

Ответы на контрольные вопросы

1. **Почему нельзя находить собственные числа матрицы A , прямо решая уравнение $\det(A - \lambda E) = 0$, а собственные векторы — «по определению», решая систему $(A - \lambda_j E)e_j = 0$?**

Так как A — матрица размера $n \times n$, то $\det(A - \lambda E)$ — многочлен степени n относительно λ , и при вычислении определителя в коэффициентах многочлена может накопиться большая ошибка, в результате чего его корни также могут быть найдены с большими ошибками.

Рассмотрим пример:

$$A = \begin{pmatrix} 6 & 1 \\ 0 & 7 \end{pmatrix}$$

В данном случае собственными числами матрицы A будут $\lambda_1 = 6$, $\lambda_2 = 7$.

Внесем возмущение 10^{-3} в элемент матрицы, равный нулю. Получим матрицу:

$$\tilde{A} = \begin{pmatrix} 6 & 1 \\ 0.001 & 7 \end{pmatrix}$$

Собственные значения полученной матрицы отличаются от исходной на 0.001 и равны $\tilde{\lambda}_1 = 5.999$, $\tilde{\lambda}_2 = 7.001$. Если размеры матрицы значительно больше (например, 10×10), таких ошибок может быть значительно больше, и собственные значения могут отличаться от истинных во много раз.

Нахождение собственных векторов невозможно по определению, так как обычно известны приближенные решения λ_i^* уравнения $\det(A - \lambda E) = 0$. Из-за этого матрица $(A - \lambda_i^* E)$ невырождена, поэтому решением системы может быть только нулевой вектор.

2. **Докажите, что ортогональное преобразование подобия сохраняет симметрию матрицы.**

Пусть A — симметричная матрица, P — ортогональная матрица, то есть

$$A^T = A \quad P^T P = E.$$

Если R — матрица, полученная с помощью ортогонального преобразования подобия, то её можно записать в виде

$$R = P^T A P.$$

Запишем R^T :

$$R^T = (P^T A P)^T = (AP)^T (P^T)^T = P^T A^T P = P^T AP = R.$$

3. Как преобразование подобия меняет собственные векторы матрицы?

Преобразование подобия является линейным преобразованием, то есть оно масштабирует и поворачивает собственные векторы.

Пусть A и B — подобные матрицы, то есть существует такая матрица P : $B = P^{-1}AP$.

Пусть x — собственный вектор матрицы A , то есть $Ax = \lambda x$; $y = P^{-1}x$. Тогда

$$By = (P^{-1}AP)y = P^{-1}APP^{-1}x = P^{-1}Ax = P^{-1}\lambda x = \lambda P^{-1}x = \lambda y.$$

Получили, что $y = P^{-1}x$ — собственный вектор матрицы B при том же собственном значении λ .

4. Почему на практике матрицу A подобными преобразованиями вращения приводят только к форме Хессенберга, но не к треугольному виду?

Приведение матрицы A к верхнетреугольному виду позволило бы сразу найти собственные значения матрицы A , но в общем случае это сделать ортогональными преобразованиями невозможно.

Рассмотрим матрицы размера 3×3 :

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

Попытаемся привести ее ортогональными преобразованиями к верхнетреугольному виду. Для этого рассмотрим преобразование подобия с матрицей T_{13} :

$$T_{13} = \begin{pmatrix} \alpha & 0 & -\beta \\ 0 & 1 & 0 \\ \beta & 0 & \alpha \end{pmatrix},$$

$$\text{где } \alpha = \frac{a_{23}}{\sqrt{a_{21}^2 + a_{23}^2}}, \beta = \frac{a_{21}}{\sqrt{a_{21}^2 + a_{23}^2}}.$$

Получим матрицу $A^{(1)}$, обнулив элемент a_{21} :

$$\begin{aligned}
A^{(1)} &= T_{13} A T_{13}^T = \begin{pmatrix} \alpha & 0 & -\beta \\ 0 & 1 & 0 \\ \beta & 0 & \alpha \end{pmatrix} \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \begin{pmatrix} \alpha & 0 & \beta \\ 0 & 1 & 0 \\ -\beta & 0 & \alpha \end{pmatrix} = \\
&= \begin{pmatrix} \alpha a_{11} - \beta a_{31} & \alpha a_{12} - \beta a_{32} & \alpha a_{13} - \beta a_{33} \\ a_{21} & a_{22} & a_{23} \\ \beta a_{11} + \alpha a_{31} & \beta a_{12} + \alpha a_{32} & \beta a_{13} + \alpha a_{33} \end{pmatrix} \begin{pmatrix} \alpha & 0 & \beta \\ 0 & 1 & 0 \\ -\beta & 0 & \alpha \end{pmatrix} = \\
&= \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} \\ a_{21}^{(1)} & a_{22}^{(1)} & a_{23}^{(1)} \\ a_{31}^{(1)} & a_{32}^{(1)} & a_{33}^{(1)} \end{pmatrix},
\end{aligned}$$

где $a_{21}^{(1)} = a_{21} \alpha - a_{23} \beta = 0$.

