MAX PLANCK GESELLSCHAFT
MAX PLANCK INSTITUTE
FOR DYNAMICS OF COMPLEX
TECHNICAL SYSTEMS
MAGDEBURG

Cross Approximation for Matrices and Tensors

CSC Reading Group

Bonan Sun and Eda Oktay

June 9, 2026





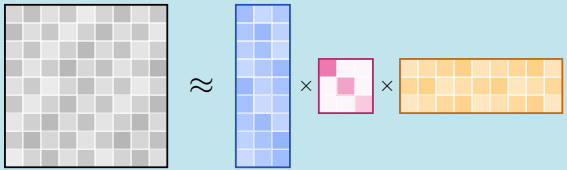
Organization

1. Motivation and notation
2. CUR/cross prototype algorithm
3. Good submatrix selection
4. Practical CUR/cross approximation algorithms
5. Connections to MOR with DEIM and tensor cross

What is and why cross approximation?

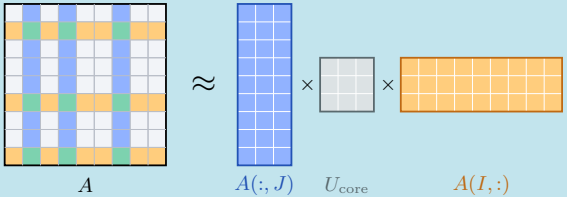
Singular value decomposition

$$A \approx A_r := U_r \Sigma_r V_r^\top$$



Cross/CUR approximation

$$A \approx C U_{\text{core}} R$$



	SVD	Cross/CUR
What it gives	Optimal dense low-rank approximation	Sample-based low-rank approximation
Advantages	Optimal error and robust standard algorithms	Interpretability, entry-query access, and tensor-friendly
Disadvantages	Loses original structure; can be expensive	Hard index selection and stability issues
When to use	Full matrix access; dense factors OK	Expensive entries; meaningful rows/columns/fibers

The cross/CUR approximation problem

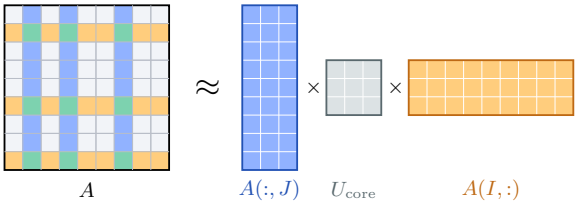
Problem formulation

Given a matrix $A \in \mathbb{R}^{m \times n}$, target rank r, solve $\min_{\substack{|I|=r,\ |J|=r \\ U_{\text{core}} \in \mathbb{R}^{r \times r}}} \|A - A(:, J) U_{\text{core}} A(I, :)\|_{\text{F}}$

Two main questions:

1. How to select the rows and columns (index sets I, J)?
2. How to choose the core U_{core}?

1. is very hard; 2. is easy once 1. is fixed.

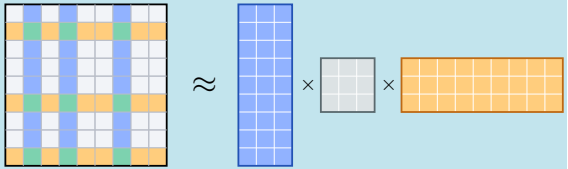


- For fixed I, J, the optimal core is $U_{\text{core}} = A(:, J)^{\dagger} A A(I, :)^{\dagger} \rightsquigarrow$ CUR approximation
- Another popular choice is $U_{\text{core}} = A(I, J)^{-1} \rightsquigarrow$ cross approximation

Two choices of the core: CUR and cross

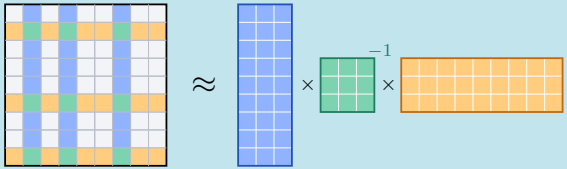
CUR (pseudoskeleton) approximation

$$A_{\mathrm{CUR}} := A(:, J)A(:, J)^{\dagger}AA(I, :)^{\dagger}A(I, :)$$



Cross (skeleton, interpolatory) approximation

$$A_{\mathrm{cross}} := A(:, J)A(I, J)^{-1}A(I, :)$$



	CUR	Cross
Core choice	Optimal for fixed I, J	Interpolatory: $A_{\mathrm{cross}}(:, J) = A(:, J)$, $A_{\mathrm{cross}}(I, :) = A(I, :)$
Core formation	Needs the whole matrix A	Only needs the intersection $A(I, J)$
Existence result: $\exists I, J$ s.t.	$\|A - A_{\mathrm{CUR}}\|_{\mathrm{F}} \leq \sqrt{2r+2}\ \|A - A_r\|_{\mathrm{F}}$	$\|A - A_{\mathrm{cross}}\|_{\mathrm{F}} \leq (r+1)\ \|A - A_r\|_{\mathrm{F}}$



