Алгоритм 4

Обозначения

В данной работе рассматривается и имплементируется предложенный Огита и Аишима итеративный метод уточнения сингулярных значений для полного сингулярного разложения [2].

Рассмотрим данный итеративный метод уточнения сингулярных значений для матрицы с действительными коэффициентами $A \in \mathbb{R}^{m \times n}$, $m \geq n$. В случае $m < n$ будет решаться задача для транспонированной матрицы A^{T}, поскольку сингулярное разложение A^{T} и A совпадает [7].

Пусть $\sigma_i \in \mathbb{R}, i = 1, 2, \ldots, n$ – сингулярные числа. Полным сингулярным разложением матрицы A будем считать такое разложение

$$A = U\Sigma V^{\mathrm{T}}, \quad U \in \mathbb{R}^{m \times m}, \quad V \in \mathbb{R}^{n \times n}, \quad \Sigma \in \mathbb{R}^{m \times n}, \text{где}$$

матрицы U и V ортогональные, матрица Σ диагональная и $\Sigma_{ii} = \sigma_i$.

Далее будем считать, что $\sigma_1 > \sigma_2 > \cdots > \sigma_n > 0$, то есть все сингулярные значения различные. Также $||\cdot||$ обозначает спектральную норму, т. е. $\sigma_{max}(A)$. E – единичная матрица. В тексте различаются приблизительные значения и посчитанные значения соответственно $\tilde{a}$ и $\hat{a}$.

Описание исходного алгоритма

Известно, что вычисление сингулярного разложения преимущественно ограниченно матричным умножением, что и является основной вычислительной сложностью алгоритма. Есть несколько подходов к матричному умножению с высокой точностью: XBLAS [3], быстрые и точные вычисления скалярного произведения [4] и произведений матриц [5], основанные на error-free transformations.

Рассматриваемый алгоритм использует такие соотношения:

1. В силу ортогональности U:

$$U^{\mathrm{T}}U = E \tag{1}$$

2. В силу ортогональности V:

$$V^{\mathrm{T}}V = E \tag{2}$$

3. В силу диагонализируемости A:

$$U^{\mathrm{T}}AV = \Sigma \tag{3}$$

Пусть $\widehat{U} \in \mathbb{R}^{m\times m}$ и $\hat{V} \in \mathbb{R}^{n\times n}$ приближенные значения матриц U и V. Корректирующими матрицами $F \in \mathbb{R}^{m\times m}$ и $G \in \mathbb{R}^{n\times n}$ будем называть такие матрицы, которые удовлетворяют равенствам $U = \widehat{U}(E+F)$ и $V = \hat{V}(E+G)$. Пусть ϵ определяется как

$$\epsilon \coloneqq \max(\epsilon_F, \epsilon_G), \quad \epsilon_F \coloneqq \|F\|, \quad \epsilon_G \coloneqq \|G\|.$$

Полагаем, что ϵ мало и $\epsilon < 1$, тогда обе матрицы $(E+F)$ и $(E+G)$ невырожденные и можно разложить в ряд Тейлора [9] обратные матрицы

$$(E+F)^{-1} = E - F + \Delta_F,\ \Delta_F \coloneqq \sum_{k=2}^{\infty}(-F)^k,\ \ ||\Delta_F|| \le \frac{\epsilon_F}{1-\epsilon_F},$$

$$(E+G)^{-1} = E - G + \Delta_G,\ \Delta_G \coloneqq \sum_{k=2}^{\infty}(-G)^k,\ \ ||\Delta_G|| \le \frac{\epsilon_G}{1-\epsilon_G}.$$

Подставив $U = \widehat{U}(E+F)$ в (1), получим

$$(E+F^T)\widehat{U}^T\widehat{U}(E+F) = E \ \ \Rightarrow\ \ \widehat{U}^T\widehat{U} = (E+F^T)^{-1}(E+F)^{-1}$$

$$\widehat{U}^T\widehat{U} = (E - F^T + \Delta_F^T)(E - F + \Delta_F),$$

из последнего следует

$$F + F^T = E - \widehat{U}^T\widehat{U} + \Delta_1, \qquad \Delta_1 = \Delta_F + \Delta_F^T + (F-\Delta_F)^T(F-\Delta_F). \tag{4}$$

Аналогичное выражение можно получить, подставив $V = \hat{V}(E+G)$ в (2)

$$G + G^T = E - \hat{V}^T\hat{V} + \Delta_2, \qquad \Delta_2 = \Delta_G + \Delta_G^T + (G-\Delta_G)^T(G-\Delta_G). \tag{5}$$

