Math 5601 Independent Study Project

Jacob Hauck

1 Introduction

2 Theory

Definition 1. (Index slice) *Let $m \leq n$, where m and n are integers. Define the **index slice from** m **to** n by the sequence*

$$m : n = \{i\}_{i=m}^{n}. \tag{1}$$

Definition 2. (Submatrix) *Let $A \in \mathbf{R}^{m\times n}$ be a matrix. Let $I = \{I_i\}_{i=1}^{r}$ be a sequence of distinct row indices of A, and let $J = \{J_j\}_{j=1}^{c}$ be a sequence of distinct column indices of A. The **submatrix of** A **with rows** I **and columns** J is the matrix $A(I,J) \in \mathbf{R}^{r\times c}$ with entries*

$$[A(I,J)]_{ij} = A_{I_i J_j}. \tag{2}$$

If the special symbol $:$ is used as row indices or column indices, it means the entire sequence $1 : m$ or $1 : n$.

If I or J is a single integer i or j instead of a sequence, we take this to mean $I = i : i$ or $J = j : j$, the sequence consisting of that one integer.

Definition 3. (Skeleton) *Let $A \in \mathbf{R}^{m\times n}$, and let $B = A(I,J) \in \mathbf{R}^{r\times r}$ be a nonsingular, square submatrix of A. Then the **skeleton of** A **with core** $G = B^{-1}$ is given by*

$$\mathscr{S}_G = A(:,J)A(I,J)^{-1}A(I,:) \in \mathbf{R}^{m\times n}. \tag{3}$$

Theorem 1. *Let $A \in \mathbf{R}^{m\times n}$. If $B = A(I,J) \in \mathbf{R}^{r\times r}$ is a square submatrix of A with rank r, then*

$$A(I,:) = \mathscr{S}_G(I,:), \qquad A(:,J) = \mathscr{S}_G(:,J), \tag{4}$$

where $G = B^{-1}$.

Proof. For $i \in 1 : r$, $j \in 1 : n$,

$$\mathscr{S}_G(I,:)_{ij} = \mathscr{S}_G(I_i, j) = \sum_{k=1}^{r}\sum_{\ell=1}^{r} A_{I_i J_k} G_{k\ell} A_{I_\ell j} = \sum_{k=1}^{r}\sum_{\ell=1}^{r} A(I,J)_{ik} G_{k\ell} A(I,:)_{\ell j} \tag{5}$$

$$= \left[A(I,J)GA(I,:)\right]_{ij} = [A(I,:)]_{ij}. \tag{6}$$

For $i \in 1:m$, $j \in 1:r$,

$$\mathscr{S}_G(:,J)_{ij} = \mathscr{S}_G(i,J_j) = \sum_{k=1}^{r}\sum_{\ell=1}^{r} A_{iJ_k} G_{k\ell} A_{I_\ell J_j} = \sum_{k=1}^{r}\sum_{\ell=1}^{r} A(:,J)_{ik} G_{k\ell} A(I,J)_{\ell j} \tag{7}$$

$$= \big[A(:,J) G A(I,J)\big]_{ij} = [A(:,J)]_{ij}. \tag{8}$$

□

Definition 4. **(Standard basis)** *Let $e_j \in \mathbf{R}^n$ denote the jth standard basis vector in $\mathbf{R}^n$.*

Theorem 2. **(Skeleton decomposition)** *Let $A \in \mathbf{R}^{m\times n}$ be a matrix with rank r. If $B = A(I,J) \in \mathbf{R}^{r\times r}$ is a square submatrix of A with rank r, and $G = B^{-1}$, then*

$$A = \mathscr{S}_G, \tag{9}$$

and $\mathscr{S}_G$ is called a ***skeleton decomposition*** *of A with* ***core*** *G.*

Proof. The columns of A at indices J (that is, $\{A(:,J_j)\}_{j=1}^{r}$) are linearly independent because

$$\sum_{j=1}^{r} \alpha_j A(:,J_j) = 0 \implies \sum_{j=1}^{r} \alpha_j A(I,J_j) = 0 \implies \alpha_j = 0, \quad j \in 1:r \tag{10}$$

because the columns $\{A(I,J_j)\}_{j=1}^{r}$ of $A(I,J)$ must be linearly independent by the fact that $A(I,J)$ has rank r.

Thus, since A has rank r, every other column of A must be a linear combination of the columns at indices J. That is, there exists $\{\alpha_{\ell j}\}$ for $j \in 1:r$ and $\ell \in 1:n$ such that

$$A(:,\ell) = \sum_{j=1}^{r} \alpha_{\ell j} A(:,J_j). \tag{11}$$

Define $\varphi : \mathbf{R}^r \to \operatorname{span}\{A(:,J_j) \mid j \in 1:r\}$ by $\varphi(e_j) = A(:,J_j)$. Clearly, φ is linear and onto. By the linear independence of $\{A(:,J_j)\}$, φ maps an r-dimensional space onto an r-dimensional space, so φ must also be one-to-one. Thus, φ is invertible, with $\varphi^{-1}(A(:,J_j)) = e_j$ for $j \in 1:r$.

Let $x \in \mathbf{R}^n$. Viewing A and $A(I,J)$ as linear mappings defined by matrix-vector multiplication, we have

$$(A(I,J) \circ \varphi^{-1} \circ A)(x) = (A(I,J) \circ \varphi^{-1}) \left(\sum_{\ell=1}^{n} A(:,\ell) x_\ell \right) = \sum_{\ell=1}^{n} x_\ell A(I,J) \varphi^{-1}(A(:,\ell)) \tag{12}$$

$$= \sum_{\ell=1}^{n} x_\ell A(I,J) \varphi^{-1} \left(\sum_{j=1}^{r} \alpha_{\ell j} A(:,J_j) \right) \tag{13}$$

$$= \sum_{\ell=1}^{n} x_\ell A(I,J) \sum_{j=1}^{r} \alpha_{\ell j} e_j = \sum_{\ell=1}^{n} x_\ell \sum_{j=1}^{r} \alpha_{\ell j} A(I,J_j) \tag{14}$$

$$= \sum_{\ell=1}^{n} x_\ell \left(\sum_{j=1}^{r} \alpha_{\ell j} A(:,J_j) \right)(I,:) = \sum_{\ell=1}^{n} A(I,\ell) x_\ell \tag{15}$$

$$= A(I,:)x. \tag{16}$$

Since x was arbitrary, and $A(I,J)$ and φ^{-1} are invertible, it follows that

$$A = \varphi \circ A(I,J)^{-1} \circ A(I,:) \tag{17}$$

as a linear map.

For any $x \in \mathbf{R}^n$, we can write $A(I,J)^{-1}A(I,:)x$ as a linear combination of $\{e_j\}_{j=1}^{r}$; that is, there exists $\{\beta_j\}$ such that

$$A(I,J)^{-1} A(I,:)x = \sum_{j=1}^{r} \beta_j e_j. \tag{18}$$

Then

$$Ax = \varphi \left(\sum_{j=1}^{r} \beta_j e_j \right) = \sum_{j=1}^{r} \beta_j A(:,J_j) = A(:,J) \sum_{j=1}^{r} \beta_j e_j = A(:,J) A(I,J)^{-1} A(I,:)x. \tag{19}$$

Since x was arbitrary, (9) follows. $\square$

Definition 5. **(Chebyshev Norm)** *If $A \in \mathbf{R}^{m \times n}$, define the* ***Chebyshev norm*** *of A by*

$$\|A\|_\infty = \max_{i,j} |A_{ij}|. \tag{20}$$

Definition 6. **(Volume)** *Let $A \in \mathbf{R}^{r \times r}$ be a square matrix. Then the* ***volume*** *of A is defined to be*

$$\mathcal{V}(A) = |\det(A)|. \tag{21}$$

Definition 7. **(Maximum volume submatrix)** *Let $A \in \mathbf{R}^{m \times n}$. A submatrix $A_\blacksquare = A(I,J) \in \mathbf{R}^{r \times r}$ of A is a* ***rank-r maximum volume submatrix*** *of A if*

$$\mathcal{V}(A_\blacksquare) = \max \Big\{ \mathcal{V}(A(I',J')) \mid A(I',J') \in \mathbf{R}^{r \times r} \text{ is a submatrix of } A \Big\}. \tag{22}$$

We will typically denote maximum volume submatrices of A by $A_\blacksquare$.

