} = \bar{a}_{ij}$, то можно вместо всей матрицы A хранить только ее верхний треугольник: a_{ij} , $i \leq j$, $i, j = 1, \dots, n$.

Формулы (8) таковы, что при вычислении элемента r_{ij} используются значения элемента a_{ij} и вычисленных ранее элементов r_{km} , $k < i$. Это позволяет хранить верную треугольную матрицу R на месте верхнего треугольника матрицы A : $r_{ij} \equiv a_{ij}$, $i \leq j$, $i, j = 1, \dots, n$. Матрица D обычно хранится в виде отдельного вектора $d = (d_{11}, \dots, d_{nn})^t$.

Замечание 5. Случай положительной матрицы A .

В силу замечания 2 в этом случае разложение матрицы имеет более простой вид $A = R^*R$. Формулы (8) тоже упрощаются:

$$\begin{cases} r_{ii} = \sqrt{(a_{ii} - \sum_{k=1}^{i-1} |r_{ki}|^2)}, & i = 1, \dots, n, \\ r_{ij} = (a_{ij} - \sum_{k=1}^{i-1} \bar{r}_{ki} r_{kj}) / r_{ii}, & i < j, \quad i, j = 1, \dots, n. \end{cases} \quad (9)$$

Если матрица A еще и вещественная, то в формулах (9) можно убрать знак модуля и комплексного сопряжения:

$$\begin{cases} r_{ii} = \sqrt{(a_{ii} - \sum_{k=1}^{i-1} r_{ki}^2)}, & i = 1, \dots, n, \\ r_{ij} = (a_{ij} - \sum_{k=1}^{i-1} r_{ki} r_{kj}) / r_{ii}, & i < j, \quad i, j = 1, \dots, n. \end{cases}$$

§ 10.3. Оценка количества арифметических операций в алгоритме построения разложения Холецкого

1. Вычисление элемента d_{ii} , $i = 1, \dots, n$ по формулам (8) требует $i - 1$ мультипликативных и столько же аддитивных операций. Следовательно, вычисление всех элементов матрицы D требует $\sum_{i=1}^n (i - 1) = n(n - 1)/2 = O(n^2)$ мультипликативных и столько же аддитивных операций.

2. Вычисление элемента r_{ii} по формулам (8) требует одной операции извлечения корня и $i - 1$ мультипликативных и столько же аддитивных операций (умножение на $d_{ii} \in \{-1, 1\}$ мы за операцию не считаем). При фиксированном $i = 1, \dots, n$ вычисление элементов r_{ij} для всех $j = i + 1, \dots, n$ по формулам (8) требует $1 + \sum_{j=i+1}^n (i - 1) = (n - i)(i - 1) + 1$ мультипликативных и $\sum_{j=i+1}^n (i - 1) = (n - i)(i - 1)$ аддитивных операций. Следовательно, вычисление всех элементов матрицы R требует n операций извлечения корня, $\sum_{i=1}^n ((n - i)(i - 1) + (i - 1) + 1) = n^3/6 + O(n^2)$ ($n \rightarrow \infty$) мультипликативных и $\sum_{i=1}^n ((n - i)(i - 1) + (i - 1)) = n^3/6 + O(n^2)$ ($n \rightarrow \infty$) аддитивных операций (подробное вычисление см. при подсчете количества арифметических операций для алгоритма LU -разложения).

На вычислительных машинах, имеющих аппаратную поддержку для вычисления трансцендентных функций вещественного аргумента, операция извлечения корня выполняется за время, по порядку равное времени деления двух вещественных чисел. Поэтому на таких вычислительных системах операции извлечения корня в описанном выше алгоритме займут время, равное времени выполнения $O(n)$ мультипликативных операций.

На вычислительных машинах, не имеющих аппаратной поддержки для вычисления трансцендентных функций вещественного аргумента, операции извлечения корня в описанном выше алгоритме займут время, равное времени выполнения $O(\text{const } n) = O(n)$ мультипликативных операций.

Таким образом, алгоритм построения разложения Холецкого требует для своего проведения выполнения $n^3/6 + O(n^2)$ ($n \rightarrow \infty$) мультипликативных и столько же аддитивных операций, а в сумме – $n^3/3 + O(n^2)$ ($n \rightarrow \infty$) арифметических операций, т.е. асимптотически вдвое меньше, чем в методе Гаусса или алгоритме построения LU -разложения.

§ 11. МЕТОД ОРТОГОНАЛИЗАЦИИ

Пусть требуется решить линейную систему

$$Ax = b \tag{1}$$

с матрицей $A = (a_{ij}) \in \mathbf{M}_n$ и правой частью $b = (b_1, \dots, b_n)^t$. Пусть $x = (x_1, \dots, x_n)^t$ – точное решение этой системы. Введем следующие вектора из \mathbf{C}^{n+1} :

$$Y = (x_1, \dots, x_n, 1)^t \in \mathbf{C}^{n+1}, \tag{2}$$

$$A_i = (a_{i1}, \dots, a_{in}, -b_i)^t \in \mathbf{C}^{n+1}, i = 1, \dots, n. \quad (3)$$

С использованием этих обозначений система (1) может быть записана в виде n условий ортогональности

$$(Y, A_i) = 0, \quad i = 1, \dots, n, \quad (4)$$

где (\cdot, \cdot) – обычное евклидово скалярное произведение в \mathbf{C}^{n+1} . Соотношения (4) дают новое определение понятию решения системы (1): решить систему (1) – это значит найти вектор Y вида (2), ортогональный всем векторам (3).

Будем строить последовательность подпространств

$$\mathbf{C}^{n+1} = \mathbf{E}^{(0)} \supset \mathbf{E}^{(1)} \supset \dots \supset \mathbf{E}^{(n)},$$

где $\mathbf{E}^{(k)} = \langle e_{k+1}^{(k)}, \dots, e_{n+1}^{(k)} \rangle$ – $(n-k+1)$ -мерное подпространство в \mathbf{C}^{n+1} , состоящее из векторов, ортогональных векторам A_1, \dots, A_k (в частности, все вектора одномерного подпространства $\mathbf{E}^{(n)}$ ортогональны A_1, \dots, A_n , т.е. $\mathbf{E}^{(n)}$ содержит решение, которое выделяется из всех других векторов этого подпространства тем, что его последняя координата равна 1).

В качестве базиса начального пространства $\mathbf{E}^{(0)}$ возьмем стандартный базис \mathbf{C}^{n+1} : $e_1^{(0)} = (1, 0, \dots, 0), \dots, e_{n+1}^{(0)} = (0, \dots, 0, 1)$.

Для всех $k = 1, \dots, n$ базис $e_{k+1}^{(k)}, \dots, e_{n+1}^{(k)}$ каждого следующего подпространства $\mathbf{E}^{(k)}$ строится по предыдущему базису $e_k^{(k-1)}, \dots, e_{n+1}^{(k-1)}$ подпространства $\mathbf{E}^{(k-1)}$ по формулам

$$e_i^{(k)} = e_i^{(k-1)} - \frac{(A_k, e_i^{(k-1)})}{(A_k, e_k^{(k-1)})} e_k^{(k-1)}, \quad i = k+1, \dots, n. \quad (5)$$

Проверим, что $\mathbf{E}^{(k)}$ состоит из векторов, ортогональных векторам A_1, \dots, A_k .

а) Базис пространства $\mathbf{E}^{(k)}$ получается как линейная комбинация базиса пространства $\mathbf{E}^{(k-1)}$, все элементы которого ортогональны векторам A_1, \dots, A_{k-1} . Следовательно, вектора из $\mathbf{E}^{(k)}$ ортогональны векторам A_1, \dots, A_{k-1} .

б) Проверим, что все элементы базиса пространства $\mathbf{E}^{(k)}$ ортогональны вектору A_k . $(A_k, e_i^{(k)}) = (A_k, e_i^{(k-1)}) - \frac{(A_k, e_i^{(k-1)})}{(A_k, e_k^{(k-1)})} (A_k, e_k^{(k-1)}) = (A_k, e_i^{(k-1)}) - (A_k, e_i^{(k-1)}) = 0, \quad i = k+1, \dots, n.$

Лемма 1. Для всех $k = 0, \dots, n$ и всех $i = k+1, \dots, n+1$ вектор $e_i^{(k)}$ имеет не более $k+1$ отличных от нуля компонент. Именно, могут быть отличными от нуля компоненты этого вектора с номерами $1, \dots, k$ и i .

Доказательство. При $k = 0$ это следует из выбора $e_i^{(0)}, i = 1, \dots, n$. Предположим, что утверждение леммы справедливо для $e_i^{(k-1)}, i = k, \dots, n$. Покажем,

что оно верно для $e_i^{(k)}$, $i = k+1, \dots, n$. Действительно, по предположению у вектора $e_k^{(k-1)}$ только компоненты $1, \dots, k$ могут быть отличны от нуля. Поэтому из формул (5) получаем, что вектор $e_i^{(k)}$ получается из вектора $e_i^{(k-1)}$ изменением не более чем первых k компонент. По предположению у вектора $e_i^{(k-1)}$ возможно отличны от нуля только компоненты $1, \dots, k-1$ и i . Следовательно, у $e_i^{(k)}$, $i = k+1, \dots, n$ могут быть отличны от нуля компоненты с номерами $1, \dots, k$ и i .

Следствие 1. По доказанному в лемме 1 вектор $e_i^{(k)}$ получается из $e_i^{(k)}$ изменением компонент $1, \dots, k$ (остальные компоненты у вектора $e_k^{(k-1)}$ нулевые). Следовательно, у всех $e_{n+1}^{(k)}$, $k = 1, \dots, n$ ($n+1$ -я компонента равна 1). Поэтому вектор $e_{n+1}^{(n)}$ является решением задачи (4).

Оценка количества арифметических операций в методе ортогонализации

Оценим трудоемкость вычислений по формулам (5) для фиксированного k , а затем просуммируем полученные оценки по всем $k = 1, \dots, n$.

1. Знаменатель $(A_k, e_k^{(k-1)})$ в (5) от i не зависит и вычисляется один раз для каждого k . В силу леммы 1 у вектора $e_k^{(k-1)}$ только компоненты $1, \dots, k$ могут быть отличны от нуля, поэтому на вычисление скалярного произведения $(A_k, e_k^{(k-1)})$ потребуется k операций умножения и $k-1$ операций сложения.

2. На вычисление скалярного произведения $(A_k, e_i^{(k-1)})$ в (5) по лемме 1 потребуется k операций умножения и $k-1$ операций сложения (поскольку у вектора $e_i^{(k-1)}$ только компоненты $1, \dots, k-1$ и i могут быть отличны от нуля). Следовательно, на вычисление этих скалярных произведений для всех $i = k+1, \dots, n$ потребуется $\sum_{i=k+1}^n k = k(n-k)$ операций умножения и $\sum_{i=k+1}^n (k-1) = (k-1)(n-k)$ операций сложения.

3. На вычисление дроби $\frac{(A_k, e_i^{(k-1)})}{(A_k, e_k^{(k-1)})}$ для всех $i = k+1, \dots, n$ в (5) потребуется $n-k$ операций деления.

4. На вычисление вектора $e_i^{(k)}$ по формуле (5) при условии, что дробь $\frac{(A_k, e_i^{(k-1)})}{(A_k, e_k^{(k-1)})}$ уже вычислена, потребуется k операций умножения и столько же операций сложения (поскольку в силу леммы 1 у вектора $e_k^{(k-1)}$ только компоненты $1, \dots, k$ могут быть отличны от нуля). Следовательно, на проведение этих вычислений для всех $i = k+1, \dots, n$ потребуется $\sum_{i=k+1}^n k = k(n-k)$ операций умножения и столько же операций сложения.

Итак, при фиксированном $k = 1, \dots, n$ трудоемкость формул (5) составляет $k + k(n-k) + (n-k) + k(n-k) = 2k(n-k) + n$ мультипликативных операций и $k-1 + (k-1)(n-k) + k(n-k) = 2k(n-k) + 2k - n - 1$ аддитивных операций. Следовательно, трудоемкость всего метода ортогонализации составляет

$\sum_{k=1}^n (2k(n-k) + n) = 2n \sum_{k=1}^n k - 2 \sum_{k=1}^n k^2 + n^2 = 2n \cdot n(n+1)/2 - 2n(n+1)(2n+1)/6 + n^2 = n^3 + O(n^2) - \frac{2}{3}n^3 + O(n^2) = n^3/3 + O(n^2)$ мультипликативных операций и $\sum_{k=1}^n (2k(n-k) + 2k - n - 1) = n^3/3 + O(n^2)$ аддитивных операций. Таким образом, метод ортогонализации асимптотически требует такого же количества арифметических операций (суммарно $\frac{2}{3}n^3 + O(n^2)$ ($n \rightarrow \infty$)), как и метод Гаусса.

МЕТОДЫ РЕШЕНИЯ ЛИНЕЙНЫХ СИСТЕМ, ОСНОВАННЫЕ НА УНИТАРНЫХ ПРЕОБРАЗОВАНИЯХ МАТРИЦ

Каждый из изложенных выше методов решения линейных систем может быть представлен в виде последовательности элементарных преобразований матрицы (см., например, такое представление в §4 для метода Гаусса). Каждое из преобразований задается некоторой матрицей P , так что применение этого преобразования эквивалентно умножению (слева) исходной матрицы A на матрицу P . Таким образом, каждый шаг приведенных выше алгоритмов есть переход от матрицы A к матрице $A := PA$. О числе обусловленности этой новой матрицы $A := PA$ можно лишь утверждать, что $\kappa(PA) \leq \kappa(P)\kappa(A)$. Поэтому может случиться так, что в процессе проведения преобразований число обусловленности матрицы возрастает и на каждом шаге метод будет вносить все большую вычислительную погрешность. В результате может оказаться, что исходная матрица имела приемлемое число обусловленности, однако после нескольких шагов алгоритма она уже имеет слишком большое число обусловленности, так что последующие шаги алгоритма приведут к появлению очень большой вычислительной погрешности.

Возникает идея подбирать матрицы преобразования P так, чтобы число обусловленности матрицы в процессе преобразований не возрастало. Лемма 1.5 указывает нам пример таких матриц: если матрица преобразования P унитарна (ортогональна в вещественном случае), то относительно спектральной нормы $\kappa(PA) = \kappa(A)$.

Излагаемые ниже метод вращений и метод отражений представляют собой алгоритмы подбора унитарных матриц преобразований P , таких, что в результате всех этих преобразований исходная матрица A приводится к треугольному виду. Система с треугольной матрицей затем решается, например, обратным ходом метода Гаусса. Несмотря на то, что трудоемкость этих методов больше, чем метода Гаусса (соответственно в 3 и 2 раза), эти методы получили широкое распространение в вычислительной практике благодаря своей устойчивости к накоплению вычислительной погрешности.

§ 12. МЕТОД ВРАЩЕНИЙ

В этом методе в качестве элементарного преобразования матрицы выбирается умножение ее на матрицу вращения.

§ 12.1. Матрица элементарного вращения и ее свойства

Определение. Элементарным вращением $T_{ij} = T_{ij}(\varphi)$ называется преобразование пространства, задаваемое матрицей $T_{ij} = (t_{kl})_{k,l=1,\dots,n}$, в которой только следующие элементы отличны от нуля: $t_{ii} = \cos \varphi$, $t_{jj} = \cos \varphi$, $t_{ij} = -\sin \varphi$, $t_{ji} = \sin \varphi$, $t_{kk} = 1$ для всех $k = 1, \dots, n, k \neq i, j$:

$$T_{ij} = \begin{pmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & & \cos \varphi & & -\sin \varphi \\ & & & & 1 & \\ & & & & & \ddots \\ & & & & & & 1 \\ & & & & & & & \ddots \\ & & & & & & & & 1 \end{pmatrix}. \quad (1)$$

Если $\langle e_1, \dots, e_n \rangle$ – базис \mathbf{C}^n ($e_k = (\underbrace{0, \dots, 0}_{k-1}, 1, 0, \dots, 0)^t$), то T_{ij} является вращением в подпространстве $\langle e_i, e_j \rangle$ и не изменяет подпространства $\langle e_1, \dots, e_{i-1}, e_{i+1}, \dots, e_{j-1}, e_{j+1}, \dots, e_n \rangle$ (другими словами, T_{ij} изменяет только i -ю и j -ю координаты векторов). Поэтому для изучения свойств преобразования T_{ij} достаточно изучить свойства преобразования

$$T = \begin{pmatrix} \cos \varphi & -\sin \varphi \\ \sin \varphi & \cos \varphi \end{pmatrix}$$

в двумерном пространстве.

Лемма 1. Матрица T_{ij} является ортогональной матрицей.

Доказательство. Следует из ортогональности матрицы T .

Лемма 2. Для всякого вектора $\mathbf{r} = \begin{pmatrix} x \\ y \end{pmatrix} \in \mathbf{R}^2$, $\mathbf{r} \neq 0$ существует матрица $T = \begin{pmatrix} \cos \varphi & -\sin \varphi \\ \sin \varphi & \cos \varphi \end{pmatrix}$, такая, что $T\mathbf{r} = \|\mathbf{r}\| \begin{pmatrix} 1 \\ 0 \end{pmatrix}$, где $\|\mathbf{r}\| = \sqrt{x^2 + y^2}$ – евклидова длина вектора \mathbf{r} . При этом трудоемкость построения матрицы T составляет 4 мультипликативные операции, одну аддитивную и одну операцию извлечения корня.

Доказательство. Достаточно положить

$$\cos \varphi = \frac{x}{\sqrt{x^2 + y^2}}, \quad \sin \varphi = -\frac{y}{\sqrt{x^2 + y^2}}.$$

Лемма 3. Для всякого вектора $x \in \mathbf{R}^n$, $x \neq 0$ существуют $n - 1$ матриц $T_{12} = T_{12}(\varphi_{12})$, $T_{13} = T_{13}(\varphi_{13})$, ..., $T_{1n} = T_{1n}(\varphi_{1n})$, таких, что $T_{1n} \dots T_{13} T_{12} x = \|x\| e_1$, где $\|x\| = \|x\|_2 = \sqrt{\sum_{k=1}^n x_k^2}$ – евклидова длина вектора x , $e_1 = (1, 0, \dots, 0)^t$ – первый координатный орт.

Доказательство. Если вектор $\mathbf{r} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \neq 0$, то по лемме 2 существует матрица элементарного вращения

$$T = T(\varphi_{12}) = \begin{pmatrix} \cos \varphi_{12} & -\sin \varphi_{12} \\ \sin \varphi_{12} & \cos \varphi_{12} \end{pmatrix},$$

такая, что $T\mathbf{r} = \|\mathbf{r}\| \begin{pmatrix} 1 \\ 0 \end{pmatrix}$. Тогда матрица

$$T_{12} = T_{12}(\varphi_{12}) = \begin{pmatrix} \cos \varphi_{12} & -\sin \varphi_{12} & & & \\ \sin \varphi_{12} & \cos \varphi_{12} & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{pmatrix}$$

переводит вектор x в вектор $x^{(2)} = T_{12}x = (\sqrt{x_1^2 + x_2^2}, 0, x_3, \dots, x_n)^t$. Если вектор $\mathbf{r} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = 0$, то преобразование не осуществляется ($T_{12} = I$ – единичной матрице).

После $k - 1$ шагов этого процесса ($k = 1, \dots, n - 1$) вектор x преобразован к виду $x^{(k)} = T_{1k} \dots T_{12}x = (\sqrt{\sum_{i=1}^k x_i^2}, 0, \dots, 0, x_{k+1}, \dots, x_n)^t$. Если вектор

$$\mathbf{r} = \begin{pmatrix} \sqrt{\sum_{i=1}^k x_i^2} \\ x_{k+1} \end{pmatrix} \in \mathbf{R}^2, \quad \mathbf{r} \neq 0,$$

то по лемме 2 существует матрица элементарного вращения

$$T = T(\varphi_{1,k+1}) = \begin{pmatrix} \cos \varphi_{1,k+1} & -\sin \varphi_{1,k+1} \\ \sin \varphi_{1,k+1} & \cos \varphi_{1,k+1} \end{pmatrix},$$

такая, что $\text{Tr} = \|\mathbf{r}\| \begin{pmatrix} 1 \\ 0 \end{pmatrix}$. Тогда матрица

$$T_{1,k+1} = T_{1,k+1}(\varphi_{1,k+1}) = \begin{pmatrix} \cos \varphi_{1,k+1} & -\sin \varphi_{1,k+1} & & & & \\ & 1 & & & & \\ & & \ddots & & & \\ & & & 1 & & \\ & \sin \varphi_{1,k+1} & & & \cos \varphi_{1,k+1} & \\ & & & & & 1 \\ & & & & & & \ddots \\ & & & & & & & 1 \end{pmatrix}_{k+1 \times k+1}$$

переводит вектор $x^{(k)}$ в вектор

$$x^{(k+1)} = T_{1,k+1}x^{(k)} = T_{1,k+1}T_{1k} \dots T_{12}x = (\sqrt{\sum_{i=1}^{k+1} x_i^2}, 0, \dots, 0, x_{k+2}, \dots, x_n)^t.$$

Если вектор $\mathbf{r} = 0$, то преобразование не осуществляется ($T_{1,k+1} = I$ – единичной матрице).

После $n-1$ шагов этого процесса вектор x будет преобразован к виду $x^{(n)} = T_{1n} \dots T_{12}x = (\sqrt{\sum_{i=1}^n x_i^2}, 0, \dots, 0)^t = \|x\| e_1$.

Лемма 4. *Произведение матрицы элементарного вращения на вектор может быть вычислено за 4 умножения и 2 сложения.*

Доказательство. Произведение $y = T_{ij}(\varphi_{ij})x$ матрицы элементарного вращения T_{ij} на вектор x имеет следующие компоненты:

$$y_k = x_k, \quad k = 1, \dots, n, \quad k \neq i, j, \quad y_i = x_i \cos \varphi_{ij} - x_j \sin \varphi_{ij}, \quad y_j = x_i \sin \varphi_{ij} + x_j \cos \varphi_{ij}.$$

При осуществлении вычислений по этим формулам надо выполнить 4 умножения и 2 сложения.

Замечание 1. Для вычисления произведения матрицы из \mathbf{M}_n общего вида на вектор требуется выполнить n^2 умножений и $n(n-1)$ сложений.

Лемма 5. *Произведение матрицы элементарного вращения $T_{ij} \in \mathbf{M}_n$ на матрицу размера $n \times m$ может быть вычислено за $4tm$ умножений и $2tm$ сложений.*

Доказательство. Пусть $n \times m$ матрица $Y = T_{ij}X$ есть произведение матрицы элементарного вращения $T_{ij} \in \mathbf{M}_n$ на $n \times m$ матрицу X . Запишем матрицы $X = (x_{ij})$ и $Y = (y_{ij})$ через их столбцы: $X = [x^{(1)}, \dots, x^{(m)}]$, $Y = [y^{(1)}, \dots, y^{(m)}]$,

где $x^{(k)} = (x_{1k}, \dots, x_{nk})^t$, $y^{(k)} = (y_{1k}, \dots, y_{nk})^t$, $k = 1, \dots, m$. Согласно определению произведения матриц $Y = T_{ij}X = [T_{ij}x^{(1)}, \dots, T_{ij}x^{(m)}]$, т.е. $y^{(k)} = T_{ij}x^{(k)}$, $k = 1, \dots, m$. Таким образом, для вычисления матрицы $Y = T_{ij}X$ надо вычислить m произведений $T_{ij}x^{(k)}$ матрицы T_{ij} на вектора $x^{(k)}$, $k = 1, \dots, m$. Доказываемое утверждение теперь вытекает из леммы 4.

Замечание 2. Для вычисления произведения двух матриц из \mathbf{M}_n общего вида требуется выполнить n^3 умножений и $n^2(n - 1)$ сложений.

§ 12.2. Алгоритм метода вращений

Пусть требуется решить линейную систему $Ax = b$, $A \in \mathbf{M}_n(\mathbf{R}^n)$ вида (4.1).

Обозначим $a_1 = (a_{11}, \dots, a_{n1})^t$ – первый столбец матрицы A . Согласно лемме 3 существуют $n - 1$ матриц $T_{12} = T_{12}(\varphi_{12})$, $T_{13} = T_{13}(\varphi_{13})$, \dots , $T_{1n} = T_{1n}(\varphi_{1n})$, таких, что $T_{1n} \dots T_{13}T_{12}a_1 = \|a_1\| e_1$ (причем значения углов φ_{1k} , $k = 2, \dots, n$ определяются леммами 2, 3). Умножим систему $Ax = b$ на $T_{1n} \dots T_{13}T_{12}$ слева, получим

$$A^{(1)}x = b^{(1)},$$

где

$$A^{(1)} = T_{1n} \dots T_{13}T_{12}A = \begin{pmatrix} \|a_1\| & c_{12} & \dots & c_{1n} \\ 0 & a_{22}^{(1)} & \dots & a_{2n}^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & a_{n2}^{(1)} & \dots & a_{nn}^{(1)} \end{pmatrix}, \quad b^{(1)} = T_{1n} \dots T_{13}T_{12}b.$$

Далее процесс применяется к подматрице $(a_{ij}^{(1)})_{i,j=2,\dots,n}$.

Пусть сделаны $k - 1$, $k = 1, \dots, n - 1$ шагов этого процесса, т.е. система преобразована к виду

$$A^{(k-1)}x = b^{(k-1)}, \tag{2}$$

где

$$A^{(k-1)} = \prod_{i=k-1}^1 \prod_{j=n}^{i+1} T_{ij}A, \quad b^{(k-1)} = \prod_{i=k-1}^1 \prod_{j=n}^{i+1} T_{ij}b, \tag{3}$$

$$A^{(k-1)} = \begin{pmatrix} \|a_1\| & c_{12} & c_{13} & \dots & c_{1,k-1} & c_{1k} & \dots & c_{1n} \\ \|a_1^{(1)}\| & c_{23} & \dots & c_{2,k-1} & c_{2k} & \dots & c_{2n} \\ \|a_1^{(2)}\| & \dots & c_{3,k-1} & c_{3k} & \dots & c_{3n} \\ \ddots & \vdots & \ddots & \ddots & \ddots & \vdots \\ \|a_1^{(k-2)}\| & c_{k-1,k} & \dots & c_{k-1,n} \\ a_{kk}^{(k-1)} & \dots & a_{kn}^{(k-1)} \\ \vdots & \ddots & \vdots \\ a_{nk}^{(k-1)} & \dots & a_{nn}^{(k-1)} \end{pmatrix}. \tag{4}$$

(здесь $\prod_{j=n}^{i+1}$ означает, что сомножители берутся в порядке $n, \dots, i+1$).

Обозначим

$$a_1^{(k-1)} = (a_{kk}^{(k-1)}, \dots, a_{nk}^{(k-1)})^t \quad (5)$$

— первый столбец подматрицы $(a_{ij}^{(k-1)})_{i,j=k,\dots,n}$. Согласно лемме 3 существуют $n-k$ матриц

$$T_{k,k+1} = T_{k,k+1}(\varphi_{k,k+1}), \quad T_{k,k+2} = T_{k,k+2}(\varphi_{k,k+2}), \quad \dots, \quad T_{k,n} = T_{kn}(\varphi_{kn}),$$

таких, что

$$T_{kn} \dots T_{k,k+2} T_{k,k+1} a_1^{(k-1)} = \|a_1^{(k-1)}\| e_1^{(n-k+1)}, \quad (6)$$

(значения углов φ_{kj} , $j = k+1, \dots, n$ определяются леммами 2, 3), здесь $e_1^{(m)} = (1, 0, \dots, 0)^t \in \mathbf{R}^m$. Умножим систему (2) на $T_{kn} \dots T_{k,k+2} T_{k,k+1}$ слева, получим

$$A^{(k)}x = b^{(k)},$$

где

$$A^{(k)} = \prod_{j=n}^{k+1} T_{kj} A^{(k-1)} = \prod_{i=k}^1 \prod_{j=n}^{i+1} T_{ij} A, \quad b^{(k)} = \prod_{j=n}^{k+1} T_{kj} b^{(k-1)} = \prod_{i=k}^1 \prod_{j=n}^{i+1} T_{ij} b, \quad (7)$$

$$A^{(k)} = \begin{pmatrix} \|a_1\| & c_{12} & c_{13} & \dots & c_{1,k-1} & c_{1k} & c_{1,k+1} & \dots & c_{1n} \\ & \|a_1^{(1)}\| & c_{23} & \dots & c_{2,k-1} & c_{2k} & c_{2,k+1} & \dots & c_{2n} \\ & & \|a_1^{(2)}\| & \dots & c_{3,k-1} & c_{3k} & c_{3,k+1} & \dots & c_{3n} \\ & & & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ & & & & \|a_1^{(k-2)}\| & c_{k-1,k} & c_{k-1,k+1} & \dots & c_{k-1,n} \\ & & & & & \|a_1^{(k-1)}\| & c_{k,k+1} & \dots & c_{k,n} \\ & & & & & & a_{k+1,k+1}^{(k)} & \dots & a_{k+1,n}^{(k)} \\ & & & & & & \vdots & \ddots & \vdots \\ & & & & & & a_{n,k+1}^{(k)} & \dots & a_{nn}^{(k)} \end{pmatrix}. \quad (8)$$

Отметим, что в (7) каждая из $n-k$ матриц элементарных вращений T_{kj} такова, что $j > k$ и потому в (7) она умножается только на подматрицу $(a_{ij}^{(k-1)})_{i,j=k,\dots,n}$ матрицы $A^{(k-1)}$ размера $n-k+1$ (остальная часть $A^{(k-1)}$ в преобразовании (7) не участвует).

После $n-1$ шагов этого процесса (т.е. перехода от матриц и правых частей (3), (4) к (7), (8)) система примет вид

$$Rx = y, \quad (9)$$

где

$$R = A^{(n-1)} = \prod_{i=n-1}^1 \prod_{j=n}^{i+1} T_{ij} A, \quad y = b^{(n-1)} = \prod_{i=n-1}^{n-1} \prod_{j=n}^{i+1} T_{ij} b, \quad (10)$$

$$R = \begin{pmatrix} \|a_1\| & c_{12} & c_{13} & \dots & c_{1,n-2} & c_{1,n-1} & c_{1n} \\ & \|a_1^{(1)}\| & c_{23} & \dots & c_{2,n-2} & c_{2,n-1} & c_{2n} \\ & & \|a_1^{(2)}\| & \dots & c_{3,n-2} & c_{3,n-1} & c_{3n} \\ & & & \ddots & \vdots & \vdots & \vdots \\ & & & & \|a_1^{(n-3)}\| & c_{n-2,n-1} & c_{n-2,n} \\ & & & & & \|a_1^{(n-2)}\| & c_{n-1,n} \\ & & & & & & a_{nn}^{(n-1)} \end{pmatrix} \quad (11)$$

(напомним, определения векторов $a_1^{(k-1)}$, $k = 1, \dots, n-1$ даются в (5), где считаем, что $a_1^{(0)} = a_1$).

Система (9) с верхней треугольной матрицей R решается обратным ходом метода Гаусса.

§ 12.3. Оценка количества арифметических операций в методе вращений

Оценим трудоемкость k -го шага алгоритма, а затем просуммируем полученные оценки по всем $k = 1, \dots, n-1$.

1. На вычисление $n-k$ матриц $T_{k,k+1}, \dots, T_{kn}$, участвующих в (6), согласно лемме 2 требуется $4(n-k)$ мультипликативных, $(n-k)$ аддитивных и $n-k$ операций извлечения корня.

2. На вычисление компонент k, \dots, n k -го столбца матрицы $A^{(k)}$, равных компонентам вектора $\|a_1^{(k-1)}\| e_1^{(n-k+1)}$ требуется (для вычисления длины вектора (5)) $n-k+1$ операций умножения, $n-k$ операций сложения и одна операция извлечения корня. Столбец k вычисляется именно этим способом (а не по общим формулам (7)) для сокращения количества арифметических операций и уменьшения вычислительной погрешности.

3. Поскольку в формуле (7) каждая из $n-k$ матриц элементарных вращений умножается на подматрицу $(a_{ij}^{(k-1)})_{i=k, \dots, n, j=k+1, \dots, n}$ матрицы $A^{(k-1)}$ размера $(n-k+1) \times (n-k)$ (k -й столбец матрицы $A^{(k)}$ уже вычислен в пункте 2), то согласно лемме 5 на это требуется $(n-k)4(n-k) = 4(n-k)^2$ умножений и $(n-k)2(n-k) = 2(n-k)^2$ сложений.

4. На вычисление новой правой части по формуле (7) согласно лемме 4 требуется $4(n-k)$ умножений и $(n-k)$ сложений.

Итак, на k -ом шаге алгоритма требуется выполнить $4(n-k) + (n-k+1) + 4(n-k)^2 + 4(n-k) = 4(n-k)^2 + 9(n-k) + 1$ мультипликативных операций, $(n-k) + (n-k) + 2(n-k)^2 + (n-k) = 2(n-k)^2 + 3(n-k)$ аддитивных операций и $n-k+1$ операций извлечения корня.

Следовательно, всего для проведения алгоритма требуется выполнить

$$\sum_{k=1}^{n-1} (4(n-k)^2 + 9(n-k) + 1) = 4n(n-1)(2n-1)/6 + 9n(n-1)/2 + n - 1$$

$$= \frac{4}{3}n^3 + O(n^2) \quad (n \rightarrow \infty)$$

мультипликативных операций, $\sum_{k=1}^{n-1} (2(n-k)^2 + 3(n-k)) = \frac{2}{3}n^3 + O(n^2)$ ($n \rightarrow \infty$) аддитивных операций и $\sum_{k=1}^{n-1} (n - k + 1) = O(n^2)$ ($n \rightarrow \infty$) операций извлечения корня (которые по трудоемкости по порядку можно сравнить с операциями деления).

На решение системы (9) с верхней треугольной матрицей R обратным ходом метода Гаусса требуется $O(n^2)$ ($n \rightarrow \infty$) арифметических операций.

Таким образом, на решение линейной системы методом вращений требуется $\frac{4}{3}n^3 + O(n^2)$ ($n \rightarrow \infty$) мультипликативных операций (что в 4 раза больше, чем в методе Гаусса), и $\frac{2}{3}n^3 + O(n^2)$ ($n \rightarrow \infty$) аддитивных операций (что в 2 раза больше, чем в методе Гаусса).

Теорема 1 (О QR-разложении). *Всякая невырожденная вещественная матрица A может быть представлена в виде $A = QR$, где матрица Q – ортогональная, а матрица R – верхняя треугольная с положительными элементами на главной диагонали. Это разложение единствено.*

Доказательство. Проведем для матрицы A алгоритм метода вращений, осуществимый для всякой невырожденной матрицы. Обозначим в (10) $\hat{Q} = \prod_{i=n-1}^1 \prod_{j=n}^{i+1} T_{ij}$. Как произведение ортогональных матриц, матрица \hat{Q} ортогональна.

Тогда (10) имеет вид $R = \hat{Q}A$, откуда $A = (\hat{Q})^{-1}R = QR$, где $Q = (\hat{Q})^t = (\hat{Q})^{-1}$. Если $a_{nn}^{(n-1)} > 0$, то матрица R , имеющая вид (11), удовлетворяет условиям теоремы. Если $a_{nn}^{(n-1)} < 0$, то положим $D = \text{diag}(1, \dots, 1, \text{sign}(a_{nn}^{(n-1)}))$. Матрица D ортогональна и $D^2 = I$. Поэтому $A = (QD)(DR)$, где $Q := QD$ и $R := DR$ удовлетворяют условиям теоремы.

Предположим, что возможно два различных разложения $A = QR$ и $A = Q'R'$, удовлетворяющих условиям теоремы. Тогда $QR = Q'R'$ и $(Q')^{-1}Q = R^{-1}R'$. В левой части последнего равенства стоит ортогональная матрица, а в правой – верхняя треугольная. Пересечение группы ортогональных матриц и группы верхних треугольных матриц состоит из матриц вида $D = \text{diag}(d_1, \dots, d_n)$, где $d_i \in \{-1, 1\}$, $i = 1, \dots, n$ (проверить самостоятельно). Поскольку диагональные элементы матрицы $R^{-1}R'$ равны произведениям диагональных элементов матриц R и R' , то они положительны. Следовательно $R^{-1}R' = I$, т.е. $R = R'$. Также $(Q')^{-1}Q = I$, т.е. $Q = Q'$. Полученное противоречие доказывает теорему.

Замечание 3. QR-разложение матрицы A может быть использовано, например, для тех же целей, что и LU-разложение. Именно, пусть стоит задача решить серию систем вида $Ax_j = b_j$, $j = 1, \dots, m$ с одной и той же матрицей A и разными правыми частями b_j . Построим QR-разложение матрицы A (которое, в отличие от LU-разложения, существует для всякой невырожденной матрицы). Для ортогональной матрицы Q легко находится обратная $Q^{(-1)} = Q^t$. Поэтому

x_j находятся как решения системы $R x_j = Q^t b_j$ с верхней треугольной матрицей R , например, обратным ходом метода Гаусса.

Замечание 4. QR -разложение матрицы A используется в QR -алгоритме нахождения собственных значений матрицы A .