Проделаем те же операции и подставим $U = \widehat{U}(E+F)$ и $V = \hat{V}(E+G)$ в уравнение (3)

$$\Sigma - F^T\Sigma - \Sigma G = \widehat{U}^T A\hat{V} + \Delta_3,\ \ \Delta_3 = -\Sigma\Delta_3 - \Delta_F^T\Sigma - (\mathrm{F}-\Delta_F)^T\Sigma(G-\Delta_G). \tag{6}$$

Проведем оценку остаточных членов:

$$\|\Delta_1\| \le \frac{(3-2\epsilon_F)\epsilon_F^2}{(1-\epsilon_F)^2} \le \chi(\epsilon)\epsilon^2,$$

$$\|\Delta_2\| \le \frac{(3-2\epsilon_G)\epsilon_G^2}{(1-\epsilon_G)^2} \le \chi(\epsilon)\epsilon^2,$$

$$\|\Delta_3\| \le \frac{\epsilon_F^2 + \epsilon_G^2 + (1-\epsilon_F-\epsilon_G)\epsilon_F\epsilon_G}{(1-\epsilon_F)(1-\epsilon_G)}\|\Sigma\| \le \chi(\epsilon)\epsilon^2\|A\|,\text{где}$$

$$\chi(\epsilon) = \frac{3-2\epsilon}{(1-\epsilon)^2}.$$

Опустим слагаемые второго порядка у Δ_1, Δ_2, Δ_3 в (4), (5), (6) и получим систему матричных уравнений для $\tilde{F} = (\tilde{f}_{ij}) \in \mathbb{R}^{m\times m}, \tilde{G} = (\tilde{g}_{ij}) \in \mathbb{R}^{n\times n}$ и $\tilde{\Sigma} = diag(\tilde{\sigma}_i) \in \mathbb{R}^{m\times n}$

$$\begin{cases} \tilde{F} + \tilde{F}^T = R, \; R = E - \hat{U}^T\hat{U} \\ \tilde{G} + \tilde{G}^T = S, \; S = E - V^T\hat{V} \\ \tilde{\Sigma} - \tilde{F}^T\tilde{\Sigma} - \tilde{\Sigma}\tilde{G} = T, \; T = \hat{U}^T A \hat{V} \end{cases}. \tag{7}$$

Таким образом, остается решить систему уравнений и найти $\tilde{F}, \tilde{G}, \tilde{\Sigma}$. Эффективнее всего это сделать, представив матрицы $\tilde{F}, \tilde{\Sigma}, R, T$ как блочные:

$$\tilde{F} = \begin{bmatrix} \tilde{F}_{11} & \tilde{F}_{12} \\ \underbrace{\tilde{F}_{21}}_{n} & \underbrace{\tilde{F}_{22}}_{m-n} \end{bmatrix} \begin{matrix} \}_n \\ \}_{m-n} \end{matrix}, \quad \tilde{\Sigma} = \begin{bmatrix} \overbrace{\tilde{\Sigma}_n}^{n} \\ 0 \end{bmatrix} \begin{matrix} \}_n \\ \}_{m-n} \end{matrix}$$

$$R = \begin{bmatrix} R_{11} & R_{12} \\ \underbrace{R_{21}}_{n} & \underbrace{R_{22}}_{m-n} \end{bmatrix} \begin{matrix} \}_n \\ \}_{m-n} \end{matrix}, \quad \mathrm{T} = \begin{bmatrix} \overbrace{\mathrm{T}_1}^{n} \\ \mathrm{T}_2 \end{bmatrix} \begin{matrix} \}_n \\ \}_{m-n} \end{matrix}$$

Тогда первое уравнение системы можно переписать:

$$\tilde{F}_{11} + \tilde{F}_{11}^T = R_{11}, \tag{7a}$$

$$\tilde{F}_{21} + \tilde{F}_{12}^T = R_{21}, \tag{7b}$$

$$\tilde{F}_{22} + \tilde{F}_{22}^T = R_{22}, \tag{7c}$$

третье уравнение системы перепишется так

$$\tilde{\Sigma}_n - \tilde{F}_{11}^T\tilde{\Sigma}_n - \tilde{\Sigma}_n\tilde{G} = \mathrm{T}_1, \tag{7d}$$

$$\tilde{F}_{12}^T\tilde{\Sigma}_n = -\mathrm{T}_2 \Leftrightarrow \tilde{\Sigma}_n\tilde{F}_{12} = -T_2^T. \tag{7e}$$

Очевидно, что диагональные элементы матриц $\tilde{F}$ и $\tilde{G}$ достаточно просто выражаются

$$\tilde{f}_{ii} = \frac{r_{ii}}{2}, \quad \tilde{g}_{ii} = \frac{s_{ii}}{2}, 1 \le i \le n.$$