Definition 8. (Pseudo-skeleton decomposition) *Let $A \in \mathbf{R}^{m\times n}$ be a matrix, and let I and J be sequences of row indices of A of length r. If $G \in \mathbf{R}^{r\times r}$, then*

$$B = A(:,J)GA(I,:) \tag{23}$$

*is called a **pseudo-skeleton decomposition** of A with **core** G, **row indices** I, and **column indices** J.*

Lemma 1. (Submatrix of a product) *Let $A \in \mathbf{R}^{m\times r}$, and let $B \in \mathbf{R}^{r\times n}$. Then for any row indices I of A,*

$$(AB)(I,:) = A(I,:)B, \tag{24}$$

and for any column indices J of B,

$$(AB)(:,J) = AB(:,J). \tag{25}$$

Lemma 2. *For any square matrix $A \in \mathbf{R}^{n\times n}$,*

$$\|A\|_2 \leq n\,\|A\|_\infty \tag{26}$$

where $\|\cdot\|_2$ is the spectral norm. This inequality is sharp.

Proof. Let $x \in \mathbf{R}^n$. Overloading $\|\cdot\|_2$ to also mean the Euclidean vector norm in $\mathbf{R}^n$, the Cauchy-Schwarz inequality implies that

$$\|Ax\|_2 = \sqrt{\sum_{i=1}^{n} |Ax_i|^2} = \sqrt{\sum_{i=1}^{n} \left|\sum_{j=1}^{n} A_{ij}x_j\right|^2} \leq \sqrt{\sum_{i=1}^{n} \left(\sum_{j=1}^{n} |A_{ij}|^2\right)\left(\sum_{j=1}^{n} |x_j|^2\right)}. \tag{27}$$

By definition, $|A_{ij}|^2 \leq \|A\|_\infty^2$, so

$$\|Ax\|_2 \leq \sqrt{\sum_{i=1}^{n} n\,\|A\|_\infty^2\,\|x\|_2^2} = n\,\|A\|_\infty\,\|x\|_2\,. \tag{28}$$

If $\|x\|_2 \neq 0$, then

$$\frac{\|Ax\|_2}{\|x\|_2} \leq n\,\|A\|_\infty\,. \tag{29}$$

Taking the supremum over $x \neq 0$ on both sides completes the proof of the inequality.

For the sharpness, take $x \in \mathbf{R}^n$ such that $x_j = n^{-\frac{1}{2}}$ for $j \in 1:n$, so that $\|x\|_2 = 1$, and take $A \in \mathbf{R}^{n\times n}$ such that $A_{ij} = 1$ for all $i,j \in 1:n$. Then $\|A\|_\infty = 1$, and

$$\|Ax\|_2 = \sqrt{\sum_{i=1}^{n} \left|\sum_{j=1}^{n} n^{-\frac{1}{2}}\right|^2} = \sqrt{n^2} = n = n\,\|A\|_\infty\,. \tag{30}$$

This implies that $\|A\|_2 \geq n\,\|A\|_\infty$, so the inequality is sharp. $\square$

Lemma 3. (Submatrix determinants) *If A and D are square matrices, then*

$$\det\left(\begin{bmatrix} A & B \\ C & D \end{bmatrix}\right) = \begin{cases} \det(A)\det(D - CA^{-1}B) & \text{if } A^{-1} \text{ exists,} \\ \det(D)\det(A - BD^{-1}C) & \text{if } D^{-1} \text{ exists.} \end{cases} \tag{31}$$

Proof. See, for example, the proof of (6.2.1) by Meyer [2]. □

Lemma 4. (Submatrix Inversion) *If A and D are square matrices, and A^{-1} exists, then the block matrix*

$$X = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \tag{32}$$

is invertible if and only if $\Gamma = D - CA^{-1}B$ is invertible, and

$$X^{-1} = \begin{bmatrix} A^{-1} + A^{-1}B\Gamma^{-1}CA^{-1} & -A^{-1}B\Gamma^{-1} \\ -\Gamma^{-1}CA^{-1} & \Gamma^{-1} \end{bmatrix}. \tag{33}$$

Proof. We can show that Γ is invertible implies that X is invertible by showing that the formula given for X^{-1} is a left inverse of X by direct computation, which, coincidentally, proves the correctness of the formula as well:

$$\begin{aligned} &\begin{bmatrix} A^{-1} + A^{-1}B\Gamma^{-1}CA^{-1} & -A^{-1}B\Gamma^{-1} \\ -\Gamma^{-1}CA^{-1} & \Gamma^{-1} \end{bmatrix} \cdot \begin{bmatrix} A & B \\ C & D \end{bmatrix} \\ &\qquad = \begin{bmatrix} I + A^{-1}B\Gamma^{-1}C - A^{-1}B\Gamma^{-1}C & A^{-1}B + A^{-1}B\Gamma^{-1}CA^{-1}B - A^{-1}B\Gamma^{-1}D \\ -\Gamma^{-1}C + \Gamma^{-1}C & -\Gamma^{-1}CA^{-1}B + \Gamma^{-1}D \end{bmatrix} \\ &\qquad = \begin{bmatrix} I & A^{-1}B + A^{-1}B\Gamma^{-1}\left(CA^{-1}B - D\right) \\ 0 & \Gamma^{-1}\left(D - CA^{-1}B\right) \end{bmatrix} \\ &\qquad = \begin{bmatrix} I & A^{-1}B - A^{-1}B \\ 0 & I \end{bmatrix} = I. \end{aligned} \tag{34}$$

If X is not invertible, then by Lemma 3 we have $0 = \det(X) = \det(A)\det(D - CA^{-1}B) = \det(A)\det(\Gamma)$, which implies that Γ is not invertible because $\det(A) \neq 0$. □

Lemma 5. (Cauchy interlacing theorem) *Let $A \in \mathbf{R}^{n\times n}$ be a symmetric matrix with eigenvalues $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$. Let $A(I, I) \in \mathbf{R}^{r\times r}$ be a submatrix with the same row and column indices*[1] *I of A. If $\mu_1 \leq \mu_2 \leq \cdots \leq \mu_r$ are the eigenvalues of $A(I, I)$, then*

$$\lambda_k \leq \mu_k \leq \lambda_{k+(n-r)}, \qquad k \in 1 : r. \tag{35}$$

The condition in (35) is known as the ***interlacing of the eigenvalues****.*

Proof. See, for example, the proof of Parlett's Theorem 10.1.1 and the following Remark 10.1.1 [3]. □

[1] Usually called a **principal submatrix**.

Lemma 6. (Interlacing of singular values) *Let $A \in \mathbf{R}^{m\times n}$, and let $A(I,J) \in \mathbf{R}^{r\times r}$ be a submatrix of A with row indices I and column indices J. If $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_\ell$ are the singular values of A, where $\ell = \min\{m,n\}$, and $\rho_1 \geq \rho_2 \geq \cdots \rho_r$ are the singular values of $A(I,J)$, then*

$$\sigma_k \geq \rho_k, \qquad k \in 1:r \tag{36}$$
$$\rho_k \geq \sigma_{k+m+n-2r}, \qquad k \in 1:(\max\{m,n\}-2r). \tag{37}$$

Proof. Define
$$M = \begin{bmatrix} 0 & A^T \\ A & 0 \end{bmatrix} \in \mathbf{R}^{(m+n)\times(m+n)}. \tag{38}$$

We note that $\lambda \neq 0$ is an eigenvalue of M if and only if

$$0 = \det(M - \lambda I_{(m+n)\times(m+n)}) = \det \begin{bmatrix} -\lambda I_{n\times n} & A^T \\ A & -\lambda I_{m\times m} \end{bmatrix} \tag{39}$$
$$= \det(-I_{n\times n}\lambda)\det(-I_{n\times n}\lambda + \lambda^{-1}A^TA) \tag{40}$$

by Lemma 3. The lattermost expression is 0 if and only if $\det(A^TA - \lambda^2 I_{n\times n}) = 0$, that is, if and only if λ is a singular value of A. Thus, the nonzero singular values of A and the nonzero eigenvalues of M are the same.