Покажем, что в общем случае ни одно из преобразований $T_{13}^{(1)}$, $T_{23}^{(1)}$, $T_{12}^{(1)}$ не сохраняет элемент $a_{21}^{(1)} = 0$ при попытке обнулить $a_{31}^{(1)}$.

$$T_{13}^{(1)} = \begin{pmatrix} \alpha & 0 & -\beta \\ 0 & 1 & 0 \\ \beta & 0 & \alpha \end{pmatrix}.$$

Тогда получим:

$$\begin{aligned}
A^{(2)} &= T_{13}^{(1)} A^{(1)} (T_{13}^{(1)})^T = \begin{pmatrix} \alpha & 0 & -\beta \\ 0 & 1 & 0 \\ \beta & 0 & \alpha \end{pmatrix} \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} \\ a_{31}^{(1)} & a_{32}^{(1)} & a_{33}^{(1)} \end{pmatrix} \begin{pmatrix} \alpha & 0 & \beta \\ 0 & 1 & 0 \\ -\beta & 0 & \alpha \end{pmatrix} = \\
&= \begin{pmatrix} \alpha a_{11}^{(1)} - \beta a_{31}^{(1)} & \alpha a_{12}^{(1)} - \beta a_{32}^{(1)} & \alpha a_{13}^{(1)} - \beta a_{33}^{(1)} \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} \\ \beta a_{11}^{(1)} + \alpha a_{31}^{(1)} & \beta a_{12}^{(1)} + \alpha a_{32}^{(1)} & \beta a_{13}^{(1)} + \alpha a_{33}^{(1)} \end{pmatrix} \begin{pmatrix} \alpha & 0 & \beta \\ 0 & 1 & 0 \\ -\beta & 0 & \alpha \end{pmatrix} = \\
&= \begin{pmatrix} a_{11}^{(2)} & a_{12}^{(2)} & a_{13}^{(2)} \\ a_{21}^{(2)} & a_{22}^{(2)} & a_{23}^{(2)} \\ a_{31}^{(2)} & a_{32}^{(2)} & a_{33}^{(2)} \end{pmatrix}.
\end{aligned}$$

Рассмотрим элементы $a_{21}^{(2)}$ и $a_{31}^{(2)}$:

$$\begin{aligned}
a_{21}^{(2)} &= -\beta a_{23}^{(1)}, \\
a_{31}^{(2)} &= \alpha(\beta a_{11}^{(1)} + \alpha a_{31}^{(1)}) - \beta(\beta a_{13}^{(1)} + \alpha a_{33}^{(1)}).
\end{aligned}$$

Заметим, что $a_{21}^{(2)}$ остается равным 0 только в том случае, если $\beta = 0$. Но тогда $a_{31}^{(2)} = \alpha^2 a_{31}^{(1)}$. Так как $\alpha = \pm 1$ из условия $\alpha^2 + \beta^2 = 1$, то $a_{31}^{(2)} = a_{31}^{(1)} \neq 0$. Таким образом, не удается занулить одновременно $a_{31}^{(1)}$ и сохранить равным 0 $a_{21}^{(1)}$.

Рассмотрим вращение $T_{12}^{(1)}$:

$$T_{12}^{(1)} = \begin{pmatrix} \alpha & -\beta & 0 \\ \beta & \alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Тогда матрица $A^{(2)}$ будет иметь вид:

$$A^{(2)} = T_{12}^{(1)} A^{(1)} (T_{12}^{(1)})^T = \begin{pmatrix} a_{11}^{(2)} & a_{12}^{(2)} & a_{13}^{(2)} \\ a_{21}^{(2)} & a_{22}^{(2)} & a_{23}^{(2)} \\ a_{31}^{(2)} & a_{32}^{(2)} & a_{33}^{(2)} \end{pmatrix},$$

где после умножения матриц получим:

$$\begin{aligned} a_{21}^{(2)} &= \alpha\beta a_{11}^{(1)} - \beta(\alpha a_{22}^{(1)} + \beta a_{12}^{(1)}), \\ a_{31}^{(2)} &= \alpha a_{31}^{(1)} - \beta a_{32}^{(1)}. \end{aligned}$$

При таком преобразовании с учетом, что $a_{31}^{(2)} = 0$, выберем $\alpha = \frac{a_{32}^{(1)}}{\sqrt{(a_{31}^{(1)})^2 + (a_{32}^{(1)})^2}}$,

$\beta = \frac{a_{31}^{(1)}}{\sqrt{(a_{31}^{(1)})^2 + (a_{32}^{(1)})^2}}$. Но тогда в общем случае $a_{21}^{(2)} \neq 0$.