§ 12.4. Построение QR -разложения методом вращений

Построение QR -разложения. Хранение матриц Q и R в памяти. Пусть стоит задача построить QR -разложение для матрицы A . Будем действовать как в теореме 1. Проведем для матрицы A метод вращений и получим в результате матрицу R из (11). При этом матрица Q равна (см. доказательство теоремы 1)

$$Q = \left(\prod_{i=n-1}^1 \prod_{j=n}^{i+1} T_{ij} \right)^t = \prod_{i=1}^{n-1} \prod_{j=i+1}^n T_{ij}^t. \quad (12)$$

Возможны два способа хранения матриц Q и R в памяти.

1. Матрица R хранится на месте верхнего треугольника матрицы A и получается из нее последовательным применением элементарных вращений (см. выше алгоритм метода вращений). Для хранения матрицы Q выделяется отдельная матрица Q , которая равна единичной перед первым шагом алгоритма. На шаге k , $k = 1, \dots, n-1$ эта матрица умножается справа на матрицу $\prod_{j=n}^{k+1} T_{kj}$:

$$Q := Q \prod_{k+1}^{j=n} T_{kj}$$

(см. (7), (12)). Произведение матрицы элементарного вращения на матрицу вычисляется по алгоритму из леммы 5 с затратой $4n$ умножений и $2n$ сложений. Следовательно, произведение $n(n-1)/2$ матриц вращения в (12) может быть вычислено за $2n^2(n-1) = 2n^3 + O(n^2)$ ($n \rightarrow \infty$) умножений и $n^2(n-1) = n^3 + O(n^2)$ ($n \rightarrow \infty$) сложений.

2. Как и в первом способе, матрица R хранится на месте верхнего треугольника матрицы A . Для хранения же матрицы Q отдельная память не выделяется. Заметим, что на шаге k , $k = 1, \dots, n-1$ мы использовали $n-k$ элементарных вращений $T_{k,k+1}, \dots, T_{kn}$ и каждая из этих матриц целиком определяется единственным параметром – значением угла φ_{kj} : $T_{kj} = T_{kj}(\varphi_{kj})$, $j = k+1, \dots, n$. При этом после преобразования (6), (7), т.е. перехода от матрицы (4) к матрице (8), в k -ом столбце матрицы $A^{(k)}$ образовались $n-k$ нулевых элементов $a_{jk}^{(k)} = 0$, $j = k+1, \dots, n$. Поэтому возможно вместо матрицы Q вида (12) хранить на месте нижнего треугольника матрицы A набор параметров, с помощью которых можно вычислять тригонометрические функции углов φ_{ij} , $j < i$, $i = 2, \dots, n$, $j = 1, \dots, n-1$, задающих матрицы T_{ij} . Конечно, проще всего было

бы хранить сами эти углы φ_{ij} , но это требует вычисления обратных тригонометрических функций, что довольно медленно и вносит большую вычислительную погрешность. На практике на месте a_{ij} , $j < i$, $i = 2 \dots, n$, $j = 1, \dots, n - 1$ хранят $\cos \varphi_{ij}$ или $\sin \varphi_{ij}$ — тот который имеет наименьший модуль. При этом на месте двух младших битов мантиссы этой величины хранятся признак того, что было запомнено: \sin или \cos , и знак не запомненной тригонометрической функции. Изменение двух младших битов мантиссы у $\cos \varphi_{ij}$ или $\sin \varphi_{ij}$ вносит погрешность, намного меньшую чем погрешность, с которой они вычислены. Запоминание значения $\cos \varphi_{ij}$ или $\sin \varphi_{ij}$ с наименьшим модулем уменьшает погрешность при вычислении по формулам $\sin \varphi_{ij} = \pm \sqrt{1 - \cos^2 \varphi_{ij}}$ или $\cos \varphi_{ij} = \pm \sqrt{1 - \sin^2 \varphi_{ij}}$. Большинство современных микропроцессоров (Intel 80x86, Motorola 68xxx, SPARC, PowerPC) поддерживают стандарт ANSI/IEEE 754-1985 при работе с данными с плавающей точкой. Для таких процессоров младшие биты мантиссы являются младшими битами числа с плавающей точкой.

При втором способе хранения матрицы Q не только экономится n^3 ячеек памяти, но и экономится $2n^3 + O(n^2)$ ($n \rightarrow \infty$) умножений и $n^3 + O(n^2)$ ($n \rightarrow \infty$) сложений на построение матрицы Q . Второму способу хранения благоприятствует также то обстоятельство, что редко требуется знать матрицу Q "саму по себе". Обычно требуется уметь вычислять ее произведения на вектор и матрицу. Для того, чтобы вычислить произведение матрицы Q вида (12) на некоторую матрицу B требуется вычислить $n(n-1)/2$ произведений матриц элементарных вращений T_{ij} на B :

$$QB = \prod_{i=1}^{n-1} \prod_{j=i+1}^n (T_{ij}B).$$

По лемме 5 на это потребуется $2n^2(n-1)$ умножений и $n^2(n-1)$ сложений. По сравнению с количеством операций, необходимых для вычисления произведения двух матриц Q и B произвольного вида, число умножений тут в 2 раза больше, а число сложений совпадает. Если таких произведений требуется вычислить не очень много, то второй способ предпочтительнее первого.

§ 12.5. Оценка количества арифметических операций в алгоритме построения QR -разложения методом вращений

Трудоемкость алгоритма построения QR -разложения складывается из количества арифметических операций, необходимых для проведения алгоритма метода вращений, и количества арифметических операций, необходимых для построения матрицы Q .

Если для Q используется второй способ хранения, то дополнительных действий для ее построения не требуется. Следовательно, в этом случае для построения QR -разложения надо выполнить $\frac{4}{3}n^3 + O(n^2)$ ($n \rightarrow \infty$) мультипликативных операций и $\frac{2}{3}n^3 + O(n^2)$ ($n \rightarrow \infty$) аддитивных операций.

Если для Q используется первый способ хранения, то как показано выше для ее построения дополнительно к $\frac{4}{3}n^3 + O(n^2)$ ($n \rightarrow \infty$) мультипликативным и $\frac{2}{3}n^3 + O(n^2)$ ($n \rightarrow \infty$) аддитивным операциям, необходимых для проведения алгоритма метода вращений, требуется $2n^3 + O(n^2)$ ($n \rightarrow \infty$) умножений и $n^3 + O(n^2)$ ($n \rightarrow \infty$) сложений, всего $\frac{10}{3}n^3 + O(n^2)$ ($n \rightarrow \infty$) мультипликативных операций и $\frac{5}{3}n^3 + O(n^2)$ ($n \rightarrow \infty$) аддитивных операций.

§ 13. МЕТОД ОТРАЖЕНИЙ

Всюду в данном параграфе под нормой вектора будет пониматься евклидова норма, а под нормой матрицы – спектральная норма.

Лемма 1. Спектральная норма всякой унитарной (ортогональной в вещественном случае) матрицы равна 1.

Доказательство. Поскольку унитарные матрицы сохраняют евклидову длину вектора, по определению спектральной нормы получаем для всякой унитарной матрицы U :

$$\|U\| = \sup_{x \neq 0} \frac{\|Ux\|}{\|x\|} = \sup_{x \neq 0} \frac{\|x\|}{\|x\|} = 1.$$

Лемма 2. Собственные значения всякой унитарной матрицы по модулю равны 1.

Доказательство. Пусть λ – произвольное собственное значение матрицы U . По лемме 1.4 $|\lambda| \leq \|U\| = 1$ – по предыдущей лемме. С другой стороны, λ^{-1} является собственным значением матрицы U^{-1} , которая тоже унитарна. Опять по лемме 1.4 и лемме 1 $|\lambda^{-1}| \leq \|U^{-1}\| = 1$, т.е. $|\lambda| \geq 1$. Следовательно, $|\lambda| = 1$.

Лемма 3. Собственные значения всякой самосопряженной (симметричной в вещественном случае) матрицы A (т.е. $A^* = A$) вещественны.

Доказательство. Пусть λ – произвольное собственное значение матрицы A , $x \neq 0$ – отвечающий ему собственный вектор, т.е. $Ax = \lambda x$. Умножим это равенство скалярно на x : $(Ax, x) = \lambda(x, x)$, откуда $\lambda = (Ax, x)/\|x\|^2$. В силу замечания 9.1 выражение (Ax, x) вещественно для самосопряженной матрицы A . Следовательно, λ вещественно.

§ 13.1. Матрица отражения и ее свойства

Определение. *Матрицей отражения* называется матрица вида $U = U(x) = I - 2xx^*$, где x – единичный вектор (т.е. $\|x\| = 1$). (Напомним, что $x^* = (\bar{x}_1, \dots, \bar{x}_n)$ – "матрица" размера $1 \times n$, $x = (x_1, \dots, x_n)^t$ "матрица" размера $n \times 1$ и потому xx^* – матрица размера $n \times n$.)

Установим основные свойства матрицы отражения.

Лемма 4. *Матрица отражения является самосопряженной матрицей.*

Доказательство. Вычислим сопряженную матрицу для матрицы отражения $U(x)$

$$(U(x))^* = (I - 2xx^*)^* = I - 2(x^*)^*x^* = I - 2xx^* = U(x),$$

что и означает самосопряженность матрицы $U(x)$.

Лемма 5. *Матрица отражения является унитарной матрицей.*

Доказательство. Вычислим для матрицы отражения $U(x)$

$$U(x)U^*(x) = U(x)^2 = (I - 2xx^*)(I - 2xx^*) = I - 4xx^* + 4xx^*xx^* = I - 4xx^* + 4x1x^* = I,$$

поскольку $x^*x = (x, x) = \|x\|^2 = 1$. Это равенство и означает унитарность матрицы $U(x)$.

Лемма 6. *Собственные значения матрицы отражения равны либо 1, либо -1 .*

Доказательство. Из лемм 2 и 4 вытекает, что собственные значения матрицы отражения по модулю равны 1. Из лемм 3 и 5 следует, что они вещественны. Значит, собственные значения есть либо 1 либо -1 .

Лемма 7. *Матрица отражения $U(x)$ имеет собственное значение -1 кратности 1, которому отвечает собственный вектор x , и собственное значение 1 кратности $n - 1$, которому отвечает собственное подпространство $\langle x \rangle^\perp = \{y : (y, x) = 0\}$.*

Доказательство. Имеем

$$U(x)x = (I - 2xx^*)x = x - 2xx^*x = x - 2x = -x,$$

поскольку $x^*x = (x, x) = \|x\|^2 = 1$. Следовательно, x – собственный вектор, отвечающий собственному значению -1 .

Далее, для всех $y \in \langle x \rangle^\perp$

$$U(x)y = (I - 2xx^*)y = y - 2xx^*y = y,$$

поскольку $x^*y = (y, x) = 0$. Следовательно, y – собственный вектор, отвечающий собственному значению 1. Такие векторы $y \in \langle x \rangle^\perp$ образуют $(n - 1)$ -мерное подпространство.

Лемма 8. Геометрический смысл преобразования, задаваемого матрицей отражения $U(x)$: отражение относительно гиперплоскости $\langle x \rangle^\perp$.

Доказательство. Всякий вектор $z \in \mathbf{C}^n$ может быть представлен в виде $z = \alpha x + y$, где $y \in \langle x \rangle^\perp$. Здесь компонента αx параллельна x , а компонента y ортогональна x , т.е. лежит в гиперплоскости $\langle x \rangle^\perp$. В силу леммы 7 $U(x)z = U(x)(\alpha x + y) = -\alpha x + y$, т.е. вектор z отразился относительно гиперплоскости $\langle x \rangle^\perp$.

Лемма 9. Пусть e – произвольный единичный вектор: $\|e\| = 1$. Тогда для всякого вектора $y \in \mathbf{C}^n$ существует вектор $x \in \mathbf{C}^n$, $\|x\| = 1$ такой, что $U(x)y = \|y\|e$.

Доказательство. Так как вектора y и $\|y\|e$ должны быть получены друг из друга отражением относительно гиперплоскости $\langle x \rangle^\perp$, то вектор $y - \|y\|e$ должен быть параллелен x , т.е. $x = \alpha(y - \|y\|e)$. Коэффициент α найдем из условия $\|x\| = 1$. Получаем

$$x = \pm \frac{y - \|y\|e}{\|y - \|y\|e\|}.$$

Лемма 10. Произведение матрицы отражения на вектор может быть вычислено за $2n + O(1)$ ($n \rightarrow \infty$) сложений и столько же умножений (точнее, за $2n + 1$ умножение и $2n - 1$ сложение).

Доказательство. Для матрицы отражения $U(x)$ и произвольного вектора y имеем

$$U(x)y = (I - 2xx^*)y = y - 2x(x^*y) = y - 2x(y, x).$$

На вычисление скалярного произведения (y, x) требуется n умножений и $n - 1$ сложение. На вычисление коэффициента $\alpha = 2(y, x)$ требуется еще одно умножение. На вычисление линейной комбинации $y - \alpha x$ требуется n умножений и столько же сложений. Складывая эти оценки, находим, что всего необходимо $n + 1 + n = 2n + 1$ умножение и $n - 1 + n = 2n - 1$ сложений.

Лемма 11. Произведение матрицы отражения $U(x) \in \mathbf{M}_n$ на матрицу размера $n \times m$ может быть вычислено за $2nm + O(m)$ ($n, m \rightarrow \infty$) сложений и столько же умножений (точнее, за $(2n + 1)m$ умножение и $(2n - 1)m$ сложение).

Доказательство. Пусть $n \times m$ матрица $B = U(x)A$ есть произведение матрицы отражения $U(x) \in \mathbf{M}_n$ на $n \times m$ матрицу A . Запишем матрицы $A = (a_{ij})$ и $B = (b_{ij})$ через их столбцы: $A = [a^{(1)}, \dots, a^{(m)}]$, $B = [b^{(1)}, \dots, b^{(m)}]$, где $a^{(k)} = (a_{1k}, \dots, a_{nk})^t$, $b^{(k)} = (b_{1k}, \dots, b_{nk})^t$, $k = 1, \dots, m$. Согласно определению произведения матриц $B = U(x)A = [U(x)a^{(1)}, \dots, U(x)a^{(m)}]$, т.е. $b^{(k)} = U(x)a^{(k)}$, $k = 1, \dots, m$. Таким образом, для вычисления матрицы $B = U(x)A$ надо вычислить m произведений $U(x)a^{(k)}$ матрицы $U(x)$ на вектора $a^{(k)}$, $k = 1, \dots, m$. Доказываемое утверждение теперь вытекает из леммы 10.

§ 13.2. Алгоритм метода отражений

Пусть требуется решить линейную систему $Ax = b$, $A \in \mathbf{M}_n$ вида (4.1).

Обозначим $a_1 = (a_{11}, \dots, a_{n1})^t$ – первый столбец матрицы A . Согласно лемме 9 существует вектор $x^{(1)} \in \mathbf{C}^n$, равный

$$x^{(1)} = \pm \frac{a_1 - \|a_1\|e_1}{\|a_1 - \|a_1\|e_1\|},$$

такой, что $U(x^{(1)})a_1 = \|a_1\|e_1$, где $e_1 = (1, 0, \dots, 0) \in \mathbf{C}^n$, $U_1 = U(x^{(1)})$ – матрица отражения. Умножим систему $Ax = b$ на $U(x^{(1)})$ слева, получим

$$A^{(1)}x = b^{(1)},$$

где

$$A^{(1)} = U(x^{(1)})A = \begin{pmatrix} \|a_1\| & c_{12} & \dots & c_{1n} \\ 0 & a_{22}^{(1)} & \dots & a_{2n}^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & a_{n2}^{(1)} & \dots & a_{nn}^{(1)} \end{pmatrix}, \quad b^{(1)} = U(x^{(1)})b.$$

Далее процесс применяется к подматрице $(a_{ij}^{(1)})_{i,j=2,\dots,n}$.

Пусть сделаны $k-1$, $k = 1, \dots, n$ шагов этого процесса, т.е. система преобразована к виду

$$A^{(k-1)}x = b^{(k-1)}, \quad (1)$$

где

$$A^{(k-1)} = \prod_{i=k-1}^1 U_i A, \quad b^{(k-1)} = \prod_{i=k-1}^1 U_i b, \quad (2)$$

$$A^{(k-1)} = \begin{pmatrix} \|a_1\| & c_{12} & c_{13} & \dots & c_{1,k-1} & c_{1k} & \dots & c_{1n} \\ \|a_1^{(1)}\| & c_{23} & \dots & c_{2,k-1} & c_{2k} & \dots & c_{2n} \\ \|a_1^{(2)}\| & \dots & c_{3,k-1} & c_{3k} & \dots & c_{3n} \\ \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \|a_1^{(k-2)}\| & c_{k-1,k} & \dots & c_{k-1,n} \\ a_{kk}^{(k-1)} & \dots & a_{kn}^{(k-1)} \\ \vdots & \ddots & \vdots \\ a_{nk}^{(k-1)} & \dots & a_{nn}^{(k-1)} \end{pmatrix}, \quad (3)$$

$$U_i = \begin{pmatrix} I_{i-1} & 0 \\ 0 & U(x^{(i)}) \end{pmatrix},$$

здесь $I_{i-1} \in \mathbf{M}_{i-1}$ – единичная матрица размера $(i-1) \times (i-1)$, $U(x^{(i)}) \in \mathbf{M}_{n-i+1}$ – матрица отражения размера $(n-i+1) \times (n-i+1)$, построенная по вектору

$$x^{(i)} = \pm \frac{a_1^{(i-1)} - \|a_1^{(i-1)}\|e_1^{(n-i+1)}}{\|a_1^{(i-1)} - \|a_1^{(i-1)}\|e_1^{(n-i+1)}\|} \in \mathbf{C}^{n-i+1},$$

где $e_1^{(m)} = (1, 0, \dots, 0) \in \mathbf{C}^m$.

Обозначим

$$a_1^{(k-1)} = (a_{kk}^{(k-1)}, \dots, a_{nk}^{(k-1)})^t \in \mathbf{C}^{n-k+1} \quad (4)$$

— первый столбец подматрицы $(a_{ij}^{(k-1)})_{i,j=k,\dots,n}$. Согласно лемме 9 существует матрица отражения

$$U(x^{(k)}) = I - 2x^{(k)}(x^{(k)})^*, \quad x^{(k)} = \pm \frac{a_1^{(k-1)} - \|a_1^{(k-1)}\|e_1^{(n-k+1)}}{\|a_1^{(k-1)} - \|a_1^{(k-1)}\|e_1^{(n-k+1)}\|} \in \mathbf{C}^{n-k+1}, \quad (5)$$

такая, что

$$U(x^{(k)})a_1^{(k-1)} = \|a_1^{(k-1)}\|e_1^{(n-k+1)}. \quad (6)$$

Положим

$$U_k = \begin{pmatrix} I_{k-1} & 0 \\ 0 & U(x_k) \end{pmatrix}. \quad (7)$$

Покажем, что матрица U_k является унитарной, т.е. $U_k^* = U_k^{-1}$. По правилам перемножения блочных матриц

$$U_k^* = \begin{pmatrix} I_{k-1}^* & 0 \\ 0 & U^*(x_k) \end{pmatrix} = \begin{pmatrix} I_{k-1} & 0 \\ 0 & U(x_k) \end{pmatrix} = U_k, \quad (8)$$

$$U_k U_k^* = U_k U_k = \begin{pmatrix} I_{k-1}^2 & 0 \\ 0 & U^2(x_k) \end{pmatrix} = \begin{pmatrix} I_{k-1} & 0 \\ 0 & I_{n-k+1} \end{pmatrix} = I_n = I, \quad (9)$$

что и означает $U_k^* = U_k^{-1}$, т.е. унитарность матрицы U_k .

Умножим систему (1) на U_k слева, получим

$$A^{(k)}x = b^{(k)},$$

где

$$A^{(k)} = U_k A^{(k-1)} = \prod_{i=k}^1 U_i A, \quad b^{(k)} = U_k b^{(k-1)} = \prod_{i=k}^1 U_i b, \quad (10)$$

$$A^{(k)} = \left(\begin{array}{ccccccccc} \|a_1\| & c_{12} & c_{13} & \dots & c_{1,k-1} & c_{1k} & c_{1,k+1} & \dots & c_{1n} \\ \|a_1^{(1)}\| & c_{23} & \dots & c_{2,k-1} & c_{2k} & c_{2,k+1} & \dots & c_{2n} \\ \|a_1^{(2)}\| & \dots & c_{3,k-1} & c_{3k} & c_{3,k+1} & \dots & c_{3n} \\ \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ & \|a_1^{(k-2)}\| & c_{k-1,k} & c_{k-1,k+1} & \dots & c_{k-1,n} \\ & \|a_1^{(k-1)}\| & c_{k,k+1} & \dots & c_{k,n} \\ & a_{k+1,k+1}^{(k)} & \dots & a_{k+1,n}^{(k)} \\ & \vdots & \ddots & \vdots \\ & a_{n,k+1}^{(k)} & \dots & a_{nn}^{(k)} \end{array} \right). \quad (11)$$

Отметим, что при умножении матрицы U_k вида (5) на матрицу $A^{(k-1)}$ вида (3) она умножается только на подматрицу $(a_{ij}^{(k-1)})_{i,j=k,\dots,n}$ матрицы $A^{(k-1)}$ размера $n - k + 1$ (остальная часть $A^{(k-1)}$ в преобразовании (10) не участвует).

Вычисления по формулам (5) осуществляются следующим образом: вначале вычисляются числа

$$s_k = \sum_{j=k+1}^n |a_{jk}^{(k-1)}|^2, \quad (12)$$

$$\|a_1^{(k-1)}\| = \sqrt{|a_{kk}^{(k-1)}|^2 + s_k}. \quad (13)$$

затем – вектор

$$x^{(k)} = (a_{kk}^{(k-1)} - \|a_1^{(k-1)}\|, a_{k+1,k}^{(k-1)}, \dots, a_{nk}^{(k-1)})^t \in \mathbf{C}^{n-k+1} \quad (14)$$

и его норма

$$\|x^{(k)}\| = \sqrt{|x_1^{(k)}|^2 + s_k}. \quad (15)$$

Теперь можно вычислить искомый вектор $x^{(k)}$:

$$x^{(k)} := x^{(k)} / \|x^{(k)}\|, \quad \text{т.е.} \quad x_j^{(k)} := x_j^{(k)} / \|x^{(k)}\|, \quad j = 1, \dots, n - k + 1. \quad (16)$$

После n шагов этого процесса (т.е. перехода от матриц и правых частей (2), (3) к (10), (11)) система примет вид

$$R x = y, \quad (17)$$

где

$$R = A^{(n)} = \prod_{i=n}^1 U_i A, \quad y = b^{(n)} = \prod_{i=n}^1 U_i b, \quad (18)$$

$$R = \begin{pmatrix} \|a_1\| & c_{12} & c_{13} & \dots & c_{1,n-2} & c_{1,n-1} & c_{1n} \\ \|a_1^{(1)}\| & c_{23} & \dots & c_{2,n-2} & c_{2,n-1} & c_{2n} & \\ \|a_1^{(2)}\| & \dots & c_{3,n-2} & c_{3,n-1} & c_{3n} & & \\ \ddots & \vdots & & \vdots & & \vdots & \\ \|a_1^{(n-3)}\| & c_{n-2,n-1} & c_{n-2,n} & & & & \\ \|a_1^{(n-2)}\| & & c_{n-1,n} & & & & \\ \|a_1^{(n-1)}\| & & & & & & \end{pmatrix} \quad (19)$$

(напомним, определения векторов $a_1^{(k-1)}$, $k = 1, \dots, n$ даются в (4), где считаем, что $a_1^{(0)} = a_1$).

Система (17) с верхней треугольной матрицей R решается обратным ходом метода Гаусса.

§ 13.3. Оценка количества арифметических операций в методе отражений

Оценим трудоемкость k -го шага алгоритма, а затем просуммируем полученные оценки по всем $k = 1, \dots, n$.

1. На вычисление матрицы $U(x_k)$ по формулам (5) требуется

а) $n - k$ умножений и $n - k - 1$ сложений для вычисления s_k в (12);

б) одно умножение, одно сложение и одна операция извлечения корня для вычисления $\|a_1^{(k-1)}\|$ в (13);

в) одно вычитание для построения вектора $x^{(k)}$ в (14);

г) одно умножение, одно сложение и одна операция извлечения корня для вычисления $\|x^{(k)}\|$ в (15);

д) $n - k + 1$ делений для построения вектора $x^{(k)}$ в (16).

Всего для построения матрицы $U(x_k)$ требуется $(n - k) + 1 + 1 + (n - k + 1) = 2(n - k) + 3$ мультипликативных, $(n - k - 1) + 1 + 1 + 1 = n - k + 2$ аддитивных операций и $1 + 1 = 2$ операции извлечения корня.

2: Компоненты k, \dots, n k -го столбца матрицы $A^{(k)}$, равные компонентам вектора $\|a_1^{(k-1)}\| e_1^{(n-k+1)}$, уже вычислены в (13). Столбец k вычисляется не по общим формулам (10) для сокращения количества арифметических операций и уменьшения вычислительной погрешности.

3. Поскольку в формуле (10) матрица U_k вида (5) умножается на матрицу $A^{(k-1)}$ вида (3), то при вычислениях по (10) надо умножить матрицу отражения $U(x^{(k)}) \in M_{n-k+1}$ на подматрицу $(a_{ij}^{(k-1)})_{i=k, \dots, n, j=k+1, \dots, n}$ матрицы $A^{(k-1)}$ размера $(n-k+1) \times (n-k)$ (k -й столбец матрицы $A^{(k)}$ уже вычислен в пункте 2). Согласно лемме 11 на это требуется $2(n-k)(n-k+1) + O(n-k) = 2(n-k)^2 + O(n-k)$ ($n \rightarrow \infty$) умножений и столько же сложений.

4. На вычисление новой правой части по формуле (10) согласно лемме 10 требуется $2(n - k + 1) + O(1)$ ($n \rightarrow \infty$) умножений и столько же сложений.

Итак, на k -ом шаге алгоритма требуется выполнить $2(n - k) + 3 + 2(n - k)^2 + O(n - k) + 2(n - k + 1) + O(1) = 2(n - k)^2 + O(n - k)$ ($n \rightarrow \infty$) мультипликативных операций, $n - k + 2 + 2(n - k)^2 + O(n - k) + 2(n - k + 1) + O(1) = 2(n - k)^2 + O(n - k)$ ($n \rightarrow \infty$) аддитивных операций и 2 операции извлечения корня.

Следовательно, всего для проведения алгоритма требуется выполнить

$$\sum_{k=1}^n (2(n - k)^2 + O(n - k)) = 2n(n - 1)(2n - 1)/6 + O(n^2) = \frac{2}{3}n^3 + O(n^2) \quad (n \rightarrow \infty)$$

мультипликативных и столько же аддитивных операций, и $\sum_{k=1}^n (2) = 2n$ операций извлечения корня (которые по трудоемкости по порядку можно сравнить с операциями деления).

На решение системы (17) с верхней треугольной матрицей R обратным ходом метода Гаусса требуется $O(n^2)$ ($n \rightarrow \infty$) арифметических операций.

Таким образом, на решение линейной системы методом отражений требуется $\frac{2}{3}n^3 + O(n^2)$ ($n \rightarrow \infty$) мультипликативных и столько же аддитивных операций (что в 2 раза больше, чем в методе Гаусса).

Теорема 1 (О QR-разложении). Всякая невырожденная матрица $A \in M_n$ может быть представлена в виде $A = QR$, где матрица Q – унитарная, а матрица R – верхняя треугольная с вещественными положительными элементами на главной диагонали. Это разложение единствено.

Доказательство проходит аналогично доказательству теоремы 12.1. Приведем для матрицы A алгоритм метода отражений, осуществимый для всякой невырожденной матрицы. Обозначим в (18) $\hat{Q} = \prod_{i=1}^n U_i$. Как произведение унитарных матриц, матрица \hat{Q} унитарна. Тогда (5) имеет вид $R = \hat{Q}A$, откуда $A = (\hat{Q})^{-1}R = QR$, где $Q = (\hat{Q})^t = (\hat{Q})^{-1}$. Здесь матрица Q унитарна, а матрица R имеет вид (19) и потому удовлетворяет условиям теоремы.

Предположим, что возможно два различных разложения $A = QR$ и $A = Q'R'$, удовлетворяющих условиям теоремы. Тогда $QR = Q'R'$ и $(Q')^{-1}Q = R^{-1}R'$. В левой части последнего равенства стоит унитарная матрица, а в правой – верхняя треугольная. Пересечение группы унитарных матриц и группы верхних треугольных матриц состоит из матриц вида $D = \text{diag}(d_1, \dots, d_n)$, где $d_j = e^{i\varphi_j}$, $j = 1, \dots, n$ (проверить самостоятельно). Поскольку диагональные элементы матрицы $R^{-1}R'$ равны произведениям диагональных элементов матриц R^{-1} и R' , то они вещественны и положительны. Следовательно $R^{-1}R' = I$, т.е. $R = R'$. Полученное противоречие доказывает теорему.

Замечание 1. Справедливы замечания 12.3 и 12.4 о применении QR-разложения.

§ 13.4. Построение QR-разложения методом отражений

Построение QR-разложения. Хранение матриц Q и R в памяти. Пусть стоит задача построить QR-разложение для матрицы A . Будем действовать как в теореме 1. Проведем для матрицы A метод отражений и получим в результате матрицу R из (19). При этом матрица Q равна (см. доказательство теоремы 1)

$$Q = \left(\prod_{i=1}^n U_i \right)^t = \prod_{i=1}^n U_i^t = \prod_{i=1}^n \bar{U}_i. \quad (20)$$

Возможны два способа хранения матриц Q и R в памяти.

1. Матрица R хранится на месте верхнего треугольника матрицы A и получается из нее последовательным применением матриц отражения (см. выше алгоритм метода отражений). Для хранения матрицы Q выделяется отдельная матрица Q , которая равна единичной перед первым шагом алгоритма. На шаге k , $k = 1, \dots, n$ эта матрица умножается справа на матрицу U_k :

$$Q := Q U_k$$

(см. (10), (20)). Матрица U_k вида (5) умножается по алгоритму из леммы 11 на матрицу Q произвольного вида за $2n(n-k+1)+O(n) = 2n(n-k)+O(n)$ ($n \rightarrow \infty$) умножений и такого же количества сложений (поскольку для вычисления произведения QU_k матрицы Q на матрицу U_k вида (5) надо вычислить произведение подматрицы $(q_{ij})_{i=1,\dots,n, j=k,\dots,n}$ размера $n \times (n - k + 1)$ на матрицу отражения $U(x^{(k)}) \in \mathbf{M}_{n-k+1}$ размера $(n - k + 1) \times (n - k + 1)$).

Следовательно, произведение n матриц отражения в (20) может быть вычислено за $\sum_{k=1}^n (2n(n - k) + O(n)) = 2nn(n - 1)/2 + O(n^2) = n^3 + O(n^2)$ ($n \rightarrow \infty$) умножений и столько же сложений.

2. Как и в первом способе, матрица R хранится на месте верхнего треугольника матрицы A . Для хранения же матрицы Q отдельная память не выделяется. Заметим, что на шаге k , $k = 1, \dots, n$ мы использовали матрицу U_k , получающуюся в (7) из матрицы отражения $U(x^{(k)})$, которая в свою очередь целиком определяется вектором $x^{(k)} \in \mathbf{C}^{n-k+1}$ из (5). При этом после преобразования (10), т.е. перехода от матрицы (3) к матрице (11), в k -ом столбце матрицы $A^{(k)}$ образовались $n - k$ нулевых элементов $a_{jk}^{(k)} = 0$, $j = k + 1, \dots, n$. Поэтому возможно вместо матрицы Q вида (20) хранить на месте нижнего треугольника матрицы A набор векторов $x^{(k)}$, $k = 1, \dots, n$, задающий матрицы отражения $U(x^{(k)})$. Формула (14) подсказывает удобный способ организации такого хранения: на шаге k $x_1^{(k)} \equiv a_{kk}^{(k-1)}, \dots, x_{n-k+1}^{(k)} \equiv a_{nk}^{(k-1)}$, а элемент $a_{kk}^{(k)} = \|a_1^{(k-1)}\|$ хранится в виде $(k - 1)$ -ой компоненты дополнительного вектора D . В итоге после n шагов процесса на месте исходной $n \times n$ матрицы A и дополнительного вектора D длины n будет находиться следующая информация: верхний треугольник матрицы R : $r_{ij} = a_{ij}$, $i < j$, $i = 1, \dots, n$, $j = 2, \dots, n$, диагональ матрицы R : $r_{ii} = d_i$, $i = 1, \dots, n$, набор векторов $x^{(k)}$, $k = 1, \dots, n$, $x_1^{(k)} \equiv a_{kk}, \dots, x_{n-k+1}^{(k)} \equiv a_{nk}$.

При втором способе хранения матрицы Q не только экономится n^3 ячеек памяти, но и экономится $n^3 + O(n^2)$ ($n \rightarrow \infty$) умножений и такое же количество сложений на построение матрицы Q . Второму способу хранения благоприятствует также то обстоятельство, что редко требуется знать матрицу Q "саму по себе". Обычно требуется уметь вычислять ее произведения на вектор и матрицу. Для того, чтобы вычислить произведение матрицы Q вида (20) на некоторую матрицу B требуется вычислить $QB = \prod_{i=1}^n (U_i B)$. На это нужно $n^3 + O(n^2)$ ($n \rightarrow \infty$) умножений и столько же сложений (см. подсчет количества операций при рассмотрении первого способа хранения, в котором фактически вычислялось произведение матрицы вида (20) и единичной матрицы). Это количество совпадает с количеством арифметических операций, необходимых для вычисления произведения двух матриц Q и B произвольного вида. В силу этого почти всегда используется второй способ хранения матрицы Q .

§ 13.5. Оценка количества арифметических операций в алгоритме построения QR -разложения методом отражений

Трудоемкость алгоритма построения QR -разложения складывается из количества арифметических операций, необходимых для проведения алгоритма метода отражений, и количества арифметических операций, необходимых для построения матрицы Q .

Если для Q используется второй способ хранения, то дополнительных действий для ее построения не требуется. Следовательно, в этом случае для построения QR -разложения надо выполнить $\frac{2}{3}n^3 + O(n^2)$ ($n \rightarrow \infty$) мультипликативных операций и такое же количество аддитивных операций.

Если для Q используется первый способ хранения, то как показано выше для ее построения дополнительно к $\frac{2}{3}n^3 + O(n^2)$ ($n \rightarrow \infty$) мультипликативным и $\frac{2}{3}n^3 + O(n^2)$ ($n \rightarrow \infty$) аддитивным операциям, необходимых для проведения алгоритма метода отражений, требуется $n^3 + O(n^2)$ ($n \rightarrow \infty$) умножений и $n^3 + O(n^2)$ ($n \rightarrow \infty$) сложений, всего $\frac{5}{3}n^3 + O(n^2)$ ($n \rightarrow \infty$) мультипликативных и столько же аддитивных операций.

Приведение матрицы к почти треугольному виду унитарным подобием

Определение. Матрица B называется *подобной* матрице A , если существует невырожденная матрица C такая, что $A = C B C^{-1}$.

В курсе алгебры доказывается, что подобные матрицы имеют один и тот же набор собственных значений.

Определение. Матрица B называется *унитарно подобной* матрице A , если матрица C в определении выше унитарная.

В силу свойства 6 числа обусловленности (см. §3) у унитарно подобных матриц числа обусловленности совпадают. Поэтому именно преобразование унитарного подобия будет вносить наименьшую вычислительную погрешность.

Отметим еще одно свойство унитарного подобия: если матрица A самосопряженная, то унитарно подобная ей матрица B также самосопряженная. Действительно, $B^* = (CAC^{-1})^* = (C^{-1})^* A^* C^* = CAC^{-1} = B$.

Рассмотренные выше алгоритмы решения линейных систем работали единственнообразным способом: они приводили исходную матрицу к более простому виду (треугольному) с помощью преобразований, сохраняющих решение системы; затем решение системы с более простой матрицей находилось в явном виде. Пусть стоит задача найти собственные значения матрицы. Попробуем действовать по той же схеме: приведем исходную матрицу к более простому виду с помощью преобразований подобия, сохраняющих собственные значения; затем для этой более простой матрицы тем или иным способом найдем ее собственные значения, которые совпадают с собственными значениями исходной матрицы. Для того, чтобы вносить меньшую вычислительную погрешность, будем использовать преобразования унитарного подобия.

Простейшее рассмотрение алгоритмов метода вращений и отражений показывает, что вид этой более простой матрицы не может быть треугольным. Действительно, пусть, например, в методе вращений при умножении на матрицу T_{12} слева элемент $(2, 1)$ исходной матрицы становится равным нулю. Тогда при умножении на матрицу $T_{12}^{-1} = T_{12}^t$ справа этот элемент может измениться и перестать быть равным нулю.

Определение. Матрица $A = (a_{ij})$ называется *почти треугольной*, если $a_{ij} = 0$ при $i > j + 1$, $j = 1, \dots, n - 2$, $i = 3, \dots, n$.

Оказывается, всякую матрицу можно привести к почти треугольному виду с помощью унитарного подобия.

§ 14. ПРИВЕДЕНИЕ МАТРИЦЫ К ПОЧТИ ТРЕУГОЛЬНОМУ ВИДУ УНИТАРНЫМ ПОДОБИЕМ МЕТОДОМ ВРАЩЕНИЙ

Пусть требуется привести вещественную матрицу A к почти треугольному виду.