Define
$$N = \begin{bmatrix} 0 & A(I,J)^T \\ A(I,J) & 0 \end{bmatrix}. \tag{41}$$

By nearly identical reasoning to that above, we can show that the nonzero eigenvalues of N and the singular values of $A(I,J)$ are the same.

Noting that M is symmetric, and N is a submatrix of M with the row and column indices

$$I' = \{J_1,\ldots,J_r,I_1,\ldots,I_r\}, \qquad J' = \{J_1,\ldots,J_r,I_1,\ldots,I_r\} = I', \tag{42}$$

we can apply Lemma 5 to M and N. Let $\lambda_1 \leq \lambda_2 \cdots \leq \lambda_{m+n}$ be the eigenvalues of M, and let $\mu_1 \leq \mu_2 \leq \cdots \leq \mu_{2r}$ be the eigenvalues of N. Accounting for the fact that the nonzero eigenvalues of N are $\rho_r \leq \rho_{r-1} \leq \cdots \leq \rho_1$, and the nonzero eigenvalues of M are $\sigma_\ell \leq \sigma_{\ell-1} \leq \cdots \leq \sigma_1$, we must have
$$\lambda_{m+n-k} = \sigma_{k+1}, \quad k \in 0:(\ell-1), \qquad \mu_{2r-k} = \rho_{k+1}, \quad k \in 0:(r-1), \tag{43}$$

and all the other eigenvalues of M and N are 0.

Then, by Lemma 5,

$$\lambda_k \leq \mu_k \leq \lambda_{k+(m+n-2r)}, \qquad k \in 1:2r \tag{44}$$
$$\Longrightarrow \lambda_{2r-k} \leq \mu_{2r-k} \leq \lambda_{m+n-k}, \qquad k \in 0:(2r-1) \tag{45}$$
$$\Longrightarrow \lambda_{m+n-(m+n-2r+k)} \leq \rho_{k+1} \leq \sigma_{k+1}, \qquad k \in 0:(r-1) \tag{46}$$
$$\Longrightarrow \rho_k \leq \sigma_k, \quad k \in 1:r, \qquad \sigma_{m+n-2r+k} \leq \rho_k, \quad k \in 1:r \text{ and } m+n-2r+k \leq \ell \tag{47}$$
$$\Longrightarrow \rho_k \leq \sigma_k, \quad k \in 1:r, \qquad \sigma_{m+n-2r+k} \leq \rho_k, \quad 1 \leq k \leq 2r - \max\{m,n\}. \tag{48}$$

□

Theorem 3. (Maximum volume pseudo-skeleton) *Let $A \in \mathbf{R}^{m\times n}$ be a matrix with singular values $\{\sigma_i\}$ in nonascending order. If $A_{\blacksquare} = A(I,J)$ is a rank-r maximum volume submatrix of A, then*

$$\left\| A - \mathscr{S}_{A_{\blacksquare}^{-1}} \right\|_\infty \leq (r+1)\sigma_{r+1}, \tag{49}$$

where $\sigma_{\min\{m,n\}+1} = 0$ by convention, and $\mathscr{S}_{A_{\blacksquare}^{-1}}$ is the pseudo-skeleton decomposition of A using rows and columns I and J, with core $A_{\blacksquare}^{-1}$.

Proof. Define $E = A - \mathscr{S}_{A_{\blacksquare}^{-1}}$. By Theorem 1, $E_{ij} = 0$ if $i = I_{i'}$ for some i' or $j = J_{j'}$ for some j'. If i does not appear in the sequence I and j does not appear in the sequence J, then define $\gamma = E_{ij}$, so that

$$\begin{aligned} \gamma = A(i,j) - \mathscr{S}_{A_{\blacksquare}^{-1}}(i,j) &= A(i,j) - \sum_{k=1}^{r}\sum_{\ell=1}^{r} A(i,J_k)A_{\blacksquare}^{-1}(k,\ell)A(I_\ell,j) \qquad (50)\\ &= A(i,j) - A(i,J)A_{\blacksquare}^{-1}A(I,j). \qquad (51) \end{aligned}$$

If we can show that this arbitrary (potentially) nonzero element of E satisfies $|\gamma| \leq (r+1)\sigma_{r+1}$, then $\|E\|_\infty \leq (r+1)\sigma_{r+1}$, and the proof is complete.

Extend I to I' by setting $I'_{r+1} = i$, and extend J to J' by setting $J'_{r+1} = j$. Define the matrix $\widehat{A} = A(I',J')$. By construction, we have

$$\widehat{A} = \begin{bmatrix} A_{\blacksquare} & A(I,j) \\ A(i,J) & A(i,j) \end{bmatrix}. \tag{52}$$

We note that by Lemma 4, the matrix $\widehat{A}$ is invertible if and only if $\gamma = A(i,j) - A(i,J)A_{\blacksquare}^{-1}A(I,j)$ is invertible, that is, nonzero. If $\gamma = 0$, then certainly $|\gamma| \leq (r+1)\sigma_{r+1}$.

Suppose that $\gamma \neq 0$. Then $\widehat{A}$ is invertible, and by Lemma 4,

$$\widehat{A}^{-1} = \begin{bmatrix} A_{\blacksquare}^{-1} + A_{\blacksquare}^{-1}A(I,j)\gamma^{-1}A(i,J)A_{\blacksquare}^{-1} & -A_{\blacksquare}^{-1}A(I,j)\gamma^{-1} \\ -\gamma^{-1}A(i,J)A_{\blacksquare}^{-1} & \gamma^{-1} \end{bmatrix}. \tag{53}$$

Hence $\left\|\widehat{A}^{-1}\right\|_\infty \geq \left|\gamma^{-1}\right|$. On the other hand, by Lemma 1, for $\ell \in 1:(r+1)$, the column $\widehat{A}^{-1}(:,\ell)$ satisfies the equation

$$\widehat{A}\widehat{A}^{-1}(:,\ell) = I_{(r+1)\times(r+1)}(:,\ell) = e_\ell. \tag{54}$$

Let $k \in 1:(r+1)$. Since $\widehat{A}$ is invertible, Cramer's ruler implies that

$$\widehat{A}^{-1}(k,\ell) = \frac{\det(M)}{\det\left(\widehat{A}\right)}, \tag{55}$$

where M is the matrix with $M(:,k) = e_\ell$, and $M(:,k') = \widehat{A}(:,k')$ if $k' \neq \ell$. That is,

$$M = \begin{bmatrix} \widehat{A}(:,1) & \cdots & \widehat{A}(:,k-1) & e_\ell & \widehat{A}(:,k+1) & \cdots \widehat{A}(:,r+1) \end{bmatrix}. \tag{56}$$

Let $I'' = \{1, 2, \ldots k-1, k+1, \ldots, r+1\}$. Expanding by cofactors on the kth column of M, we get $|\det(M)| = |\det(M')|$, where $M' = M(I'', I'')$. Since M coincides with $\widehat{A}$ on all but the kth column, it follows that $M' = M(I'', I'') = \widehat{A}(I'', I'') = A(I', J')(I'', I'')$. Hence, M' is an $r \times r$ submatrix of A.

By the maximality of the volume of $A_\blacksquare$, it follows that

$$\left|\widehat{A}^{-1}(k,\ell)\right| = \frac{|\det(M')|}{\left|\det\left(\widehat{A}\right)\right|} = |\det(M')| \cdot \left|\det\left(\widehat{A}^{-1}\right)\right| \le |\det(A_\blacksquare)| \cdot \left|\det\left(\widehat{A}^{-1}\right)\right|. \tag{57}$$

By Lemma 3,

$$\det\left(\widehat{A}\right) = \det(A_\blacksquare)\left(A(i,j) - A(i,J)A_\blacksquare^{-1}A(I,j)\right) = \det(A_\blacksquare)\,\gamma, \tag{58}$$

so

$$|\det(A_\blacksquare)| \cdot \left|\det\left(\widehat{A}^{-1}\right)\right| = \left|\gamma^{-1}\right|. \tag{59}$$

Therefore, $\left|\widehat{A}^{-1}(k,\ell)\right| \le \left|\gamma^{-1}\right|$. Since k and ℓ were arbitrary, it follows that $\left\|\widehat{A}^{-1}\right\|_\infty \le \left|\gamma^{-1}\right|$. Thus, $\left\|\widehat{A}^{-1}\right\|_\infty = \left|\gamma^{-1}\right|$.

