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

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: $x_k \to x$ при $k \to \infty$. 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: $\lambda_k \to \lambda$ при $k \to \infty$. 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 $A \in \mathbf{M}_n$ при $n \geq 5$. Другими словами, за конечное число арифметических операций нельзя найти все собственные значения произвольной матрицы $A \in \mathbf{M}_n$ при $n \geq 5$.*

§ 2. ЛОКАЛИЗАЦИЯ СОБСТВЕННЫХ ЗНАЧЕНИЙ

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

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

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

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

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

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 $x = (x_1, \ldots, x_n) \neq 0$ – соответствующий собственный вектор: $Ax = \lambda x$. Обозначим $|x_p| = \max_{i=1,\ldots,n} |x_i| \neq 0$. 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 $D = \mathrm{diag}(a_{11}, \ldots, a_{nn}) \in \mathbf{M}_n$, $B = A - D$; $A_\varepsilon = D + \varepsilon B$, $\varepsilon > 0$. 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

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

причем по условию

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

Рассмотрим введенные множества для матрицы A_ε:

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

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

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

причем множество $G_k(A_1) = G_k(A)$ связно по условию. В силу (2), (3) $G_k(A_\varepsilon) \bigcap G_{n-k}(A_\varepsilon) = \emptyset$ для всех $\varepsilon \in [0,1]$. Обозначим через $\lambda_i(A_\varepsilon)$ i-е собственное значение матрицы A_ε, $\sigma(A_\varepsilon) = \{\lambda_1(A_\varepsilon), \ldots, \lambda_n(A_\varepsilon)\}$ – 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 фунциями параметра ε. Таким образом, как композиции непрерывных функций, собственные значения $\lambda_i(A_\varepsilon)$ матрицы A_ε являются непрерывными фунциями параметра ε.

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

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

По доказанному выше

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

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

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

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

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

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 $A \in \mathbf{M}_n$ Пусть ß– алгоритм нахождения собственных значений, $\widehat{\sigma}(A) = ß(A)$ – 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 $\widehat{\sigma}(A)$, т.е. $\widehat{\sigma}(A) = \sigma(A+E)$. Оценим погрешность при нахождении собственных значений (т.е. $\widehat{\sigma}(A) - \sigma(A)$) через величину матрицы E.

Лемма 1. *Пусть $D = \operatorname{diag}(\lambda_1, \dots, \lambda_n)$ – диагональная матрица, $E = (e_{ij}) \in \mathbf{M}_n$. Тогда для всякого $\widehat{\lambda} \in \sigma(D+E)$ – собственного значения матрицы $D+E$, существует $\lambda = \lambda_i \in \sigma(D)$ – собственное значение матрицы D, такое, что*

$$|\widehat{\lambda} - \lambda| \le \|E\|_\infty$$

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

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

$$\widehat{\lambda} \in \left\{ z \in \mathbf{C}^n \ : \ |z - \lambda_i - e_{ii}| \le R_i'(E) = \sum_{\substack{j=1 \\ j \ne i}}^{n} |e_{ij}| \right\}.$$

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

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

Но

$$\sum_{j=1}^{n} |e_{ij}| \le \max_{i=1,\dots,n} \sum_{j=1}^{n} |e_{ij}| = \|E\|_\infty.$$

Поэтому

$$\widehat{\lambda} \in \{ z \in \mathbf{C}^n \ : \ |z - \lambda_i| \le \|E\|_\infty \}$$

что доказывает утверждение леммы.

Теорема 1. *Пусть $A \in \mathbf{M}_n$ – диагонализируемая матрица, т.е. существуют невырожденная матрица C и диагональная матрица $\Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$ такие, что $A = C\Lambda C^{-1}$. Пусть $E = (e_{ij}) \in \mathbf{M}_n$. Тогда для всякого $\widehat{\lambda} \in \sigma(A + E)$ – собственного значения матрицы $A + E$, существует $\lambda = \lambda_i \in \sigma(A)$ – собственное значение матрицы A, такое, что*

$$|\widehat{\lambda} - \lambda| \le \kappa_\infty(C)\|E\|_\infty$$

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 $\widehat{\lambda} \in \sigma(A + E) = \sigma(\Lambda + C^{-1}EC)$ существует $\lambda = \lambda_i \in \sigma(\Lambda) = \sigma(A)$ такое, что

$$|\widehat{\lambda} - \lambda| \le \|C^{-1}EC\|_\infty \le \|C^{-1}\|_\infty\|E\|_\infty\|C\|_\infty = \kappa_\infty(C)\|E\|_\infty.$$

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, $i = 1, \ldots, n$: $Ae_i = \lambda_i e_i$, $(e_i, e_j) = \delta_{ij}$, причем $|\lambda_1| > |\lambda_2| \ge |\lambda_3| \ge \ldots \ge |\lambda_n$ (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 $(x^{(0)}, e_1) \ne 0$, итерационный процесс*

$$x^{(k+1)} = Ax^{(k)}, \qquad \lambda_1^{(k)} = \frac{(x^{(k+1)}, x^{(k)})}{(x^{(k)}, x^{(k)})}, \qquad k = 0, 1, \ldots \tag{1}$$

сходится к собственному значению λ_1 (собственному значению с максимальным модулем), причем

$$\lambda_1^{(k)} = \lambda_1 + O\left(\lambda_1 \left|\frac{\lambda_2}{\lambda_1}\right|^k\right) \quad (k \to \infty),$$

а величины $e_1^{(k)} = \dfrac{x^{(k)}}{\|x^{(k)}\|}$ *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 $e_1, \ldots, e_n$ образуют базис в C^n, то $x^{(0)} = \sum\limits_{i=1}^{n} c_i e_i$, 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 $x^{(k)} = c_1\lambda_1^k e_1 + O\left(|\lambda_2^k|\right)$, $x^{(k+1)} = c_1\lambda_1^{k+1} e_1 + O\left(|\lambda_2^{k+1}|\right)$. Далее, вычислим

$$\begin{aligned}
\|x^{(k)}\| = (x^{(k)}, x^{(k)}) &= \left(c_1\lambda_1^k e_1 + O\left(|\lambda_2^k|\right), c_1\lambda_1^k e_1 + O\left(|\lambda_2^k|\right)\right) \\
&= |c_1|^2|\lambda_1|^{2k} + O\left(|\lambda_1^k||\lambda_2^k|\right); \\
(x^{(k+1)}, x^{(k)}) &= \left(c_1\lambda_1^{k+1} e_1 + O\left(|\lambda_2^{k+1}|\right), c_1\lambda_1^k e_1 + O\left(|\lambda_2^k|\right)\right) \\
&= \lambda_1\left(|c_1|^2|\lambda_1|^{2k} + O\left(|\lambda_1^k||\lambda_2^k|\right)\right).
\end{aligned}$$

Следовательно,

$$\begin{aligned}
\lambda_1^{(k)} = \frac{(x^{(k+1)}, x^{(k)})}{(x^{(k)}, x^{(k)})} = \frac{\lambda_1\left(|c_1|^2|\lambda_1|^{2k} + O\left(|\lambda_1^k||\lambda_2^k|\right)\right)}{|c_1|^2|\lambda_1|^{2k} + O\left(|\lambda_1^k||\lambda_2^k|\right)} &= \lambda_1 \frac{1 + O\left(\frac{1}{|c_1|^2}\left|\frac{\lambda_2}{\lambda_1}\right|^k\right)}{1 + O\left(\frac{1}{|c_1|^2}\left|\frac{\lambda_2}{\lambda_1}\right|^k\right)} \\
&= \lambda_1 + O\left(\lambda_1\left|\frac{\lambda_2}{\lambda_1}\right|^k\right)
\end{aligned}$$

Аналогично,

$$\begin{aligned}
e_1^{(k)} = \frac{x^{(k)}}{\|x^{(k)}\|} = \frac{c_1\lambda_1^k e_1 + O\left(|\lambda_2^k|\right)}{\left(|c_1|^2|\lambda_1|^{2k} + O\left(|\lambda_1^k||\lambda_2^k|\right)\right)^{1/2}} &= \frac{\frac{c_1\lambda_1^k}{|c_1||\lambda_1|^k} e_1 + O\left(\frac{1}{|c_1|}\left|\frac{\lambda_2}{\lambda_1}\right|^k\right)}{\left(1 + O\left(\frac{1}{|c_1|}\left|\frac{\lambda_2}{\lambda_1}\right|^k\right)\right)^{1/2}} \\
&= e^{i\varphi} e_1 + O\left(\left|\frac{\lambda_2}{\lambda_1}\right|^k\right),
\end{aligned}$$

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 $\|x^{(k)}\| = c_1\lambda_1^k e_1 + O\left(|\lambda_2^k|\right) \to \infty$ при $k \to \infty$; Если $|\lambda_1| < 1$, то $\|x^{(k)}\| = c_1\lambda_1^k e_1 + O\left(|\lambda_2^k|\right) \to 0$ при $k \to \infty$. 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 $x^{(k+1)} = \dfrac{x^{(k+1)}}{\|x^{(k+1)}\|}$. 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

$$\Sigma(B) = \sum_{\substack{i,j \\ i \neq j}}^{n} |b_{ij}|^2$$

– сумма квадратов внедиагольных 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 $\widehat{O} \in O(n)$ и диагональная матрица $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ такие, что $A = \widehat{O}\Lambda\widehat{O}^*$. Отсюда $\Lambda = \widehat{O}^* A \widehat{O} = OAO^*$, где обозначено $O = \widehat{O}^* \in O(n)$. Очевидно, что для диагональной матрицы $\Sigma(\Lambda) = 0$ и для всякой матрицы B $\Sigma(B) \geq 0$. Следовательно

а) $\Sigma(OAO^*) \geq 0$ для всякой $O \in O(n)$;

б) $\Sigma(OAO^*) = \Sigma(\Lambda) = 0$, если $O = \widehat{O}^*$.