Всюду ниже мы будем часто пользоваться тем фактом, что при умножении матрицы A на матрицу элементарного вращения T_{ij} слева изменяются только строки i и j матрицы A , а при умножении на T_{ij} справа изменяются только столбцы i и j матрицы A .

§ 14.1. Случай произвольной матрицы

Обозначим $a_1 = (a_{21}, \dots, a_{n1})^t$. Согласно лемме 12.3 существуют $n - 2$ матриц $T_{2j} = T_{2j}(\varphi_{2j})$, $j = 3, \dots, n$, таких, что $T_{2n} \dots T_{24} T_{23} a_1 = \|a_1\| e_1^{(n-1)}$ (причем значения углов φ_{2j} , $j = 3, \dots, n$ определяются леммами 12.2, 12.3). Умножим матрицу A на $T_{2n} \dots T_{24} T_{23}$ слева, получим

$$\widehat{A}^{(1)} = T_{2n} \dots T_{24} T_{23} A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ \|a_1\| & \hat{a}_{22}^{(1)} & \dots & \hat{a}_{2n}^{(1)} \\ 0 & \hat{a}_{32}^{(1)} & \dots & \hat{a}_{3n}^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \hat{a}_{n2}^{(1)} & \dots & \hat{a}_{nn}^{(1)} \end{pmatrix}. \quad (1)$$

Умножим матрицу $\widehat{A}^{(1)}$ на $(T_{2n} \dots T_{24} T_{23})^* = T_{23}^t T_{24}^t \dots T_{2n}^t$ справа, получим (с учетом того, что при умножении справа на T_{2j} , $j = 3, \dots, n$ первый столбец

матрицы $\widehat{A}^{(1)}$ не изменяется)

$$A^{(1)} = \widehat{A}^{(1)} T_{23}^t T_{24}^t \dots T_{2n}^t = \begin{pmatrix} a_{11} & a_{12}^{(1)} & \dots & a_{1n}^{(1)} \\ \|a_1\| & a_{22}^{(1)} & \dots & a_{2n}^{(1)} \\ 0 & a_{32}^{(1)} & \dots & a_{3n}^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & a_{n2}^{(1)} & \dots & a_{nn}^{(1)} \end{pmatrix}. \quad (2)$$

Пусть сделаны $k - 1$, $k = 1, \dots, n - 1$ шагов этого процесса, т.е. матрица преобразована к виду

$$A^{(k-1)} = \prod_{i=k}^2 \prod_{j=n}^{i+1} T_{ij} A \prod_{i=2}^k \prod_{j=i+1}^n T_{ij}^t, \quad (3)$$

$$A^{(k-1)} = \begin{pmatrix} a_{11} & c_{12} & c_{13} & \dots & c_{1,k-1} & a_{1k}^{(k-1)} & \dots & a_{1n}^{(k-1)} \\ \|a_1\| & a_{22}^{(1)} & c_{23} & \dots & c_{2,k-1} & a_{2k}^{(k-1)} & \dots & a_{2n}^{(k-1)} \\ \|a_1^{(1)}\| & a_{33}^{(2)} & \dots & c_{3,k-1} & a_{3k}^{(k-1)} & \dots & a_{3n}^{(k-1)} \\ \|a_1^{(2)}\| & \ddots & \vdots & & \vdots & \ddots & & \vdots \\ \ddots & a_{k-1,k-1}^{(k-2)} & a_{k-1,k}^{(k-1)} & \dots & a_{k-1,n}^{(k-1)} & & & \\ \|a_1^{(k-2)}\| & a_{kk}^{(k-1)} & \dots & a_{kn}^{(k-1)} & & & & \\ a_{k+1,k}^{(k-1)} & \dots & a_{k+1,n}^{(k-1)} & & & & & \\ \vdots & \ddots & \vdots & & & & & \\ a_{nk}^{(k-1)} & \dots & a_{nn}^{(k-1)} & & & & & \end{pmatrix}. \quad (4)$$

Обозначим

$$a_1^{(k-1)} = (a_{kk}^{(k-1)}, \dots, a_{nk}^{(k-1)})^t \in \mathbf{R}^{n-k} \quad (5)$$

– часть первого столбца подматрицы $(a_{ij}^{(k-1)})_{i,j=k,\dots,n}$. Согласно лемме 12.3 существуют $n - k - 1$ матриц $T_{k+1,j} = T_{k+1,j2}(\varphi_{k+1,j})$, $j = k + 2, \dots, n$ таких, что

$$T_{k+1,n} \dots T_{k+1,k+3} T_{k+1,k+2} a_1^{(k-1)} = \|a_1^{(k-1)}\| e_1^{(n-k)}, \quad (6)$$

(значения углов $\varphi_{k+1,j}$, $j = k + 2, \dots, n$ определяются леммами 12.2, 12.3). Умножим матрицу (3) на $T_{k+1,n} \dots T_{k+1,k+3} T_{k+1,k+2}$ слева, получим

$$\widehat{A}^{(k)} = \prod_{j=n}^{k+2} T_{k+1,j} A^{(k-1)}, \quad (7)$$

где

$$\widehat{A}^{(k)} = \begin{pmatrix} a_{11} & c_{12} & c_{13} & \dots & c_{1,k-1} & c_{1k} & a_{1,k+1}^{(k-1)} & \dots & a_{1n}^{(k-1)} \\ \|a_1\| & a_{22}^{(1)} & c_{23} & \dots & c_{2,k-1} & c_{2k} & a_{2,k+1}^{(k-1)} & \dots & a_{2n}^{(k-1)} \\ \|a_1^{(1)}\| & a_{33}^{(2)} & \dots & c_{3,k-1} & c_{3k} & a_{3,k+1}^{(k-1)} & \dots & a_{3n}^{(k-1)} \\ \|a_1^{(2)}\| & \ddots & \vdots & & \vdots & & \vdots & \ddots & \vdots \\ & \ddots & a_{k-1,k-1}^{(k-2)} & c_{k-1,k} & a_{k-1,k+1}^{(k-1)} & \dots & a_{k-1,n}^{(k-1)} \\ \|a_1^{(k-2)}\| & a_{kk}^{(k-1)} & a_{k,k+1}^{(k-1)} & \dots & a_{kn}^{(k-1)} & & & & \\ \|a_1^{(k-1)}\| & \widehat{a}_{k+1,k+1}^{(k)} & \dots & \widehat{a}_{k+1,n}^{(k)} \\ & \widehat{a}_{k+2,k+1}^{(k)} & \dots & \widehat{a}_{k+2,n}^{(k)} \\ & \vdots & \ddots & \vdots \\ & \widehat{a}_{n,k+1}^{(k)} & \dots & \widehat{a}_{nn}^{(k-1)} \end{pmatrix}. \quad (8)$$

Отметим, что в (7) каждая из $n - k - 1$ матриц элементарных вращений $T_{k+1,j}$ такова, что $j > k + 1$ и потому в (7) она умножается только на подматрицу $(a_{ij}^{(k-1)})_{i=k+1,\dots,n, j=k,\dots,n}$ матрицы $A^{(k-1)}$ размера $(n - k) \times (n - k + 1)$ (остальная часть $A^{(k-1)}$ в преобразовании (7) не участвует).

Умножим матрицу $\widehat{A}^{(k)}$ на $(T_{k+1,n} \dots T_{k+1,k+2})^* = T_{k+1,k+2}^* \dots T_{k+1,n}^*$ справа, получим из (8) (с учетом того, что при умножении справа на $T_{k+1,j}$, $j = k + 2, \dots, n$ столбцы $1, \dots, k$ матрицы $\widehat{A}^{(k)}$ не изменяются)

$$A^{(k)} = \widehat{A}^{(k)} \prod_{j=k+2}^n T_{k+1,j}^t = \prod_{j=n}^{k+2} T_{k+1,j} A^{(k-1)} \prod_{j=k+2}^n T_{k+1,j}^t, \quad (9)$$

$$A^{(k)} = \begin{pmatrix} a_{11} & c_{12} & c_{13} & \dots & c_{1,k-1} & c_{1k} & a_{1,k+1}^{(k)} & \dots & a_{1n}^{(k)} \\ \|a_1\| & a_{22}^{(1)} & c_{23} & \dots & c_{2,k-1} & c_{2k} & a_{2,k+1}^{(k)} & \dots & a_{2n}^{(k)} \\ \|a_1^{(1)}\| & a_{33}^{(2)} & \dots & c_{3,k-1} & c_{3k} & a_{3,k+1}^{(k)} & \dots & a_{3n}^{(k)} \\ \|a_1^{(2)}\| & \ddots & \vdots & & \vdots & & \vdots & \ddots & \vdots \\ & \ddots & a_{k-1,k-1}^{(k-2)} & c_{k-1,k} & a_{k-1,k+1}^{(k)} & \dots & a_{k-1,n}^{(k)} \\ \|a_1^{(k-2)}\| & a_{kk}^{(k-1)} & a_{k,k+1}^{(k)} & \dots & a_{kn}^{(k)} & & & & \\ \|a_1^{(k-1)}\| & a_{k+1,k+1}^{(k)} & \dots & a_{k+1,n}^{(k)} \\ & a_{k+2,k+1}^{(k)} & \dots & a_{k+2,n}^{(k)} \\ & \vdots & \ddots & \vdots \\ & a_{n,k+1}^{(k)} & \dots & a_{nn}^{(k-1)} \end{pmatrix}. \quad (10)$$

Отметим, что в (9) каждая из $n - k - 1$ матриц элементарных вращений $T_{k+1,j}^t(\varphi_{k+1,j}) = T_{k+1,j}(-\varphi_{k+1,j})$ такова, что $j > k + 1$ и потому в (9) она умножается только на подматрицу $(\widehat{a}_{ij}^{(k-1)})_{i=1,\dots,n, j=k+1,\dots,n}$ матрицы $\widehat{A}^{(k-1)}$ размера $n \times (n - k)$ (остальная часть $\widehat{A}^{(k-1)}$ в преобразовании (9) не участвует).

После $n - 2$ шагов этого процесса (т.е. перехода от матриц (3), (4) к (9), (10)) матрица примет требуемый почти треугольный вид

$$R = A^{(n-2)} = \prod_{i=n-1}^2 \prod_{j=n}^{i+1} T_{ij} A \prod_{i=2}^{n-1} \prod_{j=i+1}^n T_{ij}^t, \quad (11)$$

$$R = \begin{pmatrix} a_{11} & c_{12} & c_{13} & \dots & c_{1,n-2} & a_{1,n-1}^{(n-2)} & a_{1n}^{(n-2)} \\ \|a_1\| & a_{22}^{(1)} & c_{23} & \dots & c_{2,n-2} & a_{2,n-1}^{(n-2)} & a_{2n}^{(n-2)} \\ \|a_1^{(1)}\| & a_{33}^{(2)} & \dots & c_{3,n-2} & a_{3,n-1}^{(n-2)} & a_{3n}^{(n-2)} \\ & \|a_1^{(2)}\| & \ddots & \vdots & \vdots & \vdots & \vdots \\ & & \ddots & a_{n-2,n-2}^{(n-3)} & a_{n-2,n-1}^{(n-2)} & a_{n-2,n}^{(n-2)} \\ & & & \|a_1^{(n-3)}\| & a_{n-1,n-1}^{(n-2)} & a_{n-1,n}^{(n-2)} \\ & & & & a_{n,n-1}^{(n-2)} & a_{nn}^{(n-2)} \end{pmatrix} \quad (12)$$

(напомним, определения векторов $a_1^{(k-1)}$, $k = 1, \dots, n - 2$ даются в (5), где считаем, что $a_1^{(0)} = a_1$).

Оценка количества арифметических операций в алгоритме приведения матрицы к почти треугольному виду унитарным подобием методом вращений

Оценим трудоемкость k -го шага алгоритма, а затем просуммируем полученные оценки по всем $k = 1, \dots, n - 2$.

1. На вычисление $n - k - 1$ матриц $T_{k,k+1}, \dots, T_{kn}$, участвующих в (6), согласно лемме 12.2 требуется $4(n - k - 1)$ мультипликативных, $2(n - k - 1)$ аддитивных и $n - k$ операций извлечения корня.

2. На вычисление компонент $k + 1, \dots, n$ k -го столбца матрицы $\hat{A}^{(k)}$, равных компонентам вектора $\|a_1^{(k-1)}\| e_1^{(n-k)}$ требуется (для вычисления длины вектора (5)) $n - k$ операций умножения, $n - k - 1$ операций сложения и одна операция извлечения корня. Столбец k вычисляется именно этим способом (а не по общим формулам (7)) для сокращения количества арифметических операций и уменьшения вычислительной погрешности.

3. Поскольку в формуле (7) каждая из $n - k - 1$ матриц элементарных вращений умножается на подматрицу $(a_{ij}^{(k-1)})_{i=k+1, \dots, n, j=k+1, \dots, n}$ матрицы $A^{(k-1)}$ размера $(n - k) \times (n - k)$ (k -й столбец матрицы $\hat{A}^{(k)}$ уже вычислен в пункте 2), то согласно лемме 12.5 на это требуется $(n - k - 1)4(n - k) = 4(n - k)^2 - 4(n - k)$ умножений и $(n - k - 1)2(n - k) = 2(n - k)^2 - 2(n - k)$ сложений.

4. Поскольку матрица, транспонированная к матрице элементарного вращения, опять является матрицей элементарного вращения:

$$T_{k+1,j}^t(\varphi_{k+1,j}) = T_{k+1,j}(-\varphi_{k+1,j})$$

и в формуле (9) каждая из этих $n - k - 1$ матриц элементарных вращений умножается только на подматрицу $(\hat{a}_{ij}^{(k-1)})_{i=1, \dots, n, j=k+1, \dots, n}$ матрицы $\hat{A}^{(k-1)}$ размера

$n \times (n - k)$, то согласно лемме 12.5 на это требуется $(n - k - 1)4n = 4n(n - k) - 4n$ умножений и $(n - k - 1)2n = 2n(n - k) - 2n$ сложений.

Итак, на k -ом шаге алгоритма требуется выполнить $4(n - k - 1) + (n - k) + 4(n - k)^2 - 4(n - k) + 4n(n - k) - 4n = 4n(n - k) + 4(n - k)^2 + (n - k) - 4n - 4$ мультипликативных операций, $2(n - k - 1) + (n - k - 1) + 2(n - k)^2 - 2(n - k) + 2n(n - k) - 2n = 2n(n - k) + 2(n - k)^2 + (n - k) - 2n - 3$ аддитивных операций и $n - k$ операций извлечения корня.

Следовательно, всего для проведения алгоритма требуется выполнить

$$\begin{aligned} \sum_{k=1}^{n-2} (4n(n - k) + 4(n - k)^2 + (n - k) - 4n - 4) &= 4n((n - 1)(n - 2)/2 - 1) \\ + 4((n - 1)(n - 2)(2n - 3)/6 - 1) + (n - 1)(n - 2)/2 - 1 - 4(n - 2)(n + 1) & \\ = 2n^3 + O(n^2) + \frac{4}{3}n^3 + O(n^2) + O(n^2) &= \frac{10}{3}n^3 + O(n^2) \quad (n \rightarrow \infty) \end{aligned}$$

мультипликативных операций, $\sum_{k=1}^{n-2} (2n(n - k) + 2(n - k)^2 + (n - k) - 2n - 3) = \frac{5}{3}n^3 + O(n^2)$ ($n \rightarrow \infty$) аддитивных операций и $\sum_{k=1}^{n-2} (n - k) = O(n^2)$ ($n \rightarrow \infty$) операций извлечения корня (которые по трудоемкости по порядку можно сравнить с операциями деления).

Таким образом, на приведение матрицы к почти треугольному виду унитарным подобием методом вращений требуется $\frac{10}{3}n^3 + O(n^2)$ ($n \rightarrow \infty$) мультипликативных операций и $\frac{5}{3}n^3 + O(n^2)$ ($n \rightarrow \infty$) аддитивных операций. Заметим, что это количество операций в два с половиной раза больше, чем нужно для решения линейной системы методом вращений.

Теорема 1. *Всякая невырожденная вещественная матрица A может быть представлена в виде $A = QRQ^t$, где матрица Q – ортогональная, а матрица R – верхняя почти треугольная.*

Доказательство. Проведем для матрицы A изложенный выше алгоритм, осуществимый для всякой невырожденной матрицы. Обозначим в (11) $\hat{Q} = \prod_{i=n-1}^2 \prod_{j=n}^{i+1} T_{ij}$. Как произведение ортогональных матриц, матрица \hat{Q} ортогональна. Тогда (11) имеет вид $R = \hat{Q}A\hat{Q}^t$, откуда $A = (\hat{Q})^{-1}R(\hat{Q}^t)^{-1} = QRQ^t$, где $Q = (\hat{Q})^t = (\hat{Q})^{-1}$ – ортогональная матрица. Матрица R , имеющая вид (12), удовлетворяет условиям теоремы.

Замечание 1. Как отмечалось выше, построенное в теореме 1 разложение используется в ряде алгоритмов нахождения собственных значений матрицы.

Хранение матриц Q и R в памяти осуществляется одним из способов, изложенных при обсуждении алгоритма построения QR -разложения для матрицы A методом вращений.

Трудоемкость алгоритма построения описанного выше разложения складывается из количества арифметических операций, необходимых для проведения самого алгоритма, и количества арифметических операций, необходимых

для построения матрицы Q . Подробные выкладки были проведены при обсуждении алгоритма построения QR -разложения методом вращений.

§ 14.2. Случай симметричной матрицы

Рассмотрим ситуацию, когда описанный выше метод приведения к почти треугольному виду применяется к симметричной матрице $A \in \mathbf{M}_n$. Согласно (1), (2) $A^{(1)} = Q_1 A Q_1^t$, где $Q_1 = T_{2n} \dots T_{24} T_{23}$ – ортогональная матрица, т.е. $A^{(1)}$ и A – унитарно подобны. Следовательно, $A^{(1)}$ – симметричная матрица. Согласно (7), (9) на k -ом ($k = 1, \dots, n - 2$) шаге алгоритма $A^{(k)} = Q_k A^{(k-1)} Q_k^t$, где $Q_k = T_{k+1,n} \dots T_{k+1,k+3} T_{k+1,k+2}$ – ортогональная матрица. Следовательно, $A^{(k)}$ и A унитарно подобны, и $A^{(k)}$ – симметричная матрица для всякого $k = 1, \dots, n - 2$. Таким образом, $R = A^{(n-2)}$ – почти треугольная и симметричная, т.е. трехдиагональная матрица.

Запишем описанный выше процесс приведения симметричной матрицы к трехдиагональному виду так, чтобы максимально уменьшить объем вычислительной работы за счет использования симметрии.

Лемма 1. Для всякой матрицы элементарного вращения $T_{ij} \in \mathbf{M}_n$ и всякой симметричной матрицы $A \in \mathbf{M}_n$ матрица $B = T_{ij} A T_{ij}^t$ может быть вычислена за $4n + 8$ умножений и $2n + 4$ сложений.

Доказательство. Матрица $\hat{B} = T_{ij} A$ согласно лемме 12.5 вычисляется за $4n$ умножений и $2n$ сложений. Матрица $B = (b_{kl}) = T_{ij} A T_{ij}^t$ унитарно подобна A и потому симметрична:

$$b_{kl} = b_{lk}, \quad k, l = 1, \dots, n.$$

С другой стороны, матрица $B = \hat{B} T_{ij}^t$ получается из матрицы $\hat{B} = (\hat{b}_{kl})$ изменением элементов, расположенных только в i -ом и j -ом столбцах:

$$b_{kl} = \hat{b}_{kl}, \quad l \neq i, j, \quad k, l = 1, \dots, n.$$

Из последних двух равенств получаем:

$$b_{kl} = \hat{b}_{kl}, \quad (k, l) \neq (i, i), (i, j), (j, i) (j, j) \quad k, l = 1, \dots, n,$$

т.е. у матриц B и \hat{B} отличаются только 4 элемента с индексами (i, i) , (i, j) , (j, i) , (j, j) . Эти элементы получаются умножением i -й и j -й строки матрицы \hat{B} на матрицу T_{ij} справа и вычисляются по формулам

$$\begin{aligned} b_{ii} &= \hat{b}_{ii} \cos \varphi_{ij} + \hat{b}_{ij} \sin \varphi_{ij}, & b_{ij} &= \hat{b}_{ii} \sin \varphi_{ij} - \hat{b}_{ij} \cos \varphi_{ij}, \\ b_{ji} &= \hat{b}_{ji} \cos \varphi_{ij} + \hat{b}_{jj} \sin \varphi_{ij}, & b_{jj} &= \hat{b}_{jj} \sin \varphi_{ij} - \hat{b}_{ji} \cos \varphi_{ij}. \end{aligned}$$

Эти вычисления требуют дополнительно 8 умножений и 4 сложения. Складывая это с трудоемкостью построения матрицы \hat{B} , получаем требуемую оценку.

Замечание 2. Для несимметричной матрицы A вычисление матрицы $B = T_{ij}AT_{ij}^t$ требует $8n$ умножений и $4n$ сложений (см. лемму 12.5).

Обозначим $a_1 = (a_{21}, \dots, a_{n1})^t$. Согласно лемме 12.3 существуют $n-2$ матриц $T_{2j} = T_{2j}(\varphi_{2j})$, $j = 3, \dots, n$, таких, что $T_{2n} \dots T_{24}T_{23}a_1 = \|a_1\| e_1^{(n-1)}$ (причем значения углов φ_{2j} , $j = 3, \dots, n$ определяются леммами 12.2, 12.3). Обозначим $A_3^{(1)} = T_{23}AT_{23}^t$, $A_4^{(1)} = T_{24}A_3^{(1)}T_{24}^t$, ..., $A_n^{(1)} = T_{2n}A_{n-1}^{(1)}T_{2n}^t$. Все матрицы $A_j^{(1)}$, $j = 3, 4, \dots, n$ унитарно подобны A и потому симметричны. Согласно (1), (2) $A^{(1)} = A_n^{(1)}$. В силу симметрии матрицы $A^{(1)}$ вместо (2) для нее справедливо более точное равенство

$$A^{(1)} = T_{2n}(\dots(T_{24}(T_{23}AT_{23}^t)T_{24}^t)\dots)T_{2n}^t = \begin{pmatrix} a_{11} & \|a_1\| & 0 & \dots & 0 \\ \|a_1\| & a_{22}^{(1)} & a_{23}^{(1)} & \dots & a_{2n}^{(1)} \\ 0 & a_{32}^{(1)} & a_{33}^{(1)} & \dots & a_{3n}^{(1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & a_{n2}^{(1)} & a_{n3}^{(1)} & \dots & a_{nn}^{(1)} \end{pmatrix}. \quad (13)$$

Таким образом, у матрицы $A^{(1)}$ необходимо с помощью леммы 1 вычислить только подматрицу $(a_{ij}^{(1)})_{i,j=2,\dots,n} \in \mathbf{M}_{n-1}$ (так как остальные элементы уже вычислены).

Пусть сделаны $k-1$, $k = 1, \dots, n-1$ шагов этого процесса, т.е. матрица преобразована к виду (3), где

$$A^{(k-1)} = \begin{pmatrix} a_{11} & \|a_1\| & & & \\ \|a_1\| & a_{22}^{(1)} & \|a_1^{(1)}\| & & \\ & \|a_1^{(1)}\| & a_{33}^{(2)} & \ddots & \\ & & \|a_1^{(2)}\| & \ddots & \|a_1^{(k-3)}\| \\ & & & \ddots & a_{k-1,k-1}^{(k-2)} & \|a_1^{(k-2)}\| \\ & & & & \|a_1^{(k-2)}\| & a_{kk}^{(k-1)} & \dots & a_{kn}^{(k-1)} \\ & & & & & a_{k+1,k}^{(k-1)} & \dots & a_{k+1,n}^{(k-1)} \\ & & & & & \vdots & \ddots & \vdots \\ & & & & & a_{nk}^{(k-1)} & \dots & a_{nn}^{(k-1)} \end{pmatrix}. \quad (14)$$

Введем обозначение (5). Согласно лемме 12.3 существуют $n-k-1$ матриц $T_{k+1,j} = T_{k+1,j2}(\varphi_{k+1,j})$, $j = k+2, \dots, n$ таких, что справедливо равенство (6).

Обозначим $A_{k+2}^{(k)} = T_{k+1,k+2}A^{(k-1)}T_{k+1,k+2}^t$, $A_{k+3}^{(k)} = T_{k+1,k+3}A_{k+2}^{(k)}T_{k+1,k+3}^t$, ..., $A_n^{(k)} = T_{k+1,n}A_{n-1}^{(k)}T_{k+1,n}^t$. Все матрицы $A_j^{(k)}$, $j = k+1, k+2, \dots, n$ унитарно подобны $A^{(k-1)}$ и потому симметричны. Согласно (7), (9) $A^{(k)} = A_n^{(k)}$. В силу симметрии матрицы

$$A^{(k)} = T_{k+1,n}(\dots(T_{k+1,k+3}(T_{k+1,k+2}A^{(k-1)}T_{k+1,k+2}^t)T_{k+1,k+3}^t)\dots)T_{k+1,n}^t \quad (15)$$

вместо (10) для нее справедливо более точное равенство

$$A^{(k)} = \begin{pmatrix} a_{11} & \|a_1\| & & & & \\ \|a_1\| & a_{22}^{(1)} & \|a_1^{(1)}\| & & & \\ & \|a_1^{(1)}\| & a_{33}^{(2)} & \ddots & & \\ & & \|a_1^{(2)}\| & \ddots & \|a_1^{(k-3)}\| & \\ & & & \ddots & a_{k-1,k-1}^{(k-2)} & \|a_1^{(k-2)}\| \\ & & & & \|a_1^{(k-2)}\| & a_{kk}^{(k-1)} \\ & & & & & \|a_1^{(k-1)}\| & a_{k+1,k+1}^{(k)} & \cdots & a_{k+1,n}^{(k)} \\ & & & & & & a_{k+2,k+1}^{(k)} & \cdots & a_{k+2,n}^{(k)} \\ & & & & & & \vdots & & \vdots \\ & & & & & & a_{n,k+1}^{(k)} & \cdots & a_{nn}^{(k-1)} \end{pmatrix}. \quad (16)$$

Таким образом, у матрицы $A^{(k)}$ необходимо с помощью леммы 1 вычислить только подматрицу $(a_{ij}^{(k)})_{i,j=k+1,\dots,n} \in \mathbf{M}_{n-k-1}$ (так как остальные элементы уже вычислены).

После $n-2$ шагов этого процесса (т.е. перехода от матриц (3), (14) к (15), (16)) матрица примет требуемый трехдиагональный вид (11), где

$$R = \begin{pmatrix} a_{11} & \|a_1\| & & & & \\ \|a_1\| & a_{22}^{(1)} & \|a_1^{(1)}\| & & & \\ & \|a_1^{(1)}\| & a_{33}^{(2)} & \ddots & & \\ & & \|a_1^{(2)}\| & \ddots & \|a_1^{(n-4)}\| & \\ & & & \ddots & a_{n-2,n-2}^{(n-3)} & \|a_1^{(n-3)}\| \\ & & & & \|a_1^{(n-3)}\| & a_{n-1,n-1}^{(n-2)} & a_{n,n-1}^{(n-2)} \\ & & & & & a_{n,n-1}^{(n-2)} & a_{nn}^{(n-2)} \end{pmatrix} \quad (17)$$

(напомним, определения векторов $a_1^{(k-1)}$, $k = 1, \dots, n-2$ даются в (5), где считаем, что $a_1^{(0)} = a_1$).

Оценка количества арифметических операций в алгоритме приведения симметричной матрицы к трехдиагональному виду унитарным подобием методом вращений

Оценим трудоемкость k -го шага алгоритма, а затем просуммируем полученные оценки по всем $k = 1, \dots, n-2$.

1. На вычисление $n-k-1$ матриц $T_{k,k+1}, \dots, T_{kn}$, участвующих в (6), согласно лемме 12.2 требуется $4(n-k-1)$ мультипликативных, $2(n-k-1)$ аддитивных и $n-k$ операций извлечения корня.

2. На вычисление компонент $k+1, \dots, n$ k -го столбца и k -ой строки матрицы $A^{(k)}$, равных компонентам вектора $\|a_1^{(k-1)}\| e_1^{(n-k)}$ требуется (для вычисления

длины вектора (5)) $n - k$ операций умножения, $n - k - 1$ операций сложения и одна операция извлечения корня. k -й столбец и k -ая строка вычисляются именно этим способом для сокращения количества арифметических операций и уменьшения вычислительной погрешности.

3. Вычисления по формуле (15) можно представить как последовательное вычисление матриц $A_l^{(k)}$, $l = k + 1, k + 2, \dots, n$, у которых нужно вычислить только подматрицу $((a_l^{(k)})_{ij})_{i,j=k+1,\dots,n} \in M_{n-k-1}$. Согласно лемме 1 на вычисление каждой подматрицы требуется $4(n - k - 1) + 8 = 4(n - k) + 4$ умножений и $2(n - k - 1) + 4 = 2(n - k) + 2$ сложений. На вычисление всех $n - k - 1$ подматриц требуется, таким образом, $(n - k - 1)(4(n - k) + 4) = 4(n - k)^2 - 4$ умножений и $(n - k - 1)(2(n - k) + 2) = 2(n - k)^2 - 2$ сложений.

Итак, на k -ом шаге алгоритма требуется выполнить $4(n - k - 1) + (n - k) + 4(n - k)^2 - 4 = 4(n - k)^2 + 5(n - k) - 8$ мультипликативных операций, $2(n - k - 1) + (n - k - 1) + 2(n - k)^2 - 2 = 2(n - k)^2 - 3(n - k) - 5$ аддитивных операций и $n - k$ операций извлечения корня.

Следовательно, всего для проведения алгоритма требуется выполнить

$$\begin{aligned} \sum_{k=1}^{n-2} (4(n - k)^2 + 5(n - k) - 8) &= 4(n(n - 1)(2n - 1)/6 - 1) + 5(n(n - 1)/2 - 1) - 8(n - 2) \\ &= \frac{4}{3}n^3 + O(n^2) + O(n^2) + O(n) = \frac{4}{3}n^3 + O(n^2) \quad (n \rightarrow \infty) \end{aligned}$$

мультипликативных операций, $\sum_{k=1}^{n-2} (2(n - k)^2 - 3(n - k) - 5) = \frac{2}{3}n^3 + O(n^2)$ ($n \rightarrow \infty$) аддитивных операций и $\sum_{k=1}^{n-2} (n - k) = O(n^2)$ ($n \rightarrow \infty$) операций извлечения корня (которые по трудоемкости по порядку можно сравнить с операциями деления).

Таким образом, на приведение симметричной матрицы к трехдиагональному виду унитарным подобием методом вращений требуется $\frac{4}{3}n^3 + O(n^2)$ ($n \rightarrow \infty$) мультипликативных операций и $\frac{2}{3}n^3 + O(n^2)$ ($n \rightarrow \infty$) аддитивных операций. Заметим, что это количество операций в два с половиной раза меньше, чем требуется для приведения произвольной матрицы к почти треугольному виду унитарным подобием методом вращений и совпадает количеством операций, необходимым для решения линейной системы методом вращений.

Теорема 2. *Всякая невырожденная вещественная симметричная матрица A может быть представлена в виде $A = QRQ^t$, где матрица Q – ортогональная, а матрица R – трехдиагональная.*

Доказательство Совпадает с доказательством теоремы 1.

Хранение матриц Q и R в памяти осуществляется одним из способов, изложенных при обсуждении алгоритма построения QR -разложения для матрицы A методом вращений. Для симметричных матриц A удобно применять второй способ. Действительно, так как на шаге k , $k = 1, \dots, n - 2$ мы использовали $n - k - 1$ элементарных вращений $T_{k+1,k+2}, \dots, T_{k+1,n}$ для получения нулевых

элементов $a_{jk}^{(k)} = 0$, $a_{kj}^{(k)} = 0$, $j = k + 2, \dots, n$, то можем хранить, например, $\cos \varphi_{k+1,j}$ на месте $a_{jk}^{(k)} = 0$, а $\sin \varphi_{k+1,j}$ — на месте $a_{kj}^{(k)} = 0$.

Трудоемкость алгоритма построения описанного выше разложения складывается из количества арифметических операций, необходимых для проведения самого алгоритма, и количества арифметических операций, необходимых для построения матрицы Q . Подробные выкладки были проведены при обсуждении алгоритма построения QR -разложения методом вращений.

§ 15. ПРИВЕДЕНИЕ МАТРИЦЫ К ПОЧТИ ТРЕУГОЛЬНОМУ ВИДУ УНИТАРНЫМ ПОДОБИЕМ МЕТОДОМ ОТРАЖЕНИЙ

Пусть требуется привести матрицу A (не обязательно вещественную) к почти треугольному виду.

Всюду ниже мы будем часто пользоваться следующими фактами.

1. Если по произвольной матрице $U_k \in \mathbf{M}_k$ (размера $k \times k$) построить матрицу $U \in \mathbf{M}_n$ (размера $n \times n$) по формуле

$$U = \begin{pmatrix} I_{n-k} & 0 \\ 0 & U_k \end{pmatrix}, \quad (1)$$

где $I_{n-k} \in \mathbf{M}_{n-k}$ — единичная матрица размера $(n-k) \times (n-k)$, то при умножении матрицы A на матрицу U слева изменяются только последние k строк матрицы A , а при умножении на U справа изменяются только последние k столбцов матрицы A . Это следует из определения умножения матриц.

2. Если матрица $U_k \in \mathbf{M}_k$ в (1) самосопряженная, то матрица $U \in \mathbf{M}_n$, полученная в (1), также самосопряженная. Это доказано при рассмотрении алгоритма метода отражений, см. (13.8).

3. Если матрица $U_k \in \mathbf{M}_k$ в (1) унитарна, то матрица $U \in \mathbf{M}_n$, полученная в (1), также унитарна. Это доказано при рассмотрении алгоритма метода отражений, см. (13.9).

§ 15.1. Случай произвольной матрицы

Обозначим $a_1 = (a_{21}, \dots, a_{n1})^t$. Согласно лемме 13.9 существует вектор $x^{(1)} \in \mathbf{C}^n$, равный

$$x^{(1)} = \pm \frac{a_1 - \|a_1\|e_1}{\|a_1 - \|a_1\|e_1\|},$$

такой, что $U(x^{(1)})a_1 = \|a_1\|e_1$, где $e_1 = (1, 0, \dots, 0) \in \mathbf{C}^{n-1}$, $U(x^{(1)}) \in \mathbf{M}_{n-1}$ — матрица отражения. Положим

$$U_1 = \begin{pmatrix} 1 & 0 \\ 0 & U(x^{(1)}) \end{pmatrix}. \quad (2)$$

Как отмечалось выше, матрица U_1 является унитарной.

Умножим матрицу A на U_1 слева, получим матрицу $\widehat{A}^{(1)}$ вида (14.1) (поскольку первая строка матрицы A не изменяется). Умножим матрицу $\widehat{A}^{(1)}$ на $U_1^* = U_1$ справа, получим матрицу (14.2) (с учетом того, что при умножении справа на U_1 первый столбец матрицы $\widehat{A}^{(1)}$ не изменяется).

Пусть сделаны $k - 1$, $k = 1, \dots, n - 1$ шагов этого процесса, т.е. матрица преобразована к виду

$$A^{(k-1)} = \prod_{i=k}^2 U_i A \prod_{i=2}^k U_i \quad (3)$$

где $A^{(k-1)}$ имеет вид (14.4),

$$U_i = \begin{pmatrix} I_i & 0 \\ 0 & U(x^{(i)}) \end{pmatrix},$$

здесь $I_i \in \mathbf{M}_i$ – единичная матрица размера $i \times i$, $U(x^{(i)}) \in \mathbf{M}_{n-i}$ – матрица отражения размера $(n - i) \times (n - i)$, построенная по вектору

$$x^{(i)} = \pm \frac{a_1^{(i-1)} - \|a_1^{(i-1)}\|e_1^{(n-i)}}{\|a_1^{(i-1)} - \|a_1^{(i-1)}\|e_1^{(n-i)}\|} \in \mathbf{C}^{n-i},$$

где $e_1^{(m)} = (1, 0, \dots, 0) \in \mathbf{C}^m$,

$$a_1^{(i-1)} = (a_{i+1,i}^{(i-1)}, \dots, a_{ni}^{(i-1)})^t \in \mathbf{C}^{n-i}.$$

Обозначим через $a_1^{(k-1)}$ часть первого столбца подматрицы $(a_{ij}^{(k-1)})_{i,j=k,\dots,n}$, см. (14.5). Согласно лемме 13.9 существует матрица отражения

$$U(x^{(k)}) = I - 2x^{(k)}(x^{(k)})^*, \quad x^{(k)} = \pm \frac{a_1^{(k-1)} - \|a_1^{(k-1)}\|e_1^{(n-k)}}{\|a_1^{(k-1)} - \|a_1^{(k-1)}\|e_1^{(n-k)}\|} \in \mathbf{C}^{n-k}, \quad (4)$$

такая, что

$$U(x^{(k)})a_1^{(k-1)} = \|a_1^{(k-1)}\|e_1^{(n-k)}. \quad (5)$$

Положим

$$U_k = \begin{pmatrix} I_k & 0 \\ 0 & U(x_k) \end{pmatrix}. \quad (6)$$

Как отмечалось выше, матрица U_k является самосопряженной и унитарной.