Recall that $\sigma_1 \ge \sigma_2 \ge \cdots \ge \sigma_{\min\{m,n\}}$ are the singular values of A, and let $\rho_1 \ge \rho_2 \ge \cdots \ge \rho_{r+1}$ be the singular values of $\widehat{A}$. By Lemma 6, we must have $\sigma_{r+1} \ge \rho_{r+1}$. Since $\{\rho_k^{-1}\}_{k=1}^{r+1}$ are the singular values of $\widehat{A}^{-1}$, it follows that ρ_{r+1}^{-1} is the largest singular value of $\widehat{A}^{-1}$; hence, by Lemma 2,

$$\sigma_{r+1}^{-1} \le \rho_{r+1}^{-1} = \left\|\widehat{A}^{-1}\right\|_2 \le (r+1)\left\|\widehat{A}^{-1}\right\|_\infty = (r+1)\left|\gamma^{-1}\right|. \tag{60}$$

It follows that

$$|\gamma| \le (r+1)\sigma_{r+1}. \tag{61}$$

Since γ was an arbitrary nonzero element of E, we conclude that

$$\|E\|_\infty \le (r+1)\sigma_{r+1}. \tag{62}$$

□

Theorem 4. (Quasi-maximum volume pseudo-skeleton) *Let $A \in \mathbf{R}^{m\times n}$ be a matrix with singular values $\{\sigma_i\}$ in nonascending order. If $A_\blacksquare = A(I,J)$ is a rank-r maximum volume submatrix of A, and $B = A(I', J')$ is a rank-r submatrix of A that has quasi-maximal volume in A in the sense that there exists $\nu > 0$ such that*

$$\mathcal{V}(B) \ge \nu\mathcal{V}(A), \tag{63}$$

then

$$\|A - \mathscr{S}_{B^{-1}}\|_\infty \le \nu^{-1}(r+1)\sigma_{r+1}. \tag{64}$$

Proof. We proceed in a manner nearly the same as in the proof of Theorem 3. If we define $E = A - \mathscr{S}_{B^{-1}}$, then E is zero at row indices I' and column indices J' by Theorem 1. If we take an arbitrary nonzero entry $\gamma = E_{ij}$, then i and j do not occur in I' or J'.

Define

$$\widehat{B} = \begin{bmatrix} B & B(I,j) \\ B(i,J) & B(i,j) \end{bmatrix}. \tag{65}$$

By reasoning analogous to that used in the proof of Theorem 3, we can show that for any $k, \ell \in 1 : (r+1)$,

$$\left|\widehat{B}^{-1}(k,\ell)\right| = |\det(M')| \cdot \left|\det\left(\widehat{B}^{-1}\right)\right| \tag{66}$$

for some $r \times r$ submatrix M' of A. Applying the quasi-maximal volume property of B, we get

$$\left|\widehat{B}^{-1}(k,\ell)\right| \leq |\det(A_{\blacksquare})| \cdot \left|\det\left(\widehat{B}^{-1}\right)\right| \leq \nu^{-1} |\det(B)| \cdot \left|\det\left(\widehat{B}^{-1}\right)\right|. \tag{67}$$

Applying Lemma 3 in the same way we did in Theorem 3, we get

$$\left|\gamma^{-1}\right| = |\det(B)| \cdot \left|\det\left(\widehat{B}^{-1}\right)\right|. \tag{68}$$

Therefore,

$$\left|\widehat{B}^{-1}(k,\ell)\right| \leq \nu^{-1}\left|\gamma^{-1}\right|. \tag{69}$$

This implies that $\left\|\widehat{B}^{-1}\right\|_\infty \leq \nu^{-1}\left|\gamma^{-1}\right|$, as k and ℓ were arbitrary. On the other hand, we can also obtain the estimate

$$\sigma_{r+1}^{-1} \leq (r+1)\left\|\widehat{B}^{-1}\right\|_\infty \tag{70}$$

by the same reasoning that was used in Theorem 3. Thus,

$$|\gamma| \leq \nu^{-1}(r+1)\sigma_{r+1}. \tag{71}$$

Since γ was an arbitrary nonzero element of E, it follows that $\|E\|_\infty \leq \nu^{-1}(r+1)\sigma_{r+1}$. $\square$

Definition 9. (Dominant submatrix of a tall matrix) *Let $A \in \mathbf{R}^{m\times r}$ have rank r (which means that $m \geq r$). A nonsingular, square submatrix $A_\square = A(I,:) \in \mathbf{R}^{r\times r}$ of A is a **dominant submatrix** of A if*

$$\left\|AA_\square^{-1}\right\|_\infty \leq 1. \tag{72}$$

We will typically denote dominant submatrices of A by $A_\square$.

Proof. Observe that

$$[(AB)(I,:)]_{ij} = (AB)_{I_i j} = \sum_{k=1}^{r} A_{I_i k}B_{kj} = \sum_{k=1}^{r} A(I,:)_{ik}B_{kj} = [A(I,:)B]_{ij}, \quad i \in 1:m,\ j \in 1:n, \tag{73}$$

so $(AB)(I,:) = A(I,:)B$.

Similarly,

$$[(AB)(:,J)]_{ij} = (AB)_{iJ_j} = \sum_{k=1}^{r} A_{ik}B_{kJ_j} = \sum_{k=1}^{r} A_{ik}B(:,J)_{kj} = [AB(:,J)]_{ij}, \quad i \in 1:m,\ j \in 1:n, \tag{74}$$

so $(AB)(:,J) = AB(:,J)$. $\square$

Lemma 7. *Let $A \in \mathbf{R}^{m\times r}$, and let $B \in \mathbf{R}^{r\times r}$ be nonsingular. If $A(I,:), A(I',:) \in \mathbf{R}^{r\times r}$ are square submatrices of A, and $A(I',:)$ is nonsingular, then $(AB)(I',:)$ is nonsingular, and*

$$\frac{\mathcal{V}(A(I,:))}{\mathcal{V}(A(I',:))} = \frac{\mathcal{V}((AB)(I,:))}{\mathcal{V}((AB)(I',:))}. \tag{75}$$

Proof. By Lemma 1, $(AB)(I,:) = A(I,:)B$, and $(AB)(I',:) = A(I',:)B$. Thus,

$$\det((AB)(I',:)) = \det(A(I',:)B) = \det(A(I',:))\det(B) \neq 0. \tag{76}$$

Similarly, $\det((AB)(I,:)) = \det(A(I,:))\det(B)$. Therefore,

$$\frac{\det((AB)(I,:))}{\det((AB)(I',:))} = \frac{\det(A(I,:))\det(B)}{\det(A(I',:))\det(B)} = \frac{\det(A(I,:))}{\det(A(I',:))}. \tag{77}$$

Taking the absolute value on both sides gives (75). □

Lemma 8. (Hadamard's Inequality) *Let $A \in \mathbf{R}^{m\times m}$. Then*

$$\mathcal{V}(A) \leq \prod_{j=1}^{m} \|A(:,j)\|_2\,, \tag{78}$$

where $\|\cdot\|_2$ is the Euclidean vector norm.

Proof. See Example 6.1.4 in Meyer's linear algebra textbook [2]. □

Theorem 5. (Approximation by dominant submatrix) *Let $A \in \mathbf{R}^{m\times r}$ have rank r, and let $A_\blacksquare$ be a maximum volume submatrix of A. Then*

$$\mathcal{V}(A_\square) \geq r^{-\frac{r}{2}}\mathcal{V}(A_\blacksquare) \tag{79}$$

for all dominant submatrices $A_\square$ of A. The inequality is sharp.