Рассмотрим вращение $T_{23}^{(1)}$:

$$T_{23}^{(1)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \alpha & -\beta \\ 0 & \beta & \alpha \end{pmatrix}.$$

Тогда матрица $A^{(2)}$ будет иметь вид:

$$A^{(2)} = T_{23}^{(1)} A^{(1)} (T_{23}^{(1)})^T = \begin{pmatrix} a_{11}^{(2)} & a_{12}^{(2)} & a_{13}^{(2)} \\ a_{21}^{(2)} & a_{22}^{(2)} & a_{23}^{(2)} \\ a_{31}^{(2)} & a_{32}^{(2)} & a_{33}^{(2)} \end{pmatrix},$$

где $a_{21}^{(2)} = -\beta a_{31}^{(1)}$, $a_{31}^{(2)} = \alpha a_{31}^{(1)}$. В таком случае $\alpha = \beta = 0$, что противоречит условию $\alpha^2 + \beta^2 = 1$.

Также стоит отметить, что выполнение одной итерации QR-алгоритма для матрицы Хессенберга требует $O(n^2)$ арифметических операций, в то время как обычный QR-алгоритм требует $O(n^3)$ операций.

5. **Оцените количество арифметических операций, необходимое для приведения произвольной квадратной матрицы A к форме Хессенберга.**

Рассмотрим произвольную квадратную матрицу A размера $n \times n$. Оценим количество операций на k -м шаге алгоритма:

- 5.1. На вычисление $n - k - 1$ матриц вида $T_{k,k+1} \dots T_{kn}$ потребуется $4(n - k - 1)$ мультипликативных, $2(n - k - 1)$ аддитивных операций и $n - k$ операций извлечения корня.
- 5.2. Для вычисления компонент $k+1, \dots, n$ k -го столбца матрицы $\hat{A}^{(k)} = \prod_{j=n}^{k+2} T_{k+1,j} A^{(k-1)}$ требуется $n - k$ операций умножения, $n - k - 1$ операций сложения и 1 операция извлечения корня.
- 5.3. Каждая из $n - k - 1$ матриц элементарных вращений умножается на подматрицу $(a_{ij}^{(k-1)})_{i=k+1, \dots, n, j=k+1, \dots, n}$ матрицы $A^{(k-1)}$ размера $(n - k) \times (n - k)$, то на это потребуется $(n - k - 1)4(n - k) = 4(n - k)^2 - 4(n - k)$ умножений и $(n - k - 1)2(n - k) = 2(n - k)^2 - 2(n - k)$ сложений.
- 5.4. Матрица, транспонированная к матрице элементарных вращений, является матрицей элементарных вращений, и каждая из этих $n - k - 1$ матриц умножается на подматрицу $(\hat{a}_{ij}^{(k-1)})_{i=k+1, \dots, n, j=k+1, \dots, n}$ матрицы $\hat{A}^{(k-1)}$ размера $n \times (n - k)$, то на это потребуется $(n - k - 1)4n = 4n(n - k) - 4n$ умножений и $(n - k - 1)2n = 2n(n - k) - 2n$ сложений.

Итого на k -м шаге потребуется выполнить $4n(n - k) + 4(n - k)^2 + (n - k) - 4n - 4$ мультипликативных операций, $2n(n - k) + 2(n - k)^2 + (n - k) - 2n - 3$ аддитивных операций и $n - k$ операций извлечения корня.

Тогда всего требуется выполнить:

- $\sum_{k=1}^{n-2} (4n(n - k) + 4(n - k)^2 + (n - k) - 4n - 4) = 4n\left(\frac{(n-1)(n-2)}{2} - 1\right) + 4\left(\frac{(n-1)(n-2)(2n-3)}{6} - 1\right) + \frac{(n-1)(n-2)}{2} - 1 - 4(n-2)(n+1) = 2n^3 + O(n^2) + \frac{4}{3}n^3 + O(n^2) + O(n^2) = \frac{10}{3}n^3 + O(n^2) \quad (n \rightarrow \infty)$ мультипликативных операций;
- $\sum_{k=1}^{n-2} (2n(n - k) + 2(n - k)^2) + (n - k) - 2n - 3 = \frac{5}{3} + O(n^2) \quad (n \rightarrow \infty)$ аддитивных операций;
- $\sum_{k=1}^{n-2} (n - k) = O(n^2) \quad (n \rightarrow \infty)$ операций извлечения квадратного корня.