Следовательно, матрица $O = \widehat{O}^*$ является решением задачи минимизации функционала $\Sigma(OAO^*)$ на группе ортогональных матриц:

$$\Sigma(OAO^*) \to \min_{O \in O(n)}.$$

Если мы найдем какое-то решение O_1 этой задачи, т.е. $\Sigma(O_1 A O_1^*) = 0$, то матрица $\Sigma(O_1 A O_1^*)$ диагональна и ортогонально подобна матрице A. Следовательно, на ее диагонали стоят искомые собственные значения матрицы A.

Будем строить последовательность симметричных матриц

$$A = A_0, A_1, \ldots, A_k, \ldots \tag{1}$$

такую, что для всякого $k = 1, 2, \ldots$:

1) Следующая матрица ортогонально подобна предыдущей (и потому ортогонально подобна исходной)

$$A_k = O_k A_{k-1} O_k^*, \qquad O_k \in O(n). \tag{2}$$

2) Сумма квадратов внедиагональных элементов следующей матрицы строго меньше суммы квадратов внедиагональных элементов предыдущей матрицы:

$$\Sigma(A_k) < \Sigma(A_{k-1}) \tag{3}$$

т.е. последовательность $\{\Sigma(A_k)\}_{k=1}^{\infty}$ строго монотонно убывает, что в силу $\Sigma(A_k) \geq 0$ гарантирует существование предела $\lim\limits_{k\to\infty} \Sigma(A_k)$.

3) Этот предел равен нулю:

$$\lim_{k\to\infty} \Sigma(A_k) = 0. \tag{4}$$

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 $0 \le \Sigma(A_{k_0}) < \varepsilon$. 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, $1 \le i \le n$, такое, что

$$|\lambda - a_{ii}^{(k_0)}| \le R_i'(A_{k_0}) = \sum_{\substack{j=1\\ j\ne i}}^{n} |a_{ij}^{(k_0)}| \le \sqrt{\sum_{\substack{j=1\\ j\ne i}}^{n} 1^2}\ \sqrt{\sum_{\substack{j=1\\ j\ne i}}^{n} |a_{ij}^{(k_0)}|^2}$$
$$\le \sqrt{n-1}\sqrt{\Sigma(A_{k_0})} < \sqrt{n-1}\sqrt{\varepsilon}$$

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

$$\Sigma(B) - \Sigma(A) = \sum_{\substack{l,m=1\\ l\ne m}}^{n} (b_{lm}^2 - a_{lm}^2).$$

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 $l \ne i, j$, $m \ne i, j$. Следовательно,

$$\Sigma(B) - \Sigma(A) = \sum_{\substack{m=1\\ m\ne i,j}}^{n} (b_{im}^2 - a_{im}^2) + \sum_{\substack{m=1\\ m\ne i,j}}^{n} (b_{jm}^2 - a_{jm}^2) + \sum_{\substack{l=1\\ l\ne i,j}}^{n} (b_{li}^2 - a_{li}^2) + \sum_{\substack{l=1\\ l\ne i,j}}^{n} (b_{lj}^2 - a_{lj}^2)$$
$$+ (b_{ij}^2 - a_{ij}^2) + (b_{ji}^2 - a_{ji}^2)$$
$$= \sum_{\substack{m=1\\ m\ne i,j}}^{n} ((b_{im}^2 + b_{jm}^2) - (a_{im}^2 + a_{jm}^2)) + \sum_{\substack{l=1\\ l\ne i,j}}^{n} ((b_{li}^2 + b_{lj}^2) - (a_{li}^2 + a_{lj}^2)) + 2(b_{ij}^2 - a_{ij}^2)$$

(здесь мы использовали симметричность матриц A и B: $a_{ij} = a_{ji}$, $b_{ij} = b_{ji}$ для всех $i, j = 1, \ldots, n$). 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 $m, l = 1, \ldots, n$, $m, l \ne i, j$:

$$\begin{pmatrix} b_{im} \\ b_{jm} \end{pmatrix} = \begin{pmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{pmatrix} \begin{pmatrix} a_{im} \\ a_{jm} \end{pmatrix} \qquad (5)$$

и

$$(b_{li}, b_{lj}) = (a_{li}, a_{lj}) \begin{pmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{pmatrix}^t \tag{6}$$

Матрицы преобразования в (5) и (6) ортогональны и потому сохраняют длины (двумерных) векторов. Поэтому

$$\begin{aligned} b_{im}^2 + b_{jm}^2 &= a_{im}^2 + a_{jm}^2 \qquad m = 1, \ldots, n, \;\; m \neq i, j; \\ b_{li}^2 + b_{lj}^2 &= a_{li}^2 + a_{lj}^2 \qquad l = 1, \ldots, n, \;\; l \neq i, j. \end{aligned}$$

Следовательно,

$$\Sigma(B) - \Sigma(A) = 2(b_{ij}^2 - a_{ij}^2). \tag{7}$$

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

$$\begin{cases} \operatorname{tg} 2\varphi = -\dfrac{2a_{ij}}{a_{ii} - a_{jj}}, & \text{если } \; a_{ii} \neq a_{jj}, \\ \varphi \quad\;\; = \dfrac{\pi}{4}, & \text{если } \; a_{ii} = a_{jj}. \end{cases}$$

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

$$\cos\varphi = \left(\frac{1}{2}\left(1 + \frac{1}{(1 + \operatorname{tg}^2 2\varphi)^{1/2}}\right)\right)^{1/2},$$

$$\sin\varphi = \operatorname{sign}(\operatorname{tg} 2\varphi)\left(\frac{1}{2}\left(1 - \frac{1}{(1 + \operatorname{tg}^2 2\varphi)^{1/2}}\right)\right)^{1/2}$$

Обозначим $x = -2a_{ij}$, $y = a_{ii} - a_{jj}$. Тогда

$$\operatorname{tg} 2\varphi = \frac{x}{y}, \quad \cos 2\varphi = \frac{1}{\left(1 + \dfrac{x^2}{y^2}\right)^{1/2}} = \frac{|y|}{(x^2+y^2)^{1/2}},$$

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 $|y| \to \infty$ числа 1 и $\dfrac{|y|}{(x^2+y^2)^{1/2}}$ могут оказаться близки и при вычислении их разности возникает большая вычислительная погрешность. Поэтому $\sin\varphi$ вычисляют по формуле

$$\sin\varphi = \frac{\sin 2\varphi}{2\cos\varphi} = \frac{\operatorname{tg} 2\varphi \cos 2\varphi}{2\cos\varphi} = \frac{\dfrac{x}{y}\dfrac{|y|}{(x^2+y^2)^{1/2}}}{2\cos\varphi}$$
$$= \frac{x\operatorname{sign}(y)}{2\cos\varphi\,(x^2+y^2)^{1/2}} = \frac{\operatorname{sign}(xy)|x|}{2\cos\varphi\,(x^2+y^2)^{1/2}}$$

Окончательно, расчетные формулы имеют вид:

$$\begin{array}{llll} \cos\varphi = \dfrac{1}{\sqrt{2}}, & \sin\varphi = \dfrac{1}{\sqrt{2}} & \text{при } y = 0 \\ \cos\varphi = \left(\dfrac{1}{2}\left(1 + \dfrac{|y|}{(x^2+y^2)^{1/2}}\right)\right)^{1/2}, & \sin\varphi = \dfrac{\operatorname{sign}(xy)|x|}{2\cos\varphi\,(x^2+y^2)^{1/2}} & \text{при } y \neq 0 \end{array} \tag{8}$$

(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}:

$$|a_{ij}^{(k-1)}| = \max_{\substack{l,m=1 \\ l\neq m}} |a_{lm}^{(k-1)}| \tag{9}$$

Лемма 1. *При выборе* (9) *обнуляемого элемента условие* (4) *выполнено.*

Доказательство. При любой стратегии выбора элемента $a_{ij}^{(k-1)}$ из (7) следует, что

$$\Sigma(A_k) = \Sigma(A_{k-1}) - |a_{ij}^{(k-1)}|^2. \tag{10}$$

Пусть $a_{ij}^{(k-1)}$ выбран как (9). Тогда

$$\Sigma(A_{k-1}) = \sum_{\substack{l,m=1 \\ l\neq m}}^{n} |a_{ml}^{(k-1)}|^2 \le \sum_{\substack{l,m=1 \\ l\neq m}}^{n} |a_{ij}^{(k-1)}|^2 = n(n-1)|a_{ij}^{(k-1)}|^2.$$