Proof. Let $A_\square$ be a dominant submatrix of A, and let $B = AA_\square^{-1}$. By definition, $\|B\|_\infty \leq 1$. Thus, if we take r rows of B at indices I, then $\|B(I,:)\|_\infty \leq 1$ as well, which implies that $\|B(I,j)\|_2 \leq \sqrt{r}$. By Hadamard's inequality (Lemma 8), then,

$$\mathcal{V}(B(I,:)) \leq \prod_{j=1}^{r} \|B(I,j)\|_2 \leq r^{\frac{r}{2}}, \tag{80}$$

with equality holding if $\{B(I,j)\}_{j=1}^{r}$ forms an orthogonal set.

In particular, choose I such that $A_\blacksquare = A(I,:)$. By Lemma 1, we have

$$r^{\frac{r}{2}} \geq \mathcal{V}(B(I,:)) = |\det(B(I,:))| = |\det(A(I,:))\det(A_\square^{-1})| = \frac{\mathcal{V}(A_\blacksquare)}{\mathcal{V}(A_\square)}. \tag{81}$$

Then (79) follows.

If we choose $A = (1,1)^T$, then the maximum volume submatrix of A is $A_\blacksquare = [1]$, with volume 1. If we set $A_\square = A_\blacksquare$ and note that $r = 1$ for this choice of A, we see that $\mathcal{V}(A_\square) = 1 = r^{-\frac{r}{2}}\mathcal{V}(A_\blacksquare)$. □

Theorem 6. (Maximum volume submatrices are dominant) *Let $A \in \mathbf{R}^{m\times r}$ have rank r, and let $A_{\blacksquare} \in \mathbf{R}^{r\times r}$ be a maximum volume submatrix of A. Then $A_{\blacksquare}$ is a dominant submatrix of A.*

Proof. Since the rank of A is r, there must be a set of r linearly independent rows of A, say at indices I'. Then $A(I',:)$ is nonsingular, and $\mathcal{V}(A(I',:)) > 0$. This implies that $\mathcal{V}(A_{\blacksquare}) \geq \mathcal{V}(A(I',:)) > 0$.

Since $\mathcal{V}(A_{\blacksquare}) > 0$, it follows that $A_{\blacksquare}$ is invertible. Define $B = AA_{\blacksquare}^{-1}$. There is some row index sequence I such that $A_{\blacksquare} = A(I,:)$. By Lemma 7, $A_{\blacksquare}$ has maximal volume in A if and only if $B(I,:)$ has maximal volume in B, as multiplication by the invertible matrix $A_{\blacksquare}^{-1}$ preserves the ratios of $r\times r$ submatrix volumes.

Furthermore, $B(I,:)$ is the identity matrix $I_{r\times r}$ because, by Lemma 1,

$$B(I,:) = \left(AA_{\blacksquare}^{-1}\right)(I,:) = A(I,:)A_{\blacksquare}^{-1} = A_{\blacksquare}A_{\blacksquare}^{-1} = I_{r\times r}. \tag{82}$$

Thus, $B(I,:)$ is dominant in B if and only if $\left\|BB(I,:)^{-1}\right\|_{\infty} = \|B\|_{\infty} = \left\|AA_{\blacksquare}^{-1}\right\|_{\infty} \leq 1$, that is, if and only if $A_{\blacksquare}$ is dominant in A.

We now prove the claim by contradiction. Suppose that $A_{\blacksquare}$ is not dominant in A. Then $B(I,:)$ is not dominant in B; that is, there exists $k \in 1:m$ and $j \in 1:r$ such that $|B_{kj}| > 1$.

Let $I'_i = I_i$ if $i \neq j$, and let $I'_j = k$. Then every row of $B(I',:)$ is a row of $I_{r\times r}$ except for the jth row, which is the kth row of B. That is,

$$B(I',:) = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ & & B(k,:)\ (j\text{th row}) & & \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix}. \tag{83}$$

Expanding by cofactors and expanding on the jth row of $B(I',:)$ last shows that

$$|\det(B(I',:))| = |B_{kj}| > 1. \tag{84}$$

This means that $\mathcal{V}(B(I',:)) > 1 = \mathcal{V}(B(I,:))$, so $B(I,:)$ is not maximal in B. Then $A_{\blacksquare}$ is not maximal in A, which is a contradiction.

Hence, $A_{\blacksquare}$ is dominant in A. $\square$

3 The `maxvol` Algorithm

Definition 10. (δ-dominant submatrix) *Let $A \in \mathbf{R}^{m\times r}$ have rank r (which means that $m \geq r$), and let $\delta > 0$. A nonsingular, square submatrix $A_{\square} = A(I,:) \in \mathbf{R}^{r\times r}$ of A is a δ-**dominant submatrix** of A if*

$$\left\|AA_{\square}^{-1}\right\|_{\infty} \leq 1 + \delta. \tag{85}$$

Theorem 7. (Correctness of `maxvol`) *Let $A_{\boxdot}^{(k)}$ be the matrix $A_{\boxdot}$ in the `maxvol` algorithm after k steps of the loop. Then*

Algorithm 1: `maxvol`

Input: A matrix $A \in \mathbf{R}^{n\times r}$ of rank r
Input: Tolerance $\delta \geq 0$
Output: A matrix $A_{\square} \in \mathbf{R}^{r\times r}$ that is δ-dominant in A
1 Initialize a nonsingular submatrix $A_{\boxdot}$ of A;
2 **repeat**
3 $\quad B \leftarrow AA_{\boxdot}^{-1}$;
4 $\quad i, j \leftarrow \underset{i',j'}{\operatorname{argmax}} |B_{i'j'}|$;
5 $\quad$ **if** $|B_{ij}| > 1+\delta$ **then**
6 $\quad\quad A_{\boxdot}(j,:) \leftarrow A(i,:)$;
7 $\quad$ **end**
8 **until** $|B_{ij}| \leq 1+\delta$;
9 $A_{\square} \leftarrow A_{\boxdot}$;

1. *$A_{\boxdot}^{(k)}$ is invertible,*
2. *the sequence of volumes $\left\{\mathcal{V}\left(A_{\boxdot}^{(k)}\right)\right\}$ is strictly increasing,*
3. *the* `maxvol` *algorithm terminates in a finite number of steps c,*
4. *the output $A_{\square}$ is δ-dominant in A,*
5. *if $\delta > 0$, then the number of steps c before the algorithm terminates is bounded by*

$$c \leq \frac{\log(\mathcal{V}(A_{\square})) - \log\left(\mathcal{V}\left(A_{\boxdot}^{(0)}\right)\right)}{\log(1+\delta)} \leq \frac{\log(\mathcal{V}(A_{\blacksquare})) - \log\left(\mathcal{V}\left(A_{\boxdot}^{(0)}\right)\right)}{\log(1+\delta)}. \tag{86}$$

Proof. The first matrix $A_{\boxdot}^{(0)}$ is invertible by construction (the initialization of $A_{\boxdot}^{(0)}$ as an invertible submatrix can always be done because the rank of A is r). Suppose for induction that $A_{\boxdot}^{(k)}$ is invertible for some $k \geq 0$. If $k = c$, then we are done. Otherwise, if $(i,j) = \underset{i',j'}{\operatorname{argmax}} |B_{i'j'}|$, where $B = A\left(A_{\boxdot}^{(k)}\right)^{-1}$, then $|B_{ij}| \geq 1+\delta$.