Третье уравнение системы для диагональных элементов выглядит так

$$(1 - \tilde{f}_{ii} - \tilde{g}_{ii})\tilde{\sigma}_i = \left(1 - \frac{r_{ii} + s_{ii}}{2}\right)\tilde{\sigma}_i = t_{ii}, \quad 1 \le i \le n.$$

$$\tilde{\sigma}_i = \frac{t_{ii}}{1 - {(r_{ii} + s_{ii})}/{2}}, \text{если } r_{ii} + s_{ii} \ne 2. \tag{8}$$

Замечание. В общем случае возможно равенство $r_{ii}+s_{ii}=2$, однако на самом деле R и S это невязки с точки зрения ортогональности U и V, поэтому зачастую на практике $|r_{ii}|\ll 1$, $|s_{ii}|\ll 1$ и $|r_{ii}+s_{ii}|\ll 1$.

Остается найти недиагональные значения матриц $\tilde{F}_{11}$ и $\tilde{G}$. Рассмотрим систему уравнений (7) и (8):

$$\begin{cases} \tilde{f}_{ij}+\tilde{f}_{ji}=r_{ij} \\ \tilde{g}_{ij}+\tilde{g}_{ji}=s_{ij} \\ \tilde{\sigma}_i\tilde{f}_{ij}+\tilde{\sigma}_j\tilde{g}_{ji}=-t_{ji} \\ \tilde{\sigma}_j\tilde{f}_{ji}+\tilde{\sigma}_i\tilde{g}_{ij}=-t_{ij} \end{cases},\quad 1\le i,j\le n, i\ne j.$$

Умножим третье и четвертое уравнения системы на $\tilde{\sigma}_i$ и $\tilde{\sigma}_j$ соответственно,

$$\tilde{\sigma}_i^2\tilde{f}_{ij}+\tilde{\sigma}_i\tilde{\sigma}_j\tilde{g}_{ji}=-\tilde{\sigma}_i t_{ji},$$

$$\tilde{\sigma}_j^2\tilde{f}_{ji}+\tilde{\sigma}_i\tilde{\sigma}_j\tilde{g}_{ij}=-\tilde{\sigma}_j t_{ij}.$$

Сложив полученные уравнения и подставив второе уравнение, получим

$$\tilde{\sigma}_i^2\tilde{f}_{ij}+\tilde{\sigma}_j^2\tilde{f}_{ji}+\tilde{\sigma}_i\tilde{\sigma}_j\left(\tilde{g}_{ji}+\tilde{g}_{ij}\right)=-\tilde{\sigma}_i t_{ji}-\tilde{\sigma}_j t_{ij} \;\Rightarrow$$

$$\Rightarrow \tilde{\sigma}_i^2\tilde{f}_{ij}+\tilde{\sigma}_j^2\tilde{f}_{ji}=-\tilde{\sigma}_i t_{ji}-\tilde{\sigma}_j t_{ij}-\tilde{\sigma}_i\tilde{\sigma}_j s_{ij}.$$

В найденное выражение подставим первое уравнение системы

$$\left(\tilde{\sigma}_j^2-\tilde{\sigma}_i^2\right)\tilde{f}_{ij}=\tilde{\sigma}_j^2 r_{ij}+\tilde{\sigma}_i t_{ji}+\tilde{\sigma}_j t_{ij}+\tilde{\sigma}_i\tilde{\sigma}_j s_{ij}=\tilde{\sigma}_j\left(t_{ij}+\tilde{\sigma}_j r_{ij}\right)+\tilde{\sigma}_i\left(t_{ji}+\tilde{\sigma}_j s_{ij}\right).$$

Аналогичное выражение можно получить и для $\tilde{G}$

$$\left(\tilde{\sigma}_j^2-\tilde{\sigma}_i^2\right)\tilde{g}_{ij}=\tilde{\sigma}_i\left(t_{ij}+\tilde{\sigma}_j r_{ij}\right)+\tilde{\sigma}_j\left(t_{ji}+\tilde{\sigma}_j s_{ij}\right).$$

Таким образом, получаем выражения для недиагональных элементов

$$\tilde{f}_{ij}=\frac{\alpha_{ij}\tilde{\sigma}_j+\beta_{ij}\tilde{\sigma}_i}{\tilde{\sigma}_j^2-\tilde{\sigma}_i^2},\quad \tilde{g}_{ij}=\frac{\alpha_{ij}\tilde{\sigma}_i+\beta_{ij}\tilde{\sigma}_j}{\tilde{\sigma}_j^2-\tilde{\sigma}_i^2},\;\; \tilde{\sigma}_i\ne\tilde{\sigma}_j \text{ для } 1\le i,j\le n, i\ne j, \quad (9)$$

где $\alpha_{ij}=t_{ij}+\tilde{\sigma}_j r_{ij}$ и $\beta_{ij}=t_{ji}+\tilde{\sigma}_j s_{ij}$.