Итого необходимо $5n^3 + O(n^2) \quad (n \rightarrow \infty)$ операций для приведения матрицы к виду матрицы Хессенберга.

6. Сойдется ли алгоритм обратных итераций, если в качестве начального приближения взять собственный вектор, соответствующий другому собственному значению? Что будет в этой ситуации в методе обратной итерации, использующем отношение Рэлея?

Разложим произвольный начальный вектор и вектор решения по базису из собственных векторов:

$$x = \sum_{k=1}^n \alpha_k e_k$$

$$y = \sum_{k=1}^n \beta_k e_k$$

Тогда рассмотрим выражение:

$$(A - \lambda_i^* E)y = Ay - \lambda_i^* y = A \sum_{k=1}^n \beta_k e_k - \lambda_i^* \sum_{k=1}^n \beta_k e_k = \sum_{k=1}^n (\lambda_k - \lambda_i^*) \beta_k e_k$$

В силу единственности разложения по базису получим:

$$(\lambda_k - \lambda_i^*) \beta_k = \alpha_k$$

Тогда коэффициенты при собственных векторах $\beta_k = \frac{\alpha_k}{\lambda_k - \lambda_i^*}$.

В качестве начального приближения $x^{(0)}$ возьмем собственный вектор $e_j, j \neq i$. Если он найден точно, то

$$y^{(1)} = \frac{1}{\lambda_j - \lambda_i^*} e_j.$$

Выполнив нормировку, получим $x^{(1)} = e_j$. Таким образом, в случае точного решения алгоритм обратных итераций никогда не сойдется к собственному вектору, соответствующему собственному значению λ_i .

Пусть вектор $x^{(0)} = e_j^*$ известен лишь приближенно, и его разложение по базису собственных векторов имеет вид:

$$e_j^* = \varepsilon_1 e_1 + \varepsilon_2 e_2 + \dots + \delta e_j + \dots + \varepsilon_n e_n,$$

при этом $|\varepsilon_k| \ll 1, k = \overline{1, n}, k \neq j, \delta \approx 1$.

Тогда вектор решения $y^{(1)}$ будет иметь следующее разложение:

$$y^{(1)} = \frac{\varepsilon_1}{\lambda_1 - \lambda_i^*} e_1 + \frac{\varepsilon_2}{\lambda_2 - \lambda_i^*} e_2 + \dots + \frac{\varepsilon_i}{\lambda_i - \lambda_i^*} e_i + \dots + \frac{\delta}{\lambda_j - \lambda_i^*} e_j + \dots + \frac{\varepsilon_n}{\lambda_n - \lambda_i^*} e_n.$$

Так как $|\lambda_i - \lambda_i^*| \ll 1$, то коэффициент при e_i будет много больше коэффициентов при других собственных векторах (за исключением, может быть, коэффициента $\frac{\delta}{\lambda_j - \lambda_i^*}$ при собственном векторе e_j), а нормировка приведет только к домножению всех компонент на какое-то число. Дальнейшие шаги алгоритма приведут к возрастанию коэффициента при e_i , и, начиная с некоторого шага, он будет самым большим среди коэффициентов разложения. В результате последовательность векторов $\{x^{(j)}\}_{j=0}^\infty$ сойдется к собственному вектору e_i .

Если мы будем использовать отношение Рэлея для $x^{(0)} = e_j^*$, получим, что $\lambda^{(0)} = (Ae_j^*, e_j^*) = \lambda_j^*$, где λ_j^* — приближенное собственное значение, соответствующее собственному вектору e_j^* . Тогда решение $y^{(1)}$ будет иметь разложение:

$$y^{(1)} = \frac{\varepsilon_1}{\lambda_1 - \lambda_j^*} e_1 + \frac{\varepsilon_2}{\lambda_2 - \lambda_j^*} e_2 + \dots + \frac{\varepsilon_i}{\lambda_i - \lambda_j^*} e_i + \dots + \frac{\delta}{\lambda_j - \lambda_j^*} e_j + \dots + \frac{\varepsilon_n}{\lambda_n - \lambda_j^*} e_n.$$

Таким образом, коэффициент при e_j всегда будет наибольшим, поэтому последовательность векторов $\{x^{(j)}\}_{j=0}^\infty$ сойдется к собственному вектору e_j .