Let $I^{(k)}$ be the row indices in A of the submatrix $A_{\boxdot}^{(k)}$, and let $I^{(k+1)}$ be the row indices in A of $A_{\boxdot}^{(k+1)}$. Then, by line 6 of Algorithm 1, $I_{\ell}^{(k+1)} = I_{\ell}^{(k)}$ if $\ell \neq j$, and $I_j^{(k+1)} = i$. By Lemma 1,

$$B\left(I^{(k)},:\right) = A\left(I^{(k)},:\right)\left(A_{\boxdot}^{(k)}\right)^{-1} = A_{\boxdot}^{(k)}\left(A_{\boxdot}^{(k)}\right)^{-1} = I_{r\times r}, \tag{87}$$

and

$$B\left(I^{(k+1)},:\right) = A\left(I^{(k+1)},:\right)\left(A_{\boxdot}^{(k)}\right)^{-1} = A_{\boxdot}^{(k+1)}\left(A_{\boxdot}^{(k)}\right)^{-1}. \tag{88}$$

On the other hand, since $I^{(k+1)}$ differs from $I^{(k)}$ only in the jth entry, we must have

$$B\left(I^{(k+1)},:\right) = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ & & B(i,:)\ (j\text{th row}) & & \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix}. \tag{89}$$

Expanding by cofactors and expanding on the jth row last, we obtain

$$\frac{\mathcal{V}\left(A_{\boxdot}^{(k+1)}\right)}{\mathcal{V}\left(A_{\boxdot}^{(k)}\right)} = \left|\det\left(A_{\boxdot}^{(k+1)}\left(A_{\boxdot}^{(k)}\right)^{-1}\right)\right| = \left|\det\left(B\left(I^{(k+1)},:\right)\right)\right| = |B_{ij}| > 1+\delta. \tag{90}$$

This implies that

$$\mathcal{V}\left(A_{\boxdot}^{(k+1)}\right) > (1+\delta)\mathcal{V}\left(A_{\boxdot}^{(k)}\right) > 0, \tag{91}$$

which shows that $A_{\boxdot}^{(k+1)}$ is invertible. By induction, $A_{\boxdot}^{(k)}$ is invertible for all k. Moreover, (91) also shows that $\left\{\mathcal{V}\left(A_{\boxdot}^{(k)}\right)\right\}$ is strictly increasing.

Since the volume of $A_{\boxdot}^{(k)}$ increases with k, each $A_{\boxdot}^{(k)}$ is distinct. There are only finitely many submatrices of A, and at least one is δ-dominant (one of the maximum volume submatrices certainly is). Since each $A_{\boxdot}^{(k)}$ is distinct, $A_{\boxdot}^{(k)}$ must eventually be δ-dominant for some k. The stopping criterion on line 8 of Algorithm 1 is satisfied if and only if $A_{\boxdot}^{(k)}$ is δ-dominant, so the algorithm terminates in finitely many steps c, and the output is δ-dominant.

Iterating the inequality in (91), we obtain

$$\mathcal{V}\left(A_{\boxdot}^{(c)}\right) \geq (1+\delta)^c\mathcal{V}\left(A_{\boxdot}^{(0)}\right), \tag{92}$$

which implies the first inequality in (86) because $A_{\square} = A_{\boxdot}^{(c)}$. The second inequality is trivially true by the maximality of $\mathcal{V}\left(A_{\blacksquare}\right)$. $\square$

Lemma 9. (Sherman-Morrison formula) *Suppose that $A \in \mathbf{R}^{n\times n}$ is invertible, and let $u, v \in \mathbf{R}^{n\times 1}$ be nonzero column vectors. Then $A + uv^T$ is invertible if and only if $1 + vA^{-1}u^T \neq 0$, and*

$$\left(A + uv^T\right)^{-1} = A^{-1} - \frac{\left(A^{-1}u\right)\left(v^TA^{-1}\right)}{1 + v^TA^{-1}u}. \tag{93}$$

Proof. See Section 2 in an old paper by Bartlett [1]. $\square$

Theorem 8. (Complexity of `maxvol`) *Let $B^{(k)}$ be the matrix B and let $A_{\boxdot}^{(k)}$ be the matrix $A_{\boxdot}$ in the `maxvol` algorithm after k steps. Then $B^{(k+1)}$ and $A_{\boxdot}^{(k+1)}$ can be computed in $\mathcal{O}(nr)$ operations from $B^{(k)}$ and $A_{\boxdot}^{(k)}$. Therefore, the overall cost of the iterative portion of `maxvol` is $\mathcal{O}(cnr)$, where c is the number of iteration steps.*

Proof. We can write $A_{\boxdot}^{(k+1)}$ in terms of $A_{\boxdot}^{(k)}$ as

$$A_{\boxdot}^{(k+1)} = A_{\boxdot}^{(k)} + e_j \left(A(i,:) - A_{\boxdot}^{(k)}(j,:)\right), \tag{94}$$

where e_j is the jth standard basis vector as a column vector. If we define $q = e_j$, and if we define $v = \left(A(i,:) - A_{\boxdot}^{(k)}(j,:)\right)^T$, then

$$A_{\boxdot}^{(k+1)} = A_{\boxdot}^{(k)} + qv^T. \tag{95}$$

By the Sherman-Morrison formula (Lemma 9),

$$\left(A_{\boxdot}^{(k+1)}\right)^{-1} = \left(A_{\boxdot}^{(k)}\right)^{-1} - \frac{\left(A_{\boxdot}^{(k)}\right)^{-1} qv^T \left(A_{\boxdot}^{(k)}\right)^{-1}}{1 + v^T \left(A_{\boxdot}^{(k)}\right)^{-1} q}. \tag{96}$$

By Lemma 1,

$$\begin{aligned} v^T \left(A_{\boxdot}^{(k)}\right)^{-1} &= \left(A(i,:) - A_{\boxdot}^{(k)}(j,:)\right) \left(A_{\boxdot}^{(k)}\right)^{-1} && (97) \\ &= A(i,:) \left(A_{\boxdot}^{(k)}\right)^{-1} - e_j^T && (98) \\ &= B^{(k)}(i,:) - e_j^T. && (99) \end{aligned}$$

Therefore, $v^T \left(A_{\boxdot}^{(k)}\right)^{-1} q = B^{(k)}(i,:)e_j - e_j^T e_j = B_{ij}^{(k)} - 1$. Multiplying both sides of (96) by A gives

$$\begin{aligned} B^{(k+1)} &= B^{(k)} - \frac{1}{B_{ij}^{(k)}} B^{(k)} e_j v^T \left(A_{\boxdot}^{(k)}\right)^{-1} && (100) \\ &= B^{(k)} - \frac{1}{B_{ij}^{(k)}} B^{(k)}(:,j) \left(B^{(k)}(i,:) - e_j^T\right). && (101) \end{aligned}$$

The update rule for $B^{(k)}$ given in (101) involves a $n \times 1$ by $1 \times r$ matrix multiplication, scalar and $n \times r$ matrix multplication, and $n \times r$ matrix subtraction. Hence, it requires $\mathcal{O}(nr)$ operations to complete. The update rule for $A_{\boxdot}^{(k)}$ is $\mathcal{O}(1)$ because we only need to keep track of which rows of A are in $A_{\boxdot}^{(k)}$, and on each step only one row changes.

□

4 Implementation in `NumPy`

4.1 Practical update rules

Most of `maxvol` is trivial to implement in `NumPy`. The trickiest part is the efficient updating of B and $A_{\boxdot}$ (lines 3 and 6 in Algorithm 1). Let $B^{(k)}$ be the matrix B in `maxvol` after k steps, and let $I^{(k)}$ be the row indices in A of $A_{\boxdot}$ after k steps. Let $J^{(k)}$ be the other row indices of A that do not occur in the sequence $I^{(k)}$.

Updating $A_\boxdot$

To update $A_\boxdot$, we only need to update $I^{(k)}$. Let i, j be the indices obtained on line 4 of Algorithm 1. The update for $A_\boxdot$ is that the jth row of $A_\boxdot$ becomes the ith row of A, so

$$I_\ell^{(k+1)} = \begin{cases} I_\ell^{(k)} & \ell \neq j \\ i & \ell = j. \end{cases} \tag{102}$$

If we reuse the memory for $I^{(k)}$ for $I^{(k+1)}$, this means only doing one update operation.

Using Z instead of B

We know from our analysis that on each step $B^{(k)}\left(I^{(k)}, :\right) = I_{r\times r}$ (recall (87)). Therefore, it would be more efficient to store only the rows of $B^{(k)}$ at indices $J^{(k)}$. Let $Z^{(k)} = B^{(k)}\left(J^{(k)}, :\right)$ denote the matrix of these rows.

Let $Z_{i'j'}^{(k)}$ be the maximum modulus element of $Z^{(k)}$. If we used $Z_{i'j'}^{(k)}$ in place of $B_{ij}^{(k)}$ in Algorithm 1, then the result of the algorithm would be unchanged. Indeed, on the kth loop iteration, there are two possibilities.