Все эти рассуждения применимы для нахождения элементов остальных матриц $\tilde{F}_{12},\tilde{F}_{21},\tilde{F}_{22}$. Например, совместив уравнения (7*e*) и (8), значения матрицы $\tilde{F}_{12}$ получатся

$$\tilde{f}_{ij}=-\frac{t_{ji}}{\tilde{\sigma}_i},\tilde{\sigma}_i\ne 0, 1\le i\le n, n+1\le j\le m.$$

Из (7*b*) уравнения выражается $\tilde{F}_{21}$: $\tilde{F}_{21}=R_{21}-\tilde{F}_{12}^T$.

И значения элементов соответственно

$$\tilde{f}_{ij} = r_{ij} - \tilde{f}_{ji} = r_{ij} + \frac{t_{ij}}{\tilde{\sigma}_j}, \tilde{\sigma}_j \neq 0, n+1 \leq i \leq m, 1 \leq j \leq n.$$

Благодаря условию ($7c$) определяется $\tilde{F}_{22}$.

$$\tilde{f}_{ij} = \frac{r_{ij}}{2}, \qquad n+1 \leq i, j \leq m, i \neq j.$$

Замечание. В этом алгоритме мы полагаем, что $\tilde{\sigma}_i \neq \tilde{\sigma}_j$ для всех пар (i, j). Если же это условие не выполняется, то существует подход для решения этой проблемы [6].

Модифицированный алгоритм

Для реализации алгоритма [1] выполним часть вычислений исходного алгоритма в матричном виде. Этот подход увеличивает сложность алгоритма за счёт дополнительного матричного умножения, но дает возможность получить высокую точность при помощи точных матричных умножений [5].

Заменим вычисление коэффициентов α_{ij} и β_{ij} в формуле (9) на аналогичное вычисление коэффициентов в матричном виде. Тогда получится

$$\mathrm{C}\alpha = T_1 + R_{11} * \tilde{\Sigma}_n \text{ и } \mathrm{C}\beta = T_1^T + S * \tilde{\Sigma}_n.$$

Введем матрицы D и E

$$D = \tilde{\Sigma}_n * \mathrm{C}\alpha + \mathrm{C}\beta * \tilde{\Sigma}_n,$$

$$E = \mathrm{C}\alpha * \tilde{\Sigma}_n + \tilde{\Sigma}_n * \mathrm{C}\beta.$$

Тогда значение недиагональных элементов матриц $\tilde{F}_{11}$ и $\tilde{G}$ примет вид

$$\tilde{f}_{ij} = \frac{e_{ij}}{\tilde{\sigma}_j^2 - \tilde{\sigma}_i^2} \text{ и } \tilde{g}_{ij} = \frac{d_{ij}}{\tilde{\sigma}_j^2 - \tilde{\sigma}_i^2}.$$

```
Вход: A ∈ ℝ^{m×n}, m ≥ n, Û ∈ ℝ^{m×m}, V̂ ∈ ℝ^{n×n}
Выход: Û' ∈ ℝ^{m×m}, V̂' ∈ ℝ^{n×n}, Σ̃' = diag(σ̃_i) ∈ ℝ^{m×n}
R ← E − Û^T Û;   S ← E − V̂^T V̂;   T ← Û^T A V̂
# счет приблизительных сингулярных значений
for  i ← 1 to n do
     sigma[i] ← t[i][i]/(1 − (r[i][i] + s[i][i])/2)
sigma_arr ← diag(sigma)

# счет диагональных элементов матриц F̃_11 и G̃
for  i ← 1 to n do
     f[i][i] ← r[i][i]/2
     g[i][i] ← g[i][i]/2

# Выделение квадратных матриц n x n из T и R
T1 = T.topRows(n)
R11 = R.topLeftCorner(n, n)

# Вычисление матриц коэффициентов
Ca ← (T1 + R11 ∗ sigma_arr)
Cb ← (T1.transpose() + S ∗ sigma_arr)

D ← (sigma_arr ∗ Ca + Cb ∗ sigma_arr)
E ← (Ca ∗  sigma_arr + sigma_arr ∗  Cb)
for i ← 1 to n do
    # Вычисление недиагональных элементов F̃_11 и G̃
    for j← 1 to n do
        if i != j then
            f[i][j] ← (E[i][j])/(sigma[j] − sigma[i])
            g[i][j] ← (D[i][j])/(sigma[j] − sigma[i])
        endif
    # Вычисление F̃_12
    for j ← n + 1 to m do
        f[i][j] ← −t[j][i]/sigma[i]

for i ← n + 1 to m do
    # Вычисление F̃_21
    for j ← 1 to n do
        f[i][j] ← r[i][j] − f[j][i]
    # Вычисление F̃_22
    for j ← n + 1 to m do
        f[i][j] ← r[i][j]/2

# Обновление значений Û', V̂'
U_new ← U + U ∗ F
V_new ← V + V ∗ G
```