Organization

1. Motivation and notation
2. CUR/cross prototype algorithm
3. Good submatrix selection
4. Practical CUR/cross approximation algorithms
5. Connections to MOR with DEIM and tensor cross

From CUR/cross to two related problems

CUR approximation

$$\min_{|I|=r,\ |J|=r} \left\| A - \underbrace{A(:,J)A(:,J)^\dagger}_{:=\Pi_J} A \underbrace{A(I,:)^\dagger A(I,:)}_{:=\Pi_I} \right\|_{\mathrm{F}}$$

- Need selected J s.t. $\mathrm{col}(A(:,J)) \approx \mathrm{col}(A)$
- This is the **column subset selection (CSSP)**:

$$\min_{|J|=r} \left\| A - \underbrace{A(:,J)A(:,J)^\dagger}_{:=\Pi_J} A \right\|_{\mathrm{F}}$$

Cross approximation

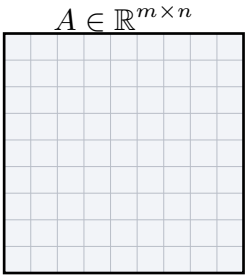
$$\min_{|I|=r,\ |J|=r} \left\| A - A(:,J)A(I,J)^{-1}A(I,:) \right\|_{\mathrm{F}}$$

- Not only need subspace approx., but also a stable intersection $A(I,J)$
- **Good submatrix selection**: for fixed J, choose I s.t.

$$\min_{|I|=r} \left\| A(I,J)^{-1} \right\|_2$$

CUR asks for good subspaces; Cross asks for good subspaces + a stable intersection.

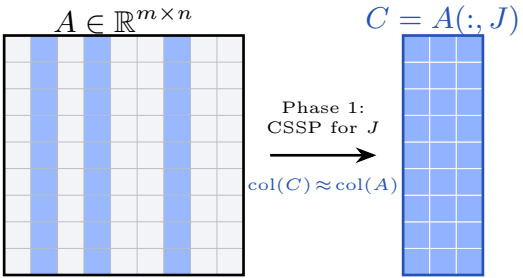
CUR/cross approximation algorithm



A typical prototype CUR/cross approximation algorithm for $A \in \mathbb{R}^{m \times n}$

1: Phase 1: choose columns J and get $C = A(:, J)$ (**CSSP step**)
2: Phase 2: choose rows I from $C = A(:, J)$ and get $R = A(I, :)$ (**Good submatrix step**)
3: Phase 3: choose a core U_{core}, e.g., $C^{\dagger}AR^{\dagger}$ for CUR or $A(I, J)^{-1}$ for cross
4: **return** I, J, U_{core}

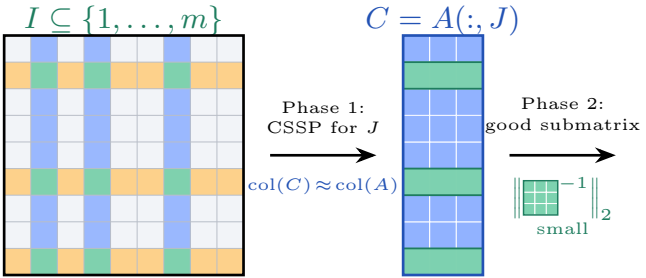
CUR/cross approximation algorithm



A typical prototype CUR/cross approximation algorithm for $A \in \mathbb{R}^{m \times n}$

1: Phase 1: choose columns J and get $C = A(:, J)$ (**CSSP step**)
2: Phase 2: choose rows I from $C = A(:, J)$ and get $R = A(I, :)$ (**Good submatrix step**)
3: Phase 3: choose a core U_{core}, e.g., $C^{\dagger}AR^{\dagger}$ for CUR or $A(I, J)^{-1}$ for cross
4: **return** I, J, U_{core}

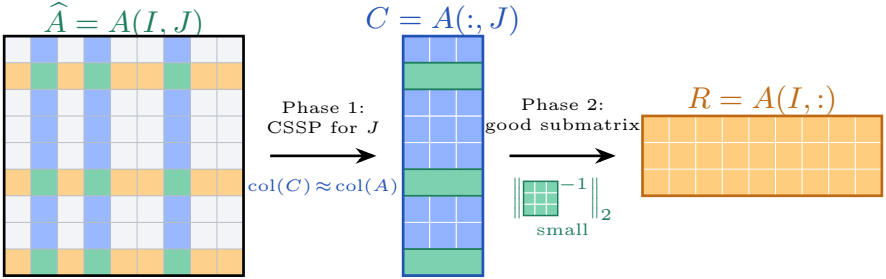
CUR/cross approximation algorithm



A typical prototype CUR/cross approximation algorithm for $A \in \mathbb{R}^{m \times n}$