1. $Z_{i'j'}^{(k)} = B_{ij}^{(k)}$, in which case we may take $j = j'$ and $i = J_{i'}^{(k)}$. The rest of the loop iteration proceeds as it would using $B_{ij}^{(k)}$.
2. $Z_{i'j'}^{(k)}$ is not the maximum modulus element in $B^{(k)}$. In this case, $\left|Z_{i'j'}^{(k)}\right| \leq \left|B_{ij}^{(k)}\right| = 1 \leq 1+\delta$, so whether we use $Z_{i'j'}^{(k)}$ or $B_{ij}^{(k)}$, the loop should exit immediately.

In any case, then, using $Z_{i'j'}^{(k)}$ in place of $B_{ij}^{(k)}$ has no effect on the result of the algorithm.

Updating Z

Now updating $B^{(k)}$ amounts to updating $Z^{(k)}$. Recall the efficient, rank-1 update rule for B:

$$B^{(k+1)} = B^{(k)} - \frac{1}{B_{ij}^{(k)}} B^{(k)}(:, j)\left(B^{(k)}(i, :) - e_j^T\right), \tag{103}$$

where i, j are the indices obtained on line 4 of Algorithm 1. Taking the submatrix with row indices $J^{(k+1)}$ on both sides of (103), applying Lemma 1 and using $Z_{i'j'}^{(k)}$ in place of $B_{ij}^{(k)}$ as discussed above, we get

$$Z^{(k+1)} = B^{(k)}\left(J^{(k+1)}, :\right) - \frac{1}{Z_{i'j'}^{(k)}} B^{(k)}\left(J^{(k+1)}, j\right)\left(B^{(k)}(i, :) - e_j^T\right). \tag{104}$$

Evidently, we will also need to keep track of $J^{(k)}$ for all k in order to find $Z^{(k+1)}$. To update $J^{(k)}$, we need to ensure that $J^{(k+1)}$ contains all indices not in $I^{(k+1)}$. Only one index in $I^{(k+1)}$ is different from $I^{(k)}$; namely, $I_j^{(k+1)} = i$ instead of $I_j^{(k)}$. Thus, we need to remove i from $J^{(k+1)}$ and replace it with $I_j^{(k)}$. Let i' be the index such that $J_{i'}^{(k)} = i$. Then we can obtain $J^{(k+1)}$ by the rule

$$J_\ell^{(k+1)} = \begin{cases} J_\ell^{(k)} & \ell \neq i', \\ I_j^{(k)} & \ell = i'. \end{cases} \tag{105}$$

Like the update rule for $I^{(k)}$, this also only requires one operation if we reuse the memory for $J^{(k)}$ for $J^{(k+1)}$.

With this rule in place, we can relate $Z^{(k+1)}$ to $Z^{(k)}$ using (104). Let $L = \{1, 2, \ldots, i'-1, i'+1, \ldots, n-r\}$. Then $J^{(k+1)}_{L_\ell} = J^{(k)}_{L_\ell}$ for all ℓ. Therefore,

$$\left(B^{(k)}\left(J^{(k+1)}, :\right)\right)(L, :) = \left(B^{(k)}\left(J^{(k)}, :\right)\right)(L, :) = Z^{(k)}(L, :). \tag{106}$$

Taking the submatrix with row indices L on both sides of (104), we obtain

$$Z^{(k+1)}(L, :) = Z^{(k)}(L, :) - \frac{1}{Z^{(k)}_{i'j'}} Z^{(k)}(L, j)\left(B^{(k)}(i, :) - e_j^T\right). \tag{107}$$

Recalling that $j = j'$ and $i = J^{(k)}_{i'}$, we get

$$Z^{(k+1)}(L, :) = Z^{(k)}(L, :) - \frac{1}{Z^{(k)}_{i'j}} Z^{(k)}(L, j)\left(Z^{(k)}(i', :) - e_j^T\right). \tag{108}$$

Similarly, if we take the submatrix with row indices $\{i'\}$ on both sides of (104), then, by the definition of $J^{(k+1)}$, we get

$$Z^{(k+1)}(i', :) = B^{(k)}\left(I^{(k)}_j, :\right) - \frac{1}{Z^{(k)}_{i'j}} B^{(k)}\left(I^{(k)}_j, j\right)\left(B^{(k)}(i, :) - e_j^T\right) \tag{109}$$

$$= e_j^T - \frac{1}{Z^{(k)}_{i'j}}\left(Z^{(k)}(i', :) - e_j^T\right) \tag{110}$$

because $B^{(k)}\left(I^{(k)}_j, :\right) = \left(B^{(k)}\left(I^{(k)}, :\right)\right)(j, :) = I_{r\times r}(j, :) = e_j^T$.

Let D be a matrix with $D(L, :) = Z^{(k)}(L, :)$ and $D(i', :) = e_j^T$. Then, using D, we can incorporate the update rule (110) into (108):

$$Z^{(k+1)} = D - \frac{1}{Z^{(k)}_{i'j}} D(:, j)\left(Z^{(k)}(i', :) - e_j^T\right). \tag{111}$$

Thus, we can compute $Z^{(k+1)}$ from $Z^{(k)}$ using this update rule.

4.2 Step-by-step design in `NumPy`

Function signature

We begin with the signature of the `maxvol` function. We need the matrix A, of course, which we will store in a `NumPy` array called `a`. Next, we need the parameter δ, which we will store in the variable `delta`, and an iteration limit, which store in the variable `max_iter`. We also allow for an optional initial submatrix, specified by a list or array of row indices, which we name `initial_rows`. The return value should be the δ-dominant matrix $A_\square$, which we return in terms of its row indices in the given matrix A. Thus, we arrive at the signature in Listing 1.

Listing 1: function signature

```
def maxvol(
    a: NDArray[np.float],  # shape = (n, r)
    initial_rows: Optional[NDArray[np.int]] = None,  # shape = (r,)
    delta: float = 1e-2,
    max_iter: int = 100
) -> Optional[NDArray[np.int]]:  # shape = (r,)
```

If we are given a square matrix A, then the rest of the algorithm will generate indexing errors; in any case, the maximum volume submatrix of a square tall matrix is the matrix itself, so we can return early if a square matrix is given. If `n` and `r` are the numbers of rows and columns of A, then this check is given by Listing 2.

Listing 2: square matrix check

```
if n == r:
    return np.arange(r)
```

We note that `np.arange(r)` generates an array whose entries are $1, 2, \ldots, r$. This is precisely the sequence of row indices of the entire square matrix, as desired.

Initialization of $A_{\boxdot}^{(0)}$

Next, we move on to the issue of initializing the nonsingular starting submatrix $A_{\boxdot}^{(0)}$. If `initial_rows` is supplied, then we will assume that the user has ensured that `initial_rows` determines a nonsingular submatrix. If `initial_rows` is not supplied, then it is up to us to determine a nonsingular submatrix. This can be done easily by applying Gaussian elimination with partial pivoting (that is, with row pivotoing). We note that Gaussian elimination on the $n \times r$ matrix A requires r elimination steps, each of which requires $\mathcal{O}(nr)$ computations, giving the entire process a computational complexity of $\mathcal{O}(nr^2)$, which is acceptable (we are mainly concerned with linear complexity in n).

If `initial_rows` is given, then we need to compute the indices of the rows of A not in the initial submatrix as well, as we need them to work with $Z^{(k)}$. We can do Gaussian elimination using the `scipy.linalg.lu` function, which will return the permutation of the rows obtained by partial pivoting. This is given as an array of row indices; the first r elements of the permutation determine a nonsingular submatrix, and the remaining elements give us the rows of A not in the submatrix. We store the current submatrix row indices in the variable `submat_rows`, and the current remaining row indices in `other_rows`. Thus, the initialization is given in Listing 3.