Следовательно,

$$|a_{ij}^{(k-1)}|^2 \ge \frac{1}{n(n-1)}\Sigma(A_{k-1}).$$

Поэтому из (10)

$$\Sigma(A_k) \le \Sigma(A_{k-1}) - \frac{2}{n(n-1)}\Sigma(A_{k-1}) = \left(1 - \frac{2}{n(n-1)}\right)\Sigma(A_{k-1})$$
$$\le \left(1 - \frac{2}{n(n-1)}\right)\left(1 - \frac{2}{n(n-1)}\right)\Sigma(A_{k-2}) \le \ldots \le \left(1 - \frac{2}{n(n-1)}\right)^k \Sigma(A_0).$$

Поскольку $q = 1 - \dfrac{2}{n(n-1)} \in (0,1)$, то $q^k \to 0$ при $k \to \infty$, и

$$\Sigma(A_k) \le q^k \Sigma(A_0) \to 0 \quad \text{при} \quad k \to \infty.$$

Следствие 1. *При $k \to \infty$ 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 $\dfrac{n(n-1)}{2}$ 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 $\dfrac{n(n-1)}{2}$ 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:

$$a_{12}, a_{13}, \ldots, a_{1n}, a_{23}, a_{24}, \ldots, a_{2n}, a_{34}, \ldots, a_{3n}, \ldots, a_{n-1,n},$$

и в качестве обнуляемого элемента на шаге k выбирается элемент с номером $k(\mathrm{mod}\left(\dfrac{n(n-1)}{2}\right)$ из этого списка. Другими словами, в качестве номера (i,j) обнуляемого элемента последовательно выбирают

$$(1,2), (1,3), \ldots, (1,n), (2,3), (2,4), \ldots, (2,n), (3,4), \ldots, (3,n), \ldots, (n-1,n).$$

Затем этот цикл повторяется, и так до тех пор, пока на некотором шаге k_0 не бедет 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

$$\sum_{\substack{j=1\\ j\neq i}}^{n} |a_{ij}^{(k-1)}|^2 = \max_{l=1,\ldots,n} \sum_{\substack{j=1\\ j\neq l}}^{n} |a_{lj}^{(k-1)}|^2,$$

а j — из условия

$$|a_{ij}^{(k-1)}| = \max_{\substack{m=1,\ldots,n\\ m\neq i}} |a_{im}^{(k-1)}|. \tag{11}$$

Для уменьшения вычислительных расходов поступают следующим образом. Кроме матрицы A_k хранится вектор $b^{(k)}$, компоненты которого на k-м шаге равны

$$b_l^{(k)} = \sum_{\substack{j=1\\ j\neq l}}^{n} |a_{lj}^{(k)}|^2, \tag{12}$$

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

$$b_l^{(k)} = b_l^{(k-1)}, \quad l = 1, \ldots, n, \ \ l \neq i, j. \tag{13}$$

С использованием вектора $b^{(k-1)}$ выбор очередного обнуляемого элемента $a_{ij}^{(k-1)}$ осуществляется следующим образом: i определяется из условия

$$|b_i^{(k-1)}| = \max_{l=1,\ldots,n} |b_l^{(k-1)}| \tag{14}$$

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 $T_{ij}(\varphi)AT_{ij}^t(\varphi)$ 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 значения 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 $b_i - a_i > \varepsilon$ $(i = 0, 1, \ldots)$ будем вычислять $c_{i+1} = \dfrac{a_i + b_i}{2}$. 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 $\lambda_k = \dfrac{a_i + b_i}{2}$ (которое является $(n_-(b_i) - n_-(a_i))$-кратным собственным значением).

На каждом шаге $i = 0, 1, \ldots$ выполнено $\lambda_k \in (a_i, b_i)$, при этом длина интервала $b_i - a_i = \dfrac{b_0 - a_0}{2^i}$, $i = 0, 1, \ldots$. Следовательно, для достижения точности ε (т.е. $b_i - a_i < \varepsilon$) надо сделать число шагов m, определяемое из условия $\dfrac{b_0 - a_0}{2^m} < \varepsilon$, 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 $[a,\dfrac{a+b}{2}]$ с точностью ε;

определить все собственные значения на отрезке $[\dfrac{a+b}{2},b]$ с точностью ε.

иначе, если $n_-(b)-n_-(a)\neq 0$, то $\dfrac{a+b}{2}$ 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 програмным 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 $\lambda_{k_1+1},\ldots,\lambda_{k_2}$ с учетом кратности). Положим $\lambda_{k_1}=a$.

Для всех $k=k_1+1,\ldots,k_2$ 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 значений 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 $1,\delta_1,\delta_2,\ldots,\delta_n$ 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 $1, \delta_1, \delta_2, \ldots, \delta_n$ 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 — соответсвенно 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 $\det L_k = \prod\limits_{j=1}^{n} l_{jj}$, $\det U_k = 1$. Поэтому $\delta_k = l_{11} \ldots l_{kk}$ и число перемен знака в последовательности $1, \delta_1, \delta_2, \ldots, \delta_n$ равно числу перемен знака в последовательности $1, l_{11}, l_{22}, \ldots, l_{nn}$.

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 реккурентных формул

Пусть требуется вычислить число перемен знака в последовательности $1, \delta_1, \delta_2, \ldots, \delta_n$ 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 $k = 3, \ldots, n-1$ $\delta_k = \det A_k$, $\delta_{k-1} = \det A_{k-1}$ 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

$$\alpha = 4\max\left\{\max_{i=1,\ldots,n}\{|a_i|\},\ \max_{i=1,\ldots,n-1}\{|b_i|\}\right\}$$

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 $\operatorname{sign}(xy) < 0$, число перемен знака $m = 1$, иначе $m = 0$.

2) Для всех $k = 2, 3, \ldots, n$ вычисляем:

   а) $a = a_k$, $b = b_{k-1}$;

   б) $\gamma = (1/\varepsilon_{mash})/\max\{|x|, |b(by)|\}$;

   в) $u = \gamma(ax - b^2y)$, $v = \gamma x$;

   г) если $\operatorname{sign}(ux) < 0$, 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 $L \in \mathrm{LT}(n)$ с 1 на главной диагонали, $R \in \mathrm{RT}(n)$.*

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 $\widehat{L} \in \mathrm{LT}(n)$, $\widehat{U} \in \mathrm{RT}(n)$ с 1 на главной диагонали. Следовательно, $A = \widehat{U}^t\widehat{L}^t \equiv LR$, где $L = \widehat{U}^t \in \mathrm{LT}(n)$ с 1 на главной диагонали, $R = \widehat{L}^t \in \mathrm{RT}(n)$. 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.

$$\sum_{j=1}^{n} l_{ij}\, r_{jk} = a_{ik}, \qquad i,k = 1,\ldots,n. \tag{1}$$

Поскольку $l_{ij} = 0$ при $i < j$, $l_{jj} = 1$, $r_{jk} = 0$ 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

$$\sum_{j=1}^{\min\{i,k\}} l_{ij}\, r_{jk} = a_{ik}, \qquad i,k = 1,\ldots,n,$$

или

$$\begin{cases} \sum\limits_{j=1}^{i} l_{ij}\, r_{jk} = a_{ik}, & k \geq i, \quad i,k = 1,\ldots,n, \\ \sum\limits_{j=1}^{k} l_{ij}\, r_{jk} = a_{ik}, & k < i, \quad i,k = 1,\ldots,n. \end{cases}$$

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

$$\left[\begin{array}{l} \left[\begin{array}{ll} r_{1k} = a_{1k}, & k = 1, \ldots, n, \\ \sum\limits_{j=1}^{i-1} l_{ij}\, r_{jk} + r_{ik} = a_{ik}, & k \geq i > 1, \quad i, k = 2, \ldots, n. \end{array}\right. \\ \left[\begin{array}{ll} l_{i1} r_{11} = a_{i1}, & i = 2, \ldots, n, \\ \sum\limits_{j=1}^{k-1} l_{ij}\, r_{jk} + l_{ik} r_{kk} = a_{ik}, & 1 < k < i, \quad i, k = 2, \ldots, n, \end{array}\right. \end{array}\right.$$

Перегруппируем эти формулы:

$$\left[\begin{array}{l} \left[\begin{array}{ll} r_{1k} = a_{1k}, & k = 1, \ldots, n, \\ l_{i1} = a_{i1}/r_{11}, & i = 2, \ldots, n, \end{array}\right. \\ \left[\begin{array}{ll} r_{ik} = a_{ik} - \sum\limits_{j=1}^{i-1} l_{ij}\, r_{jk}, & k \geq i > 1, \quad i, k = 2, \ldots, n, \\ l_{ik} = (a_{ik} - \sum\limits_{j=1}^{k-1} l_{ij}\, r_{jk})/r_{kk}, & 1 < k < i, \quad i, k = 2, \ldots, n. \end{array}\right. \end{array}\right. \tag{2}$$