1: Phase 1: choose columns J and get $C = A(:, J)$ (**CSSP step**)
2: Phase 2: choose rows I from $C = A(:, J)$ and get $R = A(I, :)$ (**Good submatrix step**)
3: Phase 3: choose a core U_{core}, e.g., $C^{\dagger} A R^{\dagger}$ for CUR or $A(I, J)^{-1}$ for cross
4: **return** I, J, U_{core}

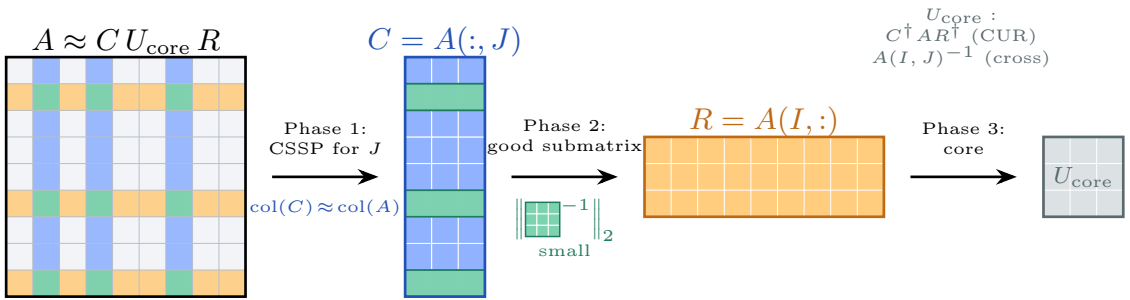
CUR/cross approximation algorithm



A typical prototype CUR/cross approximation algorithm for $A \in \mathbb{R}^{m \times n}$

1: Phase 1: choose columns J and get $C = A(:, J)$ (**CSSP step**)
2: Phase 2: choose rows I from $C = A(:, J)$ and get $R = A(I, :)$ (**Good submatrix step**)
3: Phase 3: choose a core U_{core}, e.g., $C^{\dagger}AR^{\dagger}$ for CUR or $A(I, J)^{-1}$ for cross
4: **return** I, J, U_{core}

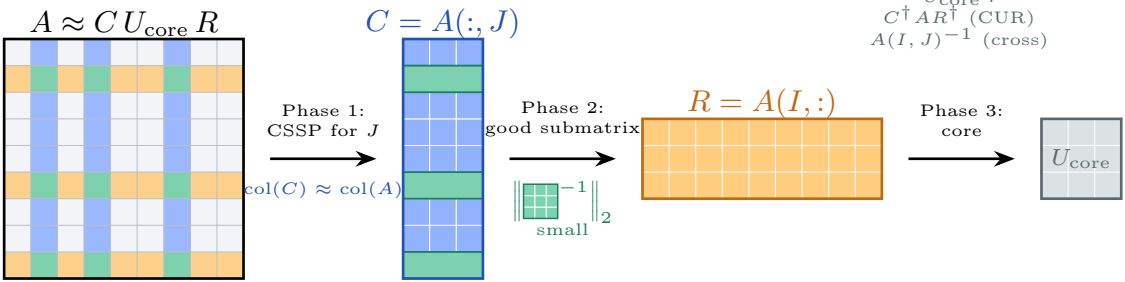
CUR/cross approximation algorithm



A typical prototype CUR/cross approximation algorithm for $A \in \mathbb{R}^{m \times n}$

1: Phase 1: choose columns J and get $C = A(:, J)$ (**CSSP step**)
2: Phase 2: choose rows I from $C = A(:, J)$ and get $R = A(I, :)$ (**Good submatrix step**)
3: Phase 3: choose a core U_{core}, e.g., $C^\dagger A R^\dagger$ for CUR or $A(I, J)^{-1}$ for cross
4: **return** I, J, U_{core}

Phase 1: CSSP through a row-space basis



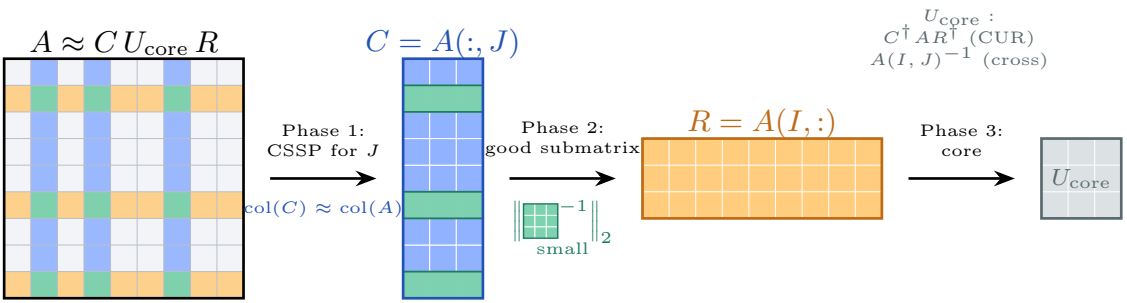
CSSP can also be done using good submatrix selection:



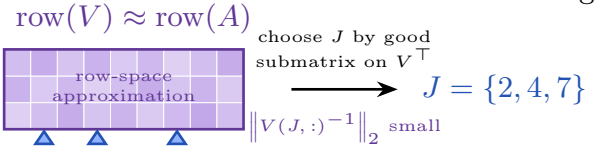
Using e.g.,

1. $[\sim, \sim, V^\top] = \texttt{truncated_svd}(A)$;
2. lazily, randomly select some rows $V = A(I', :)$.

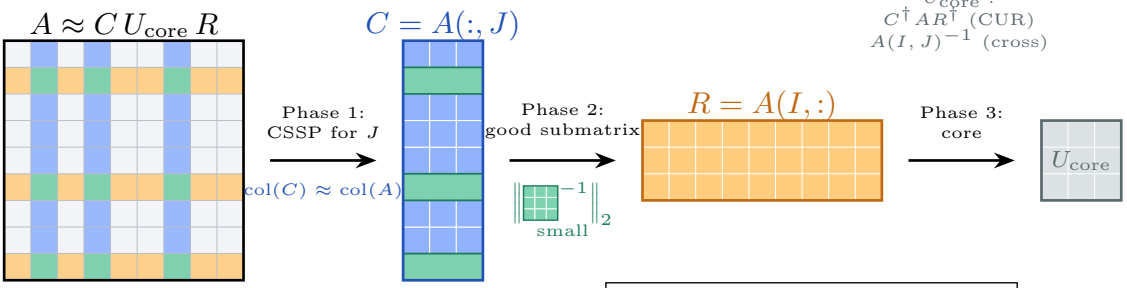
Phase 1: CSSP through a row-space basis



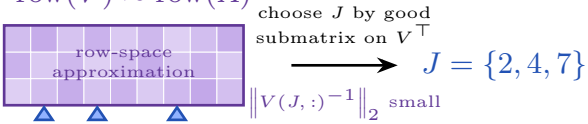
CSSP can also be done using good submatrix selection:



Phase 1: CSSP through a row-space basis



CSSP can also be done using **good submatrix selection**:



Our focus!

Organization

1. Motivation and notation
2. CUR/cross prototype algorithm
3. Good submatrix selection
4. Practical CUR/cross approximation algorithms
5. Connections to MOR with DEIM and tensor cross

How to select a good submatrix?

Good submatrix selection problem

Given $U \in \mathbb{R}^{n\times r},\ n > r$ with $U^\top U = \mathrm{Id}_r$, solve $\min\limits_{|I|=r} \left\|U(I,:)^{-1}\right\|_2$.

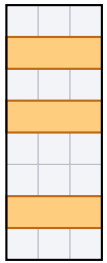
- Naive solution: compare all $\binom{n}{r}$ choices.
- In fact, this is an NP-hard problem and do not admit a PTAS.

A sufficient condition: maximum volume.

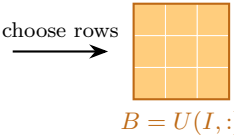
$$I \in \arg\max_{|I|=r} \mathrm{vol}(U(I,:)) \implies \left\|U(I,:)^{-1}\right\|_2 \le \sqrt{1 + r(n-r)}.$$

Here, $\mathrm{vol}(U(I,:)) := |\det U(I,:)|$.

- Maximum volume is still NP-hard.
- Practical algorithms go for greedily maximizing the volume.



$U \in \mathbb{R}^{n\times r}$



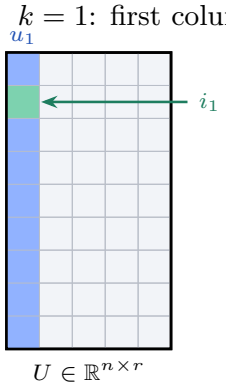
[Goreinov/Tyrtyshnikov/Zamarashkin'97], [Çivril/Magdon-Ismail'09], [Goreinov et el'10].