Листинг 1. Полный алгоритм итеративного уточнения сингулярных значений.

В представленном алгоритме есть функции, которые ранее не упоминались: $diag(sigma)$ создает диагональную матрицу с переданными значениями *sigma*, *topRows*(n) возвращает первые n строк в матрице, *topLeftCorner*(n) возвращает верхнюю левую квадратную матрицу размера $n \times n$ из исходной, *transpose*() транспонирует матрицу.

Данный алгоритм обладает квадратичной сходимостью, это следует из следующей теоремы.

Теорема. Пусть $A \in \mathbb{R}^{m\times n}$, $\widehat{U} \in \mathbb{R}^{m\times m}$, $\widehat{V} \in \mathbb{R}^{n\times n}$ при $m \geq n$. $\sigma_{n+1} \coloneqq 0$, $U = \widehat{U}(E + F)$ и $V = \widehat{V}(E + G)$, ϵ определяется как $\epsilon \coloneqq \max(\|F\|, \|G\|)$. Аналогично определим $U' = \widehat{U}'(E + F')$ и $V' = \widehat{V}'(E + G')$, $\epsilon' \coloneqq \max(\|F'\|, \|G'\|)$, где U' и V' это результат работы описанного алгоритма уточнения сингулярных значений. Если выполняется условие

$$\epsilon < \frac{\min\limits_{1\leq i\leq n}(\sigma_i - \sigma_{i+1})}{30m\|A\|},$$

тогда

$$\epsilon' < \frac{7}{10}\epsilon, \quad \limsup_{\epsilon\to 0}\, \frac{\epsilon'}{\epsilon^2} \leq \frac{18m\|A\|}{\min\limits_{1\leq i\leq n}(\sigma_i - \sigma_{i+1})}.$$

Доказательство: [2].

Воспользуемся этой теоремой. Пусть $d \coloneqq \lceil -\log_{10}\tilde{\epsilon}\rceil$, где $\tilde{\epsilon} \coloneqq \max\left(\|\tilde{F}\|, \|\tilde{G}\|\right)$. Тогда учитывая, что $F \approx \tilde{F}$ и $G \approx \tilde{G}$, получаем

$$\max\left(\frac{\|U - \widehat{U}\|}{\|\widehat{U}\|}, \frac{\|V - \widehat{V}\|}{\|\widehat{V}\|}\right) \leq \epsilon \approx \tilde{\epsilon} \approx O(10^{-d}).$$

Применяя теорему, найдем, что

$$\max\left(\frac{\|U - \widehat{U}\|}{\|\widehat{U}\|}, \frac{\|V - \widehat{V}\|}{\|\widehat{V}\|}\right) \leq \tilde{\epsilon} \leq \frac{18m\|A\|}{\min\limits_{1\leq i\leq n}(\sigma_i - \sigma_{i+1})}\epsilon^2 \approx \frac{18m\|A\|}{\min\limits_{1\leq i\leq n}(\sigma_i - \sigma_{i+1})}\tilde{\epsilon}^2 \approx$$
$$\approx O(10^{-2d}), \text{при } \epsilon \to 0.$$

Таким, образом требуемая арифметическая точность алгоритма составляет около $2d$ десятичных цифр. Хотя арифметическая точность в двух последних строках алгоритма составляет *d* десятичных цифр. Это происходит ввиду того, что только первые d десятичных цифр $\widehat{U}$ и $\widehat{V}$ точны, и только первые d десятичных цифр $\widehat{U}\tilde{F}$ и $\widehat{V}\tilde{G}$ могут повлиять на результат [1]. В итоге, вычислительная сложность алгоритма (количество требуемых операций) с точностью $2d$ и d десятичных цифр соответственно $m^3 + 2m^2n + 2mn^2 + n^3$ до $2m^3 + 2m^2n + 2mn^2 + 2n^3$ и $2m^3 + 2n^3$ [1].