Умножим матрицу (3) на U_k слева, получим

$$\widehat{A}^{(k)} = U_k A^{(k-1)}, \quad (7)$$

где матрица $\widehat{A}^{(k)}$ имеет вид (14.8). Отметим, что в (7) первые k строк у матриц $\widehat{A}^{(k)}$ и $A^{(k-1)}$ совпадают. Другими словами, преобразование (7) заключается в

умножении матрицы $U(x_k) \in \mathbf{M}_{n-k}$ на подматрицу $(a_{ij}^{(k-1)})_{i=k+1, \dots, n, j=k, \dots, n}$ матрицы $A^{(k-1)}$ размера $(n-k) \times (n-k)$ (остальная часть $A^{(k-1)}$ в преобразовании (7) не участвует).

Умножим матрицу $\widehat{A}^{(k)}$ на $U_k^* = U_k$ справа, получим из (14.8) (с учетом того, что при умножении справа на U_k столбцы $1, \dots, k$ матрицы $\widehat{A}^{(k)}$ не изменяются)

$$A^{(k)} = \widehat{A}^{(k)} U_k = U_k A^{(k-1)} U_k, \quad (8)$$

где матрица $A^{(k)}$ имеет вид (14.10). Отметим, что в (8) первые k столбцов у матриц $A^{(k)}$ и $\widehat{A}^{(k)}$ совпадают. Другими словами, преобразование (8) заключается в умножении матрицы $U(x_k) \in \mathbf{M}_{n-k}$ на подматрицу $(\widehat{a}_{ij}^{(k-1)})_{i=1, \dots, n, j=k+1, \dots, n}$ матрицы $\widehat{A}^{(k-1)}$ размера $n \times (n-k)$ (остальная часть $\widehat{A}^{(k-1)}$ в преобразовании (8) не участвует).

Вычисления по формулам (4) осуществляются следующим образом: вначале вычисляются числа

$$s_k = \sum_{j=k+2}^n |a_{jk}^{(k-1)}|^2, \quad (9)$$

$$\|a_1^{(k-1)}\| = \sqrt{|a_{k+1,k}^{(k-1)}|^2 + s_k}. \quad (10)$$

затем – вектор

$$x^{(k)} = (a_{k+1,k}^{(k-1)} - \|a_1^{(k-1)}\|, a_{k+2,k}^{(k-1)}, \dots, a_{nk}^{(k-1)})^t \in \mathbf{C}^{n-k} \quad (11)$$

и его норма

$$\|x^{(k)}\| = \sqrt{|x_1^{(k)}|^2 + s_k}. \quad (12)$$

Теперь можно вычислить искомый вектор $x^{(k)}$:

$$x^{(k)} := x^{(k)} / \|x^{(k)}\|, \quad \text{т.е.} \quad x_j^{(k)} := x_j^{(k)} / \|x^{(k)}\|, \quad j = 1, \dots, n-k. \quad (13)$$

После $n-2$ шагов этого процесса (т.е. перехода от матриц (3), (14.4) к (8), (14.10)) матрица примет требуемый почти треугольный вид (14.12), где

$$R = A^{(n-2)} = \prod_{i=n-2}^1 U_i A \prod_{i=1}^{n-2} U_i. \quad (14)$$

Оценка количества арифметических операций в алгоритме приведения матрицы к почти треугольному виду унитарным подобием методом отражений

Оценим трудоемкость k -го шага алгоритма, а затем просуммируем полученные оценки по всем $k = 1, \dots, n-2$.

1. На вычисление матрицы $U(x_k)$ по формулам (4) требуется

а) $n-k-1$ умножений и $n-k-2$ сложений для вычисления s_k в (9);

- б) одно умножение, одно сложение и одна операция извлечения корня для вычисления $\|a_1^{(k-1)}\|$ в (10);
- в) одно вычитание для построения вектора $x^{(k)}$ в (11);
- г) одно умножение, одно сложение и одна операция извлечения корня для вычисления $\|x^{(k)}\|$ в (12);
- д) $n - k$ делений для построения вектора $x^{(k)}$ в (13).

Всего для построения матрицы $U(x_k)$ требуется $(n - k - 1) + 1 + 1 + (n - k) = 2(n - k) + 1$ мультипликативных, $(n - k - 2) + 1 + 1 + 1 = n - k + 1$ аддитивных операций и $1 + 1 = 2$ операции извлечения корня.

2. Компоненты $k+1, \dots, n$ k -го столбца матрицы $A^{(k)}$, равные компонентам вектора $\|a_1^{(k-1)}\| e_1^{(n-k)}$, уже вычислены в (10). Столбец k вычисляется не по общим формулам (7) для сокращения количества арифметических операций и уменьшения вычислительной погрешности.

3. Поскольку в формуле (7) матрица $U(x_k) \in \mathbf{M}_{n-k}$ умножается на подматрицу $(a_{ij}^{(k-1)})_{i=k+1, \dots, n, j=k+1, \dots, n}$ матрицы $A^{(k-1)}$ размера $(n - k) \times (n - k)$ (k -й столбец матрицы $\widehat{A}^{(k)}$ уже вычислен в пункте 2), то согласно лемме 13.11 на это требуется $2(n - k)^2 + O(n - k)$ ($n \rightarrow \infty$) умножений и столько же сложений.

4. Поскольку в формуле (8) матрица $U(x_k) \in \mathbf{M}_{n-k}$ умножается на подматрицу $(\widehat{a}_{ij}^{(k-1)})_{i=1, \dots, n, j=k+1, \dots, n}$ матрицы $\widehat{A}^{(k-1)}$ размера $n \times (n - k)$, то согласно лемме 13.11 на это требуется $2(n - k)n + O(n - k)$ ($n \rightarrow \infty$) умножений и столько же сложений.

Итак, на k -ом шаге алгоритма требуется выполнить $2(n - k) + 1 + 2(n - k)^2 + 2n(n - k) + O(n - k) = 2n(n - k) + 2(n - k)^2 + O(n - k)$ мультипликативных операций, $n - k + 1 + 2(n - k)^2 + 2n(n - k) + O(n - k) = 2n(n - k) + 2(n - k)^2 + O(n - k)$ аддитивных операций и 2 операции извлечения корня.

Следовательно, всего для проведения алгоритма требуется выполнить

$$\begin{aligned} & \sum_{k=1}^{n-2} (2n(n - k) + 2(n - k)^2 + O(n - k)) \\ &= 2n((n - 1)(n - 2)/2) + 2((n - 1)(n - 2)(2n - 3)/6) + O(n^2) \\ &= n^3 + O(n^2) + \frac{2}{3}n^3 + O(n^2) = \frac{5}{3}n^3 + O(n^2) \quad (n \rightarrow \infty) \end{aligned}$$

мультипликативных операций, столько же аддитивных операций и $2(n - 2)$ операций извлечения корня (которые по трудоемкости по порядку можно сравнить с операциями деления).

Таким образом, на приведение матрицы к почти треугольному виду унитарным подобием методом отражений требуется $\frac{5}{3}n^3 + O(n^2)$ ($n \rightarrow \infty$) мультипликативных операций и столько же аддитивных операций. Заметим, что это количество операций в два с половиной раза больше, чем нужно для решения линейной системы методом отражений.

Теорема 1. Всякая невырожденная матрица A может быть представлена в виде $A = QRQ^t$, где матрица Q – унитарная, а матрица R – верхняя почти треугольная.

Доказательство. Проведем для матрицы A изложенный выше алгоритм, осуществимый для всякой невырожденной матрицы. Обозначим в (14) $\hat{Q} = \prod_{i=n-2}^1 U_i$. Как произведение унитарных матриц, матрица \hat{Q} унитарна. Тогда (14) имеет вид $R = \hat{Q}A\hat{Q}^t$, откуда $A = (\hat{Q})^{-1}R(\hat{Q}^t)^{-1} = QRQ^t$, где $Q = (\hat{Q})^t = (\hat{Q})^{-1}$ – унитарная матрица. Матрица R , имеющая вид (14.12), удовлетворяет условиям теоремы.

Замечание 1. Как отмечалось выше, построенное в теореме 1 разложение используется в ряде алгоритмов нахождения собственных значений матрицы.

Хранение матриц Q и R в памяти осуществляется одним из способов, изложенных при обсуждении алгоритма построения QR -разложения для матрицы A методом отражений.

Трудоемкость алгоритма построения описанного выше разложения складывается из количества арифметических операций, необходимых для проведения самого алгоритма, и количества арифметических операций, необходимых для построения матрицы Q . Подробные выкладки были проведены при обсуждении алгоритма построения QR -разложения методом отражений.

§ 15.2. Случай самосопряженной матрицы

Рассмотрим ситуацию, когда описанный выше метод приведения к почти треугольному виду применяется к самосопряженной матрице $A \in M_n$.

Согласно (14.1), (14.2) $A^{(1)} = U_1AU_1^t$, где U_1 – унитарная матрица, т.е. $A^{(1)}$ и A – унитарно подобны. Следовательно, $A^{(1)}$ – самосопряженная матрица. Согласно (7), (8) на k -ом ($k = 1, \dots, n-2$) шаге алгоритма $A^{(k)} = U_k A^{(k-1)} U_k^t$, где U_k – унитарная матрица. Следовательно, $A^{(k)}$ и A унитарно подобны, и $A^{(k)}$ – самосопряженная матрица для всякого $k = 1, \dots, n-2$. Таким образом, $R = A^{(n-2)}$ – почти треугольная и самосопряженная, т.е. трехдиагональная матрица.

Запишем описанный выше процесс приведения самосопряженной матрицы к трехдиагональному виду так, чтобы максимально уменьшить объем вычислительной работы за счет использования симметрии.

Лемма 1. Для всякой матрицы отражения $U = U(x) \in M_n$ и всякой самосопряженной матрицы $A \in M_n$ матрица $B = UAU^* = UAU$ может быть вычислена за $2n^2 + O(n)$ умножений и столько же сложений.

Доказательство. Поскольку $U(x) = I - 2xx^*$, то

$$B = (I - 2xx^*)A(I - 2xx^*) = (I - 2xx^*)(A - 2Axx^*) = A - 2Axx^* - 2xx^*A + 4xx^*Axx^*.$$

Обозначим

$$y = Ax \in \mathbf{C}^n \tag{15}$$

В силу самосопряженности матрицы A имеем $y = Ax = A^*x = (x^*A)^*$,

$$4xx^*Axx^* = 2xx^*Axx^* + 2xx^*Axx^* = 2xx^*yx^* + 2xy^*xx^*$$

и

$$B = A - 2yx^* - 2xy^* + 2xx^*yx^* + 2xy^*xx^* = A - 2(I - xx^*)yx^* - 2xy^*(I - xx^*)$$

Обозначим

$$z = 2(I - xx^*)y = 2y - x(x^*y) = 2y - 2(x, y)x. \quad (16)$$

Тогда

$$B = A - zx^* - xz^* \quad (17)$$

После этих преобразований мы можем сформулировать алгоритм вычисления матрицы B :

1) Вычисляется вектор y по формуле (15). На это требуется $n^2 + O(n)$ мультипликативных операций и столько же аддитивных операций.

2) Вычисляется вектор z по формуле (16). На вычисление $\alpha = 2(x, y)$ – удвоенного евклидова скалярного произведения, требуется $n + O(1)$ мультипликативных операций и столько же аддитивных операций; на вычисление $z = 2y - \alpha x$ требуется $2n$ мультипликативных операций и n аддитивных операций. Общее число операций, необходимое для вычисления вектора z – $3n + O(1)$ мультипликативных и $2n + O(1)$ аддитивных операций.

3) Вычисляется матрица B по формуле (17). Матрица B как унитарно подобная A самосопряжена, поэтому по формуле (17) вычисляются только $n(n+1)/2$ элементов верхнего треугольника матрицы B . На вычисление каждого элемента матрицы B по формуле (17) надо выполнить 2 умножения и 2 вычитания, поэтому трудоемкость вычисления B по формуле (17) равна $n(n+1) = n^2 + O(n)$ мультипликативным и $n^2 + O(n)$ аддитивным операциям.

Таким образом, этот алгоритм требует $n^2 + O(n) + 3n + O(1) + n^2 + O(n) = 2n^2 + O(n)$ мультипликативных и столько же аддитивных операций. Лемма доказана.

Замечание 2. Для несамосопряженной матрицы A вычисление матрицы $B = UAU$ требует $4n^2 + O(n)$ умножений столько же сложений (см. лемму 13.11).

Обозначим $a_1 = (a_{21}, \dots, a_{n1})^t$. Согласно лемме 13.9 существует вектор $x^{(1)} \in \mathbf{C}^n$, равный

$$x^{(1)} = \pm \frac{a_1 - \|a_1\|e_1}{\|a_1 - \|a_1\|e_1\|},$$

такой, что $U(x^{(1)})a_1 = \|a_1\|e_1$, где $e_1 = (1, 0, \dots, 0) \in \mathbf{C}^{n-1}$, $U(x^{(1)}) \in \mathbf{M}_{n-1}$ – матрица отражения. Введем матрицу U_1 как в (2) и вычислим матрицу

$A^{(1)} = U_1 A U_1$. В силу самосопряженности матрицы $A^{(1)}$ вместо (14.2) для нее справедливо более точное равенство

$$A^{(1)} = U_1 A U_1 = \begin{pmatrix} a_{11} & \|a_1\| & 0 & \dots & 0 \\ \|a_1\| & a_{22}^{(1)} & a_{23}^{(1)} & \dots & a_{2n}^{(1)} \\ 0 & a_{32}^{(1)} & a_{33}^{(1)} & \dots & a_{3n}^{(1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & a_{n2}^{(1)} & a_{n3}^{(1)} & \dots & a_{nn}^{(1)} \end{pmatrix}. \quad (18)$$

Таким образом, у матрицы $A^{(1)}$ необходимо с помощью леммы 1 вычислить только подматрицу $(a_{ij}^{(1)})_{i,j=2,\dots,n} \in M_{n-1}$ (так как остальные элементы уже вычислены).

Пусть сделаны $k - 1$, $k = 1, \dots, n - 1$ шагов этого процесса, т.е. матрица преобразована к виду (14.3), где матрица $A^{(k-1)}$ имеет вид (14.14).

Введем обозначение (14.5). Согласно лемме 13.9 существует матрица отражения (4) такая, что выполнено (5). Определим U_k равенством (6). Вычислим матрицу

$$A^{(k)} = U_k A^{(k-1)} U_k. \quad (19)$$

Матрица $A^{(k)}$ унитарно подобна самосопряженной матрице $A^{(k-1)}$. Поэтому она самосопряжена и вместо (14.10) для нее справедливо более точное равенство (14.16). Таким образом, у матрицы $A^{(k)}$ необходимо с помощью леммы 1 вычислить только подматрицу $(a_{ij}^{(k)})_{i,j=k+1,\dots,n} \in M_{n-k-1}$ (так как остальные элементы уже вычислены).

После $n - 2$ шагов этого процесса (т.е. перехода от матриц (14.3), (14.14) к (19), (14.16)) матрица примет требуемый трехдиагональный вид (14.17).

Оценка количества арифметических операций в алгоритме приведения самосопряженной матрицы к трехдиагональному виду унитарным подобием методом отражений

Оценим трудоемкость k -го шага алгоритма, а затем просуммируем полученные оценки по всем $k = 1, \dots, n - 2$.

1. На вычисление матрицы $U(x_k)$ по формулам (4) требуется $2(n - k) + 1$ мультипликативных, $n - k + 1$ аддитивных операций и 2 операции извлечения корня (см. вычисления при оценке количества арифметических операций в алгоритме приведения матрицы к почти треугольному виду унитарным подобием методом отражений).

2. Компоненты $k + 1, \dots, n$ k -го столбца матрицы $A^{(k)}$, равные компонентам вектора $\|a_1^{(k-1)}\| e_1^{(n-k)}$, уже вычислены в (10). Столбец k вычисляется не по общим формулам (7) для сокращения количества арифметических операций и уменьшения вычислительной погрешности.

3. Поскольку в формуле (19)

$$(a_{ij}^{(k)})_{i=k+1,\dots,n, j=k+1,\dots,n} = U(x_k)(a_{ij}^{(k-1)})_{i=k+1,\dots,n, j=k+1,\dots,n} U(x_k), \quad (20)$$

то в силу леммы 1 на вычисление подматрицы (20) матрицы $A^{(k)}$ требуется $2(n - k)^2 + O(n - k)$ мультипликативных и столько же аддитивных операций.

Итак, на k -ом шаге алгоритма требуется выполнить $n - k + 1 + 2(n - k)^2 + O(n - k) = 2(n - k)^2 + O(n - k)$ мультипликативных операций, $n - k + 1 + 2(n - k)^2 + O(n - k) = 2(n - k)^2 + O(n - k)$ аддитивных операций и 2 операции извлечения корня.

Следовательно, всего для проведения алгоритма требуется выполнить

$$\sum_{k=1}^{n-2} (2(n - k)^2 + O(n - k)) = 2((n - 1)(n - 2)(2n - 3)/6) + O(n^2) = \frac{2}{3}n^3 + O(n^2) \quad (n \rightarrow \infty)$$

мультипликативных операций, столько же аддитивных операций и $2(n - 2)$ операций извлечения корня (которые по трудоемкости по порядку можно сравнить с операциями деления).

Таким образом, на приведение матрицы к почти треугольному виду унитарным подобием методом отражений требуется $\frac{2}{3}n^3 + O(n^2)$ ($n \rightarrow \infty$) мультипликативных операций и столько же аддитивных операций. Заметим, что это количество операций в два с половиной раза меньше, чем требуется для приведения произвольной матрицы к почти треугольному виду унитарным подобием методом отражений и совпадает количеством операций, необходимым для решения линейной системы методом отражений.

Теорема 2. *Всякая невырожденная самосопряженная матрица A может быть представлена в виде $A = Q R Q^t$, где матрица Q – унитарная, а матрица R – трехдиагональная.*

Доказательство Совпадает с доказательством теоремы 1.

Хранение матриц Q и R в памяти осуществляется одним из способов, изложенных при обсуждении алгоритма построения QR -разложения для матрицы A методом отражений.

Трудоемкость алгоритма построения описанного выше разложения складывается из количества арифметических операций, необходимых для проведения самого алгоритма, и количества арифметических операций, необходимых для построения матрицы Q . Подробные выкладки были проведены при обсуждении алгоритма построения QR -разложения методом отражений.

Глава II.

МЕТОДЫ НАХОЖДЕНИЯ СОБСТВЕННЫХ ЗНАЧЕНИЙ

§ 1. ТОЧНЫЕ И ИТЕРАЦИОННЫЕ МЕТОДЫ

Определение. Метод решения линейной системы называется **точным**, если при отсутствии округлений точное решение системы находится этим методом за конечное число арифметических операций (например, для метода Гаусса это $\frac{2}{3}n^3 + O(n^2)$).

На реальной вычислительной машине точный метод дает некоторое приближение к точному решению системы. Мера близости оценена в § I.3.

Все описанные выше методы являются точными.

Определение. Метод решения линейной системы называется **итерационным**, если он состоит в вычислении последовательности $\{x_k\}$, сходящейся к точному решению: $x_k \rightarrow x$ при $k \rightarrow \infty$. Итерационный метод за конечное число арифметических операций дает только некоторое приближение x_{k_0} к точному решению.

Теория итерационных методов будет изложена в курсе "Численные методы".

Определение. Метод нахождения собственных значений называется **итерационным**, если он состоит в вычислении последовательности $\{\lambda_k\}$, сходящейся к точному собственному значению: $\lambda_k \rightarrow \lambda$ при $k \rightarrow \infty$. Итерационный метод за конечное число арифметических операций дает только некоторое приближение λ_{k_0} к точному собственному значению.

Теорема 1. (Без доказательства.) *Не может существовать точного метода нахождения всех собственных значений произвольной матрицы $A \in M_n$ при $n \geq 5$. Другими словами, за конечное число арифметических операций нельзя найти все собственные значения произвольной матрицы $A \in M_n$ при $n \geq 5$.*

§ 2. ЛОКАЛИЗАЦИЯ СОБСТВЕННЫХ ЗНАЧЕНИЙ

Для матрицы $A \in \mathbf{M}_n$ обозначим

$$R'_i(A) = \sum_{\substack{j=1 \\ j \neq i}}^n |a_{ij}|,$$

$\sigma(A) = \{\lambda_1, \dots, \lambda_n\}$ – множество собственных значений.

Теорема 1 (Гершгорин). Для всякой матрицы $A \in \mathbf{M}_n$ справедливо соотношение:

$$\sigma(A) \subset \bigcup_{i=1}^n \{z \in \mathbf{C}^n : |z - a_{ii}| \leq R'_i(A)\}. \quad (1)$$

Кроме того, если объединение k , $1 \leq k \leq n$ из этих кругов есть связная область, не пересекающаяся с остальными $n - k$ кругами, то в ней находится ровно k собственных значений матрицы A .

Доказательство. Пусть λ – собственное значение матрицы A и $x = (x_1, \dots, x_n) \neq 0$ – соответствующий собственный вектор: $Ax = \lambda x$. Обозначим $|x_p| = \max_{i=1, \dots, n} |x_i| \neq 0$. Так как $Ax = \lambda x$, то

$$\lambda x_p = (\lambda x)_p = (Ax)_p = \sum_{j=1}^n a_{pj} x_j$$

и

$$x_p(\lambda - a_{pp}) = \sum_{\substack{j=1 \\ j \neq p}}^n a_{pj} x_j$$

Следовательно,

$$|x_p| |\lambda - a_{pp}| \leq \sum_{\substack{j=1 \\ j \neq p}}^n |a_{pj}| |x_j| \leq |x_p| \sum_{\substack{j=1 \\ j \neq p}}^n |a_{pj}|$$

и

$$|\lambda - a_{pp}| \leq \sum_{\substack{j=1 \\ j \neq p}}^n |a_{pj}|.$$

Таким образом, собственное значение $\lambda \in \sigma(A)$ принадлежит объединению кругов в (1).

Докажем второе утверждение теоремы. Положим $A = D + B$, где $D = \text{diag}(a_{11}, \dots, a_{nn}) \in \mathbf{M}_n$, $B = A - D$; $A_\varepsilon = D + \varepsilon B$, $\varepsilon > 0$. Тогда $R'_i(A_\varepsilon) = R'_i(\varepsilon B) = \varepsilon R'_i(A)$. Без ограничения общности мы можем считать что первые k

кругов в (1) образуют связную область, не пересекающуюся с остальными $n - k$ кругами. Обозначим

$$G_k(A) = \bigcup_{i=1}^k \{z \in \mathbf{C}^n : |z - a_{ii}| \leq R'_i(A)\},$$

$$G_{n-k}(A) = \bigcup_{i=k+1}^n \{z \in \mathbf{C}^n : |z - a_{ii}| \leq R'_i(A)\},$$

причем по условию

$$G_k(A) \cap G_{n-k}(A) = \emptyset. \quad (2)$$

Рассмотрим введенные множества для матрицы A_ε :

$$G_k(A_\varepsilon) = \bigcup_{i=1}^k \{z \in \mathbf{C}^n : |z - a_{ii}| \leq \varepsilon R'_i(A)\},$$

$$G_{n-k}(A_\varepsilon) = \bigcup_{i=k+1}^n \{z \in \mathbf{C}^n : |z - a_{ii}| \leq \varepsilon R'_i(A)\}.$$

Тогда для всех $\varepsilon \in [0, 1]$

$$G_k(A_\varepsilon) \subset G_k(A_1) = G_k(A), \quad G_{n-k}(A_\varepsilon) \subset G_{n-k}(A_1) = G_{n-k}(A). \quad (3)$$

причем множество $G_k(A_1) = G_k(A)$ связно по условию. В силу (2), (3) $G_k(A_\varepsilon) \cap G_{n-k}(A_\varepsilon) = \emptyset$ для всех $\varepsilon \in [0, 1]$. Обозначим через $\lambda_i(A_\varepsilon)$ i -е собственное значение матрицы A_ε , $\sigma(A_\varepsilon) = \{\lambda_1(A_\varepsilon), \dots, \lambda_n(A_\varepsilon)\}$ – спектр матрицы A_ε . По построению $\lambda_i(A_1) = \lambda_i(A) = \lambda_i$, $\lambda_i(A_0) = a_{ii}$.

Собственное значение $\lambda_i(A_\varepsilon)$ матрицы A_ε является корнем характеристического многочлена этой матрицы. Корни многочлена являются непрерывными функциями его коэффициентов. Следовательно, собственные значения $\lambda_i(A_\varepsilon)$ матрицы A_ε являются непрерывными функциями элементов этой матрицы, которые, в свою очередь, являются непрерывными функциями параметра ε . Таким образом, как композиции непрерывных функций, собственные значения $\lambda_i(A_\varepsilon)$ матрицы A_ε являются непрерывными функциями параметра ε .

По доказанному первому утверждению теоремы $\sigma(A_\varepsilon) \subset G_k(A_\varepsilon) \cup G_{n-k}(A_\varepsilon)$ для всех $\varepsilon \in [0, 1]$, т.е.

$$\lambda_i(A_\varepsilon) \in G_k(A_\varepsilon) \cup G_{n-k}(A_\varepsilon). \quad (4)$$

По доказанному выше

$$G_k(A_\varepsilon) \cap G_{n-k}(A_\varepsilon) = \emptyset \quad (5)$$

для всех $\varepsilon \in [0, 1]$, причем $\lambda_i(A_1) = \lambda_i(A) = \lambda_i$, $\lambda_i(A_0) = a_{ii}$. Следовательно,

$$\lambda_i(A_0) = a_{ii} \in G_k(A_0) \text{ для } 1 \leq i \leq k, \quad \lambda_i(A_0) = a_{ii} \in G_{n-k}(A_0) \text{ для } k + 1 \leq i \leq n. \quad (6)$$

Поскольку $\lambda_i(A_\varepsilon)$ – непрерывная функция и принимает все промежуточные значения, то из (4), (5), (6) вытекает, что

$$\lambda_i(A_\varepsilon) \in G_k(A_\varepsilon) \text{ для } 1 \leq i \leq k, \quad \lambda_i(A_\varepsilon) = a_{ii} \in G_{n-k}(A_\varepsilon) \text{ для } k+1 \leq i \leq n$$

для всех $\varepsilon \in [0, 1]$. При $\varepsilon = 1$ мы получаем второе утверждение теоремы.

§ 3. ОШИБКИ ПРИ НАХОЖДЕНИИ СОБСТВЕННЫХ ЗНАЧЕНИЙ

Пусть находятся собственные значения матрицы $A \in M_n$. Пусть \mathfrak{B} – алгоритм нахождения собственных значений, $\hat{\sigma}(A) = \mathfrak{B}(A)$ – полученный этим алгоритмом спектр матрицы A . Этот спектр не совпадает с истинным значением $\sigma(A)$ спектра матрицы A , поскольку в силу теоремы 1.1 найти все собственные значения матрицы A за конечное число арифметических операций невозможно. Обозначим через $A + E$ матрицу, спектром которой является набор $\hat{\sigma}(A)$, т.е. $\hat{\sigma}(A) = \sigma(A + E)$. Оценим погрешность при нахождении собственных значений (т.е. $\hat{\sigma}(A) - \sigma(A)$) через величину матрицы E .

Лемма 1. *Пусть $D = \text{diag}(\lambda_1, \dots, \lambda_n)$ – диагональная матрица, $E = (e_{ij}) \in M_n$. Тогда для всякого $\hat{\lambda} \in \sigma(D + E)$ – собственное значение матрицы $D + E$, существует $\lambda = \lambda_i \in \sigma(D)$ – собственное значение матрицы D , такое, что*

$$|\hat{\lambda} - \lambda| \leq \|E\|_\infty$$

(где $\|\cdot\|_\infty$ – максимальная строчная норма матрицы, см. стр. 9).

Доказательство. Применяя теорему Гершгорина к матрице $D+E$, находим, что существует такое i , $1 \leq i \leq n$, что

$$\hat{\lambda} \in \left\{ z \in \mathbf{C}^n : |z - \lambda_i - e_{ii}| \leq R'_i(E) = \sum_{\substack{j=1 \\ j \neq i}}^n |e_{ij}| \right\}.$$

Так как $|z - \lambda_i - e_{ii}| \geq |z - \lambda_i| - |e_{ii}|$, то

$$\hat{\lambda} \in \left\{ z \in \mathbf{C}^n : |z - \lambda_i| \leq \sum_{j=1}^n |e_{ij}| \right\}.$$

Но

$$\sum_{j=1}^n |e_{ij}| \leq \max_{i=1, \dots, n} \sum_{j=1}^n |e_{ij}| = \|E\|_\infty.$$

Поэтому

$$\hat{\lambda} \in \{z \in \mathbf{C}^n : |z - \lambda_i| \leq \|E\|_\infty\}$$

что доказывает утверждение леммы.

Теорема 1. Пусть $A \in \mathbf{M}_n$ – диагонализируемая матрица, т.е. существуют невырожденная матрица C и диагональная матрица $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_n)$ такие, что $A = C\Lambda C^{-1}$. Пусть $E = (e_{ij}) \in \mathbf{M}_n$. Тогда для всякого $\hat{\lambda} \in \sigma(A + E)$ – собственного значения матрицы $A + E$, существует $\lambda = \lambda_i \in \sigma(A)$ – собственное значение матрицы A , такое, что

$$|\hat{\lambda} - \lambda| \leq \kappa_\infty(C) \|E\|_\infty$$

где $\kappa_\infty(C)$ – число обусловленности матрицы C относительно максимальной строчной нормы матрицы (см. стр. 13).

Доказательство. Матрицы $A + E$ и $C^{-1}(A + E)C = \Lambda + C^{-1}EC$ подобны и потому имеют одинаковые собственные значения. В силу леммы 1 для всякого $\hat{\lambda} \in \sigma(A + E) = \sigma(\Lambda + C^{-1}EC)$ существует $\lambda = \lambda_i \in \sigma(\Lambda) = \sigma(A)$ такое, что

$$|\hat{\lambda} - \lambda| \leq \|C^{-1}EC\|_\infty \leq \|C^{-1}\|_\infty \|E\|_\infty \|C\|_\infty = \kappa_\infty(C) \|E\|_\infty.$$

Теорема доказана.

§ 4. СТЕПЕННОЙ МЕТОД

Степенной метод позволяет находить максимальное по модулю собственное значение и соответствующий ему собственный вектор диагонализируемой матрицы $A \in \mathbf{M}_n$.

§ 4.1. Описание алгоритма

Теорема 1 (Степенной метод). Пусть матрица $A \in \mathbf{M}_n$ имеет полную систему ортонормированных собственных векторов e_i , $i = 1, \dots, n$: $Ae_i = \lambda_i e_i$, $(e_i, e_j) = \delta_{ij}$, причем $|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \dots \geq |\lambda_n|$ (т.е. вектора занумерованы в порядке убывания модуля собственного значения, причем собственное значение с максимальным модулем – не кратное). Тогда для всякого вектора $x^{(0)} \in \mathbf{C}^n$ такого, что $(x^{(0)}, e_1) \neq 0$, итерационный процесс

$$x^{(k+1)} = Ax^{(k)}, \quad \lambda_1^{(k)} = \frac{(x^{(k+1)}, x^{(k)})}{(x^{(k)}, x^{(k)})}, \quad k = 0, 1, \dots \quad (1)$$

сходится к собственному значению λ_1 (собственному значению с максимальным модулем), причем

$$\lambda_1^{(k)} = \lambda_1 + O\left(\lambda_1 \left|\frac{\lambda_2}{\lambda_1}\right|^k\right) \quad (k \rightarrow \infty),$$

а величины $e_1^{(k)} = \frac{x^{(k)}}{\|x^{(k)}\|}$ сходятся к собственному вектору, соответствующему λ_1 (с точностью до постоянного множителя):

$$e_1^{(k)} = e^{i\varphi} e_1 + O\left(\left|\frac{\lambda_2}{\lambda_1}\right|^k\right),$$

где $e^{i\varphi}$ – число, по модулю равное 1.

Доказательство. Поскольку вектора e_1, \dots, e_n образуют базис в C^n , то $x^{(0)} = \sum_{i=1}^n c_i e_i$, причем по условию $c_1 = (x^{(0)}, e_1) \neq 0$. Вычислим

$$x^{(k)} = A^k x^{(0)} = A^k \left(\sum_{i=1}^n c_i e_i \right) = \sum_{i=1}^n c_i A^k e_i = \sum_{i=1}^n \lambda_i^k e_i.$$

Поэтому $x^{(k)} = c_1 \lambda_1^k e_1 + O(|\lambda_2^k|)$, $x^{(k+1)} = c_1 \lambda_1^{k+1} e_1 + O(|\lambda_2^{k+1}|)$. Далее, вычислим

$$\begin{aligned} \|x^{(k)}\| &= (x^{(k)}, x^{(k)}) = (c_1 \lambda_1^k e_1 + O(|\lambda_2^k|), c_1 \lambda_1^k e_1 + O(|\lambda_2^k|)) \\ &\quad = |c_1|^2 |\lambda_1|^{2k} + O(|\lambda_1^k| |\lambda_2^k|); \\ (x^{(k+1)}, x^{(k)}) &= (c_1 \lambda_1^{k+1} e_1 + O(|\lambda_2^{k+1}|), c_1 \lambda_1^k e_1 + O(|\lambda_2^k|)) \\ &\quad = \lambda_1 (|c_1|^2 |\lambda_1|^{2k} + O(|\lambda_1^k| |\lambda_2^k|)). \end{aligned}$$

Следовательно,

$$\begin{aligned} \lambda_1^{(k)} &= \frac{(x^{(k+1)}, x^{(k)})}{(x^{(k)}, x^{(k)})} = \frac{\lambda_1 (|c_1|^2 |\lambda_1|^{2k} + O(|\lambda_1^k| |\lambda_2^k|))}{|c_1|^2 |\lambda_1|^{2k} + O(|\lambda_1^k| |\lambda_2^k|)} = \lambda_1 \frac{1 + O\left(\frac{1}{|c_1|^2} \left|\frac{\lambda_2}{\lambda_1}\right|^k\right)}{1 + O\left(\frac{1}{|c_1|^2} \left|\frac{\lambda_2}{\lambda_1}\right|^k\right)} \\ &\quad = \lambda_1 + O\left(\lambda_1 \left|\frac{\lambda_2}{\lambda_1}\right|^k\right) \end{aligned}$$

Аналогично,

$$\begin{aligned} e_1^{(k)} &= \frac{x^{(k)}}{\|x^{(k)}\|} = \frac{c_1 \lambda_1^k e_1 + O(|\lambda_2^k|)}{\left(|c_1|^2 |\lambda_1|^{2k} + O(|\lambda_1^k| |\lambda_2^k|)\right)^{1/2}} = \frac{\frac{c_1 \lambda_1^k}{|c_1| |\lambda_1|^k} e_1 + O\left(\frac{1}{|c_1|} \left|\frac{\lambda_2}{\lambda_1}\right|^k\right)}{\left(1 + O\left(\frac{1}{|c_1|} \left|\frac{\lambda_2}{\lambda_1}\right|^k\right)\right)^{1/2}} \\ &\quad = e^{i\varphi} e_1 + O\left(\left|\frac{\lambda_2}{\lambda_1}\right|^k\right), \end{aligned}$$

где $e^{i\varphi} = \frac{c_1}{|c_1|} \left(\frac{\lambda_1}{|\lambda_1|}\right)^k$. Торема доказана.

Замечание 1. При реализации на реальной ЭВМ итерационный процесс (1) сходится, даже если условие $(x^{(0)}, e_1) \neq 0$ не выполнено. Дело в том, что из-за присутствия округлений для некоторого k_0 будет выполнено $(x^{(k_0)}, e_1) \neq 0$, даже если $(x^{(0)}, e_1) = 0$. После этого описанный выше процесс работает, как если бы его начали с $x^{(0)} = x^{(k_0)}$.

Замечание 2. Если $|\lambda_1| > 1$, то $\|x^{(k)}\| = c_1 \lambda_1^k e_1 + O(|\lambda_2^k|) \rightarrow \infty$ при $k \rightarrow \infty$; Если $|\lambda_1| < 1$, то $\|x^{(k)}\| = c_1 \lambda_1^k e_1 + O(|\lambda_2^k|) \rightarrow 0$ при $k \rightarrow \infty$. Поэтому, чтобы не произошло переполнения или потери точности, надо через какое-то количество m итераций нормировать вектор $x^{(k)}$ так, чтобы $\|x^{(k)}\| = 1$. Практически это осуществляется так: если k делится на m нацело, то после вычисления $x^{(k+1)}$ и $\lambda_1^{(k)}$ полагаем $x^{(k+1)} = \frac{x^{(k+1)}}{\|x^{(k+1)}\|}$. Далее процесс продолжается, как если бы мы его начали с $x^{(0)} = x^{(k+1)}$.

§ 4.2. Оценка количества арифметических операций на один шаг алгоритма

Для выполнения шага алгоритма (1) (т.е. вычисления вектора $x^{(k+1)}$ и величины $\lambda_1^{(k)}$) требуется вычислить:

1) вектор $x^{(k+1)} = Ax^{(k)}$; для этого требуется $n^2 + O(n)$ аддитивных и столько же мультипликативных опреаций (для вычисления произведения матрицы A на вектор $x^{(k)}$;

2) скалярное произведение $a = (x^{(k)}, x^{(k)})$; для этого требуется $n + O(1)$ аддитивных и столько же мультипликативных опреаций;

3) скалярное произведение $b = (x^{(k+1)}, x^{(k)})$; для этого требуется $n + O(1)$ аддитивных и столько же мультипликативных опреаций;

4) отношение $\lambda_1 = a/b$; для этого требуется 1 мультипликативная опреация.