Forward greedy I: add one element at a time

Process column by column

- Step $k = 1$ (first column):
  Select a 1×1 submatrix s.t. the volume is maximized:

$$i_1 = \arg\max_i |\det U(i,1)| = \arg\max_i |u_{i1}| \rightsquigarrow I_1 = \{i_1\}.$$



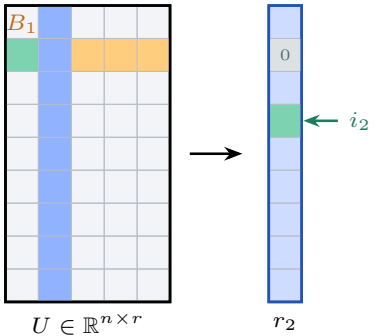
Forward greedy I: add one element at a time

Process column by column

- Step $k = 1$ (first column): $I_1 = \{\arg\max_i |u_{i1}|\}$.
- Step $k = 2$ (second column):
  Select a 2×2 submatrix containing the first one s.t. the volume is maximized:

$$
\begin{aligned}
i_2 &= \arg\max_i |\det U(I_1 \cup \{i\}, 1:2)| \\
&= \arg\max_i \left| \det \begin{pmatrix} u_{i_1,1} & u_{i_1,2} \\ u_{i,1} & u_{i,2} \end{pmatrix} \right| \\
&= \arg\max_i |u_{i_1,1}(u_{i,2} - u_{i,1}\frac{u_{i_1,2}}{u_{i_1,1}})| \\
&= \arg\max_i |u_{i,2} - u_{i,1}\frac{u_{i_1,2}}{u_{i_1,1}}| \\
&= \arg\max_i |r_2(i)|.
\end{aligned}
$$

$k = 2$: residualize column 2



$$r_2 = U(:,2) - U(:,1)B_1^{-1}U(I_1,2)$$

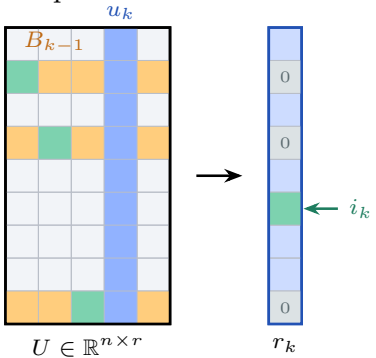
Forward greedy I: add one element at a time

Process column by column

- Step $k = 1$: $I_1 = \{\arg\max_i |u_{i1}|\}$.
- Step $k = 2$: $I_2 = I_1 \cup \{\arg\max_i |r_2(i)|\}$.
- Suppose we have selected I_{k-1} for the first $k-1$ columns and $B_{k-1} = U(I_{k-1}, 1:k-1)$.
- Step k (k-th column): Select a $k \times k$ submatrix containing the previous one s.t. the volume is maximized:

$$
\begin{aligned}
i_k &= \arg\max_i |\det U(I_{k-1} \cup \{i\}, 1:k)| \\
&= \arg\max_i \left| \det \begin{pmatrix} B_{k-1} & U(I_{k-1}, k) \\ U(i, 1:k-1) & u_{i,k} \end{pmatrix} \right| \\
&= \arg\max_i |\det B_{k-1}| |u_{i,k} - U(i, 1:k-1) B_{k-1}^{-1} U(I_{k-1}, k)| \\
&= \arg\max_i |u_{i,k} - U(i, 1:k-1) B_{k-1}^{-1} U(I_{k-1}, k)| \\
&= \arg\max_i |r_k(i)|.
\end{aligned}
$$

step k: residualize column k



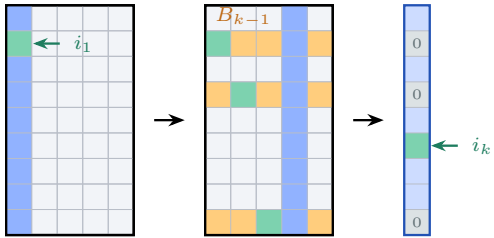
$$r_k = U(:, k) - U(:, 1:k-1) B_{k-1}^{-1} U(I_{k-1}, k)$$

Forward greedy I: add one element at a time

DEIM ([Chaturantabut/Sorensen'10])

Input: $U \in \mathbb{R}^{n\times r}$.
Output: row index set I.
$I = \{\arg\max_i |U(i,1)|\}$.
for $k = 2, \ldots, r$ **do**
    $r_k = U(:,k) - U(:,1:k-1)U(I,1:k-1)^{-1}U(I,k)$.
    $I \leftarrow I \cup \{\arg\max_i |r_k(i)|\}$.
end for

1. first column 2. old block 3. residual



Forward greedy I: add one element at a time

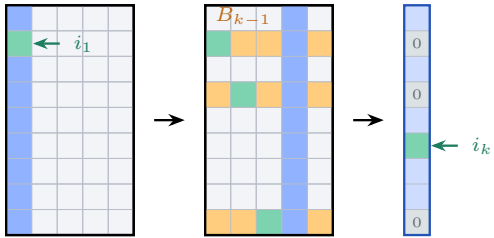
DEIM ([Chaturantabut/Sorensen'10])

Input: $U \in \mathbb{R}^{n\times r}$.
Output: row index set I.
$I = \{\arg\max_i |U(i,1)|\}$.
for $k = 2, \ldots, r$ **do**
    $r_k = U(:,k) - U(:,1:k-1)U(I,1:k-1)^{-1}U(I,k)$.
    $I \leftarrow I \cup \{\arg\max_i |r_k(i)|\}$.
end for