Тестирование алгоритма

Тестирование алгоритма проводилось на прямоугольных и квадратных матрицах разного размера. Сначала генерировались сингулярные значения в некотором интервале, по которым восстанавливались матрицы A, U, V. После в матрицы U, V добавлялся псевдослучайный шум, применялся алгоритм Огиты – Аишимы и оценивалась точность полученных ошибок для матриц U, Σ, V. Полученные результаты представлены в файле *results.csv*.

Также было проведено следующее тестирование. Аналогично прошлому варианту генерировались сингулярные значения в некотором интервале, по которым восстанавливались матрицы A, U, V. Далее применялся алгоритм сингулярного разложения *Jacobi* [8] с последующим уточнением результатов алгоритмом Огиты-Аишимы. Полученные результаты представлены в файле *results2.csv*.

При проведении тестов, представленных ниже, помимо уточнения сингулярных чисел, оценивалась невязка по фробениусовой норме исходной матрицы и восстановленной матрицы после 1-й итерации алгоритма.

Случай действительной матрицы

Приведем примеры работы алгоритма для 5 итераций на действительной матрице со следующим тестированием: сгенерируем сингулярные значения на интервале $[0, 10]$ и восстановим матрицы A, U, V. В матрицы U, V добавим псевдослучайный шум с уровнем 10^{-15}.

Исходные матрицы.

A:

$$\begin{pmatrix} 4.10336775836158208428 & 3.29563031926548808499 & 2.98311937547734791184 \\ -1.83714064901915822723 & 4.81146718285331530772 & -0.87937211634792041172 \\ -0.41307885747135677568 6 & -1.60458059115785352266 & -6.20393499587064470259 \\ -3.69810676996781650429 & -3.80611558124069507682 & 0.05841966814797664366 99 \end{pmatrix},$$

U:

$$\begin{pmatrix} -0.66119324751568990679 8 & -0.00581964040032574632407 & -0.31594546858578353857 1 & 0.68041757921542281854 1 \\ -0.19512644118361285494 & -0.62504802301705331390 4 & 0.74095297189569920144 3 & 0.14909505126737927134 5 \\ 0.54364491311871517112 7 & -0.63327920821140023523 9 & -0.45385688627338098099 9 & 0.31212430163240725237 \\ 0.47874771000059971991 1 & 0.45633161752745784954 5 & 0.38102635223993652150 8 & 0.64605031066681131647 8 \end{pmatrix},$$

S:

$$\begin{pmatrix} 8.89835108803662434059 & 0 & 0 \\ 0 & 5.85432592672243339417 & 0 \\ 0 & 0 & 4.45451069918437993515 \\ 0 & 0 & 0 \end{pmatrix},$$

V:

$$\begin{pmatrix} -0.48881783390119570376 4 & -0.05150854090206287415 62 & -0.87086393625771618817 4 \\ -0.65319789236808748595 6 & -0.64008748985155354001 9 & 0.40450033218951222496 4 \\ -0.57826433286692638987 4 & 0.76657346389610817239 6 & 0.27924090994030709188 9 \end{pmatrix}.$$

Матрицы после применения алгоритма.

$\hat{A}$:

$$\begin{pmatrix} 4.10336775836158208384 & 3.29563031926548808499 & 2.98311937547734791184 \\ -1.83714064901915822702 & 4.81146718285331530772 & -0.87937211634792041177 5 \\ -0.41307885747135677514 3 & -1.60458059115785352266 & -6.20393499587064470259 \\ -3.69810676996781650429 & -1.60458059115785352266 & 0.05841966814797664375 12 \end{pmatrix},$$

$\widehat{U}$:

$$\begin{pmatrix} -0.66119324751568990685 2 & -0.00581964040032574642868 & -0.31594546858578353851 7 & 0.68041757921542281859 5 \\ -0.19512644118361285494 & -0.62504802301705331368 8 & 0.74095297189569920160 6 & 0.14909505126737927133 2 \\ 0.54364491311871517118 1 & -0.63327920821140023534 8 & -0.45385688627338098078 2 & 0.31212430163240725234 3 \\ 0.47874771000059971996 5 & 0.45633161752745784965 3 & 0.3810263522399365214 & 0.64605031066681131642 4 \end{pmatrix},$$

$\hat{S}$:

$$\begin{pmatrix} 8.89835108803662433972 & 0 & 0 \\ 0 & 5.85432592672243339374 & 0 \\ 0 & 0 & 4.45451069918437993515 \\ 0 & 0 & 0 \end{pmatrix},$$