Суммируя эти оценки, находим, что на один шаг алгоритма требуется $n^2 + O(n)$ аддитивных и столько же мультипликативных опреаций.

§ 5. МЕТОД ВРАЩЕНИЙ ЯКОБИ

Метод вращений Якоби позволяет находить все собственные значения симметричной вещественной матрицы $A \in \mathbf{M}_n$.

§ 5.1. Описание алгоритма

Определение. Для всякой матрицы $B = (b_{ij})$ положим

$$\Sigma(B) = \sum_{\substack{i,j \\ i \neq j}} |b_{ij}|^2$$

– сумма квадратов внедиагональных элементов матрицы B .

Пусть $A = (a_{ij}) \in \mathbf{M}_n(\mathbf{R})$, $A = A^* = A^t$. Всякая симметричная вещественная матрица диагонализируема в некотором евклидовом базисе, т.е. существует ортогональная матрица $\hat{O} \in O(n)$ и диагональная матрица $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_n)$ такие, что $A = \hat{O}\Lambda\hat{O}^*$. Отсюда $\Lambda = \hat{O}^*A\hat{O} = OAO^*$, где обозначено $O = \hat{O}^* \in O(n)$. Очевидно, что для диагональной матрицы $\Sigma(\Lambda) = 0$ и для всякой матрицы B $\Sigma(B) \geq 0$. Следовательно

- $\Sigma(OAO^*) \geq 0$ для всякой $O \in O(n)$;
- $\Sigma(OAO^*) = \Sigma(\Lambda) = 0$, если $O = \hat{O}^*$.

Следовательно, матрица $O = \hat{O}^*$ является решением задачи минимизации функционала $\Sigma(OAO^*)$ на группе ортогональных матриц:

$$\Sigma(OAO^*) \rightarrow \min_{O \in O(n)} .$$

Если мы найдем какое-то решение O_1 этой задачи, т.е. $\Sigma(O_1AO_1^*) = 0$, то матрица $\Sigma(O_1AO_1^*)$ диагональна и ортогонально подобна матрице A . Следовательно, на ее диагонали стоят искомые собственные значения матрицы A .

Будем строить последовательность симметричных матриц

$$A = A_0, A_1, \dots, A_k, \dots \quad (1)$$

такую, что для всякого $k = 1, 2, \dots$:

1) Следующая матрица ортогонально подобна предыдущей (и потому ортогонально подобна исходной)

$$A_k = O_k A_{k-1} O_k^*, \quad O_k \in O(n). \quad (2)$$

2) Сумма квадратов внедиагональных элементов следующей матрицы строго меньше суммы квадратов внедиагональных элементов предыдущей матрицы:

$$\Sigma(A_k) < \Sigma(A_{k-1}) \quad (3)$$

т.е. последовательность $\{\Sigma(A_k)\}_{k=1}^\infty$ строго монотонно убывает, что в силу $\Sigma(A_k) \geq 0$ гарантирует существование предела $\lim_{k \rightarrow \infty} \Sigma(A_k)$.

3) Этот предел равен нулю:

$$\lim_{k \rightarrow \infty} \Sigma(A_k) = 0. \quad (4)$$

Матрицы $O_k \in O(n)$ в (2) подбираются на шаге k так, чтобы удовлетворить условиям (3), (4).

Теорема 1. Пусть $\varepsilon > 0$ – произвольно. Тогда в процессе (1) – (4) существует $k = k_0$ такое, что $\Sigma(A_{k_0}) < \varepsilon$. При этом для всякого собственного значения λ матрицы A существует i , $1 \leq i \leq n$, такое, что $|\lambda - a_{ii}^{(k_0)}| < \sqrt{n-1}\sqrt{\varepsilon}$, где $(a_{ij}^{(k_0)})$ – элементы матрицы A_{k_0} . Другими словами, диагональ матрицы A_{k_0} с точностью $\sqrt{n-1}\sqrt{\varepsilon}$ представляет собой набор собственных значений матрицы A .

Доказательство. В силу (4) для всякого $\varepsilon > 0$ существует $k = k_0$, такое, что $0 \leq \Sigma(A_{k_0}) < \varepsilon$. В силу (2) всякое собственное значение λ матрицы A является собственным значением матрицы A_{k_0} .

По теореме Гершгорина для всякого λ – собственного значения матрицы A_{k_0} существует i , $1 \leq i \leq n$, такое, что

$$\begin{aligned} |\lambda - a_{ii}^{(k_0)}| &\leq R'_i(A_{k_0}) = \sum_{\substack{j=1 \\ j \neq i}}^n |a_{ij}^{(k_0)}| \leq \sqrt{\sum_{\substack{j=1 \\ j \neq i}}^n 1^2} \sqrt{\sum_{\substack{j=1 \\ j \neq i}}^n |a_{ij}^{(k_0)}|^2} \\ &\leq \sqrt{n-1} \sqrt{\Sigma(A_{k_0})} < \sqrt{n-1} \sqrt{\varepsilon} \end{aligned}$$

Теорема доказана.

В методе вращений Якоби В качестве ортогональных матриц O_k в (2) используются матрицы элементарного вращения (см. стр. 43): $O_k = T_{ij}^k = T_{ij}(\varphi_k)$. Угол φ подбирается так, чтобы удовлетворить (3), а индексы i и j – так, чтобы удовлетворить (4).

§ 5.2. Выбор угла вращения

Вычислим для произвольной симметричной матрицы A и произвольной матрицы элементарного вращения T_{ij} матрицу $B = T_{ij}AT_{ij}^t$ и выражение

$$\Sigma(B) - \Sigma(A) = \sum_{\substack{l,m=1 \\ l \neq m}}^n (b_{lm}^2 - a_{lm}^2).$$

При умножении A на T_{ij} слева изменяются только строки i и j матрицы A , при умножении $T_{ij}A$ на T_{ij}^t справа изменяются только столбцы i и j матрицы $T_{ij}A$. Поэтому все элементы, не находящиеся в строках i , j и столбцах i , j , у матриц A и $B = T_{ij}AT_{ij}^t$ совпадают: $b_{lm} = a_{lm}$ при $l \neq i, j$, $m \neq i, j$. Следовательно,

$$\begin{aligned} \Sigma(B) - \Sigma(A) &= \sum_{\substack{m=1 \\ m \neq i,j}}^n (b_{im}^2 - a_{im}^2) + \sum_{\substack{m=1 \\ m \neq i,j}}^n (b_{jm}^2 - a_{jm}^2) + \sum_{\substack{l=1 \\ l \neq i,j}}^n (b_{li}^2 - a_{li}^2) + \sum_{\substack{l=1 \\ l \neq i,j}}^n (b_{lj}^2 - a_{lj}^2) \\ &\quad + (b_{ij}^2 - a_{ij}^2) + (b_{ji}^2 - a_{ji}^2) \\ &= \sum_{\substack{m=1 \\ m \neq i,j}}^n ((b_{im}^2 + b_{jm}^2) - (a_{im}^2 + a_{jm}^2)) + \sum_{\substack{l=1 \\ l \neq i,j}}^n ((b_{li}^2 + b_{lj}^2) - (a_{li}^2 + a_{lj}^2)) + 2(b_{ij}^2 - a_{ij}^2) \end{aligned}$$

(здесь мы использовали симметричность матриц A и B : $a_{ij} = a_{ji}$, $b_{ij} = b_{ji}$ для всех $i, j = 1, \dots, n$). Без ограничения общности мы можем считать, что $i < j$. В силу строения матрицы T_{ij} (см. (I.12.1)) и правил умножения матриц получаем для всех $m, l = 1, \dots, n$, $m, l \neq i, j$:

$$\begin{pmatrix} b_{im} \\ b_{jm} \end{pmatrix} = \begin{pmatrix} \cos \varphi & -\sin \varphi \\ \sin \varphi & \cos \varphi \end{pmatrix} \begin{pmatrix} a_{im} \\ a_{jm} \end{pmatrix} \quad (5)$$

и

$$(b_{li}, b_{lj}) = (a_{li}, a_{lj}) \begin{pmatrix} \cos \varphi & -\sin \varphi \\ \sin \varphi & \cos \varphi \end{pmatrix}^t \quad (6)$$

Матрицы преобразования в (5) и (6) ортогональны и потому сохраняют длины (двумерных) векторов. Поэтому

$$\begin{aligned} b_{im}^2 + b_{jm}^2 &= a_{im}^2 + a_{jm}^2 & m = 1, \dots, n, \quad m \neq i, j; \\ b_{li}^2 + b_{lj}^2 &= a_{li}^2 + a_{lj}^2 & l = 1, \dots, n, \quad l \neq i, j. \end{aligned}$$

Следовательно,

$$\Sigma(B) - \Sigma(A) = 2(b_{ij}^2 - a_{ij}^2). \quad (7)$$

Это выражение будет минимально, когда $b_{ij} = 0$.

Определим угол вращения из уравнения $\varphi = 0$. Из выражения (7) вытекает, что достаточно рассмотреть 2×2 симметричные матрицы A и B :

$$B = \begin{pmatrix} b_{ii} & b_{ij} \\ b_{ij} & b_{jj} \end{pmatrix}, \quad A = \begin{pmatrix} a_{ii} & a_{ij} \\ a_{ij} & a_{jj} \end{pmatrix}, \quad T_{ij} = \begin{pmatrix} \cos \varphi & -\sin \varphi \\ \sin \varphi & \cos \varphi \end{pmatrix}, \quad B = T_{ij} A T_{ij}^t.$$

Вычислим

$$T_{ij} A = \begin{pmatrix} \cos \varphi a_{ii} - \sin \varphi a_{ij} & \cos \varphi a_{ij} - \sin \varphi a_{jj} \\ \sin \varphi a_{ii} + \cos \varphi a_{ij} & \sin \varphi a_{ij} + \cos \varphi a_{jj} \end{pmatrix}$$

и

$$\begin{aligned} b_{ij} &= \sin \varphi (\cos \varphi a_{ii} - \sin \varphi a_{ij}) + \cos \varphi (\cos \varphi a_{ij} - \sin \varphi a_{jj}) \\ &= \sin \varphi \cos \varphi a_{ii} - \sin^2 \varphi a_{ij} + \cos^2 \varphi a_{ij} - \sin \varphi \cos \varphi a_{jj} \\ &= \frac{1}{2} \sin 2\varphi (a_{ii} - a_{jj}) + \cos 2\varphi a_{ij} \end{aligned}$$

Из условия $b_{ij} = 0$ получаем уравнение

$$\frac{1}{2} \sin 2\varphi (a_{ii} - a_{jj}) + \cos 2\varphi a_{ij} = 0$$

откуда

$$\begin{cases} \operatorname{tg} 2\varphi = -\frac{2a_{ij}}{a_{ii} - a_{jj}}, & \text{если } a_{ii} \neq a_{jj}, \\ \varphi = \frac{\pi}{4}, & \text{если } a_{ii} = a_{jj}. \end{cases}$$

Будем выбирать $\varphi \in [-\frac{\pi}{4}, \frac{\pi}{4}]$. Тогда $\cos 2\varphi \geq 0$ и $\operatorname{sign}(\sin \varphi) = \operatorname{sign}(\operatorname{tg} 2\varphi)$. Следовательно,

$$\cos 2\varphi = \frac{1}{(1 + \operatorname{tg}^2 2\varphi)^{1/2}}$$

и

$$\begin{aligned} \cos \varphi &= \left(\frac{1}{2} \left(1 + \frac{1}{(1 + \operatorname{tg}^2 2\varphi)^{1/2}} \right) \right)^{1/2}, \\ \sin \varphi &= \operatorname{sign}(\operatorname{tg} 2\varphi) \left(\frac{1}{2} \left(1 - \frac{1}{(1 + \operatorname{tg}^2 2\varphi)^{1/2}} \right) \right)^{1/2} \end{aligned}$$

Обозначим $x = -2a_{ij}$, $y = a_{ii} - a_{jj}$. Тогда

$$\operatorname{tg} 2\varphi = \frac{x}{y}, \quad \cos 2\varphi = \frac{1}{\left(1 + \frac{x^2}{y^2}\right)^{1/2}} = \frac{|y|}{(x^2 + y^2)^{1/2}},$$

и

$$\cos \varphi = \left(\frac{1}{2} \left(1 + \frac{|y|}{(x^2 + y^2)^{1/2}} \right) \right)^{1/2}, \quad \sin \varphi = \operatorname{sign}(xy) \left(\frac{1}{2} \left(1 - \frac{|y|}{(x^2 + y^2)^{1/2}} \right) \right)^{1/2}.$$

Однако при $|y| \rightarrow \infty$ числа 1 и $\frac{|y|}{(x^2 + y^2)^{1/2}}$ могут оказаться близки и при вычислении их разности возникает большая вычислительная погрешность. Поэтому $\sin \varphi$ вычисляют по формуле

$$\begin{aligned} \sin \varphi &= \frac{\sin 2\varphi}{2 \cos \varphi} = \frac{\operatorname{tg} 2\varphi \cos 2\varphi}{2 \cos \varphi} = \frac{\frac{x}{y} \frac{|y|}{(x^2 + y^2)^{1/2}}}{2 \cos \varphi} \\ &= \frac{x \operatorname{sign}(y)}{2 \cos \varphi (x^2 + y^2)^{1/2}} = \frac{\operatorname{sign}(xy)|x|}{2 \cos \varphi (x^2 + y^2)^{1/2}} \end{aligned}$$

Окончательно, расчетные формулы имеют вид:

$$\begin{aligned} \cos \varphi &= \frac{1}{\sqrt{2}}, & \sin \varphi &= \frac{1}{\sqrt{2}} && \text{при } y = 0 \\ \cos \varphi &= \left(\frac{1}{2} \left(1 + \frac{|y|}{(x^2 + y^2)^{1/2}} \right) \right)^{1/2}, & \sin \varphi &= \frac{\operatorname{sign}(xy)|x|}{2 \cos \varphi (x^2 + y^2)^{1/2}} && \text{при } y \neq 0 \end{aligned} \quad (8)$$

(где $x = -2a_{ij}$, $y = a_{ii} - a_{jj}$).

Таким образом, для выполнения условия (3) достаточно выбрать произвольный внедиагональный элемент $a_{ij}^{(k-1)} \neq 0$ и сделать преобразование $T_{ij}(\varphi)$ с углом φ , определенным по приведенным выше формулам.

§ 5.3. Стратегии выбора обнуляемого элемента

Обеспечим выполнение условия (4) за счет выбора номера (i, j) обнуляемого элемента $a_{ij}^{(k-1)}$. Это можно сделать несколькими способами, которые называют *стратегиями выбора обнуляемого элемента*. Именно стратегия выбора обнуляемого элемента в значительной степени определяет трудоемкость алгоритма метода вращений Якоби.

§ 5.3.1. Метод вращений с выбором максимального элемента

В качестве $a_{ij}^{(k-1)}$ выбираем максимальный по модулю внедиагональный элемент матрицы A_{k-1} :

$$|a_{ij}^{(k-1)}| = \max_{\substack{l,m=1 \\ l \neq m}} |a_{lm}^{(k-1)}| \quad (9)$$

Лемма 1. При выборе (9) обнуляемого элемента условие (4) выполнено.

Доказательство. При любой стратегии выбора элемента $a_{ij}^{(k-1)}$ из (7) следует, что

$$\Sigma(A_k) = \Sigma(A_{k-1}) - |a_{ij}^{(k-1)}|^2. \quad (10)$$

Пусть $a_{ij}^{(k-1)}$ выбран как (9). Тогда

$$\Sigma(A_{k-1}) = \sum_{\substack{l,m=1 \\ l \neq m}}^n |a_{ml}^{(k-1)}|^2 \leq \sum_{\substack{l,m=1 \\ l \neq m}}^n |a_{ij}^{(k-1)}|^2 = n(n-1)|a_{ij}^{(k-1)}|^2.$$

Следовательно,

$$|a_{ij}^{(k-1)}|^2 \geq \frac{1}{n(n-1)} \Sigma(A_{k-1}).$$

Поэтому из (10)

$$\begin{aligned} \Sigma(A_k) &\leq \Sigma(A_{k-1}) - \frac{2}{n(n-1)} \Sigma(A_{k-1}) = \left(1 - \frac{2}{n(n-1)}\right) \Sigma(A_{k-1}) \\ &\leq \left(1 - \frac{2}{n(n-1)}\right) \left(1 - \frac{2}{n(n-1)}\right) \Sigma(A_{k-2}) \leq \dots \leq \left(1 - \frac{2}{n(n-1)}\right)^k \Sigma(A_0). \end{aligned}$$

Поскольку $q = 1 - \frac{2}{n(n-1)} \in (0, 1)$, то $q^k \rightarrow 0$ при $k \rightarrow \infty$, и

$$\Sigma(A_k) \leq q^k \Sigma(A_0) \rightarrow 0 \quad \text{при } k \rightarrow \infty.$$

Следствие 1. При $k \rightarrow \infty$ диагональные элементы матрицы A_k сходятся к собственным значениям матрицы A . Доказательство следует из теоремы 1.

Оценка количества арифметических операций на один шаг алгоритма

Трудоемкость алгоритма складывается из трудоемкости построения матрицы $T_{ij}(\varphi)$, трудоемкости умножения матрицы A на $T_{ij}(\varphi)$ слева и $T_{ij}^t(\varphi)$ справа (не зависят от стратегии выбора обнуляемого элемента), а также трудоемкости выбора очередного обнуляемого элемента.

- 1) На выбор $a_{ij}^{(k-1)}$ по формуле (9) в силу симметричности матрицы A требуется $\frac{n(n-1)}{2}$ операций сравнения, которые по порядку сложности равны аддитивным операциям.
- 2) На построение матрицы $T_{ij}(\varphi)$ (т.е. определение $\cos \varphi$ и $\sin \varphi$) по формулам (8) требуется не зависящее от n число операций (т.е. $O(1)$).
- 3) На умножение матрицы A на $T_{ij}(\varphi)$ слева согласно лемме I.12.5 требуется $4n$ умножений и $2n$ сложений.
- 4) На умножение матрицы $T_{ij}(\varphi)A$ на $T_{ij}^t(\varphi)$ справа согласно лемме I.12.5 требуется $4n$ умножений и $2n$ сложений.

Следовательно, всего на вычисление матрицы $T_{ij}(\varphi)AT_{ij}^t(\varphi)$ требуется $8n + O(1)$ аддитивных и $4n + O(1)$ мультипликативных операций; на один шаг алгоритма требуется $8n + O(1)$ аддитивных, $4n + O(1)$ мультипликативных и $\frac{n(n-1)}{2}$ операций сравнения (которые по порядку можно сравнить с аддитивными операциями).

§ 5.3.2. Метод вращений с циклическим выбором обнуляемого элемента

Внедиагональные элементы матрицы нумеруются в следующем порядке:

$$a_{12}, a_{13}, \dots, a_{1n}, a_{23}, a_{24}, \dots, a_{2n}, a_{34}, \dots, a_{3n}, \dots, a_{n-1,n},$$

и в качестве обнуляемого элемента на шаге k выбирается элемент с номером $k \pmod{\frac{n(n-1)}{2}}$ из этого списка. Другими словами, в качестве номера (i, j) обнуляемого элемента последовательно выбирают

$$(1, 2), (1, 3), \dots, (1, n), (2, 3), (2, 4), \dots, (2, n), (3, 4), \dots, (3, n), \dots, (n-1, n).$$

Затем этот цикл повторяется, и так до тех пор, пока на некотором шаге k_0 не будет выполнено условие $\Sigma(A_{k_0}) < \varepsilon$.

Опыт показывает, что обычно нужно не более 5, 6 таких циклов для достижения максимально возможной на данной ЭВМ точности (т.е. машинной точности).

Оценка количества арифметических операций на один шаг алгоритма

В этой стратегии не требуется дополнительных вычислений для определения очередного обнуляемого элемента. Следовательно, трудоемкость шага алгоритма равна трудоемкости вычисления матрицы $T_{ij}(\varphi)AT_{ij}^t(\varphi)$, т.е. требуется $8n + O(1)$

аддитивных и $4n + O(1)$ мультипликативных операций. Однако, из-за ненаправленного выбора обнуляемого элемента этот алгоритм требует значительного числа итераций для достижения требуемой точности.

§ 5.3.3. Метод вращений с выбором оптимального элемента

В качестве обнуляемого элемента $a_{ij}^{(k-1)}$ выбираем максимальный по модулю внедиагональный элемент в строке матрицы A_{k-1} , имеющей максимальную сумму модулей внедиагональных элементов. Другими словами, i определяется из условия

$$\sum_{\substack{j=1 \\ j \neq i}}^n |a_{ij}^{(k-1)}|^2 = \max_{l=1, \dots, n} \sum_{\substack{j=1 \\ j \neq l}}^n |a_{lj}^{(k-1)}|^2,$$

а j — из условия

$$|a_{ij}^{(k-1)}| = \max_{\substack{m=1, \dots, n \\ m \neq i}} |a_{im}^{(k-1)}|. \quad (11)$$

Для уменьшения вычислительных расходов поступают следующим образом. Кроме матрицы A_k хранится вектор $b^{(k)}$, компоненты которого на k -м шаге равны

$$b_l^{(k)} = \sum_{\substack{j=1 \\ j \neq l}}^n |a_{lj}^{(k)}|^2, \quad (12)$$

Поскольку при переходе от матрицы A_{k-1} к $A_k = T_{ij} A_{k-1} T_{ij}^t$ суммы квадратов внедиагональных элементов строки l , $l \neq i, j$ не изменяются (см. вычисления при получении формулы (7)), то при переходе от A_{k-1} к $A_k = T_{ij} A_{k-1} T_{ij}^t$ надо заново пересчитывать по формуле (12) только числа $b_i^{(k)}$ и $b_j^{(k)}$: для остальных компонент вектора $b^{(k)}$ справедливо равенство

$$b_l^{(k)} = b_l^{(k-1)}, \quad l = 1, \dots, n, \quad l \neq i, j. \quad (13)$$

С использованием вектора $b^{(k-1)}$ выбор очередного обнуляемого элемента $a_{ij}^{(k-1)}$ осуществляется следующим образом: i определяется из условия

$$|b_i^{(k-1)}| = \max_{l=1, \dots, n} |b_l^{(k-1)}| \quad (14)$$

а j — из условия (11).

Оценка количества арифметических операций на один шаг алгоритма

- 1) На выбор очередного обнуляемого элемента $a_{ij}^{(k-1)}$ требуется n сравнений в формуле (14) и $n - 1$ сравнение в формуле (11).
- 2) На вычисление матрицы $T_{ij}(\varphi) A T_{ij}^t(\varphi)$ требуется $8n + O(1)$ аддитивных и $4n + O(1)$ мультипликативных операций (см. вычисления выше).

3) На вычисление вектора $b^{(k)}$ по формуле (12) (при i и j) и формуле (13) (для остальных компонент) требуется $2(n-1)$ умножение и $2(n-2)$ сложение.

Таким образом, на шаг алгоритма требуется $10n+O(1)$ аддитивных, $6n+O(1)$ мультипликативных и $2n+O(1)$ операций сравнения (которые по порядку можно сравнить с аддитивными операциями).

§ 6. МЕТОД БИСЕКЦИИ

Метод бисекции позволяет находить для произвольной действительной симметричной матрицы:

- 1) k -е по величине собственное значение,
- 2) все собственные значения на заданном интервале,
- 3) все собственные значения.

Метод основывается на следующей теореме:

Теорема 1. (Без доказательства.) Пусть $A \in \mathbf{M}_n(\mathbf{R})$, $A = A^*$, $A = (a_{ij})$ — невырожденная матрица,

$$A_k = \begin{pmatrix} a_{11} & \dots & a_{1k} \\ \vdots & \ddots & \vdots \\ a_{k1} & \dots & a_{kk} \end{pmatrix} \in M_k$$

и $\delta_k = \det A_k$ — главные угловые миноры матрицы A . Тогда количество отрицательных собственных значений матрицы A равно $S(A)$ — числу перемен знака в последовательности $1, \delta_1, \delta_2, \dots, \delta_n$.

Положим $n_-(\lambda) = S(A - \lambda I)$. Тогда по теореме 1 $n_-(0) = S(A)$ — число отрицательных собственных значений матрицы A , $n_-(\lambda)$ — число собственных значений матрицы A , меньших λ (поскольку собственные значения матрицы $A - \lambda I$ равны собственным значениям матрицы A минус λ). Следовательно, $n_-(\lambda_2) - n_-(\lambda_1)$ равно числу собственных значений матрицы A , принадлежащих интервалу (λ_1, λ_2) (здесь λ_1, λ_2 не могут быть выбраны равными собственным значениям матрицы A , так как тогда матрицы $A - \lambda_1 I$, $A - \lambda_2 I$ вырождены и теорема 1 не применима).

§ 6.1. Алгоритм вычисления k -го по величине собственного значения методом бисекции

Пусть требуется найти k -е по величине собственное значение матрицы A с точностью $\varepsilon > 0$.

- 1) Зададимся $b_0 > a_0$ такими, что $n_-(a_0) < k$, $n_-(b_0) \geq k$ (например, возьмем $a_0 = -\|A\|_\infty$, $b_0 = \|A\|_\infty$, тогда в силу леммы I.1.4 $n_-(a_0) = 0$, $n_-(b_0) = n$). В силу сказанного выше на интервале $(-\infty, a_0)$ находится меньше k собственных значений, на интервале (b_0, ∞) — больше k собственных значений. Следовательно, $\lambda_k \in (a_0, b_0)$.
- 2) До тех пор, пока $b_i - a_i > \varepsilon$ ($i = 0, 1, \dots$) будем вычислять $c_{i+1} = \frac{a_i + b_i}{2}$. Если $n_-(c_{i+1}) < k$, то полагаем $a_{i+1} = c_{i+1}$, $b_{i+1} = b_i$, иначе полагаем $a_{i+1} = a_i$, $b_{i+1} = c_{i+1}$.
- 3) Переходим к пункту 2.
- 4) По окончании этого процесса (т.е. $b_i - a_i < \varepsilon$) в качестве ответа берем $\lambda_k = \frac{a_i + b_i}{2}$ (которое является $(n_-(b_i) - n_-(a_i))$ -кратным собственным значением).

На каждом шаге $i = 0, 1, \dots$ выполнено $\lambda_k \in (a_i, b_i)$, при этом длина интервала $b_i - a_i = \frac{b_0 - a_0}{2^i}$, $i = 0, 1, \dots$. Следовательно, для достижения точности ε (т.е. $b_i - a_i < \varepsilon$) надо сделать число шагов m , определяемое из условия $\frac{b_0 - a_0}{2^m} < \varepsilon$, т.е. $(b_0 - a_0)\varepsilon^{-1} < 2^m$, $m > \log_2(b_0 - a_0)\varepsilon^{-1} = \log_2((b_0 - a_0)\varepsilon^{-1}) = \log_2(b_0 - a_0) + \log_2\varepsilon^{-1}$.

§ 6.2. Алгоритм вычисления всех собственных значений на заданном интервале методом бисекции

Пусть требуется найти все собственные значения матрицы A на интервале $[a, b]$ с точностью $\varepsilon > 0$.

Существует несколько способов организовать последовательность вычислений.

§ 6.2.1. Рекурсивный алгоритм

В рекурсивной форме алгоритм формулируется наиболее просто и при правильной реализации работает наиболее быстро.

Алгоритм определения всех собственных значений на отрезке $[a, b]$ с точностью ε :

Если $b - a > \varepsilon$ и $n_-(b) - n_-(a) \neq 0$ (т.е. число собственных значений на $[a, b]$ не равно 0), то

определить все собственные значения на отрезке $[a, \frac{a+b}{2}]$ с точностью ε ;

определить все собственные значения на отрезке $[\frac{a+b}{2}, b]$ с точностью ε .

иначе, если $n_-(b) - n_-(a) \neq 0$, то $\frac{a+b}{2}$ является $(n_-(b) - n_-(a))$ -кратным собственным значением, определенным с точностью ε .

иначе на отрезке $[a, b]$ нет собственных значений.

Однако, поскольку глубина рекурсии может достигать n и алгоритм обычно применяется к трехдиагональным матрицам, размерность которых может быть очень велика (сотни тысяч), то **рекурсию в этом алгоритме следует организовывать программным путем**.

§ 6.2.2. Алгоритм последовательного поиска собственных значений

В этой формулировке алгоритм работает медленнее, чем предыдущий, но не требует рекурсии.

Пусть $n_-(a) = k_1$, $n_-(b) = k_2$ (т.е. на отрезке $[a, b]$ находится $k_2 - k_1 \neq 0$ собственных значений $\lambda_{k_1+1}, \dots, \lambda_{k_2}$ с учетом кратности). Положим $\lambda_{k_1} = a$.

Для всех $k = k_1 + 1, \dots, k_2$ будем находить k -е по величине собственное значение, используя алгоритм § 6.1, который начинаем с $a_0 = \lambda_{k-1}$, $b_0 = b$.

§ 6.3. Алгоритм вычисления всех собственных значений методом бисекции

Положим $a = -\|A\|_\infty$, $b = \|A\|_\infty$, тогда в силу леммы I.1.4 $n_-(a) = 0$, $n_-(b_0) = n$, и на отрезке $[a, b]$ находятся все собственные значения матрицы A . Для их нахождения применяем алгоритм из § 6.2.

§ 6.4. Вычисление числа перемен знака в последовательности главных миноров

Для реализации этих алгоритмов надо уметь быстро вычислять функцию $n_-(\lambda) = S(A - \lambda I)$ — число перемен знака в последовательности $1, \delta_1, \delta_2, \dots, \delta_n$ главных миноров матрицы $A - \lambda I$. Эта функция может быть вычислена достаточно быстро только для трехдиагональных матриц. Поэтому исходную симметричную матрицу A перед началом алгоритма метода бисекции приводят к трехдиагональному виду унитарным подобием одним из описанных выше способов (см. § I.14.2 и § I.15.2).

Для вычисления функции $n_-(\lambda)$ для трехдиагональных симметричных матриц существуют несколько способов.

§ 6.4.1. Вычисление числа перемен знака в последовательности главных миноров с помощью LU-разложения

Пусть требуется вычислить число перемен знака в последовательности $1, \delta_1, \delta_2, \dots, \delta_n$ главных миноров трехдиагональной матрицы A (в алгоритме $A := A - \lambda I$).

Построим для матрицы A LU-разложение (см. § I.5.2, стр. 23). В силу определения перемножения матриц $A_k = L_k U_k$, где A_k, L_k, U_k — соответственно главные подматрицы матриц A, L, U . Следовательно, $\delta_k = \det A_k = \det L_k \det U_k$. В силу вида (I.5.2) треугольных матриц L и U получаем $\det L_k = \prod_{j=1}^n l_{jj}$, $\det U_k = 1$. Поэтому $\delta_k = l_{11} \dots l_{kk}$ и число перемен знака в последовательности $1, \delta_1, \delta_2, \dots, \delta_n$ равно числу перемен знака в последовательности $1, l_{11}, l_{22}, \dots, l_{nn}$.

Само LU-разложение матрицы A не строится, находятся лишь знаки l_{ii} . Память под матрицы L и U не выделяется, так как согласно формулам (I.5.3) для вычисления очередных элементов $l_{ii}, l_{i+1,i}, u_{i,i+1}$ нужно знать лишь элементы $l_{i-1,i-1}, l_{i,i-1}, u_{i-1,i}$.

Согласно доказанному в § I.5.2 для построения LU-разложения требуется $n - 1$ аддитивных и $2(n - 1)$ мультипликативных операций. Поскольку для вычисления числа перемен знака требуется еще не более n сложений (для суммирования знаков l_{ii}), то на вычисление функции $S(A)$ требуется $2n + O(1)$ аддитивных и столько же мультипликативных операций.

§ 6.4.2. Вычисление числа перемен знака в последовательности главных миноров с помощью рекуррентных формул

Пусть требуется вычислить число перемен знака в последовательности $1, \delta_1, \delta_2, \dots, \delta_n$ главных миноров трехдиагональной симметричной матрицы A (в алгоритме $A := A - \lambda I$):

$$A = \begin{pmatrix} a_1 & b_1 & & & \\ b_1 & a_2 & b_2 & & \\ & b_2 & a_3 & \ddots & \\ & \ddots & \ddots & b_{n-2} & \\ & & & b_{n-2} & a_{n-1} & b_{n-1} \\ & & & & b_{n-1} & a_n \end{pmatrix}.$$

Имеем: $\delta_1 = a_1$, $\delta_2 = a_1 a_2 - b_1^2$. Пусть для некоторого $k = 3, \dots, n - 1$ $\delta_k = \det A_k$, $\delta_{k-1} = \det A_{k-1}$ уже вычислены. Вычислим $\delta_{k+1} = \det A_{k+1}$. Разложим

$\det A_{k+1}$ по последнему столбцу:

$$\delta_{k+1} = a_{k+1}\delta_k - b_k \det \begin{pmatrix} a_1 & b_1 & & & \\ b_1 & a_2 & b_2 & & \\ & b_2 & a_3 & \ddots & \\ & \ddots & \ddots & b_{k-2} & \\ & & b_{k-2} & a_{k-1} & b_{k-1} \\ & & & 0 & b_k \end{pmatrix} = a_{k+1}\delta_k - b_k^2\delta_{k-1}.$$

Эта формула позволяет вычислить все δ_k , но может приводить к переполнению или потере точности. Поскольку требуются не сами числа δ_k , а только их знаки, то эти формулы домножают на специально подобранные множители так, чтобы все числа δ_k были бы не слишком велики или малы.

У матриц A и αA собственные значения различаются на множитель α и потому при $\alpha > 0$ числа перемен знака в последовательности главных миноров совпадают. Возьмем

$$\alpha = 4 \max \left\{ \max_{i=1,\dots,n} \{|a_i|\}, \max_{i=1,\dots,n-1} \{|b_i|\} \right\}$$

и будем рассматривать матрицу $A := \alpha^{-1}A$. У этой матрицы все элементы по модулю не превышают $1/4$. Если оказалось, что при некотором i элемент $|b_i| < \varepsilon_{mash}$ (где ε_{mash} — машинная точность для данной ЭВМ), то полагаем $b_i = 0$. При этом матрица распадается на две подматрицы, объединение наборов собственных значений которых дает набор собственных значений исходной матрицы. Поэтому мы можем считать, что все $|b_i| > \varepsilon_{mash}$.

Запишем для так преобразованной матрицы алгоритм вычисления числа перемен знака m в последовательности главных миноров.

- 1) Полагаем $x = a_1$, $y = 1$. Если $\text{sign}(xy) < 0$, число перемен знака $m = 1$, иначе $m = 0$.
- 2) Для всех $k = 2, 3, \dots, n$ вычисляем:
 - a) $a = a_k$, $b = b_{k-1}$;
 - б) $\gamma = (1/\varepsilon_{mash}) / \max\{|x|, |b(by)|\}$;
 - в) $u = \gamma(ax - b^2y)$, $v = \gamma x$;
 - г) если $\text{sign}(ux) < 0$, то $m = m + 1$;
 - д) $x = u$, $y = v$.

В этих формулах в начале k -го шага x равен δ_{k-1} , умноженному на некоторое положительное число γ_k , y равен δ_{k-2} , умноженному на γ_k . Множитель γ подбирается для обеспечения максимальной вычислительной устойчивости.

§ 7. LR АЛГОРИТМ

LR алгоритм позволяет находить все собственные значения матрицы $A \in \mathbf{M}_n$.

§ 7.1. LR-разложение, используемое в *LR* алгоритме

Теорема 1. (О *LR*-разложении) *Если все главные угловые миноры матрицы $A \in \mathbf{M}_n$ отличны от нуля, то матрица A допускает представление $A = LR$, где $L \in \text{LT}(n)$ с 1 на главной диагонали, $R \in \text{RT}(n)$.*

Доказательство. Для матрицы A^t все главные угловые миноры отличны от нуля. По теореме I.4.1 для матрицы A^t осуществимо *LU*-разложение $A^t = \widehat{L}\widehat{U}$, где $\widehat{L} \in \text{LT}(n)$, $\widehat{U} \in \text{RT}(n)$ с 1 на главной диагонали. Следовательно, $A = \widehat{U}^t\widehat{L}^t \equiv LR$, где $L = \widehat{U}^t \in \text{LT}(n)$ с 1 на главной диагонали, $R = \widehat{L}^t \in \text{RT}(n)$. Теорема доказана.

§ 7.1.1. Алгоритм построения *LR*-разложения для произвольной матрицы

Алгоритм построения *LR*-разложения матрицы $A \in \mathbf{M}_n$ очень похож на алгоритм построения *LU*-разложения (см. стр. 19).