Процесс вычислений по этим формулам строится следующим образом: вначале по первой из формул (2) вычисляются неизвестные элементы первой строки матрицы R: r_{1k}, $k = 1, \ldots, n$, затем по второй из формул (2) вычисляются неизвестные элементы первого столбца матрицы L: l_{i1}, $i = 2, \ldots, n$, (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}, $k = 2, \ldots, n$ (напомним, $r_{21} = 0$, так как R -верхняя треугольная)

$$r_{2k} = a_{2k} - l_{21}\, r_{1k}, \quad k = 2, \ldots, n.$$

По четвертой формуле (2) вычисляются неизвестные элементы второго столбца матрицы L: l_{i2}, $i = 3, \ldots, n$ (напомним, $l_{12} = 0$, так как L -нижняя треугольная, $l_{22} = 1$, так как L имеет единичную главную диагональ)

$$l_{i2} = (a_{i2} - l_{i1}\, r_{12})/r_{22}, \quad i = 3, \ldots, n.$$

Затем по третьей формуле (2) вычисляются неизвестные элементы третьей строки матрицы R: r_{3k}, $k = 3, \ldots, n$ и так далее. а по четвертой формуле (2) вычисляются неизвестные элементы третьего столбца матрицы L: l_{i3}, $i = 4, \ldots, n$, 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: $l_{ij} \equiv a_{ij}$, $i > j$, $i, j = 1, \ldots, n$, а вернюю треугольную матрицу R — на месте верхнего треугольника матрицы A: $r_{ij} \equiv a_{ij}$, $i \leq j$, $i, j = 1, \ldots, n$.

Оценка количества арифметических операций в алгоритме построения LR-разложения

1. При фиксированном $i = 1, \ldots, n$ вычисление элементов r_{ik} для всех $k = i, \ldots, n$ 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 $\sum_{i=1}^{n}(n-i+1)(i-1) = n\sum_{i=1}^{n}(i-1) - \sum_{i=1}^{n}(i-1)^2 = n\sum_{j=0}^{n-1} j - \sum_{j=0}^{n-1} j^2 = n^2(n-1)/2 - (n-1)n(2n-1)/6 = n^3/2 - n^3/3 + O(n^2) = n^3/6 + O(n^2)$ $(n \to \infty)$ мультипликативных и столько же аддитивных операций.

2. При фиксированном $k = 1, \ldots, n$ вычисление элементов l_{ik} для всех $i = k+1, \ldots, n$ 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 $\sum_{k=1}^{n}(n-k)k = n^2(n+1)/2 - n(n+1)(2n+1)/6 = n^3/6 + O(n^2)$ $(n \to \infty)$ мультипликативных и $\sum_{i=1}^{n}(n-k)(k-1) = n^2(n-1)/2 - n(n+1)(2n+1)/6 + n(n+1)/2 = n^3/6 + O(n^2)$ $(n \to \infty)$ аддитивных операций.

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

$$L = \begin{pmatrix} 1 & & & & \\ l_{21} & 1 & & & \\ & l_{32} & 1 & & \\ & & \ddots & \ddots & \\ & & & l_{n,n-1} & 1 \end{pmatrix}. \tag{3}$$

Формулы (2), следовательно, примут вид

$$\left[\begin{array}{l} r_{1k} = a_{1k}, \quad k = 1, \ldots, n, \\ \left[\begin{array}{ll} r_{ik} = a_{ik} - l_{i,i-1}\, r_{i-1,k}, & k \geq i > 1, \quad i,k = 2, \ldots, n, \\ l_{i,i-1} = a_{i,i-1}/r_{i-1,i-1}, & i = 2, \ldots, n. \end{array}\right. \end{array}\right. \tag{4}$$

Алгоритм построения LR-разложения по этим формулам требует для своего проведения выполнения $n^2/2 + O(n)$ $(n \to \infty)$ 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:

$$L = \begin{pmatrix} 1 & & & & \\ l_{21} & 1 & & & \\ & l_{32} & 1 & & \\ & & \ddots & \ddots & \\ & & & l_{n,n-1} & 1 \end{pmatrix}, \quad R = \begin{pmatrix} r_{11} & r_{12} & & & \\ & r_{22} & r_{23} & & \\ & & 1 & \ddots & \\ & & & \ddots & r_{n-1,n} \\ & & & & r_{nn} \end{pmatrix}. \tag{5}$$

Формулы (4), следовательно, примут вид

$$\left[\begin{array}{l} r_{1k} = a_{1k}, \quad k = 1, 2, \\ \left[\begin{array}{ll} r_{ik} = a_{ik} - l_{i,i-1}\, r_{i-1,k}, & k = i, i+1, \quad i = 2, \ldots, n, \\ l_{i,i-1} = a_{i,i-1}/r_{i-1,i-1}, & i = 2, \ldots, n. \end{array}\right. \end{array}\right. \tag{6}$$

Алгоритм построения LR-разложения по этим формулам требует для своего проведения выполнения $3n+O(1)$ $(n \to \infty)$ мультипликативных и $n+O(1)$ $(n \to \infty)$ 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 $k = 1, 2, \ldots$ матрица A_{k+1} получается из матрицы A_k следующим образом:

   а) 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 $k = 1, 2, \ldots$ 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 $k = 1, 2, \ldots$ 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 $k = 1, 2, \ldots$, причем $A_{k+1} = L_1^{-1} \ldots L_k^{-1} A_1 L_k \ldots L_1 = (L_k \ldots L_1)^{-1} A (L_k \ldots L_1)$.

Следствие 1. *Матрицы A_k, $k = 1, 2, \ldots$ 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 $k = 1, 2, \ldots$ LR-алгоритма осуществимо LR-разложение для матрицы A_k, и собственные значения $\{\lambda_i\}$ матрицы A таковы, что*

$$|\lambda_1| > |\lambda_2| > \ldots > |\lambda_n|.$$

Тогда $L_k \to I$ при $k \to \infty$, $R_k \to A_k$ при $k \to \infty$ (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:

$$a_{ii}^{(k)} \to \lambda_i \quad \textit{при} \quad k \to \infty, \quad i = 1, 2, \ldots, n.$$

Скорость сходимости матрицы A_k к треугольной дается соотношением

$$a_{ij}^{(k)} = O\left(\left|\frac{\lambda_i}{\lambda_j}\right|^k\right) \quad \textit{при} \quad k \to \infty, \quad i > j.$$

Применение алгоритма к матрице $A \in \mathbf{M}_n$ произвольного вида требует слишком большого числа арифметических операций: $(2/3)\,n^3 + O(n^2)$ $(n \to \infty)$ на построение LR-разложения матрицы A_k и не более $n^3 + O(n^2)$ $(n \to \infty)$ 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, $k = 1, 2, \ldots$ 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), $R_k \in \mathrm{RT}(n)$. 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 $R \in \mathrm{RT}(n)$, L имеет вид (3), может быть вычислено значительно быстрее, чем произведение произвольных треугольных матриц. По определению произведения матриц

$$\left[\begin{array}{ll} a_{ik} = r_{ik} + r_{i,k+1}l_{k+1,k}, & k = i, i+1, \ldots, n-1, \quad i = 1, 2, \ldots, n \\ a_{in} = r_{in}, & i = 1, 2, \ldots, n \\ a_{i,i-1} = r_{ii}l_{i,i-1}, & i = 2, 3, \ldots, n \end{array}\right. \tag{7}$$

Вычисление произведения $A = RL$ по этим формулам требует $n^2/2 + O(n)$ $(n \to \infty)$ 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 $n^2/2 + O(n)$ $(n \to \infty)$ мультипликативных и столько же аддитивных операций.

2) Вычисление произведения $A_{k+1} = R_k L_k$ по формулам (7) требует $n^2/2 + O(n)$ $(n \to \infty)$ мультипликативных и столько же аддитивных операций.

Следовательно, один шаг алгоритма для почти треугольной матрицы требует $n^2 + O(n)$ $(n \to \infty)$ 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, $k = 1, 2, \ldots$ 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

$$\left[\begin{array}{ll} a_{ii} = r_{ii} + r_{i,i+1}l_{i+1,i}, & i = 1, 2, \ldots, n-1 \\ a_{i,i+1} = r_{i,i+1}, & i = 1, 2, \ldots, n-1 \\ a_{i,i-1} = r_{ii}l_{i,i-1}, & i = 2, 3, \ldots, n \\ a_{nn} = r_{nn} & \end{array}\right. \tag{8}$$

Вычисление произведения $A = RL$ по этим формулам требует $2n+O(1)$ $(n \to \infty)$ мультипликативных и $n + O(1)$ $(n \to \infty)$ 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 $3n + O(1)$ $(n \to \infty)$ мультипликативных и $n + O(1)$ $(n \to \infty)$ аддитивных операций.