LUPP (GE with partial pivoting)

Input: $U \in \mathbb{R}^{n\times r}$.
Output: row index set I.
$I = \{\arg\max_i |U(i,1)|\}$.
for $k = 2, \ldots, r$ **do**
    $U = U - \frac{1}{u_{i_{k-1},k-1}} U(:,k-1)U(i_{k-1},:)$.
    $I \leftarrow I \cup \{\arg\max_i U(i,k)\}$.
end for

1. first column 2. old block 3. residual



$r_k = U(:,k) - U(:,1:k-1)B_{k-1}^{-1}U(I_{k-1},k)$

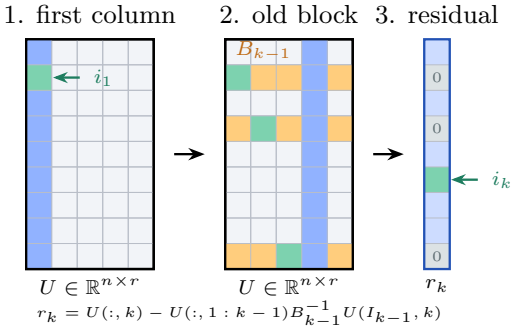
Forward greedy I: add one element at a time

DEIM ([Chaturantabut/Sorensen'10])

Input: $U \in \mathbb{R}^{n\times r}$.
Output: row index set I.
$I = \{\arg\max_i |U(i,1)|\}$.
for $k = 2, \ldots, r$ **do**
    $r_k = U(:,k) - U(:,1:k-1)U(I,1:k-1)^{-1}U(I,k)$.
    $I \leftarrow I \cup \{\arg\max_i |r_k(i)|\}$.
end for

LUPP (GE with partial pivoting)

Input: $U \in \mathbb{R}^{n\times r}$.
Output: row index set I.
$I = \{\arg\max_i |U(i,1)|\}$.
for $k = 2, \ldots, r$ **do**
    $U = U - \frac{1}{u_{i_{k-1},k-1}} U(:,k-1)U(i_{k-1},:)$.
    $I \leftarrow I \cup \{\arg\max_i U(i,k)\}$.
end for



Remark

- DEIM and LUPP are equivalent.
- DEIM can be viewed as a simplified LUPP where only the pivots are kept but not the LU factors.

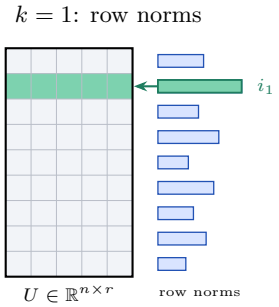
Forward greedy II: add one row at a time

Another volume-maximizing heuristic

- Step $k = 1$:
  Select a $1 \times r$ submatrix s.t. the volume is maximized:

$$i_1 = \arg\max_i \mathrm{vol}(U(i,:)) = \arg\max_i \|U(i,:)\|_2 \rightsquigarrow I_1 = \{i_1\}.$$

- Here, the volume of a rectangular matrix $M \in \mathbb{R}^{k\times r}$ is $\mathrm{vol}(M) := \sqrt{\det(MM^\top)} = \prod_{i=1}^{k} \sigma_i(M)$.



Forward greedy II: add one row at a time

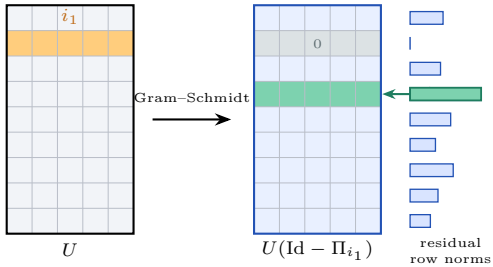
Another volume-maximizing heuristic

- Step $k = 1$: $I_1 = \{\arg\max_i \|U(i,:)\|_2\}$.
- Step $k = 2$:
  Select a $2 \times r$ submatrix containing the first one s.t. the volume is maximized:

$$
\begin{aligned}
i_2 &= \arg\max_i \operatorname{vol} \begin{bmatrix} U(i_1,:) \\ U(i,:) \end{bmatrix} \\
&= \arg\max_i \|U(i_1,:)\|_2 \left\| (\mathrm{Id} - \Pi_{i_1}) U(i,:)^\top \right\|_2 \\
&= \arg\max_i \left\| (\mathrm{Id} - \Pi_{i_1}) U(i,:)^\top \right\|_2 \\
&= \arg\max_i \left\| (\mathrm{Id} - Q_1 Q_1^\top) U(i,:)^\top \right\|_2 .
\end{aligned}
$$

Here $\Pi_{i_1} = Q_1 Q_1^\top$ with $Q_1 = U(i_1,:)^\top / \|U(i_1,:)\|_2$ is the orthogonal projection onto the first selected row direction.