Пусть требуется найти нижнюю треугольную матрицу $L = (l_{ij})$ с единицами на главной диагонали и верхнюю треугольную матрицу $R = (r_{ij})$ такую, что $A = LR$, т.е.

$$\sum_{j=1}^n l_{ij} r_{jk} = a_{ik}, \quad i, k = 1, \dots, n. \quad (1)$$

Поскольку $l_{ij} = 0$ при $i < j$, $r_{jj} = 1$, $r_{jk} = 0$ при $j > k$, то (1) есть система из n^2 уравнений относительно $n(n-1)/2$ неизвестных l_{ij} , $i \geq j$ и $n(n+1)/2$ неизвестных r_{jk} , $j < k$, всего $n(n+1)/2 + n(n-1)/2 = n^2$ неизвестных. Получим формулы для решения системы (1), которые и составляют алгоритм нахождения *LR*-разложения.

В силу $l_{ij} = 0$ при $i < j$, $r_{jk} = 0$ при $j > k$ сумма в (1) имеет вид

$$\sum_{j=1}^{\min\{i,k\}} l_{ij} r_{jk} = a_{ik}, \quad i, k = 1, \dots, n,$$

или

$$\begin{cases} \sum_{j=1}^i l_{ij} r_{jk} = a_{ik}, & k \geq i, \quad i, k = 1, \dots, n, \\ \sum_{j=1}^k l_{ij} r_{jk} = a_{ik}, & k < i, \quad i, k = 1, \dots, n. \end{cases}$$

Выделим в первой из этих сумм отдельно случай $i = 1$, а во второй - случай $k = 1$, и учтем, что $l_{ii} = 1$ для всех $i = 1, \dots, n$,

$$\begin{cases} r_{1k} = a_{1k}, & k = 1, \dots, n, \\ \sum_{j=1}^{i-1} l_{ij} r_{jk} + r_{ik} = a_{ik}, & k \geq i > 1, \quad i, k = 2, \dots, n. \\ l_{i1} r_{11} = a_{i1}, & i = 2, \dots, n, \\ \sum_{j=1}^{k-1} l_{ij} r_{jk} + l_{ik} r_{kk} = a_{ik}, & 1 < k < i, \quad i, k = 2, \dots, n, \end{cases}$$

Перегруппируем эти формулы:

$$\begin{cases} r_{1k} = a_{1k}, & k = 1, \dots, n, \\ l_{i1} = a_{i1}/r_{11}, & i = 2, \dots, n, \\ r_{ik} = a_{ik} - \sum_{j=1}^{i-1} l_{ij} r_{jk}, & k \geq i > 1, \quad i, k = 2, \dots, n, \\ l_{ik} = (a_{ik} - \sum_{j=1}^{k-1} l_{ij} r_{jk})/r_{kk}, & 1 < k < i, \quad i, k = 2, \dots, n. \end{cases} \quad (2)$$

Процесс вычислений по этим формулам строится следующим образом: вначале по первой из формул (2) вычисляются неизвестные элементы первой строки матрицы R : r_{1k} , $k = 1, \dots, n$, затем по второй из формул (2) вычисляются неизвестные элементы первого столбца матрицы L : l_{i1} , $i = 2, \dots, n$, (напомним, элемент l_{11} известен, он равен 1). Далее в вычислениях участвуют только третья и четвертая из формул (2). По третьей формуле (2) вычисляются неизвестные элементы второй строки матрицы R : r_{2k} , $k = 2, \dots, n$ (напомним, $r_{21} = 0$, так как R -верхняя треугольная)

$$r_{2k} = a_{2k} - l_{21} r_{1k}, \quad k = 2, \dots, n.$$

По четвертой формуле (2) вычисляются неизвестные элементы второго столбца матрицы L : l_{i2} , $i = 3, \dots, n$ (напомним, $l_{12} = 0$, так как L -нижняя треугольная, $l_{22} = 1$, так как L имеет единичную главную диагональ)

$$l_{i2} = (a_{i2} - l_{i1} r_{12})/r_{22}, \quad i = 3, \dots, n.$$

Затем по третьей формуле (2) вычисляются неизвестные элементы третьей строки матрицы R : r_{3k} , $k = 3, \dots, n$ и так далее. а по четвертой формуле (2) вычисляются неизвестные элементы третьего столбца матрицы L : l_{i3} , $i = 4, \dots, n$, и так далее.

Замечание 1. Организация хранения матриц L и R в памяти. Формулы (2) таковы, что при вычислении элемента l_{ij} или r_{ij} используются значения элемента a_{ij} и вычисленных ранее элементов l_{km} , $m < j$ и r_{km} , $k < i$. Это позволяет хранить нижнюю треугольную матрицу L (без единичной главной диагонали) на месте нижнего треугольника матрицы A : $l_{ij} \equiv a_{ij}$, $i > j$, $i, j = 1, \dots, n$, а верхнюю треугольную матрицу R — на месте верхнего треугольника матрицы A : $r_{ij} \equiv a_{ij}$, $i \leq j$, $i, j = 1, \dots, n$.

Оценка количества арифметических операций в алгоритме построения LR-разложения

1. При фиксированном $i = 1, \dots, n$ вычисление элементов r_{ik} для всех $k = i, \dots, n$ по третьей формуле (2) требует $\sum_{k=i}^n (i-1) = (n-i+1)(i-1)$ мультипликативных и столько же аддитивных операций. Следовательно, вычисление всех элементов матрицы R требует $\sum_{i=1}^n (n-i+1)(i-1) = n \sum_{i=1}^n (i-1) - \sum_{i=1}^n (i-1)^2 = n \sum_{j=0}^{n-1} j - \sum_{j=0}^{n-1} j^2 = n^2(n-1)/2 - (n-1)n(2n-1)/6 = n^3/2 - n^3/3 + O(n^2) = n^3/6 + O(n^2)$ ($n \rightarrow \infty$) мультипликативных и столько же аддитивных операций.

2. При фиксированном $k = 1, \dots, n$ вычисление элементов l_{ik} для всех $i = k+1, \dots, n$ по четвертой формуле (2) требует $\sum_{i=k+1}^n k = (n-k)k$ мультипликативных и $\sum_{i=k+1}^n (k-1) = (n-k)(k-1)$ аддитивных операций. Следовательно, вычисление всех элементов матрицы R требует $\sum_{k=1}^n (n-k)k = n^2(n+1)/2 - n(n+1)(2n+1)/6 = n^3/6 + O(n^2)$ ($n \rightarrow \infty$) мультипликативных и $\sum_{i=1}^n (n-k)(k-1) = n^2(n-1)/2 - n(n+1)(2n+1)/6 + n(n+1)/2 = n^3/6 + O(n^2)$ ($n \rightarrow \infty$) аддитивных операций.

Таким образом, алгоритм построения LR-разложения требует для своего проведения выполнения $n^3/3 + O(n^2)$ ($n \rightarrow \infty$) мультипликативных и столько же аддитивных операций, а в сумме — $(2/3)n^3 + O(n^2)$ ($n \rightarrow \infty$) арифметических операций.

§ 7.1.2. Алгоритм построения LR-разложения для почти треугольной матрицы

Рассмотрим случай, когда матрица $A \in \mathbf{M}_n$ в приведенном выше алгоритме почти треугольная. Из определения произведения матриц вытекает, что матрица L в LR-разложении будет двухдиагональной:

$$L = \begin{pmatrix} 1 & & & & \\ l_{21} & 1 & & & \\ & l_{32} & 1 & & \\ & & \ddots & \ddots & \\ & & & l_{n,n-1} & 1 \end{pmatrix}. \quad (3)$$

Формулы (2), следовательно, примут вид

$$\begin{cases} r_{1k} = a_{1k}, & k = 1, \dots, n, \\ \begin{cases} r_{ik} = a_{ik} - l_{i,i-1} r_{i-1,k}, & k \geq i > 1, \\ l_{i,i-1} = a_{i,i-1} / r_{i-1,i-1}, \end{cases} & i, k = 2, \dots, n, \\ i = 2, \dots, n. \end{cases} \quad (4)$$

Алгоритм построения LR-разложения по этим формулам требует для своего проведения выполнения $n^2/2 + O(n)$ ($n \rightarrow \infty$) мультипликативных и столько же аддитивных операций.

§ 7.1.3. Алгоритм построения LR-разложения для трехдиагональной матрицы

Рассмотрим случай, когда матрица $A \in \mathbf{M}_n$ в приведенном выше алгоритме трехдиагональная. Из определения произведения матриц вытекает, что матрицы L и R в LR -разложении будут двухдиагональными:

$$L = \begin{pmatrix} 1 & & & & \\ l_{21} & 1 & & & \\ & l_{32} & 1 & & \\ & & \ddots & \ddots & \\ & & & l_{n,n-1} & 1 \end{pmatrix}, \quad R = \begin{pmatrix} r_{11} & r_{12} & & & \\ & r_{22} & r_{23} & & \\ & & 1 & \ddots & \\ & & & \ddots & r_{n-1,n} \\ & & & & r_{nn} \end{pmatrix}. \quad (5)$$

Формулы (4), следовательно, примут вид

$$\left[\begin{array}{ll} r_{1k} = a_{1k}, & k = 1, 2, \\ \left[\begin{array}{lll} r_{ik} = a_{ik} - l_{i,i-1} r_{i-1,k}, & k = i, i+1, & i = 2, \dots, n, \\ l_{i,i-1} = a_{i,i-1}/r_{i-1,i-1}, & & i = 2, \dots, n. \end{array} \right. \end{array} \right. \quad (6)$$

Алгоритм построения LR -разложения по этим формулам требует для своего проведения выполнения $3n + O(1)$ ($n \rightarrow \infty$) мультипликативных и $n + O(1)$ ($n \rightarrow \infty$) аддитивных операций.

§ 7.2. LR алгоритм нахождения собственных значений

Будем строить для матрицы $A \in \mathbf{M}_n$ последовательность $\{A_k\}$ матриц $A_k \in \mathbf{M}_n$ по следующим правилам:

- 1) $A_1 = A$;
- 2) для всех $k = 1, 2, \dots$ матрица A_{k+1} получается из матрицы A_k следующим образом:
 - a) строим LR -разложение матрицы A_k : $A_k = L_k R_k$,
 - б) вычисляем матрицу A_{k+1} как произведение матриц R_k и L_k : $A_{k+1} = R_k L_k$.

Здесь мы предполагаем, что для каждого $k = 1, 2, \dots$ LR -разложение матрицы A_k существует, т.е. для нее выполнены условия теоремы 1. Если это не так, то алгоритм не применим.

Лемма 1. Для всех $k = 1, 2, \dots$ матрица A_k подобна A .

Доказательство. Имеем: $A_{k+1} = R_k L_k = (L_k^{-1} L_k) R_k L_k = L_k^{-1} (L_k R_k) L_k = L_k^{-1} A_k L_k$. Следовательно, матрица A_{k+1} подобна A_k . Поскольку $A_1 = A$, то по индукции получаем, что A_k подобна A для всех $k = 1, 2, \dots$, причем $A_{k+1} = L_1^{-1} \dots L_k^{-1} A_1 L_k \dots L_1 = (L_k \dots L_1)^{-1} A (L_k \dots L_1)$.

Следствие 1. Матрицы A_k , $k = 1, 2, \dots$ имеют те же собственные значения, что и матрица A .

Теорема 2. (Без доказательства.) Пусть матрица $A \in \mathbf{M}_n$ такова, что на каждом шаге $k = 1, 2, \dots$ LR-алгоритма осуществимо LR-разложение для матрицы A_k , и собственные значения $\{\lambda_i\}$ матрицы A таковы, что

$$|\lambda_1| > |\lambda_2| > \dots > |\lambda_n|.$$

Тогда $L_k \rightarrow I$ при $k \rightarrow \infty$, $R_k \rightarrow A_k$ при $k \rightarrow \infty$ (по норме в пространстве матриц). Тем самым диагональные элементы матрицы $A_k = (a_{ij}^{(k)})$ сходятся к собственным значениям матрицы A , причем в правильном порядке:

$$a_{ii}^{(k)} \rightarrow \lambda_i \quad \text{при } k \rightarrow \infty, \quad i = 1, 2, \dots, n.$$

Скорость сходимости матрицы A_k к треугольной дается соотношением

$$a_{ij}^{(k)} = O\left(\left|\frac{\lambda_i}{\lambda_j}\right|^k\right) \quad \text{при } k \rightarrow \infty, \quad i > j.$$

Применение алгоритма к матрице $A \in \mathbf{M}_n$ произвольного вида требует слишком большого числа арифметических операций: $(2/3)n^3 + O(n^2)$ ($n \rightarrow \infty$) на построение LR-разложения матрицы A_k и не более $n^3 + O(n^2)$ ($n \rightarrow \infty$) на вычисление матрицы A_{k+1} как произведения двух треугольных матриц. Поэтому LR-алгоритм **никогда** не применяется к матрицам произвольного вида.

§ 7.2.1. LR алгоритм нахождения собственных значений для почти треугольной матрицы

Лемма 2. Если матрица A — почти треугольная, то все матрицы A_k , $k = 1, 2, \dots$ в LR-алгоритме — почти треугольные.

Доказательство. Матрица $A_1 = A$ — почти треугольная. Предположим, что матрица A_k — почти треугольная. Тогда в LR-разложении $A_k = L_k R_k$ матрица L_k имеет вид (3) (т.е. является двухдиагональной), $R_k \in \text{RT}(n)$. Из определения произведения матриц вытекает, что матрица $A_{k+1} = R_k L_k$ является почти треугольной матрицей. Лемма доказана.

Эта лемма позволяет значительно ускорить работу LR-алгоритма. Перед его применением исходная матрица A приводится к почти треугольному виду A' унитарным подобием одним из алгоритмов, описанных в § I.14 и § I.15. Затем к матрице A' применяется LR-алгоритм.

Алгоритм вычисления произведения матриц R и L

Произведение матриц $A = RL$, где $R \in \text{RT}(n)$, L имеет вид (3), может быть вычислено значительно быстрее, чем произведение произвольных треугольных матриц. По определению произведения матриц

$$\begin{cases} a_{ik} = r_{ik} + r_{i,k+1}l_{k+1,k}, & k = i, i+1, \dots, n-1, \quad i = 1, 2, \dots, n \\ a_{in} = r_{in}, & i = 1, 2, \dots, n \\ a_{i,i-1} = r_{ii}l_{i,i-1}, & i = 2, 3, \dots, n \end{cases} \quad (7)$$

Вычисление произведения $A = RL$ по этим формулам требует $n^2/2 + O(n)$ ($n \rightarrow \infty$) мультипликативных и столько же аддитивных операций.

Оценка количества арифметических операций на один шаг LR-алгоритма для почти треугольной матрицы

1) Построение LR -разложения матрицы $A_k = L_k R_k$ по формулам (4) требует $n^2/2 + O(n)$ ($n \rightarrow \infty$) мультипликативных и столько же аддитивных операций.

2) Вычисление произведения $A_{k+1} = R_k L_k$ по формулам (7) требует $n^2/2 + O(n)$ ($n \rightarrow \infty$) мультипликативных и столько же аддитивных операций.

Следовательно, один шаг алгоритма для почти треугольной матрицы требует $n^2 + O(n)$ ($n \rightarrow \infty$) мультипликативных и столько же аддитивных операций.

§ 7.2.2. LR алгоритм нахождения собственных значений для трехдиагональной матрицы

Лемма 3. Если матрица A — трехдиагональная, то все матрицы A_k , $k = 1, 2, \dots$ в LR -алгоритме — трехдиагональные.

Доказательство. Матрица $A_1 = A$ — трехдиагональная. Предположим, что матрица A_k — трехдиагональная. Тогда в LR -разложении $A_k = L_k R_k$ матрицы L_k и R_k имеют вид (5) (т.е. являются двухдиагональными). Из определения произведения матриц вытекает, что матрица $A_{k+1} = R_k L_k$ является трехдиагональной матрицей. Лемма доказана.

Эта лемма позволяет значительно ускорить работу LR -алгоритма для самосопряженной матрицы. Перед его применением исходная матрица A приводится к трехдиагональному виду A' унитарным подобием одним из алгоритмов, описанных в § I.14 и § I.15. Затем к матрице A' применяется LR -алгоритм.

Замечание 2. LR -алгоритм сохраняет трехдиагональный вид матрицы, но не ее самосопряженность. Другими словами, если матрица A_k была самосопряженной, то после шага алгоритма матрица A_{k+1} может не быть самосопряженной.

Алгоритм вычисления произведения матриц R и L

Произведение матриц $A = RL$, где R и L имеют вид (5), может быть вычислено значительно быстрее, чем произведение произвольных треугольных матриц. По определению произведения матриц

$$\begin{cases} a_{ii} = r_{ii} + r_{i,i+1}l_{i+1,i}, & i = 1, 2, \dots, n-1 \\ a_{i,i+1} = r_{i,i+1}, & i = 1, 2, \dots, n-1 \\ a_{i,i-1} = r_{ii}l_{i,i-1}, & i = 2, 3, \dots, n \\ a_{nn} = r_{nn} \end{cases} \quad (8)$$

Вычисление произведения $A = RL$ по этим формулам требует $2n + O(1)$ ($n \rightarrow \infty$) мультипликативных и $n + O(1)$ ($n \rightarrow \infty$) аддитивных операций.

Оценка количества арифметических операций на один шаг LR -алгоритма для трехдиагональной матрицы

1) Построение LR -разложения матрицы $A_k = L_k R_k$ по формулам (6) требует $3n + O(1)$ ($n \rightarrow \infty$) мультипликативных и $n + O(1)$ ($n \rightarrow \infty$) аддитивных операций.

2) Вычисление произведения $A_{k+1} = R_k L_k$ по формулам (7) требует $2n + O(1)$ ($n \rightarrow \infty$) мультипликативных и $n + O(1)$ ($n \rightarrow \infty$) аддитивных операций.

Следовательно, один шаг алгоритма для трехдиагональной матрицы требует $5n + O(1)$ ($n \rightarrow \infty$) мультипликативных и $2n + O(1)$ ($n \rightarrow \infty$) аддитивных операций.

§ 7.3. Ускорение сходимости алгоритма

Рассмотрим способы, применяемые для ускорения сходимости последовательности матриц $\{A_k\}$ к треугольной матрице. Эти способы одинаковы как для LR -алгоритма, так и для рассматриваемых ниже алгоритма Холецкого и QR -алгоритма нахождения собственных значений. Поскольку все эти алгоритмы **никогда** не применяются для матриц произвольного вида, всюду ниже мы будем считать, что исходная матрица уже приведена унитарным подобием к почти треугольному или трехдиагональному виду. Таким образом, начальная матрица A_1 — почти треугольная (или трехдиагональная). По доказанному выше это означает, что все матрицы A_k — почти треугольные (трехдиагональные).

Замечание 3. Описанные ниже приемы не только ускоряют сходимость алгоритмов нахождения собственных значений, но и расширяют множество матриц, для которых они сходятся. Другими словами, при использовании этих приемов алгоритмы нахождения собственных значений часто работают для матриц, для которых не выполнены условия приведенных теорем о сходимости алгоритмов.

Замечание 4. Без использования описанных ниже приемов скорость сходимости алгоритмов нахождения собственных значений оказывается весьма низкой, а множество матриц, для которых они применимы, достаточно узким. Поэтому эти приемы **всегда** применяются при вычислениях по этим алгоритмам.

§ 7.3.1. Исчерпывание матрицы

На k -ом шаге матрица A_k имеет вид

$$A_k = \begin{pmatrix} a_{11}^{(k)} & a_{12}^{(k)} & \dots & a_{1,i}^{(k)} & \dots & a_{1,n-2}^{(k)} & a_{1,n-1}^{(k)} & a_{1n}^{(k)} \\ a_{21}^{(k)} & a_{22}^{(k)} & \dots & a_{2,i}^{(k)} & \dots & a_{2,n-2}^{(k)} & a_{2,n-1}^{(k)} & a_{2n}^{(k)} \\ a_{32}^{(k)} & \ddots & \vdots & \dots & \vdots & \vdots & \vdots & \vdots \\ \ddots & a_{i,i}^{(k)} & \dots & a_{i,n-2}^{(k)} & a_{i,n-1}^{(k)} & a_{i,n}^{(k)} \\ a_{i+1,i}^{(k)} & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \ddots & a_{n-2,n-2}^{(k)} & a_{n-2,n-1}^{(k)} & a_{n-2,n}^{(k)} \\ a_{n-1,n-2}^{(k)} & a_{n-1,n-1}^{(k)} & a_{n-1,n}^{(k)} \\ a_{n,n-1}^{(k)} & a_{nn}^{(k)} \end{pmatrix}$$

Если для некоторого i , $i = 1, 2, \dots, n - 1$ выполнено условие $|a_{i+1,i}^{(k)}| < \varepsilon \|A_k\|_\infty$, где ε — точность, с которой требуется найти собственные значения, то полагаем $a_{i+1,i}^{(k)} = 0$. Поскольку собственные значения являются непрерывными функциями элементов матрицы, то собственные значения матрицы A_k изменятся при этой замене на величину порядка ε . Новая матрица A_k имеет блочную структуру:

$$A_k = \begin{pmatrix} a_{11}^{(k)} & a_{12}^{(k)} & \dots & a_{1,i}^{(k)} & a_{1,i+1}^{(k)} & \dots & a_{1,n-2}^{(k)} & a_{1,n-1}^{(k)} & a_{1n}^{(k)} \\ a_{21}^{(k)} & a_{22}^{(k)} & \dots & a_{2,i}^{(k)} & a_{2,i+1}^{(k)} & \dots & a_{2,n-2}^{(k)} & a_{2,n-1}^{(k)} & a_{2n}^{(k)} \\ a_{32}^{(k)} & \ddots & \vdots & \vdots & \vdots & \dots & \vdots & \vdots & \vdots \\ \ddots & a_{i,i}^{(k)} & a_{i,i+1}^{(k)} & \dots & a_{i,n-2}^{(k)} & a_{i,n-1}^{(k)} & a_{i,n}^{(k)} \\ 0 & a_{i+1,i+1}^{(k)} & \dots & a_{i+1,n-2}^{(k)} & a_{i+1,n-1}^{(k)} & a_{i+1,n}^{(k)} \\ a_{i+2,i+1}^{(k)} & \ddots & \vdots \\ \ddots & a_{n-2,n-2}^{(k)} & a_{n-2,n-1}^{(k)} & a_{n-2,n}^{(k)} \\ a_{n-1,n-2}^{(k)} & a_{n-1,n-1}^{(k)} & a_{n-1,n}^{(k)} \\ a_{n,n-1}^{(k)} & a_{nn}^{(k)} \end{pmatrix}$$

и ее множество собственных значений равно объединению множеств собственных значений подматриц $(a_{jl}^{(k)})_{j,l=1,2,\dots,i} \in \mathbf{M}_i$ и $(a_{jl}^{(k)})_{j,l=i+1,i+2,\dots,n} \in \mathbf{M}_{n-i}$. Следовательно, можно применить алгоритм к каждой из этих подматриц по отдельности.

Часто для уменьшения вычислительных затрат вместо условия $|a_{i+1,i}^{(k)}| < \varepsilon \|A_k\|_\infty$ используют условие $|a_{i+1,i}^{(k)}| < \varepsilon \|A_1\|_\infty$.

§ 7.3.2. Сдвиги

Для матрицы $A - \lambda I$ собственные значения равны $\lambda_i - \lambda$, λ_i – собственные значения матрицы A . Скорость сходимости поддиагональных элементов $a_{i+1,i}^{(k)}$ к нулю для этой матрицы по теореме о сходимости алгоритма есть $O\left(\left|\frac{\lambda_{i+1} - \lambda}{\lambda_i - \lambda}\right|^k\right)$.

Если λ близко к λ_{i+1} , то скорость сходимости элемента $a_{i+1,i}^{(k)}$ к нулю будет очень высокой.

Модифицированный *LR*-алгоритм, основанный на этой идее, выглядит следующим образом.

Будем строить для матрицы $A \in \mathbf{M}_n$ последовательность $\{A_k\}$ матриц $A_k \in \mathbf{M}_n$ по следующим правилам:

- 1) $A_1 = A$;
- 2) для всех $k = 1, 2, \dots$ матрица A_{k+1} получается из матрицы A_k следующим образом:
 - а) определяем требуемый сдвиг s_k (его оптимальный выбор – отдельная задача),
 - б) строим *LR*-разложение матрицы $A_k - s_k I$: $A_k - s_k I = L_k R_k$,
 - в) вычисляем матрицу A_{k+1} как произведение матриц R_k и L_k плюс $s_k I$: $A_{k+1} = R_k L_k + s_k I$.

Здесь мы предполагаем, что для каждого $k = 1, 2, \dots$ *LR*-разложение матрицы $A_k - s_k I$ существует, т.е. для нее выполнены условия теоремы 1. Если это не так, то надо изменить значение сдвига s_k . На практике условия теоремы 1 не проверяют, а выполняют алгоритм построения *LR*-разложения. Если в алгоритме требуется осуществить деление на 0, то немного изменяют значение сдвига s_k и заново выполняют алгоритм построения *LR*-разложения.

Нетрудно проверить, что матрица A_{k+1} подобна A_k : $A_{k+1} = R_k L_k + s_k I = (L_k^{-1} L_k)(R_k L_k + s_k I) = L_k^{-1}(L_k R_k)L_k + s_k L_k I = L_k^{-1}(L_k R_k + s_k I)L_k = L_k^{-1} A_k L_k$ и, следовательно, все матрицы A_k , $k = 1, 2, \dots$ имеют те же собственные значения, что и матрица A .

§ 7.3.3. Практическая организация вычислений в *LR* алгоритме

Пусть требуется определить все собственные значения матрицы $A \in \mathbf{M}_n$ с точностью ε .

Вначале приводим матрицу к почти треугольному виду A_1 унитарным подобием одним из алгоритмов, описанных в § I.14 и § I.15.

Затем к матрице A_1 применяем LR -алгоритм со сдвигами. На шаге k в качестве сдвига s_k возьмем $a_{nn}^{(k)}$, т.е. $s_k = a_{nn}^{(k)}$. Поскольку $a_{nn}^{(k)} \rightarrow \lambda_n$, то s_k является приближением к λ_n и скорость сходимости к нулю элемента $a_{n,n-1}^{(k)}$ будет очень высокой. Как только на некотором шаге k будет выполнено условие $|a_{n,n-1}^{(k)}| < \varepsilon \|A\|_\infty$, в качестве λ_n берем $a_{nn}^{(k)}$ и применяем алгоритм к подматрице $(a_{ij})_{i,j=1,2,\dots,n-1} \in M_{n-1}$ на 1 меньшей размерности. Так поступаем до тех пор, пока размерность матрицы не станет равной 2. Для этой матрицы собственные значения определяются как решения соответствующего квадратного уравнения.

§ 8. МЕТОД ХОЛЕЦКОГО

Метод Холецкого позволяет находить все собственные значения самосопряженной положительно определенной матрицы $A \in M_n$ за вдвое меньшее количество арифметических операций, чем LR -алгоритм.

§ 8.1. Разложение Холецкого, используемое в методе Холецкого

Теорема 1. (О разложении Холецкого) *Если матрица $A = A^* > 0$ — самосопряженная положительно определенная, то она допускает представление $A = LL^*$, где матрица L — нижняя треугольная с вещественными положительными элементами на главной диагонали.*

Доказательство. Согласно теореме I.10.1 и замечаниям I.10.2, I.10.3 матрица A допускает представление в виде $A = R^*R$, где R — верхняя треугольная матрица с вещественными положительными элементами на главной диагонали. Положим $L = R^* \in LT(n)$. Тогда $A = LL^*$ и матрица L — нижняя треугольная с вещественными положительными элементами на главной диагонали. Теорема доказана.

§ 8.1.1. Алгоритм построения разложения Холецкого для произвольной самосопряженной матрицы

Алгоритм построения разложения Холецкого самосопряженной матрицы $A \in M_n$ очень похож на алгоритм построения разложения Холецкого, используемого в методе Холецкого решения линейных систем с самосопряженной матрицей (см. стр. 36).

Пусть для самосопряженной матрицы $A = (a_{ij})$ ($A^* = A$, т.е. $\bar{a_{ij}} = a_{ji}$) требуется найти нижнюю треугольную матрицу $L = (l_{ij})$ с вещественными положительными элементами на главной диагонали ($l_{ii} > 0$ для всех $i = 1, \dots, n$), такую, что $A = LL^*$, т.е.

$$\sum_{k=1}^n l_{ik} \bar{l}_{jk} = a_{ij}, \quad i, j = 1, \dots, n. \quad (1)$$

Поскольку матрицы A и LL^* – самосопряженные, то уравнение с номером (j, i) получается из уравнения с номером (i, j) путем комплексного сопряжения и не дает ничего нового. Поэтому система (1) эквивалентна системе

$$\sum_{k=1}^n l_{ik}\overline{l_{jk}} = a_{ij}, \quad i \leq j, \quad i, j = 1, \dots, n. \quad (2)$$

Таким образом, (2) представляет собой систему из $n(n+1)/2$ уравнений с $n(n+1)/2$ неизвестными l_{ij} , $i \geq j$ (напомним, $L \in \text{LT}(n)$ и $l_{ij} = 0$ при $i < j$, при этом $l_{kk} > 0$). Получим формулы для решения системы (2), которые и составляют алгоритм нахождения разложения Холецкого.

Перепишем (2) в виде

$$\sum_{k=1}^i l_{ik}\overline{l_{jk}} + \sum_{k=i+1}^n l_{ik}\overline{l_{jk}} = a_{ij}, \quad i \leq j, \quad i, j = 1, \dots, n. \quad (3)$$

Поскольку матрица L – нижняя треугольная, то $l_{ik} = 0$ при $i < k$ и вторая из сумм в (3) равна нулю. Следовательно, система (2) эквивалентна следующей

$$\sum_{k=1}^{i-1} l_{ik}\overline{l_{jk}} + l_{ii}\overline{l_{ji}} = a_{ij}, \quad i \leq j, \quad i, j = 1, \dots, n,$$

(здесь считается, что сумма вида $\sum_{k=1}^{i-1}$ равна нулю, если верхний предел суммирования меньше нижнего; это позволяет не рассматривать отдельно случай $i = 1$). Применим к этим уравнениям комплексное сопряжение и учтем, что $\overline{a_{ij}} = a_{ji}$ и l_{ii} – вещественный элемент:

$$\sum_{k=1}^{i-1} \overline{l_{ik}}l_{jk} + l_{ii}l_{ji} = a_{ji}, \quad i \leq j, \quad i, j = 1, \dots, n,$$

Выделим в здесь отдельно случай $i = j$:

$$\begin{cases} l_{ii}^2 = a_{ii} - \sum_{k=1}^{i-1} |l_{ik}|^2 & i = 1, \dots, n, \\ l_{ii}l_{ji} = a_{ji} - \sum_{k=1}^{i-1} \overline{l_{ik}}l_{jk} & i < j, \quad i, j = 1, \dots, n, \end{cases}$$

Отсюда получаем расчетные формулы:

$$\begin{cases} l_{ii} = \sqrt{a_{ii} - \sum_{k=1}^{i-1} |l_{ik}|^2} & i = 1, \dots, n, \\ l_{ji} = (a_{ji} - \sum_{k=1}^{i-1} \overline{l_{ik}}l_{jk})/l_{ii} & i < j, \quad i, j = 1, \dots, n. \end{cases} \quad (4)$$

Процесс вычислений по этим формулам строится следующим образом: вначале вычисляются неизвестные элементы первого столбца матрицы L :

$$l_{11} = \sqrt{a_{11}}, \quad l_{j1} = a_{j1}/l_{11}, \quad j = 2, \dots, n;$$

потом по формулам (4) при $i = 2$ вычисляются неизвестные элементы второго столбца матрицы L :

$$\begin{aligned} l_{22} &= \sqrt{a_{22} - |l_{21}|^2}, \\ l_{j2} &= (a_{j2} - \bar{l}_{21}l_{j1})/l_{22}, \quad j = 3, \dots, n; \end{aligned}$$

затем по формулам (4) при $i = 3$ вычисляются неизвестные элементы третьего столбца матрицы L и так далее.

Замечание 1. Организация хранения матриц A и L в памяти. Поскольку для самосопряженной матрицы $a_{ji} = \bar{a}_{ij}$, то можно вместо всей матрицы A хранить только ее нижний треугольник: a_{ij} , $i \geq j$, $i, j = 1, \dots, n$. Формулы (4) таковы, что при вычислении элемента l_{ji} используются значения элемента a_{ji} и вычисленных ранее элементов l_{mk} , $k < i$. Это позволяет хранить нижнюю треугольную матрицу L на месте нижнего треугольника матрицы A : $l_{ji} \equiv a_{ji}$, $i \geq j$, $i, j = 1, \dots, n$.

Оценка количества арифметических операций в алгоритме построения разложения Холецкого

Вычисление элемента l_{ii} по формулам (4) требует одной операции извлечения корня и $i - 1$ мультипликативных и столько же аддитивных операций. При фиксированном $i = 1, \dots, n$ вычисление элементов l_{ji} для всех $j = i + 1, \dots, n$ по формулам (4) требует $1 + \sum_{j=i+1}^n (i-1) = (n-i)(i-1)+1$ мультипликативных и $\sum_{j=i+1}^n (i-1) = (n-i)(i-1)$ аддитивных операций. Следовательно, вычисление всех элементов матрицы L требует n операций извлечения корня, $\sum_{i=1}^n ((n-i)(i-1) + (i-1) + 1) = n^3/6 + O(n^2)$ ($n \rightarrow \infty$) мультипликативных и $\sum_{i=1}^n ((n-i)(i-1) + (i-1)) = n^3/6 + O(n^2)$ ($n \rightarrow \infty$) аддитивных операций (подробное вычисление см. при подсчете количества арифметических операций для метода Холецкого решения линейных систем).

§ 8.1.2. Алгоритм построения разложения Холецкого для трехдиагональной матрицы

Рассмотрим случай, когда самосопряженная матрица $A \in \mathbf{M}_n$ в приведенном выше алгоритме трехдиагональная. Из определения произведения матриц вытекает, что матрица L в разложении Холецкого $A = LL^*$ будет двухдиагональной:

$$A = \begin{pmatrix} a_{11} & a_{12} & & & \\ a_{21} & a_{22} & a_{23} & & \\ & a_{32} & a_{33} & \ddots & \\ & & \ddots & \ddots & a_{n-1,n} \\ & & & a_{n,n-1} & a_{nn} \end{pmatrix}, \quad L = \begin{pmatrix} l_{11} & & & & \\ l_{21} & l_{22} & & & \\ & l_{32} & l_{33} & & \\ & & \ddots & \ddots & \\ & & & l_{n,n-1} & l_{nn} \end{pmatrix}. \quad (5)$$

Формулы (4), следовательно, примут вид

$$\begin{cases} l_{ii} = \sqrt{a_{ii} - |l_{i,i-1}|^2} & i = 1, \dots, n, \\ l_{i+1,i} = a_{i+1,i}/l_{ii} & i = 1, \dots, n-1. \end{cases} \quad (6)$$

Алгоритм построения разложения Холецкого по этим формулам требует для своего проведения выполнения n операций извлечения корня, $2n - 1$ мультипликативных и n аддитивных операций.

§ 8.2. Метод Холецкого нахождения собственных значений

Будем строить для самосопряженной положительно определенной матрицы $A \in \mathbf{M}_n$ последовательность $\{A_k\}$ матриц $A_k \in \mathbf{M}_n$ по следующим правилам:

- 1) $A_1 = A$;
- 2) для всех $k = 1, 2, \dots$ матрица A_{k+1} получается из матрицы A_k следующим образом:
 - a) строим разложение Холецкого матрицы A_k : $A_k = L_k L_k^*$,
 - b) вычисляем матрицу A_{k+1} как произведение матриц L_k^* и L_k : $A_{k+1} = L_k^* L_k$.

Лемма 1. Для всех $k = 1, 2, \dots$ матрица A_k подобна A .

Доказательство. Имеем: $A_{k+1} = L_k^* L_k = (L_k^{-1} L_k) L_k^* L_k = L_k^{-1} (L_k L_k^*) L_k = L_k^{-1} A_k L_k$. Следовательно, матрица A_{k+1} подобна A_k . Поскольку $A_1 = A$, то по индукции получаем, что A_k подобна A для всех $k = 1, 2, \dots$, причем $A_{k+1} = L_1^{-1} \dots L_k^{-1} A_1 L_k \dots L_1 = (L_k \dots L_1)^{-1} A (L_k \dots L_1)$.

Следствие 1. Матрицы A_k , $k = 1, 2, \dots$ имеют те же собственные значения, что и матрица A .

Лемма 2. Для всех $k = 1, 2, \dots$ A_k – самосопряженная положительно определенная матрица.

Доказательство. Действительно, $A_1 = A$ – самосопряженная положительно определенная матрица. Пусть сделано $k = 1, 2, \dots$ шагов описанного выше процесса. Матрица $A_k = (L_{k-1} \dots L_1)^{-1} A (L_{k-1} \dots L_1)$ подобна матрице A и имеет те же собственные значения. Поскольку $A_k = L_{k-1}^* L_{k-1}$ и $A_k^* = L_{k-1}^* (L_{k-1}^*)^* = A_k$, то A_k – самосопряженная матрица. По лемме I.9.3 все собственные значения матрицы A вещественны и положительны. Следовательно, все собственные значения матрицы A_k вещественны и положительны. По лемме I.9.3 это означает, что A_k – самосопряженная положительно определенная матрица.

Лемма 3. Разложение Холецкого осуществимо для всякого $k = 1, 2, \dots$

Доказательство. Действительно, по лемме 2 A_k — симметричная положительно определенная матрица, и для нее разложение Холецкого существует по теореме 1.