2) Вычисление произведения $A_{k+1} = R_k L_k$ по формулам (7) требует $2n + O(1)$ $(n \to \infty)$ мультипликативных и $n + O(1)$ $(n \to \infty)$ аддитивных операций.

Следовательно, один шаг алгоритма для трехдиагональной матрицы требует $5n + O(1)$ $(n \to \infty)$ мультипликативных и $2n + O(1)$ $(n \to \infty)$ 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

$$A_k = \begin{pmatrix} a_{11}^{(k)} & a_{12}^{(k)} & \dots & a_{1,i}^{(k)} & \dots & a_{1,n-2}^{(k)} & a_{1,n-1}^{(k)} & a_{1n}^{(k)} \\ a_{21}^{(k)} & a_{22}^{(k)} & \dots & a_{2,i}^{(k)} & \dots & a_{2,n-2}^{(k)} & a_{2,n-1}^{(k)} & a_{2n}^{(k)} \\ & a_{32}^{(k)} & \ddots & \vdots & \dots & \vdots & \vdots & \vdots \\ & & \ddots & a_{i,i}^{(k)} & \dots & a_{i,n-2}^{(k)} & a_{i,n-1}^{(k)} & a_{i,n}^{(k)} \\ & & & a_{i+1,i}^{(k)} & \ddots & \vdots & \vdots & \vdots \\ & & & & \ddots & a_{n-2,n-2}^{(k)} & a_{n-2,n-1}^{(k)} & a_{n-2,n}^{(k)} \\ & & & & & a_{n-1,n-2}^{(k)} & a_{n-1,n-1}^{(k)} & a_{n-1,n}^{(k)} \\ & & & & & & a_{n,n-1}^{(k)} & a_{nn}^{(k)} \end{pmatrix}$$

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:

$$A_k = \begin{pmatrix} a_{11}^{(k)} & a_{12}^{(k)} & \dots & a_{1,i}^{(k)} & a_{1,i+1}^{(k)} & \dots & a_{1,n-2}^{(k)} & a_{1,n-1}^{(k)} & a_{1n}^{(k)} \\ a_{21}^{(k)} & a_{22}^{(k)} & \dots & a_{2,i}^{(k)} & a_{2,i+1}^{(k)} & \dots & a_{2,n-2}^{(k)} & a_{2,n-1}^{(k)} & a_{2n}^{(k)} \\ & a_{32}^{(k)} & \ddots & \vdots & \vdots & \dots & \vdots & \vdots & \vdots \\ & & \ddots & a_{i,i}^{(k)} & a_{i,i+1}^{(k)} & \dots & a_{i,n-2}^{(k)} & a_{i,n-1}^{(k)} & a_{i,n}^{(k)} \\ & & & 0 & a_{i+1,i+1}^{(k)} & \dots & a_{i+1,n-2}^{(k)} & a_{i+1,n-1}^{(k)} & a_{i+1,n}^{(k)} \\ & & & & a_{i+2,i+1}^{(k)} & \ddots & \vdots & \vdots & \vdots \\ & & & & & \ddots & a_{n-2,n-2}^{(k)} & a_{n-2,n-1}^{(k)} & a_{n-2,n}^{(k)} \\ & & & & & & a_{n-1,n-2}^{(k)} & a_{n-1,n-1}^{(k)} & a_{n-1,n}^{(k)} \\ & & & & & & & a_{n,n-1}^{(k)} & a_{nn}^{(k)} \end{pmatrix}$$

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 $|a^{(k)}_{i+1,i}| < \varepsilon\|A_k\|_\infty$ используют условие $|a^{(k)}_{i+1,i}| < \varepsilon\|A_1\|_\infty$.

§ 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 $a^{(k)}_{i+1,i}$ к нулю для этой матрицы по теореме о сходимости алгоритма есть $O\left(\left|\dfrac{\lambda_{i+1}-\lambda}{\lambda_i-\lambda}\right|^k\right)$. Если λ близко к λ_{i+1}, то скорость сходимости элемента $a^{(k)}_{i+1,i}$ 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 $k = 1, 2, \ldots$ 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 $A_k - s_kI$: $A_k - s_kI = L_kR_k$,

   в) вычисляем матрицу A_{k+1} как произведение матриц R_k и L_k плюс s_kI: $A_{k+1} = R_kL_k + s_kI$.

Здесь мы предполагаем, что для каждого $k = 1, 2, \ldots$ LR-разложение матрицы $A_k - s_kI$ 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: $A_{k+1} = R_kL_k + s_kI = (L_k^{-1}L_k)(R_kL_k + s_kI) = L_k^{-1}(L_kR_k)L_k + s_kL_kI = L_k^{-1}(L_kR_k + s_kI)L_k = L_k^{-1}A_kL_k$ и, следовательно, все матрицы A_k, $k = 1, 2, \ldots$ 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 $a_{nn}^{(k)} \to \lambda_n$, 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.

§ 8. МЕТОД ХОЛЕЦКОГО

Метод Холецкого позволяет находить все собственные значения самосопряженной положительно определенной матрицы $A \in \mathbf{M}_n$ 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 $L = R^* \in \mathrm{LT}(n)$. 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 $A \in \mathbf{M}_n$ 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. $\overline{a_{ij}} = a_{ji}$) 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.

$$\sum_{k=1}^{n} l_{ik}\overline{l_{jk}} = a_{ij}, \qquad i, j = 1, \dots, n. \tag{1}$$

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

$$\sum_{k=1}^{n} l_{ik}\overline{l_{jk}} = a_{ij}, \qquad i \le j, \;\; i,j = 1,\dots,n. \tag{2}$$

Таким образом, (2) представляет собой систему из $n(n+1)/2$ уравнений с $n(n+1)/2$ неизвестными l_{ij}, $i \ge j$ (напомним, $L \in \mathrm{LT}(n)$ 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

$$\sum_{k=1}^{i} l_{ik}\overline{l_{jk}} + \sum_{k=i+1}^{n} l_{ik}\overline{l_{jk}} = a_{ij}, \qquad i \le j, \;\; i,j = 1,\dots,n. \tag{3}$$

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

$$\sum_{k=1}^{i-1} l_{ik}\overline{l_{jk}} + l_{ii}\overline{l_{ji}} = a_{ij}, \qquad i \le j, \;\; i,j = 1,\dots,n,$$

(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:

$$\sum_{k=1}^{i-1} \overline{l_{ik}} l_{jk} + l_{ii} l_{ji} = a_{ji}, \qquad i \le j, \;\; i,j = 1,\dots,n,$$

Выделим в здесь отдельно случай $i = j$:

$$\left[\begin{array}{ll} l_{ii}^2 = a_{ii} - \sum\limits_{k=1}^{i-1} |l_{ik}|^2 & i = 1,\dots,n, \\ l_{ii} l_{ji} = a_{ji} - \sum\limits_{k=1}^{i-1} \overline{l_{ik}} l_{jk} & i < j, \;\; i,j = 1,\dots,n, \end{array}\right.$$

Отсюда получаем расчетные формулы:

$$\left[\begin{array}{ll} l_{ii} = \sqrt{a_{ii} - \sum\limits_{k=1}^{i-1} |l_{ik}|^2} & i = 1,\dots,n, \\ l_{ji} = (a_{ji} - \sum\limits_{k=1}^{i-1} \overline{l_{ik}} l_{jk})/l_{ii} & i < j, \;\; i,j = 1,\dots,n. \end{array}\right. \tag{4}$$

Процесс вычислений по этим формулам строится следующим образом: вначале вычисляются неизвестные элементы первого столбца матрицы L:

$$l_{11} = \sqrt{a_{11}}, \quad , l_{j1} = a_{j1}/l_{11}, \;\; j = 2,\dots,n;$$

потом по формулам (4) при $i = 2$ вычисляются неизвестные элементы второго столбца матрицы L:

$$l_{22} = \sqrt{a_{22} - |l_{21}|^2},$$
$$l_{j2} = (a_{j2} - \overline{l_{21}} l_{j1})/l_{22}, \quad j = 3, \ldots, n;$$

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 $a_{ji} = \overline{a_{ij}}$, то можно вместо всей матрицы A хранить только ее нижний треугольник: a_{ij}, $i \geq j$, $i, j = 1, \ldots, n$. 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: $l_{ji} \equiv a_{ji}$, $i \geq j$, $i, j = 1, \ldots, n$.

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 $i = 1, \ldots, n$ вычисление элементов l_{ji} для всех $j = i + 1, \ldots, n$ 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, $\sum_{i=1}^{n} ((n-i)(i-1) + (i-1) + 1) = n^3/6 + O(n^2)$ $(n \to \infty)$ мультипликативных и $\sum_{i=1}^{n} ((n-i)(i-1) + (i-1)) = n^3/6 + O(n^2)$ $(n \to \infty)$ 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

$$\left[\begin{array}{ll} l_{ii} = \sqrt{a_{ii} - |l_{i,i-1}|^2} & i = 1, \ldots, n, \\ l_{i+1,i} = a_{i+1,i}/l_{ii} & i = 1, \ldots, n-1. \end{array}\right. \tag{6}$$

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 $k = 1, 2, \ldots$ матрица A_{k+1} получается из матрицы A_k следующим образом:

   а) строим разложение Холецкого матрицы A_k: $A_k = L_k L_k^*$,

   б) вычисляем матрицу A_{k+1} как произведение матриц L_k^* и L_k: $A_{k+1} = L_k^* L_k$.