$\widehat{V}$:

$$\begin{pmatrix} -0.48881783390119570371 & -0.05150854090206287440 36 & -0.87086393625771618817 4 \\ -0.65319789236808748595 6 & -0.64008748985155353996 5 & 0.40450033218951222518 1 \\ -0.57826433286692638987 4 & 0.76657346389610817250 4 & 0.27924090994030709164 5 \end{pmatrix}.$$

Посчитаем невязки матриц.

$$\left\|A - \hat{A}\right\|_1 = 1.54499 * 10^{-18};\ \left\|A - \hat{A}\right\|_2 = 7.65208 * 10^{-19}.$$

$$\left\|U - \widehat{U}\right\|_2 = 4.30403 * 10^{-19};\ \left\|S - \hat{S}\right\|_2 = 9.6974 * 10^{-19};$$

$$\left\|V - \widehat{V}\right\|_2 = 4.30506 * 10^{-19}.$$

Случай комплексной матрицы.

Для вычисления сингулярного разложения комплексной матрицы 6×5 используется *Jacobi* [8] с последующим уточнением алгоритмом Огиты - Аишимы.

Оригинальная матрица:

$$\begin{pmatrix} 0.127171 - 0.997497i & 0.617481 - 0.613392i & -0.0402539 + 0.170019i & 0.791925 - 0.299417i & 0.49321 + 0.64568i \\ 0.717887 - 0.651784i & 0.0270699 + 0.421003i & -0.970031 - 0.39201i & -0.271096 - 0.817194i & -0.668203 - 0.705374i \\ -0.108615 + 0.97705i & -0.990661 - 0.761834i & -0.24424 - 0.982177i & 0.142369 + 0.0633259i & 0.214331 + 0.203528i \\ 0.32609 - 0.667531i & -0.295755 - 0.0984222i & 0.215369 - 0.885922i & 0.605213 + 0.566637i & -0.3961 + 0.0397656i \\ 0.453352 + 0.751946i & 0.851436 + 0.911802i & -0.715323 + 0.0787072i & -0.529344 - 0.0758385i & -0.580798 + 0.724479i \\ 0.687307 + 0.559313i & 0.99939 + 0.993591i & -0.215125 + 0.222999i & -0.405438 - 0.467574i & -0.952513 + 0.680288i \end{pmatrix}$$

Восстановленная матрица после *Jacobi*:

$$\begin{pmatrix} 0.127171 - 0.997498i & 0.617482 - 0.613392i & -0.0402537 + 0.170019i & 0.791925 - 0.299417i & 0.49321 + 0.64568i \\ 0.717888 - 0.651784i & 0.02707 + 0.421004i & -0.970032 - 0.39201i & -0.271096 - 0.817195i & -0.668203 - 0.705374i \\ -0.108615 + 0.97705i & -0.990662 - 0.761834i & -0.244239 - 0.982177i & 0.142369 + 0.0633259i & 0.214332 + 0.203528i \\ 0.326091 - 0.667532i & -0.295755 - 0.0984224i & 0.215369 - 0.885922i & 0.605213 + 0.566638i & -0.396099 + 0.0397656i \\ 0.453353 + 0.751946i & 0.851436 + 0.911802i & -0.715324 + 0.0787074i & -0.529344 - 0.0758386i & -0.580798 + 0.724479i \\ 0.687308 + 0.559313i & 0.99939 + 0.993592i & -0.215125 + 0.222999i & -0.405439 - 0.467574i & -0.952514 + 0.680289i \end{pmatrix}$$

Восстановленная матрица после алгоритма Ogita-Aishima:

$$\begin{pmatrix} 0.127171-0.997497i & 0.617481-0.613391i & -0.040254+0.170019i & 0.791925-0.299417i & 0.49321+0.64568i \\ 0.717887-0.651784i & 0.0270699+0.421003i & -0.970031-0.39201i & -0.271096-0.817194i & -0.668203-0.705374i \\ -0.108615+0.97705i & -0.990661-0.761834i & -0.24424-0.982177i & 0.142369+0.0633259i & 0.214332+0.203528i \\ 0.32609-0.667531i & -0.295755-0.0984222i & 0.215369-0.885922i & 0.605213+0.566637i & -0.3961+0.0397657i \\ 0.453353+0.751946i & 0.851436+0.911802i & -0.715323+0.0787073i & -0.529344-0.0758386i & -0.580798+0.724479i \\ 0.687307+0.559313i & 0.99939+0.993591i & -0.215125+0.222999i & -0.405438-0.467574i & -0.952513+0.680288i \end{pmatrix}$$

Таким образом, тест на комплексных матрицах показал, что имплементированный алгоритм работает не только на действительных матрицах, но и на комплексных матрицах. Однако это подлежит дальнейшему изучению.