Теорема 2. (Без доказательства.) Пусть $A \in \mathbf{M}_n$ — симметричная положительно определенная матрица, и собственные ее значения $\{\lambda_i\}$ матрицы A таковы, что

$$\lambda_1 > \lambda_2 > \dots > \lambda_n.$$

Тогда $A_k \rightarrow \text{diag}[\lambda_1, \dots, \lambda_n]$ при $k \rightarrow \infty$ (по норме в пространстве матриц). Другими словами, диагональные элементы матрицы $A_k = (a_{ij}^{(k)})$ сходятся к собственным значениям матрицы A , причем в правильном порядке:

$$a_{ii}^{(k)} \rightarrow \lambda_i \quad \text{при } k \rightarrow \infty, \quad i = 1, 2, \dots, n.$$

Скорость сходимости матрицы A_k к диагональной дается соотношением

$$a_{ij}^{(k)} = \overline{a_{ji}^{(k)}} = O\left(\left|\frac{\lambda_i}{\lambda_j}\right|^k\right) \quad \text{при } k \rightarrow \infty, \quad i > j.$$

Применение алгоритма к произвольной самосопряженной матрице $A \in \mathbf{M}_n$ требует слишком большого числа арифметических операций: $n^3/3 + O(n^2)$ ($n \rightarrow \infty$) на построение разложения Холецкого матрицы A_k и не более $n^3 + O(n^2)$ ($n \rightarrow \infty$) на вычисление матрицы A_{k+1} как произведения двух треугольных матриц. Поэтому метод Холецкого **никогда** не применяется к матрицам произвольного вида.

§ 8.2.1. Метод Холецкого нахождения собственных значений для трехдиагональной матрицы

Лемма 4. Если матрица A — трехдиагональная, то все матрицы A_k , $k = 1, 2, \dots$ в методе Холецкого — трехдиагональные.

Доказательство. Матрица $A_1 = A$ — трехдиагональная. Предположим, что матрица A_k — трехдиагональная. Тогда в разложении Холецкого $A_k = L_k L_k^*$ матрица L_k имеет вид (5) (т.е. является двухдиагональной). Из определения произведения матриц вытекает, что матрица $A_{k+1} = L_k^* L_k$ является трехдиагональной матрицей. Лемма доказана.

Эта лемма позволяет значительно ускорить работу метода Холецкого. Перед его применением исходная матрица A приводится к трехдиагональному виду A' унитарным подобием одним из алгоритмов, описанных в § I.14 и § I.15. Затем к матрице A' применяется метод Холецкого.

Алгоритм вычисления произведения матриц L^* и L

Произведение матриц $A = L^*L$, где L имеет вид (5), может быть вычислено значительно быстрее, чем произведение произвольных треугольных матриц. По определению произведения матриц

$$\begin{cases} a_{i,i-1} = \overline{l_{ii}}l_{i,i-1}, & i = 2, 3, \dots, n \\ a_{ii} = |l_{ii}|^2 + |l_{i+1,i}|^2, & i = 1, 2, \dots, n-1 \\ a_{i,i+1} = l_{ii}\overline{l_{i,i+1}}, & i = 1, 2, \dots, n-1 \\ a_{nn} = |l_{nn}|^2 \end{cases} \quad (7)$$

Вычисление произведения $A = L^*L$ по этим формулам требует $4n + O(1)$ ($n \rightarrow \infty$) мультипликативных и $n + O(1)$ ($n \rightarrow \infty$) аддитивных операций.

Оценка количества арифметических операций на один шаг метода Холецкого для трехдиагональной матрицы

1) Построение разложения Холецкого матрицы $A_k = L_k L_k^*$ по формулам (6) требует n операций извлечения корня, $2n + O(1)$ ($n \rightarrow \infty$) мультипликативных и $n + O(1)$ ($n \rightarrow \infty$) аддитивных операций.

2) Вычисление произведения $A_{k+1} = L_k^* L_k$ по формулам (7) требует $4n + O(1)$ ($n \rightarrow \infty$) мультипликативных и $n + O(1)$ ($n \rightarrow \infty$) аддитивных операций.

Следовательно, один шаг алгоритма для трехдиагональной матрицы требует n операций извлечения корня, $6n + O(1)$ ($n \rightarrow \infty$) мультипликативных и $2n + O(1)$ ($n \rightarrow \infty$) аддитивных операций.

§ 8.3. Ускорение сходимости алгоритма

Рассмотрим способы, применяемые для ускорения сходимости последовательности матриц $\{A_k\}$ к диагональной матрице. Как отмечалось выше (см. стр. 104), эти способы во многом схожи со способами ускорения сходимости LR - и QR -алгоритмов. Также справедливы замечания 7.3 и 7.4.

Поскольку метод Холецкого **никогда** не применяется для матриц произвольного вида, всюду ниже мы будем считать, что исходная матрица уже приведена унитарным подобием к трехдиагональному виду. Таким образом, начальная матрица A_1 — трехдиагональная. По доказанному выше это означает, что все матрицы A_k — трехдиагональные.

§ 8.3.1. Исчерпывание матрицы

Идея исчерпывания матрицы для метода Холецкого та же, что и для LR -алгоритма (см. стр. 105).

§ 8.3.2. Сдвиги

Идея использования сдвигов для метода Холецкого та же, что и для LR -алгоритма (см. стр. 106). Модифицированный метод Холецкого, основанный на этой идее, выглядит следующим образом.

Будем строить для матрицы $A \in \mathbf{M}_n$ последовательность $\{A_k\}$ матриц $A_k \in \mathbf{M}_n$ по следующим правилам:

- 1) $A_1 = A$;
- 2) для всех $k = 1, 2, \dots$ матрица A_{k+1} получается из матрицы A_k следующим образом:
 - a) определяем требуемый сдвиг s_k (его оптимальный выбор – отдельная задача),
 - б) строим разложение Холецкого матрицы $A_k - s_k I$: $A_k - s_k I = L_k L_k^*$,
 - в) вычисляем матрицу A_{k+1} как произведение матриц L_k^* и L_k плюс $s_k I$: $A_{k+1} = L_k^* L_k + s_k I$.

Здесь мы предполагаем, что для каждого $k = 1, 2, \dots$ разложение Холецкого матрицы $A_k - s_k I$ существует, т.е. для нее выполнены условия теоремы 1. Если это не так, то надо изменить значение сдвига s_k . На практике условия теоремы 1 проверить невозможно, поэтому выполняют алгоритм построения разложения Холецкого. При этом, если в алгоритме требуется извлечь корень из отрицательного числа или осуществить деление на 0, то немного изменяют значение сдвига s_k и заново выполняют алгоритм построения разложения Холецкого.

Нетрудно проверить, что матрица A_{k+1} подобна A_k : $A_{k+1} = L_k^* L_k + s_k I = (L_k^{-1} L_k)(L_k^* L_k + s_k I) = L_k^{-1}(L_k L_k^*)L_k + s_k L_k I = L_k^{-1}(L_k L_k^* + s_k I)L_k = L_k^{-1} A_k L_k$ и, следовательно, все матрицы A_k , $k = 1, 2, \dots$ имеют те же собственные значения, что и матрица A .

§ 8.3.3. Практическая организация вычислений в методе Холецкого

Пусть требуется определить все собственные значения самосопряженной положительно определенной матрицы $A \in \mathbf{M}_n$ с точностью ε .

Вначале приводим матрицу к трехдиагональному виду A_1 унитарным подобием одним из алгоритмов, описанных в § I.14 и § I.15.

Затем к матрице A_1 применяем метод Холецкого со сдвигами. На шаге k в качестве сдвига s_k возьмем $a_{nn}^{(k)}$, т.е. $s_k = a_{nn}^{(k)}$. Поскольку $a_{nn}^{(k)} \rightarrow \lambda_n$, то s_k является приближением к λ_n и скорость сходимости к нулю элемента $a_{n,n-1}^{(k)}$ будет очень высокой. Как только на некотором шаге k будет выполнено условие $|a_{n,n-1}^{(k)}| < \varepsilon \|A\|_\infty$, в качестве λ_n берем $a_{nn}^{(k)}$ и применяем алгоритм к подматрице $(a_{ij})_{i,j=1,2,\dots,n-1} \in \mathbf{M}_{n-1}$ на 1 меньшей размерности. Так поступаем до тех пор, пока размерность матрицы не станет равной 2. Для этой матрицы собственные значения определяются как решения соответствующего квадратного уравнения.

§ 9. QR АЛГОРИТМ

QR алгоритм позволяет находить все собственные значения матрицы $A \in M_n$.

§ 9.1. *QR*-разложение, используемое в *QR* алгоритме

В *QR*-алгоритме используется то же *QR*-разложение, что строилось в методе вращений (см. § I.12, теорема I.12.1) и в методе отражений решения линейных систем (см. § I.13, теорема I.13.1).

§ 9.1.1. Алгоритм построения *QR*-разложения для произвольной матрицы

Алгоритм построения *QR*-разложения для произвольной матрицы был описан ранее, см. § I.12.4, “Построение *QR*-разложения методом вращений”, с. 50, или § I.13.4, “Построение *QR*-разложения методом отражений”, с. 59. Там же подсчитана вычислительная сложность этих алгоритмов.

§ 9.1.2. Алгоритм построения *QR*-разложения для почти треугольной матрицы

Рассмотрим случай, когда матрица $A \in M_n$ в алгоритме построения *QR*-разложения почти треугольная:

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1,i} & \dots & a_{1,n-2} & a_{1,n-1} & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2,i} & \dots & a_{2,n-2} & a_{2,n-1} & a_{2n} \\ a_{32} & \ddots & \vdots & \dots & \vdots & \vdots & \vdots & \vdots \\ \ddots & a_{i,i} & \dots & a_{i,n-2} & a_{i,n-1} & a_{i,n} \\ a_{i+1,i} & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \ddots & a_{n-2,n-2} & a_{n-2,n-1} & a_{n-2,n} \\ a_{n-1,n-2} & a_{n-1,n-1} & a_{n-1,n} \\ a_{n,n-1} & a_{nn} \end{pmatrix}$$

Алгоритм построения *QR*-разложения для почти треугольной матрицы методом вращений

Обозначим $a_1 = (a_{11}, a_{21}, 0, \dots, 0)^t \in \mathbf{R}^n$ – первый столбец матрицы A . Согласно лемме I.12.3 существует матрица $T_{12} = T_{12}(\varphi_{12})$, такая, что $T_{12}a_1 = \|a_1\| e_1$

(причем значение угла φ_{12} определяется леммами I.12.2, I.12.3). Умножим матрицу A на T_{12} слева, получим матрицу $A^{(1)} = T_{12}A$,

$$A^{(1)} = \begin{pmatrix} \|a_1\| & r_{12} & \dots & r_{1,n-1} & r_{1n} \\ a_{22}^{(1)} & \dots & a_{2,n-1}^{(1)} & a_{2n}^{(1)} & \\ a_{32}^{(1)} & \dots & a_{3,n-1}^{(1)} & a_{3n}^{(1)} & \\ \ddots & \vdots & \vdots & \vdots & \\ a_{n,n-1}^{(1)} & a_{nn}^{(1)} & & & \end{pmatrix}. \quad (1)$$

Далее процесс применяется к подматрице $(a_{ij}^{(1)})_{i,j=2,\dots,n}$.

Пусть сделаны $k-1$, $k = 1, \dots, n-1$ шагов этого процесса, т.е. матрица преобразована к виду

$$A^{(k-1)} = \prod_{i=k-1}^1 T_{i,i+1} A, \quad (2)$$

где

$$A^{(k-1)} = \begin{pmatrix} \|a_1\| & r_{12} & r_{13} & \dots & r_{1,k-1} & r_{1k} & \dots & r_{1,n-1} & r_{1n} \\ \|a_1^{(1)}\| & r_{23} & \dots & r_{2,k-1} & r_{2k} & \dots & r_{2,n-1} & r_{2n} & \\ \|a_1^{(2)}\| & \dots & r_{3,k-1} & r_{3k} & \dots & r_{3,n-1} & r_{3n} & \\ \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \\ \|a_1^{(k-2)}\| & r_{k-1,k} & \dots & r_{k-1,n-1} & r_{k-1,n} & & & & \\ a_{kk}^{(k-1)} & \dots & a_{k,n-1}^{(k-1)} & a_{kn}^{(k-1)} & & & & & \\ a_{k+1,k}^{(k-1)} & \dots & a_{k+1,n-1}^{(k-1)} & a_{k+1,n}^{(k-1)} & & & & & \\ \ddots & \vdots & \vdots & \vdots & & & & & \\ a_{n,n-1}^{(k-1)} & a_{nn}^{(k-1)} & & & & & & & \end{pmatrix} \quad (3)$$

(здесь $\prod_{i=k-1}^1$ означает, что сомножители берутся в порядке $k-1, \dots, 1$).

Обозначим

$$a_1^{(k-1)} = (a_{kk}^{(k-1)}, a_{k+1,k}^{(k-1)}, 0, \dots, 0)^t \in \mathbf{R}^{n-k+1} \quad (4)$$

– первый столбец подматрицы $(a_{ij}^{(k-1)})_{i,j=k,\dots,n}$. Согласно лемме I.12.3 существует матрица $T_{k,k+1} = T_{k,k+1}(\varphi_{k,k+1})$, такая, что

$$T_{k,k+1} a_1^{(k-1)} = \|a_1^{(k-1)}\| e_1^{(n-k+1)}, \quad (5)$$

(значения угла $\varphi_{k,k+1}$ определяются леммами I.12.2, I.12.3), здесь $e_1^{(m)} = (1, 0, \dots, 0)^t \in \mathbf{R}^m$. Умножим матрицу (3) на $T_{k,k+1}$ слева, получим

$$A^{(k)} = T_{k,k+1} A^{(k-1)} = \prod_{i=k}^1 T_{i,i+1} A, \quad (6)$$

где $A^{(k)} =$

$$\left(\begin{array}{ccccccccc} \|a_1\| & r_{12} & r_{13} & \dots & r_{1,k-1} & r_{1k} & r_{1,k+1} & \dots & r_{1,n-1} & r_{1n} \\ \|a_1^{(1)}\| & r_{23} & \dots & r_{2,k-1} & r_{2k} & r_{2,k+1} & \dots & r_{2,n-1} & r_{2n} \\ \|a_1^{(2)}\| & \dots & r_{3,k-1} & r_{3k} & r_{3,k+1} & \dots & r_{3,n-1} & r_{3n} \\ \ddots & \vdots & \vdots & & \vdots & \ddots & \vdots & & \vdots \\ \|a_1^{(k-2)}\| & r_{k-1,k} & r_{k-1,k+1} & \dots & r_{k-1,n-1} & r_{k-1,n} \\ \|a_1^{(k-1)}\| & r_{k,k+1} & \dots & r_{k,n-1} & r_{kn} \\ a_{k+1,k+1}^{(k)} & \dots & a_{k+1,n-1}^{(k)} & a_{k+1,n}^{(k)} \\ a_{k+2,k+1}^{(k)} & \dots & a_{k+2,n-1}^{(k)} & a_{k+2,n}^{(k)} \\ \ddots & \vdots & \vdots & & \vdots \\ a_{n,n-1}^{(k)} & & a_{nn}^{(k)} & & \end{array} \right). \quad (7)$$

Отметим, что в (6) матрица элементарного вращения $T_{k,k+1}$ умножается только на подматрицу $(a_{ij}^{(k-1)})_{i,j=k,\dots,n}$ матрицы $A^{(k-1)}$ размера $n - k + 1$ (остальная часть $A^{(k-1)}$ в преобразовании (6) не участвует). Следовательно, матрица $A^{(k)}$ получается из $A^{(k-1)}$ изменением двух строк (k -ой и $(k+1)$ -ой) длины $n - k + 1$.

После $n - 1$ шагов этого процесса (т.е. перехода от матриц (3) к (7)), матрица примет вид

$$R = A^{(n-1)} = \prod_{i=n-1}^1 T_{i,i+1} A, \quad (8)$$

где

$$R = \left(\begin{array}{ccccccccc} \|a_1\| & r_{12} & r_{13} & \dots & r_{1,n-2} & r_{1,n-1} & r_{1n} \\ \|a_1^{(1)}\| & r_{23} & \dots & r_{2,n-2} & r_{2,n-1} & r_{2n} \\ \|a_1^{(2)}\| & \dots & r_{3,n-2} & r_{3,n-1} & r_{3n} \\ \ddots & \vdots & \vdots & & \vdots & & \vdots \\ \|a_1^{(n-3)}\| & r_{n-2,n-1} & r_{n-2,n} \\ \|a_1^{(n-2)}\| & r_{n-1,n} \\ a_{nn}^{(n-1)} & & & & & & & & \end{array} \right) \quad (9)$$

(напомним, определения векторов $a_1^{(k-1)}$, $k = 1, \dots, n - 1$ даются в (4), где считаем, что $a_1^{(0)} = a_1$).

Так как матрицы вращения ортогональные, то $T_{i,i+1}^{-1} = T_{i,i+1}^t$ и из (8) получаем

$$A = \prod_{i=1}^{n-1} T_{i,i+1}^t R \equiv QR \quad (10)$$

— искомое QR -разложение.

Хранение матриц Q и R в памяти. Матрица R хранится на месте верхне-го треугольника матрицы A и получается из нее последовательным применением

элементарных вращений (как описано выше). Для хранения матрицы Q выделяются два вектора длины $n - 1$. В первом векторе хранятся значения $\cos \varphi_{i,i+1}$, $i = 1, 2, \dots, n - 1$, во втором векторе — значения $\sin \varphi_{i,i+1}$, $i = 1, 2, \dots, n - 1$.

Оценка количества арифметических операций

Оценим трудоемкость k -го шага алгоритма, а затем просуммируем полученные оценки по всем $k = 1, \dots, n - 1$.

1. На вычисление матрицы $T_{k,k+1}$, участвующей в (6), согласно лемме I.12.2 требуется 4 мультипликативные, одна аддитивная и одна операция извлечения корня.

2. На вычисление компонент k, \dots, n k -го столбца матрицы $A^{(k)}$, равных компонентам вектора $\|a_1^{(k-1)}\| e_1^{(n-k+1)}$ требуется (для вычисления длины вектора (4)) две операции умножения, одна операция сложения и одна операция извлечения корня. Столбец k вычисляется именно этим способом (а не по общим формулам (6)) для сокращения количества арифметических операций и уменьшения вычислительной погрешности.

3. Поскольку в формуле (6) матрица элементарного вращения умножается на подматрицу $(a_{ij}^{(k-1)})_{i=k, \dots, n, j=k+1, \dots, n}$ матрицы $A^{(k-1)}$ размера $(n - k + 1) \times (n - k)$ (k -й столбец матрицы $A^{(k)}$ уже вычислен в пункте 2), то согласно лемме I.12.5 на это требуется $4(n - k)$ умножений и $2(n - k)$ сложений.

Итак, на k -ом шаге алгоритма требуется выполнить $4 + 2 + 4(n - k) = 4(n - k) + 6$ мультипликативных операций, $1 + 1 + 2(n - k) = 2(n - k) + 2$ аддитивных операций и две операции извлечения корня.

Следовательно, всего для проведения алгоритма требуется выполнить

$$\sum_{k=1}^{n-1} (4(n - k) + 6) = 4n(n - 1)/2 + 6(n - 1) = 2n^2 + O(n) \quad (n \rightarrow \infty)$$

мультипликативных операций, $\sum_{k=1}^{n-1} (2(n - k) + 2) = n^2 + O(n)$ ($n \rightarrow \infty$) аддитивных операций и $\sum_{k=1}^{n-1} 2 = O(n)$ ($n \rightarrow \infty$) операций извлечения корня (которые по трудоемкости по порядку можно сравнить с операциями деления).

Алгоритм построения QR-разложения для почти треугольной матрицы методом отражений

Обозначим $a_1 = (a_{11}, a_{21}, 0, \dots, 0)^t \in \mathbf{R}^n$ — первый столбец матрицы A . Согласно лемме I.13.9 существует вектор $x^{(1)} \in \mathbf{C}^n$, равный

$$x^{(1)} = \pm \frac{a_1 - \|a_1\|e_1}{\|a_1 - \|a_1\|e_1\|}, \quad (11)$$

такой, что $U(x^{(1)})a_1 = \|a_1\|e_1$, где $e_1 = (1, 0, \dots, 0) \in \mathbf{C}^n$, $U_1 = U(x^{(1)})$ — матрица отражения. Отметим, что у вектора $x^{(1)}$ только первые две компоненты отличны от нуля. Следовательно, матрица $U(x^{(1)})$ отличается от единичной матрицы только блоком 2×2 , стоящим на главной диагонали. Умножим матрицу

A на $U(x^{(1)})$ слева, получим матрицу $A^{(1)} = U(x^{(1)})A$ вида (1). Далее процесс применяется к подматрице $(a_{ij}^{(1)})_{i,j=2,\dots,n}$.

Пусть сделаны $k - 1$, $k = 1, \dots, n$ шагов этого процесса, т.е. матрица преобразована к виду

$$A^{(k-1)} = \prod_{i=k-1}^1 U_i A, \quad (12)$$

где матрица $A^{(k-1)}$ имеет вид (3),

$$U_i = \begin{pmatrix} I_{i-1} & 0 \\ 0 & U(x^{(i)}) \end{pmatrix}, \quad (13)$$

здесь $I_{i-1} \in \mathbf{M}_{i-1}$ – единичная матрица размера $(i-1) \times (i-1)$, $U(x^{(i)}) \in \mathbf{M}_{n-i+1}$ – матрица отражения размера $(n-i+1) \times (n-i+1)$, построенная по вектору

$$x^{(i)} = \pm \frac{a_1^{(i-1)} - \|a_1^{(i-1)}\|e_1^{(n-i+1)}}{\|a_1^{(i-1)} - \|a_1^{(i-1)}\|e_1^{(n-i+1)}\|} \in \mathbf{C}^{n-i+1},$$

где $e_1^{(m)} = (1, 0, \dots, 0) \in \mathbf{C}^m$.

Введем обозначение (4) для первого столбца подматрицы $(a_{ij}^{(k-1)})_{i,j=k,\dots,n}$. Согласно лемме I.13.9 существует матрица отражения (I.13.5) такая, что выполнено соотношение (I.13.6). Введем матрицу U_k вида (I.13.7). Соотношения (I.13.8) и (I.13.9) показывают, что матрица U_k унитарна и самосопряжена. Отметим, что у вектора $x^{(k)}$ в (I.13.5) только первые две компоненты отличны от нуля. Следовательно, матрица U_k отличается от единичной матрицы только блоком 2×2 , стоящим на главной диагонали.

Умножим матрицу (3) на U_k слева, получим

$$A^{(k)} = U_k A^{(k-1)} = \prod_{i=k}^1 U_i A, \quad (14)$$

где $A^{(k)}$ имеет вид (7). Отметим, что в (14) матрица U_k умножается только на подматрицу $(a_{ij}^{(k-1)})_{i,j=k,\dots,n}$ матрицы $A^{(k-1)}$ размера $n - k + 1$ (остальная часть $A^{(k-1)}$ в преобразовании (14) не участвует). Поскольку матрица U_k отличается от единичной матрицы только блоком 2×2 , стоящим на главной диагонали в строках k и $k+1$, то матрица $A^{(k)}$ получается из $A^{(k-1)}$ изменением двух строк (k -ой и $(k+1)$ -ой) длины $n - k + 1$.

Вычисления по формулам (I.13.5) осуществляются следующим образом: вначале вычисляются числа

$$s_k = |a_{k+1,k}^{(k-1)}|^2, \quad (15)$$

$$\|a_1^{(k-1)}\| = \sqrt{|a_{kk}^{(k-1)}|^2 + s_k}. \quad (16)$$

затем – вектор

$$x^{(k)} = (a_{kk}^{(k-1)} - \|a_1^{(k-1)}\|, a_{k+1,k}^{(k-1)}, 0, \dots, 0)^t \in \mathbf{C}^{n-k+1} \quad (17)$$

и его норма

$$\|x^{(k)}\| = \sqrt{|x_1^{(k)}|^2 + s_k}. \quad (18)$$

Теперь можно вычислить искомый вектор $x^{(k)}$:

$$x^{(k)} := (x_1^{(k)} / \|x^{(k)}\|, x_2^{(k)} / \|x^{(k)}\|, 0, \dots, 0) \in \mathbf{C}^{n-k+1}. \quad (19)$$

После n шагов этого процесса (т.е. перехода от матриц (3) к (7)), матрица примет вид

$$R = A^{(n-1)} = \prod_{i=n}^1 U_i A, \quad (20)$$

где матрица R имеет вид (9).

Так как матрицы U_k унитарные и самосопряженные, то $U_i^{-1} = U_i^* = U_i$ и из (20) получаем

$$A = \prod_{i=1}^n U_i R \equiv QR \quad (21)$$

— искомое QR -разложение.

Хранение матриц Q и R в памяти. Матрица R хранится на месте верхнего треугольника матрицы A и получается из нее последовательным применением матриц отражения (как описано выше). Для хранения матрицы Q выделяются два вектора длины n . В первом векторе хранятся первые ненулевые компоненты векторов $x^{(i)}$, $i = 1, 2, \dots, n$, во втором векторе — вторые ненулевые компоненты векторов $x^{(i)}$, $i = 1, 2, \dots, n$.

Оценка количества арифметических операций

Оценим трудоемкость k -го шага алгоритма, а затем просуммируем полученные оценки по всем $k = 1, \dots, n$.

1. На вычисление матрицы $U(x_k)$ по формулам (I.13.5) требуется
 - а) 1 умножение для вычисления s_k в (15);
 - б) одно умножение, одно сложение и одна операция извлечения корня для вычисления $\|a_1^{(k-1)}\|$ в (16);
 - в) одно вычитание для построения вектора $x^{(k)}$ в (17);
 - г) одно умножение, одно сложение и одна операция извлечения корня для вычисления $\|x^{(k)}\|$ в (18);
 - д) 2 деления для построения вектора $x^{(k)}$ в (19).

Всего для построения матрицы $U(x_k)$ требуется $1 + 1 + 1 + 2 = 5$ мультипликативных, $1 + 1 + 1 = 3$ аддитивные операции и $1 + 1 = 2$ операции извлечения корня.

2. Компоненты k, \dots, n k -го столбца матрицы $A^{(k)}$, равные компонентам вектора $\|a_1^{(k-1)}\| e_1^{(n-k+1)}$, уже вычислены в (16). Столбец k вычисляется не по общим формулам (20) для сокращения количества арифметических операций и уменьшения вычислительной погрешности.

3. Поскольку в формуле (20) матрица U_k вида (I.13.5) умножается на матрицу $A^{(k-1)}$ вида (3), то при вычислениях по (20) надо умножить матрицу отражения

$U(x^{(k)}) \in \mathbf{M}_{n-k+1}$ на подматрицу $(a_{ij}^{(k-1)})_{i=k,\dots,n, j=k+1,\dots,n}$ матрицы $A^{(k-1)}$ размера $(n - k + 1) \times (n - k)$ (k -й столбец матрицы $A^{(k)}$ уже вычислен в пункте 2). Поскольку матрица $U(x^{(k)})$ отличается от единичной матрицы только блоком 2×2 , стоящим на главной диагонали в строках 1 и 2, то при вычислениях по (20) надо умножить матрицу отражения $U(x^{(k)}) \in \mathbf{M}_{n-k+1}$ на подматрицу $(a_{ij}^{(k-1)})_{i=k, k+1, j=k+1, \dots, n}$ матрицы $A^{(k-1)}$ размера $2 \times (n - k)$. Согласно лемме I.13.11 на это требуется $(n - k)(2 \cdot 2 + 1) = 5(n - k)$ умножений и $(n - k)(2 \cdot 2 - 1) = 3(n - k)$ сложений.

Итак, на k -ом шаге алгоритма требуется выполнить $5 + 5(n - k) = 5(n - k + 1)$ мультипликативных операций, $3 + 3(n - k) = 3(n - k + 1)$ аддитивных операций и 2 операции извлечения корня.

Следовательно, всего для проведения алгоритма требуется выполнить

$$\sum_{k=1}^n 5(n - k + 1) = 5n(n + 1)/2 = (5/2)n^2 + O(n) \quad (n \rightarrow \infty)$$

мультипликативных операций, $\sum_{k=1}^n 3(n - k + 1) = (3/2)n^2 + O(n)$ ($n \rightarrow \infty$) аддитивных операций и $\sum_{k=1}^n 2 = O(n)$ ($n \rightarrow \infty$) операций извлечения корня (которые по трудоемкости по порядку можно сравнить с операциями деления).

§ 9.1.3. Алгоритм построения QR-разложения для трехдиагональной матрицы

Рассмотрим случай, когда матрица $A \in \mathbf{M}_n$ в приведенном выше алгоритме трехдиагональная.

Алгоритм построения QR-разложения для трехдиагональной матрицы методом вращений

Обозначим $a_1 = (a_{11}, a_{21}, 0, \dots, 0)^t \in \mathbf{R}^n$ – первый столбец матрицы A . Согласно лемме I.12.3 существует матрица $T_{12} = T_{12}(\varphi_{12})$, такая, что $T_{12}a_1 = \|a_1\| e_1$ (причем значение угла φ_{12} определяется леммами I.12.2, I.12.3). Умножим матрицу Ab на T_{12} слева, получим матрицу

$$A^{(1)} = T_{12}A = \begin{pmatrix} \|a_1\| & r_{12} & r_{13} & & & \\ & a_{22}^{(1)} & a_{23}^{(1)} & & & \\ & a_{32}^{(1)} & a_{33}^{(1)} & \ddots & & \\ & \ddots & \ddots & & a_{n-1,n}^{(1)} & \\ & a_{n,n-1}^{(1)} & & & a_{nn}^{(1)} & \end{pmatrix}. \quad (22)$$

Далее процесс применяется к подматрице $(a_{ij}^{(1)})_{i,j=2,\dots,n}$.

Пусть сделаны $k - 1$, $k = 1, \dots, n - 1$ шагов этого процесса, т.е. матрица преобразована к виду (2), где

$$A^{(k-1)} = \begin{pmatrix} \|a_1\| & r_{12} & r_{13} & & \\ \|a_1^{(1)}\| & \ddots & \ddots & & \\ & \ddots & r_{k-2,k-1} & r_{k-2,k} & \\ & & \|a_1^{(k-2)}\| & r_{k-1,k} & r_{k-1,k+1} \\ & & & a_{kk}^{(k-1)} & a_{k,k+1}^{(k-1)} \\ & & & a_{k+1,k}^{(k-1)} & \ddots & \ddots \\ & & & & \ddots & a_{n-1,n-1}^{(k-1)} & a_{n-1,n}^{(k-1)} \\ & & & & & a_{n,n-1}^{(k-1)} & a_{nn}^{(k-1)} \end{pmatrix}. \quad (23)$$

Введем обозначение (4) для первого столбца подматрицы $(a_{ij}^{(k-1)})_{i,j=k,\dots,n}$. Согласно лемме I.12.3 существует матрица $T_{k,k+1} = T_{k,k+1}(\varphi_{k,k+1})$, такая, что выполнено соотношение (5) (значения угла $\varphi_{k,k+1}$ определяются леммами I.12.2, I.12.3). Умножим матрицу (23) на $T_{k,k+1}$ слева, получим (6), где $A^{(k)} =$

$$\begin{pmatrix} \|a_1\| & r_{12} & r_{13} & & \\ \|a_1^{(1)}\| & \ddots & \ddots & & \\ & \ddots & r_{k-2,k-1} & r_{k-2,k} & \\ & & \|a_1^{(k-2)}\| & r_{k-1,k} & r_{k-1,k+1} \\ & & & \|a_1^{(k-1)}\| & r_{k,k+1} & r_{k,k+2} \\ & & & & a_{k+1,k+1}^{(k)} & a_{k+1,k+2}^{(k)} \\ & & & & a_{k+2,k+1}^{(k)} & \ddots & \ddots \\ & & & & & \ddots & a_{n-1,n-1}^{(k)} & a_{n-1,n}^{(k)} \\ & & & & & & a_{n,n-1}^{(k)} & a_{nn}^{(k)} \end{pmatrix}. \quad (24)$$

Отметим, что в (6) матрица элементарного вращения $T_{k,k+1}$ умножается только на подматрицу $(a_{ij}^{(k-1)})_{i,j=k,\dots,n}$ матрицы $A^{(k-1)}$ размера $n - k + 1$ (остальная часть $A^{(k-1)}$ в преобразовании (6) не участвует). Следовательно, матрица $A^{(k)}$ получается из $A^{(k-1)}$ изменением двух строк (k -ой и $(k+1)$ -ой) длины $n - k + 1$, имеющими не более трех ненулевых элементов.

После $n - 1$ шагов этого процесса (т.е. перехода от матриц (23) к (24)), матрица примет вид (8), где

$$R = \begin{pmatrix} \|a_1\| & r_{12} & r_{13} & & \\ \|a_1^{(1)}\| & \ddots & \ddots & & \\ & \ddots & r_{n-2,n-1} & r_{n-2,n} & \\ & & \|a_1^{(n-2)}\| & r_{n-1,n} & \\ & & & a_{nn}^{(n-1)} & \end{pmatrix} \quad (25)$$

(напомним, определения векторов $a_1^{(k-1)}$, $k = 1, \dots, n-1$ даются в (4), где считаем, что $a_1^{(0)} = a_1$).

Так как матрицы вращения ортогональные, то $T_{i,i+1}^{-1} = T_{i,i+1}^t$ и из (8) получаем искомое QR -разложение (10).

Хранение матриц Q и R в памяти. Матрица A хранится в виде трех векторов, задающих ненулевые диагонали матрицы. Матрица R хранится на месте матрицы A и получается из нее последовательным применением элементарных вращений (как описано выше). Для хранения матрицы Q выделяются два вектора длины $n-1$. В первом векторе хранятся значения $\cos \varphi_{i,i+1}$, $i = 1, 2, \dots, n-1$, во втором векторе — значения $\sin \varphi_{i,i+1}$, $i = 1, 2, \dots, n-1$.

Оценка количества арифметических операций

Оценим трудоемкость k -го шага алгоритма, а затем просуммируем полученные оценки по всем $k = 1, \dots, n-1$.

1. На вычисление матрицы $T_{k,k+1}$, участвующей в (6), согласно лемме I.12.2 требуется 4 мультипликативные, одна аддитивная и одна операция извлечения корня.

2. На вычисление компонент k, \dots, n k -го столбца матрицы $A^{(k)}$, равных компонентам вектора $\|a_1^{(k-1)}\| e_1^{(n-k+1)}$ требуется (для вычисления длины вектора (4)) две операции умножения, одна операция сложения и одна операция извлечения корня. Столбец k вычисляется именно этим способом (а не по общим формулам (6)) для сокращения количества арифметических операций и уменьшения вычислительной погрешности.

3. Поскольку в формуле (6) матрица элементарного вращения умножается на подматрицу $(a_{ij}^{(k-1)})_{i=k, \dots, n, j=k+1, \dots, n}$ матрицы $A^{(k-1)}$ размера $(n-k+1) \times (n-k)$ (k -й столбец матрицы $A^{(k)}$ уже вычислен в пункте 2), имеющими не более двух ненулевых элементов, то согласно лемме I.12.5 на это требуется не более $4 \cdot 2 = 8$ умножений и $2 \cdot 2 = 4$ сложений.

Итак, на k -ом шаге алгоритма требуется выполнить не более $4 + 2 + 8 = 14$ мультипликативных операций, $1+1+4 = 6$ аддитивных операций и две операции извлечения корня.

Следовательно, всего для проведения алгоритма требуется выполнить не более $\sum_{k=1}^{n-1} 14 = 14(n-1)$ мультипликативных операций, $6(n-1)$ аддитивных операций и $2(n-1)$ операций извлечения корня (которые по трудоемкости по порядку можно сравнить с операциями деления).

Алгоритм построения QR -разложения для трехдиагональной матрицы методом отражений

Обозначим $a_1 = (a_{11}, a_{21}, 0, \dots, 0)^t \in \mathbf{R}^n$ — первый столбец матрицы A . Согласно лемме I.13.9 существует вектор $x^{(1)} \in \mathbf{C}^n$, вида (11), такой, что $U(x^{(1)})a_1 = \|a_1\|e_1$, где $e_1 = (1, 0, \dots, 0) \in \mathbf{C}^n$, $U_1 = U(x^{(1)})$ — матрица отражения. Отметим, что у вектора $x^{(1)}$ только первые две компоненты отличны от нуля. Следовательно, матрица $U(x^{(1)})$ отличается от единичной матрицы только

блоком 2×2 , стоящим на главной диагонали. Умножим матрицу A на $U(x^{(1)})$ слева, получим матрицу $A^{(1)} = U(x^{(1)})A$ вида (22). Далее процесс применяется к подматрице $(a_{ij}^{(1)})_{i,j=2,\dots,n}$.

Пусть сделаны $k - 1$, $k = 1, \dots, n$ шагов этого процесса, т.е. матрица преобразована к виду (12) где матрица $A^{(k-1)}$ имеет вид (23), матрица U_i построена в (13).