Лемма 1. *Для всех $k = 1, 2, \ldots$ 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 $k = 1, 2, \ldots$, причем $A_{k+1} = L_1^{-1} \ldots L_k^{-1} A_1 L_k \ldots L_1 = (L_k \ldots L_1)^{-1} A (L_k \ldots L_1)$.

Следствие 1. *Матрицы A_k, $k = 1, 2, \ldots$ имеют те же собственные значения, что и матрица A.*

Лемма 2. *Для всех $k = 1, 2, \ldots$ A_k – самосопряженная положительно определенная матрица.*

Доказательство. Действительно, $A_1 = A$ – самосопряженная положительно определенная матрица. Пусть сделано $k = 1, 2, \ldots$ шагов описанного выше процесса. Матрица $A_k = (L_{k-1} \ldots L_1)^{-1} A (L_{k-1} \ldots L_1)$ 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 $k = 1, 2, \ldots$.*

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*

$$\lambda_1 > \lambda_2 > \ldots > \lambda_n.$$

Тогда $A_k \to \operatorname{diag}[\lambda_1, \ldots, \lambda_n]$ при $k \to \infty$ (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:

$$a_{ii}^{(k)} \to \lambda_i \quad при \quad k \to \infty, \quad i = 1, 2, \ldots, n.$$

Скорость сходимости матрицы A_k к диагональной дается соотношением

$$a_{ij}^{(k)} = \overline{a_{ji}^{(k)}} = O\left(\left|\frac{\lambda_i}{\lambda_j}\right|^k\right) \quad при \quad k \to \infty, \quad i > j.$$

Применение алгоритма к произвольной самосопряженной матрице $A \in \mathbf{M}_n$ требует слишком большого числа арифметических операций: $n^3/3 + O(n^2)$ $(n \to \infty)$ на построение разложения Холецкого матрицы A_k и не более $n^3 + O(n^2)$ $(n \to \infty)$ 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, $k = 1, 2, \ldots$ 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

$$\left[\begin{array}{ll} a_{i,i-1} = \overline{l_{ii}} l_{i,i-1}, & i = 2,3,\ldots,n \\ a_{ii} = |l_{ii}|^2 + |l_{i+1,i}|^2, & i = 1,2,\ldots,n-1 \\ a_{i,i+1} = l_{ii}\overline{l_{i,i+1}}, & i = 1,2,\ldots,n-1 \\ a_{nn} = |l_{nn}|^2 & \end{array}\right. \tag{7}$$

Вычисление произведения $A = L^*L$ по этим формулам требует $4n + O(1)$ $(n \to \infty)$ мультипликативных и $n + O(1)$ $(n \to \infty)$ 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, $2n + O(1)$ $(n \to \infty)$ мультипликативных и $n + O(1)$ $(n \to \infty)$ аддитивных операций.

2) Вычисление произведения $A_{k+1} = L_k^* L_k$ по формулам (7) требует $4n + O(1)$ $(n \to \infty)$ мультипликативных и $n + O(1)$ $(n \to \infty)$ аддитивных операций.

Следовательно, один шаг алгоритма для трехдиагональной матрицы требует n операций извлечения корня, $6n + O(1)$ $(n \to \infty)$ мультипликативных и $2n + O(1)$ $(n \to \infty)$ 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 $k = 1, 2, \ldots$ матрица A_{k+1} получается из матрицы A_k следующим образом:

   а) 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 $k = 1, 2, \ldots$ 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, $k = 1, 2, \ldots$ 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 $a_{nn}^{(k)} \to \lambda_n$, 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 $(a_{ij})_{i,j=1,2,\ldots,n-1} \in \mathbf{M}_{n-1}$ 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 $A \in \mathbf{M}_n$.

§ 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 $A \in \mathbf{M}_n$ в алгоритме построения QR-разложения почти треугольная:

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1,i} & \dots & a_{1,n-2} & a_{1,n-1} & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2,i} & \dots & a_{2,n-2} & a_{2,n-1} & a_{2n} \\ & a_{32} & \ddots & \vdots & \dots & \vdots & \vdots & \vdots \\ & & \ddots & a_{i,i} & \dots & a_{i,n-2} & a_{i,n-1} & a_{i,n} \\ & & & a_{i+1,i} & \ddots & \vdots & \vdots & \vdots \\ & & & & \ddots & a_{n-2,n-2} & a_{n-2,n-1} & a_{n-2,n} \\ & & & & & a_{n-1,n-2} & a_{n-1,n-1} & a_{n-1,n} \\ & & & & & & a_{n,n-1} & a_{nn} \end{pmatrix}$$

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 $A^{(1)} = T_{12}A$,

$$A^{(1)} = \begin{pmatrix} \|a_1\| & r_{12} & \dots & r_{1,n-1} & r_{1n} \\ & a_{22}^{(1)} & \dots & a_{2,n-1}^{(1)} & a_{2n}^{(1)} \\ & a_{32}^{(1)} & \dots & a_{3,n-1}^{(1)} & a_{3n}^{(1)} \\ & & \ddots & \vdots & \vdots \\ & & & a_{n,n-1}^{(1)} & a_{nn}^{(1)} \end{pmatrix}. \tag{1}$$

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

$$A^{(k-1)} = \prod_{i=k-1}^{1} T_{i,i+1} A, \tag{2}$$

где

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

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

Обозначим

$$a_1^{(k-1)} = (a_{kk}^{(k-1)}, a_{k+1,k}^{(k-1)}, 0, \dots, 0)^t \in \mathbf{R}^{n-k+1} \tag{4}$$

– 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

$$T_{k,k+1} a_1^{(k-1)} = \|a_1^{(k-1)}\| \, e_1^{(n-k+1)}, \tag{5}$$

(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

$$A^{(k)} = T_{k,k+1} A^{(k-1)} = \prod_{i=k}^{1} T_{i,i+1} A, \tag{6}$$

где $A^{(k)} =$

$$\begin{pmatrix} \|a_1\| & r_{12} & r_{13} & \dots & r_{1,k-1} & r_{1k} & r_{1,k+1} & \dots & r_{1,n-1} & r_{1n} \\ & \|a_1^{(1)}\| & r_{23} & \dots & r_{2,k-1} & r_{2k} & r_{2,k+1} & \dots & r_{2,n-1} & r_{2n} \\ & & \|a_1^{(2)}\| & \dots & r_{3,k-1} & r_{3k} & r_{3,k+1} & \dots & r_{3,n-1} & r_{3n} \\ & & & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ & & & & \|a_1^{(k-2)}\| & r_{k-1,k} & r_{k-1,k+1} & \dots & r_{k-1,n-1} & r_{k-1,n} \\ & & & & & \|a_1^{(k-1)}\| & r_{k,k+1} & \dots & r_{k,n-1} & r_{kn} \\ & & & & & & a_{k+1,k+1}^{(k)} & \dots & a_{k+1,n-1}^{(k)} & a_{k+1,n}^{(k)} \\ & & & & & & a_{k+2,k+1}^{(k)} & \dots & a_{k+2,n-1}^{(k)} & a_{k+2,n}^{(k)} \\ & & & & & & & \ddots & \vdots & \vdots \\ & & & & & & & & a_{n,n-1}^{(k)} & a_{nn}^{(k)} \end{pmatrix}. \tag{7}$$

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

$$R = A^{(n-1)} = \prod_{i=n-1}^{1} T_{i,i+1} A, \tag{8}$$

где

$$R = \begin{pmatrix} \|a_1\| & r_{12} & r_{13} & \dots & r_{1,n-2} & r_{1,n-1} & r_{1n} \\ & \|a_1^{(1)}\| & r_{23} & \dots & r_{2,n-2} & r_{2,n-1} & r_{2n} \\ & & \|a_1^{(2)}\| & \dots & r_{3,n-2} & r_{3,n-1} & r_{3n} \\ & & & \ddots & \vdots & \vdots & \vdots \\ & & & & \|a_1^{(n-3)}\| & r_{n-2,n-1} & r_{n-2,n} \\ & & & & & \|a_1^{(n-2)}\| & r_{n-1,n} \\ & & & & & & a_{nn}^{(n-1)} \end{pmatrix} \tag{9}$$

(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

$$A = \prod_{i=1}^{n-1} T_{i,i+1}^t R \equiv QR \tag{10}$$

— 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

элементарных вращений (как описано выше). Для хранения матрицы Q выделяются два вектора длины $n-1$. В первом векторе хранятся значения $\cos\varphi_{i,i+1}$, $i=1,2,\ldots,n-1$, во втором векторе — значения $\sin\varphi_{i,i+1}$, $i=1,2,\ldots,n-1$.