Дополнительные тесты

Таблица 1. Значение характеристик для псевдослучайных действительных матриц.

Matrix Size	Relative Error (‖A - U*S*V^T‖ / ‖A‖)	max(‖F‖, ‖G‖)	max(‖R‖, ‖S‖)	Elapsed Time (s)
10×10	7.12345e-08	8.74816e-08	1.74963e-07	0.0032547
30×30	1.14187e-07	2.67087e-07	5.34173e-07	0.0243358
50×50	1.33093e-07	3.75116e-07	7.50232e-07	0.110236
60×45	6.47478e-07	3.02241e-06	6.04481e-06	0.182544

Применение безошибочного преобразования матричного умножения в алгоритме Огиты – Аишимы

Одним из существенных недостатков алгоритма (рисунок 1) является частое применение матричного умножения, что влияет на сложность алгоритма и его точность. Попробуем применить один из методов безошибочного матричного умножения [5], чтобы повысить точность алгоритма. В статье [5] применяются быстрые рутины в *BLAS*, в настоящей имплементации мы воспользуемся инструментами *eigen* [8].

Проведем тесты с применением *eigen* умножения и имплементированного алгоритма точного умножения в алгоритме Огиты-Аишимы на 10 итерациях. Проведем тестирование на матрицах размера 30×30, 60×60, 100×100 с генерацией сингулярных значений на отрезке $[0, 10]$, в восстановленные матрицы U, V добавим псевдослучайный шум с уровнем 10^{-15}.

Таблица 2. Сравнение результатов работы алгоритма Огиты-Аишимы с применением встроенного матричного умножения и безошибочного матричного умножения [5] для действительных матриц.

Размер матрицы	$\left\|A - \hat{U} * \hat{S} * \hat{V}^{\mathrm{T}}\right\|_2$	Время выполнения (мс)	Метод
30×30	$6{,}42142 * 10^{-18}$	282,256	*eigen* (c)
30×30	$9{,}53216 * 10^{-14}$	1048,95	алгоритм [5]
60×60	$1{,}11905 * 10^{-17}$	2132,03	*eigen* (c)
60×60	$1{,}42568 * 10^{-13}$	6581,68	алгоритм [5]
100×100	$1{,}82237 * 10^{-17}$	4092,89	*eigen* (c)
100×100	$8{,}11211 * 10^{-13}$	28115,3	алгоритм [5]

При тестировании алгоритма для уточнения сингулярных значений с применением точного матричного умножения были получены результаты хуже, чем с применением средств *eigen*. Вероятнее всего, алгоритм плохо протестирован и требует улучшений.

Список литературы

[1] Uchino Y., Terao T., Ozaki K.: Acceleration of iterative refinement for singular value decomposition / Numerical Algorithms. — 2024. — № 95. — С. 979–1009.

[2] Ogita, T., Aishima, K.: Iterative refinement for singular value decomposition based on matrix multiplication / J. Comput. Appl. — 2020. — №369. – 112512.

[3] Li X. S., Demmel J. W., Bailey D. H., Henry G., Hida Y., Iskandar J., Kahan W., Kang S. Y., Kapur A., Martin M. C., Thompson B. J., Tung T., Yoo D.: Design, implementation and testing of extended and mixed precision BLAS, ACM Trans. / Math. Software. —2002. — № 28. — С. 152–205.

[4] Ogita T., Rump S. M., Oishi S.: Accurate sum and dot product / SIAM J. Sci. Comput. —2005. — № 26:6. — С. 1955–1988.

[5] Ozaki K., Ogita T., Oishi S., Rump S. M.: Error-free transformations of matrix multiplication by using fast routines of matrix multiplication and its applications / Numerical Algorithms. — 2012. — 59:1. — С. 95-118.

[6] Ogita T., Aishima K.: Iterative refinement for symmetric eigenvalue decomposition / Japan J. Indust. Appl. Math. — 2018, № 35:3. — С. 1007–1035.

[7] Golub G. Matrix Computations / Gene H. Golub, Charles F. Van Loan. — 4th ed. — Baltimore : Johns Hopkins University Press, 2013. — С. 756.

[8] Eigen [Электронный ресурс]: библиотека для линейной алгебры на C++ / разработчики G. Guennebaud, B. Jacob и др. – URL: https://eigen.tuxfamily.org.

[9] Зорич В. А. Математический анализ. Ч. 1 / В. А. Зорич. — 6-е изд., испр. — М.: МЦНМО, 2012. — 564 с.