$k = 2$: residual row norms



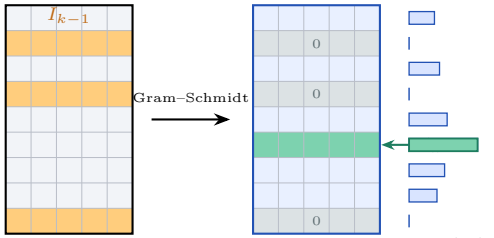
Forward greedy II: add one row at a time

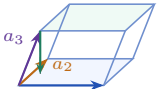
Another volume-maximizing heuristic

- Step $k = 1$: $I_1 = \{\arg\max_i \|U(i,:)\|_2\}$.
- Step $k = 2$: $I_2 = I_1 \cup \{\arg\max_i \left\|(\mathrm{Id} - Q_1 Q_1^\top)U(i,:)^\top\right\|_2\}$.
- Suppose we have selected I_{k-1} and $Q_{k-1} = \mathrm{orth}\{U(I_{k-1},:)\}$.
- Step k:
  Select a $k \times r$ submatrix containing the previous one s.t. the volume is maximized:

$$
\begin{aligned}
i_k &= \arg\max_i \mathrm{vol}\begin{bmatrix} U(I_{k-1},:) \\ U(i,:) \end{bmatrix} \\
&= \arg\max_i \mathrm{vol}(U(I_{k-1},:)) \left\|(\mathrm{Id} - \Pi_{I_{k-1}})U(i,:)^\top\right\|_2 \\
&= \arg\max_i \left\|(\mathrm{Id} - Q_{k-1}Q_{k-1}^\top)U(i,:)^\top\right\|_2 .
\end{aligned}
$$

step k: residual row norms



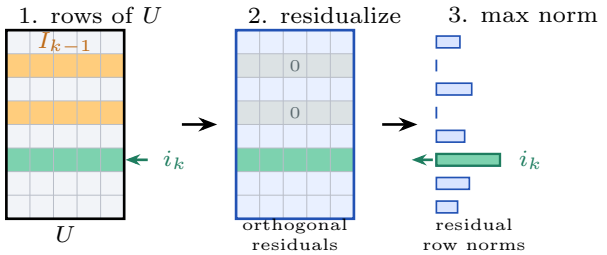


height: $(\mathrm{Id} - \Pi_{a_1,a_2})a_3$
$\mathrm{vol}([a_1, a_2, a_3])$
$= \mathrm{vol}([a_1, a_2]) \|(\mathrm{Id} - \Pi_{a_1,a_2})a_3\|_2$

Forward greedy II: add one row at a time

QRCP ([Businger/Golub'65]) on $U^\top$

Input: $U \in \mathbb{R}^{n\times r}$ with $U^\top U = \mathrm{Id}$.
Output: row index set I.
$I \leftarrow \emptyset$, $Q \leftarrow [\,]$.
for $k = 1, \ldots, r$ **do**
  $i_k \in \arg\max_i \left\|(\mathrm{Id} - QQ^\top)U(i,:)^\top\right\|_2$.
  $I \leftarrow I \cup \{i_k\}$.
  $Q \leftarrow \mathrm{orth}(Q, U(i_k,:)^\top)$.
end for



Q-DEIM ([Drmac/Gugercin'16])

Input: $U \in \mathbb{R}^{n\times r}$ with $U^\top U = \mathrm{Id}$.
Output: row index set I.
$[\sim, \sim, I] = \mathtt{qr}(U^\top, \mathtt{'vector'})$;
$I = I(1:r)$.

Remark

- Q-DEIM is QRCP applied to $U^\top$.

Why forward greedy is not enough

Limitation

Forward greedy methods work really well in practice, but their worst-case error bounds are exponential.

Recall that

$$I \in \operatorname*{arg\,max}_{|I|=r} \operatorname{vol}(U(I,:)) \implies \left\|U(I,:)^{-1}\right\|_2 \leq \sqrt{1 + r(n-r)}.$$

Theorem (Higham'02)

Let I be the index set returned by LUPP/DEIM applied to $U \in \mathbb{R}^{n\times r}$, then

$$\left\|U(I,:)^{-1}\right\|_2 \leq \sqrt{nr}\,2^{r-1}.$$

Theorem (Drmac/Gugercin'16)

Let I be the index set returned by Q-DEIM applied to $U \in \mathbb{R}^{n\times r}$ (QRCP applied to $U^\top$), then

$$\left\|U(I,:)^{-1}\right\|_2 \leq \frac{\sqrt{n-r+1}}{3}\sqrt{4^r + 6r - 1}.$$

This motivates methods that can achieve better bounds such as `maxvol` and sRRQR.