Оценка количества арифметических операций

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

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 $k,\ldots,n$ 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 $(a_{ij}^{(k-1)})_{i=k,\ldots,n,\,j=k+1,\ldots,n}$ 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

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

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

Алгоритм построения QR-разложения для почти треугольной матрицы методом отражений

Обозначим $a_1=(a_{11},a_{21},0,\ldots,0)^t\in\mathbf{R}^n$ – первый столбец матрицы A. Согласно лемме I.13.9 существует вектор $x^{(1)}\in\mathbf{C}^n$, равный

$$x^{(1)}=\pm\frac{a_1-\|a_1\|e_1}{\|a_1-\|a_1\|e_1\|},\tag{11}$$

такой, что $U(x^{(1)})a_1=\|a_1\|e_1$, где $e_1=(1,0,\ldots,0)\in\mathbf{C}^n$, $U_1=U(x^{(1)})$ – 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

$$A^{(k-1)} = \prod_{i=k-1}^{1} U_i A, \tag{12}$$

где матрица $A^{(k-1)}$ имеет вид (3),

$$U_i = \begin{pmatrix} I_{i-1} & 0 \\ 0 & U(x^{(i)}) \end{pmatrix}, \tag{13}$$

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

$$A^{(k)} = U_k A^{(k-1)} = \prod_{i=k}^{1} U_i A, \tag{14}$$

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

$$s_k = |a_{k+1,k}^{(k-1)}|^2, \tag{15}$$

$$\|a_1^{(k-1)}\| = \sqrt{|a_{kk}^{(k-1)}|^2 + s_k}. \tag{16}$$

затем – вектор

$$x^{(k)} = (a_{kk}^{(k-1)} - \|a_1^{(k-1)}\|, a_{k+1,k}^{(k-1)}, 0, \dots, 0)^t \in \mathbf{C}^{n-k+1} \tag{17}$$

и его норма

$$\|x^{(k)}\| = \sqrt{|x_1^{(k)}|^2 + s_k}. \tag{18}$$

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

$$x^{(k)} := (x_1^{(k)}/\|x^{(k)}\|, x_2^{(k)}/\|x^{(k)}\|, 0, \ldots, 0) \in \mathbf{C}^{n-k+1}. \tag{19}$$

После n шагов этого процесса (т.е. перехода от матриц (3) к (7)), матрица примет вид

$$R = A^{(n-1)} = \prod_{i=n}^{1} U_i A, \tag{20}$$

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

$$A = \prod_{i=1}^{n} U_i R \equiv QR \tag{21}$$

— 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 $x^{(i)}$, $i = 1, 2, \ldots, n$, во втором векторе — вторые ненулевые компоненты векторов $x^{(i)}$, $i = 1, 2, \ldots, n$.

Оценка количества арифметических операций

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

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 $k, \ldots, n$ 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 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

$$\sum_{k=1}^{n} 5(n-k+1) = 5n(n+1)/2 = (5/2)n^2 + O(n)\ (n\to\infty)$$

мультипликативных операций, $\sum_{k=1}^{n} 3(n-k+1) = (3/2)n^2 + O(n)\ (n\to\infty)$ аддитивных операций и $\sum_{k=1}^{n} 2 = O(n)\ (n\to\infty)$ 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

$$A^{(1)} = T_{12}A = \begin{pmatrix} \|a_1\| & r_{12} & r_{13} & & \\ & a_{22}^{(1)} & a_{23}^{(1)} & & \\ & a_{32}^{(1)} & a_{33}^{(1)} & \ddots & \\ & & \ddots & \ddots & a_{n-1,n}^{(1)} \\ & & & a_{n,n-1}^{(1)} & a_{nn}^{(1)} \end{pmatrix}. \qquad (22)$$

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 выполено 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 $a_1^{(k-1)}$, $k = 1, \ldots, n-1$ 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 $\cos\varphi_{i,i+1}$, $i = 1, 2, \ldots, n-1$, во втором векторе — значения $\sin\varphi_{i,i+1}$, $i = 1, 2, \ldots, n-1$.

Оценка количества арифметических операций

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

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 $k, \ldots, n$ 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 $(a_{ij}^{(k-1)})_{i=k,\ldots,n,\, j=k+1,\ldots,n}$ 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 $a_1 = (a_{11}, a_{21}, 0, \ldots, 0)^t \in \mathbf{R}^n$ – 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 $e_1 = (1, 0, \ldots, 0) \in \mathbf{C}^n$, $U_1 = U(x^{(1)})$ – 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 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:

   а) строим QR-разложение матрицы A_k: $A_k = Q_k R_k$,

   б) 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) $Q_k \to I$ *при* $k \to \infty$, $R_k \to A_k$ *при* $k \to \infty$. *Тем самым диагональные элементы матрицы* $A_k = (a_{ij}^{(k)})$ *сходятся к собственным значениям матрицы* A*:*

$$a_{ii}^{(k)} \to \lambda_{i_0} \quad \textit{при} \quad k \to \infty, \quad i = 1, 2, \ldots, n$$

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*

$$a_{ij}^{(k)} = O\left(\left|\frac{\lambda_i}{\lambda_j}\right|^k\right) \quad \textit{при} \quad k \to \infty, \quad i > j.$$

Замечание 1. (Без доказательства). Если для матрицы A осуществим LR-алгоритм, то при применении QR-алгоритма собственные значения A получаются в правильном порядке

$$a_{ii}^{(k)} \to \lambda_i \quad \text{при} \quad k \to \infty, \quad i = 1, 2, \ldots, n,$$

и скорость сходимости матрицы A_k к треугольной дается соотношением

$$a_{ij}^{(k)} = O\left(\left|\frac{\lambda_i}{\lambda_j}\right|^k\right) \quad \text{при} \quad k \to \infty, \quad i > j.$$

Замечание 2. QR-алгоритм сходится при существенно менее ограничительных условия, 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, $k = 1, 2, \ldots$ *в* QR*-алгоритме — почти треугольные.*

Доказательство. Согласно (10) или (21) $Q = \prod\limits_{i=1}^{n-1} T_{i,i+1}^t$ или $Q = \prod\limits_{i=1}^{n} U_i$ (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 $2n^2+O(n)$ $(n \to \infty)$ мультипликативных операций, $n^2 + O(n)$ $(n \to \infty)$ аддитивных операций и $O(n)$ $(n \to \infty)$ 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 $T^t_{i,i+1}$ 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 $\sum_{i=1}^{n-1} 4(i+1) = 4n(n+1)/2 = 2n^2 + O(n)$ $(n \to \infty)$ мультипликативных операций и $n^2 + O(n)$ аддитивных операций.

Следовательно, один шаг алгоритма для почти треугольной матрицы требует $4n^2 + O(n)$ $(n \to \infty)$ мультипликативных и $2n^2 + O(n)$ $(n \to \infty)$ 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 $(5/2)n^2 + O(n)$ $(n \to \infty)$ мультипликативных операций, $(3/2)n^2 + O(n)$ $(n \to \infty)$ аддитивных операций и $O(n)$ $(n \to \infty)$ 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 $\sum_{i=1}^{n} 5(i+1) = 5(n+1)(n+2)/2 = (5/2)n^2 + O(n)$ $(n \to \infty)$ мультипликативных операций и $(3/2)n^2 + O(n)$ аддитивных операций.

Следовательно, один шаг алгоритма для почти треугольной матрицы требует $5n^2 + O(n)$ $(n \to \infty)$ мультипликативных и $3n^2 + O(n)$ $(n \to \infty)$ 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, $k = 1, 2, \ldots$ 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, $k = 1, 2, \ldots$ в QR-алгоритме — самосопряженные трехдиагональные.*

Доказательство. В силу леммы 3 все матрицы A_k, $k = 1, 2, \ldots$ — самосопряженные. Поскольку трехдиагональная матрица A является почти треугольной, то в силу леммы 2 все матрицы A_k, $k = 1, 2, \ldots$ — почти треугольные. Итак, для всякого $k = 1, 2, \ldots$ 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 $r_{13}, r_{24}, \ldots, r_{n-2,n}$ 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) $Q = \prod\limits_{k=1}^{n-1} T_{k,k+1}^t$ или $Q = \prod\limits_{k=1}^{n} U_k$, 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 $r_{i,i+2}$, $i = 1, 2, \ldots, n-2$. 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 $T^t_{k,k+1}$ или U_k.