Введем обозначение (4) для первого столбца подматрицы $(a_{ij}^{(k-1)})_{i,j=k,\dots,n}$. Согласно лемме I.13.9 существует матрица отражения (I.13.5) такая, что выполнено соотношение (I.13.6). Введем матрицу U_k вида (I.13.7). Соотношения (I.13.8) и (I.13.9) показывают, что матрица U_k унитарна и самосопряжена. Отметим, что у вектора $x^{(k)}$ в (I.13.5) только первые две компоненты отличны от нуля. Следовательно, матрица U_k отличается от единичной матрицы только блоком 2×2 , стоящим на главной диагонали.

Умножим матрицу (3) на U_k слева, получим (14), где $A^{(k)}$ имеет вид (24). Отметим, что в (14) матрица U_k умножается только на подматрицу $(a_{ij}^{(k-1)})_{i,j=k,\dots,n}$ матрицы $A^{(k-1)}$ размера $n - k + 1$ (остальная часть $A^{(k-1)}$ в преобразовании (14) не участвует). Поскольку матрица U_k отличается от единичной матрицы только блоком 2×2 , стоящим на главной диагонали в строках k и $k+1$, то матрица $A^{(k)}$ получается из $A^{(k-1)}$ изменением двух строк (k -ой и $(k+1)$ -ой) длины $n - k + 1$, имеющими не более трех ненулевых элементов.

После n шагов этого процесса (т.е. перехода от матриц (23) к (24)), матрица примет вид (20), где матрица R имеет вид (25).

Так как матрицы U_k унитарные и самосопряженные, то $U_i^{-1} = U_i^* = U_i$ и из (20) получаем искомое QR -разложение (21).

Хранение матриц Q и R в памяти. Матрица A хранится в виде трех векторов, задающих ненулевые диагонали матрицы. Матрица R хранится на месте матрицы A и получается из нее последовательным применением матриц отражения (как описано выше). Для хранения матрицы Q выделяются два вектора длины n . В первом векторе хранятся первые ненулевые компоненты векторов $x^{(i)}$, $i = 1, 2, \dots, n$, во втором векторе — вторые ненулевые компоненты векторов $x^{(i)}$, $i = 1, 2, \dots, n$.

Оценка количества арифметических операций

Оценим трудоемкость k -го шага алгоритма, а затем просуммируем полученные оценки по всем $k = 1, \dots, n$.

1. На вычисление матрицы $U(x_k)$ по формулам (I.13.5) требуется $1+1+1+2 = 5$ мультипликативных, $1+1+1 = 3$ аддитивные операции и $1+1 = 2$ операции извлечения корня.

2. Компоненты k, \dots, n k -го столбца матрицы $A^{(k)}$, равные компонентам вектора $\|a_1^{(k-1)}\| e_1^{(n-k+1)}$, уже вычислены в (16). Столбец k вычисляется не по общим формулам (20) для сокращения количества арифметических операций и уменьшения вычислительной погрешности.

3. Поскольку в формуле (20) матрица U_k вида (I.13.5) умножается на матрицу $A^{(k-1)}$ вида (23), то при вычислениях по (20) надо умножить матрицу отражения $U(x^{(k)}) \in \mathbf{M}_{n-k+1}$ на подматрицу $(a_{ij}^{(k-1)})_{i=k,\dots,n, j=k+1,\dots,n}$ матрицы $A^{(k-1)}$ размера $(n - k + 1) \times (n - k)$ (k -й столбец матрицы $A^{(k)}$ уже вычислен в пункте 2). Поскольку матрица $U(x^{(k)})$ отличается от единичной матрицы только блоком 2×2 , стоящим на главной диагонали в строках 1 и 2, то при вычислениях по (20) надо умножить матрицу отражения $U(x^{(k)}) \in \mathbf{M}_{n-k+1}$ на подматрицу $(a_{ij}^{(k-1)})_{i=k, k+1, j=k+1, \dots, n}$ матрицы $A^{(k-1)}$ размера $2 \times (n - k)$, где в каждой строке не более двух ненулевых элементов. Согласно лемме I.13.11 на это требуется $2(2 \cdot 2 + 1) = 10$ умножений и $2(2 \cdot 2 - 1) = 6$ сложений.

Итак, на k -ом шаге алгоритма требуется выполнить $5 + 10 = 15$ мультипликативных операций, $3 + 6 = 9$ аддитивных операций и 2 операции извлечения корня.

Следовательно, всего для проведения алгоритма требуется выполнить не более $\sum_{k=1}^n 15 = 15n$ мультипликативных операций, $9n$ аддитивных операций и $2n$ операций извлечения корня (которые по трудоемкости по порядку можно сравнить с операциями деления).

§ 9.2. QR алгоритм нахождения собственных значений

Будем строить для матрицы $A \in \mathbf{M}_n$ последовательность $\{A_k\}$ матриц $A_k \in \mathbf{M}_n$ по следующим правилам:

- 1) $A_1 = A$;
- 2) для всех $k = 1, 2, \dots$ матрица A_{k+1} получается из матрицы A_k следующим образом:
 - a) строим QR-разложение матрицы A_k : $A_k = Q_k R_k$,
 - b) вычисляем матрицу A_{k+1} как произведение матриц R_k и Q_k : $A_{k+1} = R_k Q_k$.

Лемма 1. Для всех $k = 1, 2, \dots$ матрица A_k унитарно подобна A .

Доказательство. Имеем: $A_{k+1} = R_k Q_k = (Q_k^* Q_k) R_k Q_k = Q_k^* (Q_k R_k) Q_k = Q_k^* A_k Q_k$. Следовательно, матрица A_{k+1} унитарно подобна A_k . Поскольку $A_1 = A$, то по индукции получаем, что A_k унитарно подобна A для всех $k = 1, 2, \dots$, причем $A_{k+1} = Q_1^* \dots Q_k^* A_1 Q_k \dots Q_1 = (Q_k \dots Q_1)^* A (Q_k \dots Q_1)$.

Следствие 1. Матрицы A_k , $k = 1, 2, \dots$ имеют те же собственные значения, что и матрица A .

Теорема 1. (Без доказательства.) Пусть собственные значения $\{\lambda_i\}$ матрицы $A \in \mathbf{M}_n$ таковы, что

$$|\lambda_1| > |\lambda_2| > \dots > |\lambda_n|.$$

Тогда в некотором смысле (детали выходят за рамки настоящего курса) $Q_k \rightarrow I$ при $k \rightarrow \infty$, $R_k \rightarrow A_k$ при $k \rightarrow \infty$. Тем самым диагональные элементы матрицы $A_k = (a_{ij}^{(k)})$ сходятся к собственным значениям матрицы A :

$$a_{ii}^{(k)} \rightarrow \lambda_i \quad \text{при } k \rightarrow \infty, \quad i = 1, 2, \dots, n$$

при некотором i_0 (т.е. порядок собственных значений может нарушаться).

Скорость сходимости матрицы A_k к треугольной имеет порядок $O\left(\left|\frac{\lambda_i}{\lambda_j}\right|^k\right)$, но нельзя утверждать, что

$$a_{ij}^{(k)} = O\left(\left|\frac{\lambda_i}{\lambda_j}\right|^k\right) \quad \text{при } k \rightarrow \infty, \quad i > j.$$

Замечание 1. (Без доказательства). Если для матрицы A осуществим LR -алгоритм, то при применении QR -алгоритма собственные значения A получаются в правильном порядке

$$a_{ii}^{(k)} \rightarrow \lambda_i \quad \text{при } k \rightarrow \infty, \quad i = 1, 2, \dots, n,$$

и скорость сходимости матрицы A_k к треугольной дается соотношением

$$a_{ij}^{(k)} = O\left(\left|\frac{\lambda_i}{\lambda_j}\right|^k\right) \quad \text{при } k \rightarrow \infty, \quad i > j.$$

Замечание 2. QR -алгоритм сходится при существенно менее ограничительных условиях, чем это указано в теореме 1.

Применение алгоритма к матрице $A \in \mathbf{M}_n$ произвольного вида требует слишком большого числа арифметических операций ($Cn^3 + O(n^2)$, константа C зависит от метода построения QR -разложения).

§ 9.2.1. QR алгоритм нахождения собственных значений для почти треугольной матрицы

Лемма 2. Если матрица A — почти треугольная, то все матрицы A_k , $k = 1, 2, \dots$ в QR -алгоритме — почти треугольные.

Доказательство. Согласно (10) или (21) $Q = \prod_{i=1}^{n-1} T_{i,i+1}^t$ или $Q = \prod_{i=1}^n U_i$ (по утверждаемой в теоремах I.12.1) и I.13.1) единственности QR -разложения эти матрицы совпадают). Если матрица A — почти треугольная и $A = QR$ — ее QR -разложение, то матрица RQ будет почти треугольной, так как умножение треугольной матрицы R на матрицу $T_{i,i+1}^t$ или U_i справа заменяет ее i -й и $i+1$

столбцы на их линейную комбинацию, что в результате дает почти треугольную матрицу.

Эта лемма позволяет значительно ускорить работу QR -алгоритма. Перед его применением исходная матрица A приводится к почти треугольному виду A' унитарным подобием одним из алгоритмов, описанных в § I.14 и § I.15. Затем к матрице A' применяется QR -алгоритм.

Оценка количества арифметических операций на один шаг QR -алгоритма для почти треугольной матрицы для метода вращений

1) Построение QR -разложения матрицы $A_k = Q_k R_K$ требует $2n^2 + O(n)$ ($n \rightarrow \infty$) мультипликативных операций, $n^2 + O(n)$ ($n \rightarrow \infty$) аддитивных операций и $O(n)$ ($n \rightarrow \infty$) операций извлечения корня (которые по трудоемкости по порядку можно сравнить с операциями деления).

2) Подсчитаем затраты на вычисление произведения (10). Так как умножение треугольной матрицы R на матрицу $T_{i,i+1}^t$ справа изменяет ее i -й и $i+1$ столбцы, имеющие не более $i+1$ ненулевой элемент, то согласно лемме I.12.5 на это требуется $4(i+1)$ умножений и $2(i+1)$ сложений. Следовательно, на вычисление произведения (10) требуется $\sum_{i=1}^{n-1} 4(i+1) = 4n(n+1)/2 = 2n^2 + O(n)$ ($n \rightarrow \infty$) мультипликативных операций и $n^2 + O(n)$ аддитивных операций.

Следовательно, один шаг алгоритма для почти треугольной матрицы требует $4n^2 + O(n)$ ($n \rightarrow \infty$) мультипликативных и $2n^2 + O(n)$ ($n \rightarrow \infty$) аддитивных операций.

Оценка количества арифметических операций на один шаг QR -алгоритма для почти треугольной матрицы для метода отражений

1) Построение QR -разложения матрицы $A_k = Q_k R_K$ требует $(5/2)n^2 + O(n)$ ($n \rightarrow \infty$) мультипликативных операций, $(3/2)n^2 + O(n)$ ($n \rightarrow \infty$) аддитивных операций и $O(n)$ ($n \rightarrow \infty$) операций извлечения корня (которые по трудоемкости по порядку можно сравнить с операциями деления).

2) Подсчитаем затраты на вычисление произведения (21). Так как умножение треугольной матрицы R на матрицу U_i справа изменяет ее i -й и $i+1$ столбцы, имеющие не более $i+1$ ненулевой элемент, то Согласно лемме I.13.11 на это требуется $5(i+1)$ умножений и $3(i+1)$ сложений. Следовательно, на вычисление произведения (21) требуется $\sum_{i=1}^n 5(i+1) = 5(n+1)(n+2)/2 = (5/2)n^2 + O(n)$ ($n \rightarrow \infty$) мультипликативных операций и $(3/2)n^2 + O(n)$ аддитивных операций.

Следовательно, один шаг алгоритма для почти треугольной матрицы требует $5n^2 + O(n)$ ($n \rightarrow \infty$) мультипликативных и $3n^2 + O(n)$ ($n \rightarrow \infty$) аддитивных операций.

§ 9.2.2. QR алгоритм нахождения собственных значений для самосопряженной трехдиагональной матрицы

Лемма 3. Если матрица A — самосопряженная, то все матрицы A_k , $k = 1, 2, \dots$ в QR-алгоритме — самосопряженные.

Доказательство. Действительно, пусть $A_k = Q_k R_k$ — самосопряженная, т.е. $A_k^* = R_k^* Q_k^* = A_k$. Тогда $A_{k+1} = R_k Q_k$ и в силу унитарности Q_k имеем $A_{k+1}^* = Q_k^* R_k^* = Q_k^{-1} R_k^* = Q_k^{-1} R_k^* (Q_k^{-1} Q_k) = Q_k^{-1} (R_k^* Q_k) Q_k = Q_k^{-1} A_k^* Q_k = Q_k^{-1} A_k Q_k$, т.е. матрица A_{k+1} унитарно подобна самосопряженной матрице A_k и потому самосопряжена.

Лемма 4. Если матрица A — самосопряженная трехдиагональная, то все матрицы A_k , $k = 1, 2, \dots$ в QR-алгоритме — самосопряженные трехдиагональные.

Доказательство. В силу леммы 3 все матрицы A_k , $k = 1, 2, \dots$ — самосопряженные. Поскольку трехдиагональная матрица A является почти треугольной, то в силу леммы 2 все матрицы A_k , $k = 1, 2, \dots$ — почти треугольные. Итак, для всякого $k = 1, 2, \dots$ матрица A_k является самосопряженной и почти треугольной, и, следовательно, трехдиагональной.

Эти леммы позволяют значительно ускорить работу QR-алгоритма для самосопряженной матрицы. Перед его применением исходная матрица A приводится к трехдиагональному виду A' унитарным подобием одним из алгоритмов, описанных в § I.14 и § I.15. Затем к матрице A' применяется QR-алгоритм.

Шаг QR-алгоритма для самосопряженной трехдиагональной матрицы

Для целей QR-алгоритма описанное выше построение QR-разложения для трехдиагональной матрицы (см. стр. 120) может быть значительно ускорено.

1) В матрице R (25) элементы $r_{13}, r_{24}, \dots, r_{n-2,n}$ не вычисляются и не хранятся, поскольку, если $A = QR$ — самосопряженная трехдиагональная, то матрицу RQ можно вычислить, не используя эти элементы.

Действительно, поскольку при переходе от матрицы $A^{(k-1)}$ (23) к матрице $A^{(k)}$ (24) изменяются только k -я и $(k+1)$ -я строки матрицы $A^{(k-1)}$, то невычисление элемента $r_{k,k+2}$ не окажет влияния на остальные элементы матрицы $A^{(k)}$. Далее, согласно (10) или (21) $Q = \prod_{k=1}^{n-1} T_{k,k+1}^t$ или $Q = \prod_{k=1}^n U_k$, а умножение R на $T_{k,k+1}^t$ или U_k справа изменяет только k -й и $(k+1)$ -й столбцы матрицы R вида (25). Поэтому нижний треугольник произведения RQ можно вычислить, не используя элементы $r_{i,i+2}$, $i = 1, 2, \dots, n-2$. По лемме 4 матрица RQ — самосопряженная, поэтому ее верхний треугольник получается из нижнего транспонированием и комплексным сопряжением.

Это наблюдение всегда используют при реализации QR-алгоритма для самосопряженной трехдиагональной матрицы, поскольку оно не только ускоряет вычисления и экономит память ЭВМ, но обеспечивает сохранение самосопряженности матрицы вне зависимости от вносимой вычислительной погрешности.

2) При проведении QR -алгоритма для самосопряженной трехдиагональной матрицы нет необходимости хранить все матрицы, составляющие матрицу Q , достаточно хранить только последнюю (т.е. после k -го шага построения QR -разложения достаточно помнить только матрицу $T_{k,k+1}^t$ или U_k .

Действительно, после перехода от матрицы $A^{(k-1)}$ (23) к матрице $A^{(k)}$ (24) в алгоритме участвует только подматрица $(a_{ij}^{(k-1)})_{i,j=k+1,\dots,n}$ и столбцы $1, 2, \dots, k$ в дальнейших вычислениях не изменяются. Поэтому можно сразу умножить $A^{(k)}$ на матрицу $T_{k-1,k}^t$ или U_{k-1} справа (это изменяет только $(k-1)$ -й и k -й столбцы матрицы $A^{(k)}$). На последнем шаге ($k = n - 1$) надо умножить еще и на матрицу $T_{n-1,n}^t$ или U_{n-1} справа.

В результате такого процесса матрица A_k в QR -алгоритме нахождения собственных значений сразу перейдет в матрицу A_{k+1} (без построения QR -разложения матрицы A_k в явном виде).

Оценка количества арифметических операций на один шаг QR -алгоритма для самосопряженной трехдиагональной матрицы для метода вращений

1) Построение QR -разложения матрицы $A_k = Q_k R_k$ требует $14n + O(1)$ ($n \rightarrow \infty$) мультипликативных операций, $6n + O(1)$ ($n \rightarrow \infty$) аддитивных операций и $2n + O(1)$ ($n \rightarrow \infty$) операций извлечения корня (которые по трудоемкости по порядку можно сравнить с операциями деления).

2) Подсчитаем затраты на вычисление произведения (10). Так как умножение трехдиагональной матрицы R на матрицу $T_{i,i+1}^t$ справа изменяет ее i -й и $i + 1$ столбцы, имеющие не более трех ненулевых элементов, из которых один мы не вычисляем, то согласно лемме I.12.5 на это требуется 8 умножений и 4 сложения. Следовательно, на вычисление произведения (10) требуется $\sum_{i=1}^{n-1} 8 = 8(n-1)$ мультипликативных операций и $4(n-1)$ аддитивных операций.

Следовательно, один шаг алгоритма для трехдиагональной матрицы требует $22n + O(1)$ ($n \rightarrow \infty$) мультипликативных, $10n + O(1)$ ($n \rightarrow \infty$) аддитивных операций и $2n + O(1)$ ($n \rightarrow \infty$) операций извлечения корня (которые по трудоемкости по порядку можно сравнить с операциями деления).

Оценка количества арифметических операций на один шаг QR -алгоритма для самосопряженной трехдиагональной матрицы для метода отражений

1) Следовательно, всего для проведения алгоритма требуется выполнить не более $15n$ мультипликативных операций, $9n$ аддитивных операций и $2n$ операций извлечения корня (которые по трудоемкости по порядку можно сравнить с операциями деления).

2) Подсчитаем затраты на вычисление произведения (21). Так как умножение трехдиагональной матрицы R на матрицу U_i справа изменяет ее i -й и $i + 1$ столбцы, имеющие не более трех ненулевых элементов, из которых один мы не вычисляем, то согласно лемме I.13.11 на это требуется 10 умножений и 6

сложений. Следовательно, на вычисление произведения (21) требуется $\sum_{i=1}^n 10 = 10n$ мультипликативных операций и $6n$ аддитивных операций.

Следовательно, один шаг алгоритма для трехдиагональной матрицы требует $25n + O(1)$ ($n \rightarrow \infty$) мультипликативных, $15n + O(1)$ ($n \rightarrow \infty$) аддитивных операций и $2n + O(1)$ ($n \rightarrow \infty$) операций извлечения корня (которые по трудоемкости по порядку можно сравнить с операциями деления).

§ 9.3. Ускорение сходимости алгоритма

Рассмотрим способы, применяемые для ускорения сходимости последовательности матриц $\{A_k\}$ к диагональной матрице. Как отмечалось выше (см. стр. 104), эти способы во многом схожи со способами ускорения сходимости LR -алгоритма и алгоритма Холецкого. Также справедливы замечания 7.3 и 7.4.

Поскольку QR -алгоритм **никогда** не применяется для матриц произвольного вида, всюду ниже мы будем считать, что исходная матрица уже приведена унитарным подобием к почти треугольному или трехдиагональному виду. Таким образом, начальная матрица A_1 — почти треугольная (или трехдиагональная). По доказанному выше это означает, что все матрицы A_k — почти треугольные (трехдиагональные).

§ 9.3.1. Исчерпывание матрицы

Идея исчерпывания матрицы для QR -алгоритма та же, что и для LR -алгоритма (см. стр. 105).

§ 9.3.2. Сдвиги

Идея использования сдвигов для QR -алгоритма та же, что и для LR -алгоритма (см. стр. 106). Модифицированный QR -алгоритм, основанный на этой идеи, выглядит следующим образом.

Будем строить для матрицы $A \in M_n$ последовательность $\{A_k\}$ матриц $A_k \in M_n$ по следующим правилам:

- 1) $A_1 = A$;
- 2) для всех $k = 1, 2, \dots$ матрица A_{k+1} получается из матрицы A_k следующим образом:
 - a) определяем требуемый сдвиг s_k (его оптимальный выбор — отдельная задача),
 - б) строим QR -разложение матрицы $A_k - s_k I$: $A_k - s_k I = Q_k R_k$,
 - в) вычисляем матрицу A_{k+1} как произведение матриц R_k и Q_k плюс $s_k I$: $A_{k+1} = R_k L_k + s_k I$.

Нетрудно проверить, что матрица A_{k+1} (унитарно) подобна A_k : $A_{k+1} = R_k Q_k + s_k I = (Q_k^{-1} Q_k)(R_k Q_k + s_k I) = Q_k^{-1}(Q_k R_k)Q_k + s_k Q_k I = Q_k^{-1}(Q_k R_k + s_k I)Q_k = Q_k^{-1} A_k Q_k$ и, следовательно, все матрицы A_k , $k = 1, 2, \dots$ имеют те же собственные значения, что и матрица A .

§ 9.3.3. Практическая организация вычислений в QR алгоритме

Пусть требуется определить все собственные значения матрицы $A \in M_n$ с точностью ε .

Вначале приводим матрицу к почти треугольному виду A_1 унитарным подобием одним из алгоритмов, описанных в § I.14 и § I.15.

Затем к матрице A_1 применяем QR-алгоритм со сдвигами. На шаге k в качестве сдвига s_k возьмем $a_{nn}^{(k)}$, т.е. $s_k = a_{nn}^{(k)}$. Поскольку $a_{nn}^{(k)} \rightarrow \lambda_n$, то s_k является приближением к λ_n и скорость сходимости к нулю элемента $a_{n,n-1}^{(k)}$ будет очень высокой. Как только на некотором шаге k будет выполнено условие $|a_{n,n-1}^{(k)}| < \varepsilon \|A\|_\infty$, в качестве λ_n берем $a_{nn}^{(k)}$ и применяем алгоритм к подматрице $(a_{ij})_{i,j=1,2,\dots,n-1} \in M_{n-1}$ на 1 меньшей размерности. Так поступаем до тех пор, пока размерность матрицы не станет равной 2. Для этой матрицы собственные значения определяются как решения соответствующего квадратного уравнения.

§ 10. МЕТОД ОБРАТНОЙ ИТЕРАЦИИ НАХОЖДЕНИЯ СОБСТВЕННЫХ ВЕКТОРОВ

Метод обратной итерации позволяет найти собственный вектор, соответствующий однократному собственному значению λ диагонализируемой матрицы $A \in M_n$, если известно достаточно хорошее приближение $\hat{\lambda}$ к собственному значению λ .

Теорема 1 (Метод обратной итерации). *Пусть матрица $A \in M_n$ имеет полную систему ортонормированных собственных векторов e_i , $i = 1, \dots, n$: $Ae_i = \lambda_i e_i$, $(e_i, e_j) = \delta_{ij}$, причем собственное значение λ_m , $m = 1, 2, \dots, n$ однократное. Пусть $\hat{\lambda}_m \neq \lambda_m$ – достаточно хорошее приближение к λ_m :*

$$\max_{\substack{i=1,2,\dots,n \\ i \neq m}} \left| \frac{\lambda_m - \hat{\lambda}_m}{\lambda_i - \hat{\lambda}_m} \right| = q < 1. \quad (1)$$

Тогда для всякого вектора $x^{(0)} \in C^n$ такого, что $(x^{(0)}, e_m) \neq 0$, итерационный процесс

$$\begin{aligned} x^{(k+1)} &- \text{решение уравнения } (A - \hat{\lambda}_m I)x^{(k+1)} = x^{(k)} \\ e_m^{(k+1)} &= \frac{x^{(k+1)}}{\|x^{(k+1)}\|}, \quad k = 0, 1, \dots \end{aligned} \quad (2)$$

сходится к собственному вектору e_m (с точностью до постоянного множителя):

$$e_m^{(k)} \rightarrow e^{i\varphi} e_m \quad \text{при } k \rightarrow \infty$$

(где $e^{i\varphi}$ – число, по модулю равное 1). При этом существует такое $k_0 > 0$, что для всех $k \geq k_0$ выполнено неравенство

$$\|e_m^{(k)} - e^{i\varphi} e_m\| \leq C q^k, \quad (3)$$

где постоянная C не зависит от k .

Доказательство. Поскольку вектора e_1, \dots, e_n образуют базис в C^n , то $x^{(0)} = \sum_{i=1}^n c_i e_i$, причем по условию $c_m = (x^{(0)}, e_m) \neq 0$. Поскольку

$$x^{(k+1)} = (A - \hat{\lambda}_m I)^{-1} x^{(k)},$$

то

$$x^{(k+1)} = (A - \hat{\lambda}_m I)^{-k} x^{(0)}.$$

Так как собственные значения матрицы $A - \hat{\lambda}_m I$ равны $\lambda_i - \hat{\lambda}_m$, $i = 1, 2, \dots, n$, то собственные значения матрицы $(A - \hat{\lambda}_m I)^{-1}$ есть $(\lambda_i - \hat{\lambda}_m)^{-1}$, $i = 1, 2, \dots, n$, а матрицы $(A - \hat{\lambda}_m I)^{-1} = (\lambda_i - \hat{\lambda}_m)^{-k}$, $i = 1, 2, \dots, n$. Следовательно,

$$\begin{aligned} x^{(k)} &= (A - \hat{\lambda}_m I)^{-k} \sum_{i=1}^n c_i e_i = \sum_{i=1}^n \frac{c_i e_i}{(\lambda_i - \hat{\lambda}_m)^k} \\ &= \frac{c_m}{(\lambda_m - \hat{\lambda}_m)^k} \left(e_m + \sum_{\substack{i=1 \\ i \neq m}}^n \frac{c_i}{c_m} e_i \left(\frac{\lambda_m - \hat{\lambda}_m}{\lambda_i - \hat{\lambda}_m} \right)^k \right) \end{aligned}$$

В силу ортонормированности базиса $\{e_i\}$

$$\|x^{(k)}\|^2 = (x^{(k)}, x^{(k)}) = \frac{1}{(\lambda_m - \hat{\lambda}_m)^{2k}} \left(|c_m|^2 + \sum_{\substack{i=1 \\ i \neq m}}^n |c_i|^2 \left(\frac{\lambda_m - \hat{\lambda}_m}{\lambda_i - \hat{\lambda}_m} \right)^{2k} \right).$$

В силу (1) и равенства $\|x^{(0)}\|^2 = \sum_{i=1}^n |c_i|^2$ получаем

$$\frac{|c_m|}{(\lambda_m - \hat{\lambda}_m)^k} \leq \|x^{(k)}\| \leq \frac{|c_m|}{(\lambda_m - \hat{\lambda}_m)^k} \left(1 + \frac{\|x^{(0)}\|^2}{|c_m|^2} q^{2k} \right)^{1/2}$$

или

$$\left(1 + \frac{\|x^{(0)}\|^2}{|c_m|^2} q^{2k} \right)^{-1/2} \leq \frac{|c_m|}{(\lambda_m - \hat{\lambda}_m)^k} \frac{1}{\|x^{(k)}\|} \leq 1$$

Вычислим

$$\begin{aligned} \frac{x^{(k)}}{\|x^{(k)}\|} - \frac{c_m}{|c_m|} e_m &= \left(\frac{c_m}{(\lambda_m - \hat{\lambda}_m)^k} \frac{1}{\|x^{(k)}\|} - \frac{c_m}{|c_m|} \right) e_m \\ &\quad + \sum_{\substack{i=1 \\ i \neq m}}^n \frac{1}{\|x^{(k)}\|} \frac{c_m}{(\lambda_m - \hat{\lambda}_m)^k} \frac{c_i}{c_m} e_i \left(\frac{\lambda_m - \hat{\lambda}_m}{\lambda_i - \hat{\lambda}_m} \right)^k. \end{aligned}$$

В силу (1), (2) и равенств $\|e_i\| = 1$, $i = 1, 2, \dots, n$, обозначив $e^{i\varphi} = c_m/|c_m|$, находим

$$\|e_m^{(k)} - e^{i\varphi} e_m\| \leq \left(1 - \left(1 + \frac{\|x^{(0)}\|^2}{|c_m|^2} q^{2k}\right)^{-1/2}\right) + \sum_{\substack{i=1 \\ i \neq m}}^n \frac{|c_i|}{|c_m|} q^k.$$

Так как

$$\sum_{i=1}^n |c_i| \leq \sqrt{n} \left(\sum_{i=1}^n |c_i|^2 \right)^{1/2} = \sqrt{n} \|x^{(0)}\|,$$

то

$$\|e_m^{(k)} - e^{i\varphi} e_m\| \leq \left(1 - \left(1 + \frac{\|x^{(0)}\|^2}{|c_m|^2} q^{2k}\right)^{-1/2}\right) + \sqrt{n} \frac{\|x^{(0)}\|}{|c_m|} q^k$$

В силу (1) найдется $k_0 > 0$, такое, что для всех $k \geq k_0$ выполнено $\alpha \equiv \frac{\|x^{(0)}\|^2}{|c_m|^2} q^{2k} < 1$. Тогда $(1 - \alpha)^{-1/2} \geq 1 - \alpha/2$ и $1 - (1 - \alpha)^{-1/2} \leq \alpha/2 \leq \alpha^{1/2}/2$.

Поэтому для всех $k \geq k_0$ справедливо неравенство

$$\|e_m^{(k)} - e^{i\varphi} e_m\| \leq \frac{\|x^{(0)}\|}{|c_m|} (1/2 + \sqrt{n}) q^k.$$

Это неравенство является требуемой оценкой (3).

ПРОГРАММА КУРСА

1. Матричные нормы. Подчиненные матричные нормы.
2. Максимальная строчная и максимальная столбцовая нормы. Их подчиненность.
3. Спектральная норма и ее свойства. Спектральный радиус и его свойства.
4. Обратимость матрицы, близкой к обратимой (теорема Банаха).
5. Оценка относительной погрешности решения линейной системы через относительные погрешности в матрице системы и в правой части. Число обусловленности.
6. Оценка относительной погрешности решения линейной системы через ненеизьюк. Свойства числа обусловленности.
7. Метод Гаусса. Оценка числа арифметических операций.
8. Представление метода Гаусса в виде последовательности элементарных преобразований. LU -разложение.
9. Алгоритм построения LU -разложения. Оценка числа арифметических операций.
10. Критерий осуществимости метода Гаусса.
11. Метод Гаусса для ленточных матриц. Оценка числа арифметических операций.
12. Алгоритм построения LU -разложения для трехдиагональных матриц. Оценка числа арифметических операций. Организация хранения матриц в памяти ЭВМ.
13. Метод прогонки для трехдиагональных матриц. Оценка числа арифметических операций. Организация хранения матриц в памяти ЭВМ.
14. Задача обращения матрицы. Обращение матрицы с помощью LU -разложения. Оценка числа арифметических операций.

15. Метод Гаусса с выбором главного элемента. Критерий осуществимости. Способы программной реализации.
16. Метод Жордана (Гаусса-Жордана) решения систем линейных уравнений. Оценка числа арифметических операций.
17. Положительно определенные матрицы. Осуществимость LU -разложения для положительно определенных матриц.
18. Теорема о разложении Холецкого для самосопряженной матрицы.
19. Метод Холецкого (квадратного корня) решения систем линейных уравнений. Организация процесса вычислений и хранения матриц в памяти ЭВМ. Оценка числа арифметических операций.
20. Метод ортогонализации решения систем линейных уравнений. Оценка числа арифметических операций.
21. Матрица элементарного вращения и ее свойства (геометрический смысл, затраты на вычисление произведений на вектор и матрицу).
22. Метод вращений решения систем линейных уравнений. Осуществимость. Оценка числа арифметических операций.
23. Теорема о построении QR -разложения методом вращений. Единственность разложения. Способы хранения матриц Q и R в памяти ЭВМ. Оценка числа арифметических операций, необходимых для построения QR - разложения.
24. Матрица отражения и ее свойства (геометрический смысл, затраты на вычисление произведений на вектор и матрицу).
25. Метод отражений решения систем линейных уравнений. Осуществимость. Оценка числа арифметических операций.
26. Теорема о построении QR -разложения методом отражений. Единственность разложения. Способы хранения матриц Q и R в памяти ЭВМ. Оценка числа арифметических операций, необходимых для построения QR -разложения.
27. Приведение матрицы к почти треугольному виду унитарным подобием методом вращений. Осуществимость. Оценка числа арифметических операций.
28. Приведение симметричной матрицы к трехдиагональному виду унитарным подобием методом вращений. Осуществимость. Оценка числа арифметических операций.

29. Приведение матрицы к почти треугольному виду унитарным подобием методом отражений. Осуществимость. Оценка числа арифметических операций.
30. Приведение симметричной матрицы к трехдиагональному виду унитарным подобием методом отражений. Осуществимость. Оценка числа арифметических операций.
31. Локализация собственных значений. Теорема о кругах Гершгорина.
32. Оценка погрешности нахождения собственных значений через погрешность в матрице системы для диагонализируемых матриц.
33. Степенной метод поиска максимального собственного значения. Достаточные условия сходимости. Оценка числа арифметических операций на один шаг алгоритма.
34. Общий вид методов нахождения собственных значений, минимизирующих сумму квадратов внедиагональных элементов матрицы. Теорема о сходимости таких методов (достаточное условие окончания итераций).
35. Преобразование элементарного вращения, используемое в методе вращений Якоби. Вид формул, обеспечивающий меньшее накопление вычислительной погрешности.
36. Стратегии выбора очередного обнуляемого элемента в методе Якоби. Оценка скорости сходимости в случае обнуления максимального по модулю внедиагонального элемента. Оценка числа арифметических операций на один шаг алгоритма для каждой из стратегий.
37. Вычисление k -го по величине собственного значения методом бисекции. Оценка количества итераций. Способы вычисления числа перемен знака в последовательности главных миноров. Организация процесса вычислений для предотвращения переполнения или потери точности.
38. Вычисление всех собственных значений матрицы на заданном интервале методом бисекции. Оценка количества итераций. Способы вычисления числа перемен знака в последовательности главных миноров. Организация процесса вычислений для предотвращения переполнения или потери точности.
39. Теорема о LR -разложении. LR -алгоритм поиска собственных значений, его осуществимость. Достаточные условия сходимости (без доказательства).
40. Алгоритм построения LR -разложения. Сохранение почти треугольного вида матрицы в LR -алгоритме нахождения собственных значений.

41. Алгоритм построения LR -разложения для почти треугольной матрицы. Расчетные формулы LR -алгоритма нахождения собственных значений для почти треугольной матрицы. Оценка числа арифметических операций на один шаг алгоритма.
42. Ускорение сходимости LR -алгоритма с помощью исчерпывания матрицы и сдвигов (без доказательства). Пример выбора сдвига и исчерпывания матрицы.
43. Теорема о разложении Холецкого, используемом в методе Холецкого поиска собственных значений симметричной матрицы. Метод Холецкого поиска собственных значений симметричной матрицы, его осуществимость. Достаточные условия сходимости (без доказательства).
44. Алгоритм построения разложения Холецкого, используемом в методе Холецкого поиска собственных значений симметричной матрицы. Сохранение трехдиагонального вида матрицы в методе Холецкого поиска собственных значений симметричной матрицы.
45. Алгоритм построения разложения Холецкого, используемом в методе Холецкого поиска собственных значений симметричной матрицы, для трехдиагональной матрицы. Расчетные формулы метода Холецкого поиска собственных значений для трехдиагональной матрицы. Оценка числа арифметических операций на один шаг алгоритма.
46. Ускорение сходимости метода Холецкого поиска собственных значений с помощью исчерпывания матрицы и сдвигов (без доказательства). Пример выбора сдвига и исчерпывания матрицы.
47. QR -алгоритм поиска собственных значений, его осуществимость. Достаточные условия сходимости (без доказательства).
48. Сохранение почти треугольного вида матрицы в QR -алгоритме нахождения собственных значений.
49. Алгоритм построения QR -разложения для почти треугольной матрицы. Расчетные формулы QR -алгоритма нахождения собственных значений для почти треугольной матрицы. Оценка числа арифметических операций на один шаг алгоритма.
50. Ускорение сходимости QR -алгоритма с помощью исчерпывания матрицы и сдвигов (без доказательства). Пример выбора сдвига и исчерпывания матрицы.

ЛИТЕРАТУРА

1. Практикум по алгебре. Под ред. Н.С. Бахвалова, А.И. Кострикина. Изд.-во МГУ, 1983.
2. Ю.П. Размыслов, С.Я. Ищенко. Практикум по вычислительным методам алгебры. Изд.-во МГУ, 1988.
3. Н.С. Бахвалов, Н.П. Жидков, Г.М. Кобельков. Численные методы. М., "Наука", 1987.
4. Д.К. Фаддеев, В.К. Фаддеева. Вычислительные методы линейной алгебры, 1963.
5. Дж.Х. Уилкинсон. Алгебраическая проблема собственных значений. М., "Наука", 1970.
6. Р. Хорн, Ч. Джонсон. Матричный анализ. М., "Мир", 1989.
7. А.А. Самарский, А.В. Гулин. Численные методы. М., "Наука", 1989.
8. С. Писсанецки. Технология разреженных матриц. М., "Мир", 1988.