Listing 3: $A_{\boxdot}^{(0)}$ initialization

```
if initial_rows is None:
    # p_indices=True to get row index array instead of permutation matrix.
    # Return of lu is a tuple (p, l, u). We only need the p entry,
    # which is the array of row indices of the permutation.
    p = scipy.linalg.lu(a, p_indices=True)[0]

    submat_rows = p[:r]  # get first r elements
    other_rows = p[r:]  # get remaining elements
else:
```

```
    submat_rows = initial_rows

    # find other rows by building a set of all indices and set-subtracting
    # given initial submatrix row indices. Then convert to array of indices.
    other_rows_set = set(range(n)).difference(map(int, submat_rows))
    other_rows = np.array(tuple(other_rows_set))
```

Initialization of $Z^{(0)}$

We have an efficient update rule for $Z^{(k)}$, but we still need to initialize $Z^{(0)}$. The only way to do this is by using the definition, that is (by Lemma 1),

$$Z^{(0)} = B^{(0)}\left(J^{(0)}, :\right) = A\left(J^{(0)}, :\right)\left(A_{\boxdot}^{(0)}\right)^{-1}. \tag{112}$$

The most stable and efficient way to do this is by using a linear system solver (rather than computing the inverse of $A_{\boxdot}^{(0)}$ explicitly). This can be done fairly easily with `np.linalg.solve`. The main wrinkle is that we are multiplying by an inverse matrix on the *right*, and this command computes the product with an inverse matrix on the *left*. We can deal with this by transposing the inputs and transposing the output of `np.linalg.solve`.

Noting that $A\left(J^{(0)}, :\right)$ can be obtained by taking the rows of `a` stored in `other_rows`, and $A_{\boxdot}^{(0)}$ can be obtained by taking the rows of `a` stored in `submat_rows`, the initialization of $Z^{(0)}$, which we store in the variable `z`, is given in Listing 4.

Listing 4: $Z^{(0)}$ initialization

```
z = np.linalg.solve(a[submat_rows].T, a[other_rows].T).T
```

We remark that `np.linalg.solve` uses Gaussian elimination on $A_{\boxdot}^{(0)} \in \mathbf{R}^{r\times r}$ to solve n equations, so this step has a computational complexity of $\mathcal{O}(nr^2 + r^3) \subseteq \mathcal{O}(nr^2)$, which is acceptable.

Loop setup

Python doesn't have a `do-while`/`repeat-until` loop construct; since we want to terminate after `max_iter` iterations in any case, we can use a `for` loop and `if-break` to simulate the repeat-until in Algorithm 1. Furthermore, the `if` statement in Algorithm 1 is the same as the loop stopping condition, so we can use the `if-break` simultaneously to exit the loop and to do the `if` statement on line 5. Thus, the beginning of our loop computes the maximum modulus element of $Z^{(k)}$ (that is, the Chebyshev norm) and exits if its modulus is less than $1+\delta$. If we need to exit the loop, then we also need to return `submat_rows` immediately, so we can do the exit and return all at once. See Listing 5.

Listing 5: loop setup

```
# use dummy index _, as we don't need the iteration index
for _ in range(max_iter):

    # np.argmax returns the index in the flattened array, so unravel
    # to get the 2-dimensional index.
    i_rel, j = np.unravel_index(np.argmax(np.abs(z)), z.shape)
    max_mod_el = z[i_rel, j]
```

```

    if np.abs(max_mod_el) < 1 + delta:
        return submat_rows
```

Updating Z

For the Z update, we recall the update rule (111). Since most of the content of D is the same as $Z^{(k)}$, we can store the i' row of $Z^{(k)}$, then replace the j row of $Z^{(k)}$ with e_j^T, so that $Z^{(k)}$ becomes D. This requires $2r$ and r units of extra memory instead of $n - r$ copy operations and $(n - r)r$ units of extra memory. Thus, the update of `z` is given in Listing 6.

Listing 6: Z update

```
# Save i' row of z and subtract e_j^T.
right = z[i_rel].copy()
right[j] -= 1.

# Store e_j^T in the i' row of z.
z[i_rel, :] = 0.
z[i_rel, j] = 1.

# In-place update of z. Divide by max_mod_el before matrix multiply.
z -= z[:, j : j+1] @ (right[None] / max_mod_el)
```

Updating row indices

The last thing to do is to update the row index sequence of the current submatrix and the row indices of the current Z matrix. Following the update rules for $I^{(k)}$ and $J^{(k)}$, we see that this amounts to swapping the values `submat_rows[j]` and `other_rows[i_rel]`, as in Listing 7.

Listing 7: row index update

```
temp = submat_rows[j]
submat_rows[j] = other_rows[i_rel]
other_rows[i_rel] = temp
```

Return value

If the loop does not exit as a result of having found a δ-dominant submatrix, that is, if the loop completes `max_iter` iterations, then we want to return `None` to indicate the failure to converge. By default, if no `return` statement is encountered in a Python function, then the function returns `None`, so we simply leave the rest of the function after the loop blank.

4.3 Complete code

Bringing all the snippets above together, we obtain the full code for the `maxvol` algorithm (Listing 8).

Listing 8: `NumPy` implementation of `maxvol`

```
def maxvol(
        a: NDArray[np.float],
```

```
        initial_rows: Optional[NDArray[np.int]] = None,
        delta: float = 1e-2,
        max_iter: int = 100
) -> Optional[NDArray[np.int]]:
    """
    :param a: An n x r matrix of rank r.
    :param initial_rows: A set of row indices in the matrix a giving us
    an initial nonsingular r x r submatrix. Uses Gaussian elimination to
    choose an initial set of rows if initial_rows is None.
    :param delta: delta-dominance delta value.
    :param max_iter: Maximum number of maxvol iterations.
    :return: Row indices of a delta-dominant submatrix of a. None if
    convergence fails to occur within max_iter iterations.
    """
    # input validation
    n, r = a.shape
    assert r <= n

    # In the edge case that a is square, the best we can do is return a
    # itself (that is, the submatrix rows are all the rows)
    if n == r:
        return np.arange(r)

    # Initialize a nonsingular submatrix.
    # We only track the rows of the submatrix, as we do
    # not need the submatrix itself in the algorithm, and we can always
    # retrieve the submatrix from the original matrix using the row indices.
    if initial_rows is None:
        # Use Gaussian elimination with partial pivoting to get rows
        # of a nonsingular submatrix. This operation is O(nr^2).
        p = scipy.linalg.lu(a, p_indices=True)[0]

        # nonsingular submatrix rows are packed into the first r indices
        submat_rows = p[:r]
        # and other rows are in the remaining indices
        other_rows = p[r:]
    else:
        # use given rows of a
        submat_rows = initial_rows
        # get other rows of a
        other_rows_set = set(range(n)).difference(map(int, submat_rows))
        other_rows = np.array(tuple(other_rows_set))

    # Get initial z = a[other_rows] @ (a[submat_rows])^{-1}.
    # Use np.linalg.solve to avoid computing matrix inverse.
    # Note that this operation is O(nr^2).
    z = np.linalg.solve(a[submat_rows].T, a[other_rows].T).T

    for _ in range(max_iter):
        # Get rows to swap by finding the maximum modulus element of z.
```

```
        i_rel, j = np.unravel_index(np.argmax(np.abs(z)), z.shape)
        max_mod_el = z[i_rel, j]

        # Stop if the current submatrix is delta-dominant.
        if np.abs(max_mod_el) < 1 + delta:
            return submat_rows

        # Update z.
        right = z[i_rel].copy()
        right[j] -= 1.

        z[i_rel, :] = 0.
        z[i_rel, j] = 1.

        z -= z[:, j : j+1] @ (right[None] / max_mod_el)

        # Update row index sequences.
        temp = submat_rows[j]
        submat_rows[j] = other_rows[i_rel]
        other_rows[i_rel] = temp

    # default return value is None
```

5 Experiments

6 Conclusion

References

[1] M. S. Bartlett. An Inverse Matrix Adjustment Arising in Discriminant Analysis. *The Annals of Mathematical Statistics*, 22(1):107–111, March 1951.

[2] Carl D. Meyer. *Matrix Analysis and Applied Linear Algebra*. Society for Industrial and Applied Mathematics, Philadelphia, 2008.

[3] Beresford N. Parlett. *The Symmetric Eigenvalue Problem*. Number 20 in Classics in Applied Mathematics. Society for Industrial and Applied Mathematics, Philadelphia, 1998.