Действительно, после перехода от матрицы $A^{(k-1)}$ (23) к матрице $A^{(k)}$ (24) в алгоритме участвует только подматрица $(a^{(k-1)}_{ij})_{i,j=k+1,\dots,n}$ и столбцы $1, 2, \dots, k$ в дальнейших вычислениях не изменяются. Поэтому можно сразу умножить $A^{(k)}$ на матрицу $T^t_{k-1,k}$ 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 $T^t_{n-1,n}$ 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 $14n+O(1)$ ($n \to \infty$) мультипликативных операций, $6n + O(1)$ ($n \to \infty$) аддитивных операций и $2n + O(1)$ ($n \to \infty$) 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 $T^t_{i,i+1}$ 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 $22n+O(1)$ ($n \to \infty$) мультипликативных, $10n+O(1)$ ($n \to \infty$) аддитивных операций и $2n+O(1)$ ($n \to \infty$) 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 $25n+O(1)$ $(n \to \infty)$ мультипликативных, $15n+O(1)$ $(n \to \infty)$ аддитивных операций и $2n+O(1)$ $(n \to \infty)$ 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 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 $k = 1, 2, \ldots$ матрица A_{k+1} получается из матрицы A_k следующим образом:

   а) 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: $A_{k+1} = R_kQ_k + s_kI = (Q_k^{-1}Q_k)(R_kQ_k + s_kI) = Q_k^{-1}(Q_kR_k)Q_k + s_kQ_kI = Q_k^{-1}(Q_kR_k + s_kI)Q_k = Q_k^{-1}A_kQ_k$ и, следовательно, все матрицы A_k, $k = 1, 2, \ldots$ 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 $A \in \mathbf{M}_n$ 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 QR-алгоритм со сдвигами. На шаге k в качестве сдвига s_k возьмем $a_{nn}^{(k)}$, т.е. $s_k = a_{nn}^{(k)}$. Поскольку $a_{nn}^{(k)} \to \lambda_n$, 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 $(a_{ij})_{i,j=1,2,\ldots,n-1} \in \mathbf{M}_{n-1}$ 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.

§ 10. МЕТОД ОБРАТНОЙ ИТЕРАЦИИ НАХОЖДЕНИЯ СОБСТВЕННЫХ ВЕКТОРОВ

Метод обратной итерации позволяет найти собственный вектор, соответствующий однократному собственному значению λ диагонализируемой матрицы $A \in \mathbf{M}_n$, если известно достаточно хорошее приближение $\widehat{\lambda}$ к собственному значению λ.

Теорема 1 (Метод обратной итерации). *Пусть матрица $A \in \mathbf{M}_n$ имеет полную систему ортонормированных собственных векторов e_i, $i = 1, \ldots, n$: $Ae_i = \lambda_i e_i$, $(e_i, e_j) = \delta_{ij}$, причем собственное значение λ_m, $m = 1, 2, \ldots, n$ однократное. Пусть $\widehat{\lambda}_m \neq \lambda_m$ – достаточно хорошее приближение к λ_m:*

$$\max_{\substack{i=1,2,\ldots,n \\ i\neq m}} \left| \frac{\lambda_m - \widehat{\lambda}_m}{\lambda_i - \widehat{\lambda}_m} \right| = q < 1. \tag{1}$$

Тогда для всякого вектора $x^{(0)} \in \mathbf{C}^n$ такого, что $(x^{(0)}, e_m) \neq 0$, итерационный процесс

$$\begin{gathered} x^{(k+1)} - \textit{решение уравнения } (A - \widehat{\lambda}_m I)x^{(k+1)} = x^{(k)} \\ e_m^{(k+1)} = \frac{x^{(k+1)}}{\|x^{(k+1)}\|}, \qquad k = 0, 1, \ldots \end{gathered} \tag{2}$$

сходится к собственному вектору e_m (с точностью до постоянного множителя):

$$e_m^{(k)} \to e^{i\varphi} e_m \quad \textit{при } k \to \infty$$

(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

$$\|e_m^{(k)} - e^{i\varphi} e_m\| \leq C q^k, \tag{3}$$

где постоянная C не зависит от k.

Доказательство. Поскольку вектора $e_1, \ldots, e_n$ образуют базис в C^n, то $x^{(0)} = \sum\limits_{i=1}^{n} c_i e_i$, причем по условию $c_m = (x^{(0)}, e_m) \neq 0$. Поскольку

$$x^{(k+1)} = (A - \widehat{\lambda}_m I)^{-1} x^{(k)},$$

то

$$x^{(k+1)} = (A - \widehat{\lambda}_m I)^{-k} x^{(0)}.$$

Так как собственные значения матрицы $A - \widehat{\lambda}_m I$ равны $\lambda_i - \widehat{\lambda}_m$, $i = 1, 2, \ldots, n$, то собственные значения матрицы $(A - \widehat{\lambda}_m I)^{-1}$ есть $(\lambda_i - \widehat{\lambda}_m)^{-1}$, $i = 1, 2, \ldots, n$, а матрицы $(A - \widehat{\lambda}_m I)^{-1}$ — $(\lambda_i - \widehat{\lambda}_m)^{-k}$, $i = 1, 2, \ldots, n$. Следовательно,

$$x^{(k)} = (A - \widehat{\lambda}_m I)^{-k} \sum_{i=1}^{n} c_i e_i = \sum_{i=1}^{n} \frac{c_i e_i}{(\lambda_i - \widehat{\lambda}_m)^k}$$

$$= \frac{c_m}{(\lambda_m - \widehat{\lambda}_m)^k} \left(e_m + \sum_{\substack{i=1 \\ i \neq m}}^{n} \frac{c_i}{c_m} e_i \left(\frac{\lambda_m - \widehat{\lambda}_m}{\lambda_i - \widehat{\lambda}_m} \right)^k \right)$$

В силу ортонормированности базиса $\{e_i\}$

$$\|x^{(k)}\|^2 = (x^{(k)}, x^{(k)}) = \frac{1}{(\lambda_m - \widehat{\lambda}_m)^{2k}} \left(|c_m|^2 + \sum_{\substack{i=1 \\ i \neq m}}^{n} |c_i|^2 \left(\frac{\lambda_m - \widehat{\lambda}_m}{\lambda_i - \widehat{\lambda}_m} \right)^{2k} \right).$$

В силу (1) и равенства $\|x^{(0)}\|^2 = \sum\limits_{i=1}^{n} |c_i|^2$ получаем

$$\frac{|c_m|}{(\lambda_m - \widehat{\lambda}_m)^k} \leq \|x^{(k)}\| \leq \frac{|c_m|}{(\lambda_m - \widehat{\lambda}_m)^k} \left(1 + \frac{\|x^{(0)}\|^2}{|c_m|^2} q^{2k} \right)^{1/2}$$

или

$$\left(1 + \frac{\|x^{(0)}\|^2}{|c_m|^2} q^{2k} \right)^{-1/2} \leq \frac{|c_m|}{(\lambda_m - \widehat{\lambda}_m)^k} \frac{1}{\|x^{(k)}\|} \leq 1$$

Вычислим

$$\frac{x^{(k)}}{\|x^{(k)}\|} - \frac{c_m}{|c_m|} e_m = \left(\frac{c_m}{(\lambda_m - \widehat{\lambda}_m)^k} \frac{1}{\|x^{(k)}\|} - \frac{c_m}{|c_m|} \right) e_m$$

$$+ \sum_{\substack{i=1 \\ i \neq m}}^{n} \frac{1}{\|x^{(k)}\|} \frac{c_m}{(\lambda_m - \widehat{\lambda}_m)^k} \frac{c_i}{c_m} e_i \left(\frac{\lambda_m - \widehat{\lambda}_m}{\lambda_i - \widehat{\lambda}_m} \right)^k.$$

В силу (1), (2) и равенств $\|e_i\| = 1$, $i = 1, 2, \ldots, n$, обозначив $e^{i\varphi} = c_m/|c_m|$, находим

$$\left\| e_m^{(k)} - e^{i\varphi} e_m \right\| \le \left(1 - \left(1 + \frac{\|x^{(0)}\|^2}{|c_m|^2} q^{2k} \right)^{-1/2} \right) + \sum_{\substack{i=1 \\ i \ne m}}^{n} \frac{|c_i|}{|c_m|}\, q^k \,.$$

Так как

$$\sum_{i=1}^{n} |c_i| \le \sqrt{n} \left(\sum_{i=1}^{n} |c_i|^2 \right)^{1/2} = \sqrt{n}\, \|x^{(0)}\|,$$

то

$$\left\| e_m^{(k)} - e^{i\varphi} e_m \right\| \le \left(1 - \left(1 + \frac{\|x^{(0)}\|^2}{|c_m|^2} q^{2k} \right)^{-1/2} \right) + \sqrt{n}\, \frac{\|x^{(0)}\|}{|c_m|}\, q^k$$

В силу (1) найдется $k_0 > 0$, такое, что для всех $k \ge k_0$ выполнено $\alpha \equiv \dfrac{\|x^{(0)}\|^2}{|c_m|^2} q^{2k} < 1$. Тогда $(1-\alpha)^{-1/2} \ge 1 - \alpha/2$ и $1 - (1-\alpha)^{-1/2} \le \alpha/2 \le \alpha^{1/2}/2$. Поэтому для всех $k \ge k_0$ справедливо неравенство

$$\left\| e_m^{(k)} - e^{i\varphi} e_m \right\| \le \frac{\|x^{(0)}\|}{|c_m|} (1/2 + \sqrt{n})\, q^k.$$

Это неравенство является требуемой оценкой (3).

ПРОГРАММА КУРСА

1. Матричные нормы. Подчиненые 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 невязку. 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.