7. Сформулируйте и обоснуйте критерий останова для QR-алгоритма отыскания собственных значений матрицы.

Поиск собственных чисел с помощью QR-алгоритма может быть реализован следующим образом: когда поддиагональные элементы последней строки станут равными или близкими к 0, мы принимаем элемент $a_{nn}^{(k)}$ за собственное число λ_n , затем переходим к матрице размера $(n-1) \times (n-1)$ и повторяем всё то же самое, и так далее, пока мы не найдём все собственные значения.

Критерием таких переходов может стать условие $\sum_{i=1}^{m-1} |a_{mi}^{(k)}| < \varepsilon$, $m = 2, \dots, n$.

Если же матрица приведена к форме Хессенберга, то критерий останова может быть записан как $|a_{i,i-1}^{(k)}|, \varepsilon$.

8. Предложите возможные варианты условий перехода к алгоритму со сдвигами. Предложите алгоритм выбора величины сдвига.

К алгоритму со сдвигами можно прибегать тогда, когда после длительной работы QR-алгоритма, некоторые модули элементов главной диагонали $|a_{ii}^{(k)}|$ и $|a_{jj}^{(k)}|$ оказываются достаточно близкими по значению. Поскольку $a_{ii}^{(k)} \xrightarrow[k \rightarrow \infty]{} \lambda_i$ и $a_{jj}^{(k)} \xrightarrow[k \rightarrow \infty]{} \lambda_j$, то близость модулей этих значений будет означать то, что величины $|\lambda_i|$ и $|\lambda_j|$ достаточно близки, а значит $\left| \frac{\lambda_i}{\lambda_j} \right| \approx 1$, из чего, в свою очередь, следует достаточно медленная скорость сходимости. Соответственно, чтобы эту проблему решить, следует произвести сдвиг. Величину сдвига надо выбирать таким образом, чтобы дробь $\left| \frac{\lambda_i - \sigma}{\lambda_j - \sigma} \right| \ll 1$, то есть $\lambda_i \approx \sigma$, тогда скорость сходимости в разы возрастет.

Алгоритм со сдвигами можно совместить с понижением размерности задачи. То есть, можно выбрать параметр σ такой, что $|\tilde{\lambda}_n| = |\lambda_n - \sigma| \ll 1$. Тогда алгоритм будет сходиться быстрее, и после некоторого количества итераций можно произвести обратный сдвиг и перейти к задаче меньшей размерности. В таком случае, работая с подматрицей $m \times m$ ($m < n$), в качестве параметра сдвига выбирается $\sigma = a_{mm}^{(k)}$, поскольку этот элемент сходится к собственному числу λ_m .

9. Для чего нужно на каждой итерации нормировать приближение к собственному вектору?

Нормировка позволяет сделать решение задачи более устойчивым. Рассмотрим разложение вектора $y^{(1)}$ по собственным векторам через коэффициенты разложения вектора $x^{(0)}$:

$$y^{(1)} = \sum_{k=1}^n \frac{\alpha_k^{(0)}}{\lambda_k - \lambda_i^*} e_k = \sum_{k=1}^n \alpha_k^{(1)} e_k$$

Теперь запишем разложение вектора $y^{(2)}$ по собственным векторам через коэффициенты разложения вектора $x^{(1)}$:

$$y^{(2)} = \sum_{k=1}^n \frac{\alpha_k^{(1)}}{\lambda_k - \lambda_i^*} e_k = \sum_{k=1}^n \frac{\alpha_k^{(0)}}{(\lambda_k - \lambda_i^*)^2} e_k$$

Рассмотрим коэффициент при e_i : т.к. $|\lambda_i - \lambda_i^*| \ll 1$, то на каждом шаге $\alpha_i^{(k)} = \frac{\alpha_i^{(0)}}{(\lambda_k - \lambda_i^*)^k}$ будет неограниченно расти. Это может привести к переполнению памяти и, как следствие, к неверному результату.

10. Приведите примеры использования собственных чисел и собственных векторов в численных методах.

- Оценка числа обусловленности матрицы A : $\text{cond}A \geq \frac{|\lambda_{\max}|}{|\lambda_{\min}|}$
- Нахождение оптимального параметра τ при решении СЛАУ с матрицей системы $A = A^T > 0$ методом простых итераций: $\tau_{opt} = \frac{2}{\lambda_{\max} + \lambda_{\min}}$