Greedy maximum volume algorithms

A sufficient condition for a good submatrix: maximum volume.

$$I \in \underset{|I|=r}{\arg\max}\, \mathrm{vol}(U(I,:)) \implies \left\|U(I,:)^{-1}\right\|_2 \leq \sqrt{1 + r(n-r)}.$$



Algorithms that greedily maximize the volume

Forward greedy	**Exchange greedy**
$\{i_1\} \rightsquigarrow \{i_1, i_2\} \rightsquigarrow \cdots \rightsquigarrow I$	$\{i_1, \ldots, i_k, \ldots, i_r\} \rightsquigarrow \{i_1, \ldots, \tilde{i}_k, \ldots, i_r\} \rightsquigarrow \cdots \rightsquigarrow I$
LUPP/DEIM, QRCP/Q-DEIM	maxvol/sRRQR

Exchange greedy: maxvol

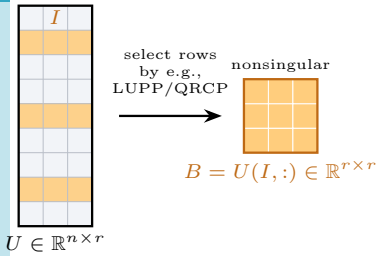
Exchange one row at a time to increase the volume

- Step 0: initialize the index set I.
- It starts from a nonsingular $r \times r$ submatrix.

$$B = U(I, :) \in \mathbb{R}^{r \times r},$$

- The initialization can be done using e.g., LUPP or QRCP as discussed.
- The question is whether an unselected row can increase the volume.
- WLOG, assume $I = \{1, \ldots, r\}$.

start from a full block



Exchange greedy: maxvol

Exchange one row at a time to increase the volume

- Step 0: initialize the index set I by e.g. applying LUPP or QRCP.
- WLOG, assume $I = \{1, \dots, r\}$ and $I_c = \{1, \dots, n\} \setminus I$.
- Consider B_{new} being B with row i replaced by $U(\alpha, :)$, $\alpha \in I_c$.
- The exchange should increase the volume as much as possible:

$$(\alpha, i) \in \arg\max_{\alpha, i} \operatorname{vol}(B_{\text{new}}) / \operatorname{vol}(B).$$

Let's compute the new volume:

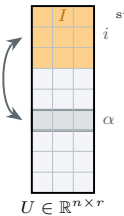
$$\det(B_{\text{new}}) = \det \begin{pmatrix} B(1,:) \\ \vdots \\ U(\alpha,:) \\ \vdots \\ B(r,:) \end{pmatrix} = \sum_{\ell=1}^{r} t_{\alpha\ell} \det \begin{pmatrix} B(1,:) \\ \vdots \\ B(\ell,:) \\ \vdots \\ B(r,:) \end{pmatrix} = t_{\alpha i} \det B$$

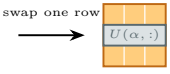
Here $U(\alpha, :) = t_\alpha^\top B$. Therefore,

$$\operatorname{vol}(B_{\text{new}}) / \operatorname{vol}(B) = |t_{\alpha i}|.$$

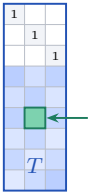
$t_\alpha^\top = U(\alpha, :) B^{-1} \rightsquigarrow t_{\alpha i}$ is the entry of $U(I_c, :) B^{-1}$.

one candidate exchange





$$\frac{\operatorname{vol}(B_{\text{new}})}{\operatorname{vol}(B)} = |t_{\alpha,i}|$$



$$= UU(I,:)^{-1} = \begin{bmatrix} \mathrm{Id}_r \\ T \end{bmatrix}$$

Exchange greedy: maxvol

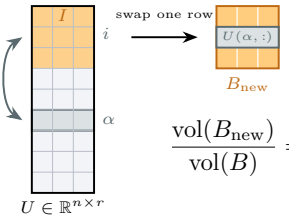
Exchange one row at a time to increase the volume

- Step 0: initialize the index set I by e.g. applying LUPP or QRCP. Reorder I so that $I = \{1, \dots, r\}$.
- Step 1: compute the $UU(I,:)^{-1} = \begin{bmatrix} \mathrm{Id}_r \\ T \end{bmatrix}$ matrix.
- Step 2: find the largest entry $t_{\alpha i}$ of T.
- Step 3: If $|t_{\alpha i}| > 1$, the volume is increased by $|t_{\alpha i}|$, exchange i and α; otherwise, return.

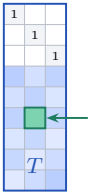
Practically, stopping criterion: $|t_{\alpha i}| \leq \mu$ for $\mu > 1$.

$$\begin{aligned} \|U(I,:)^{-1}\|_2 = \|UU(I,:)^{-1}\|_2 &= \left\| \begin{bmatrix} \mathrm{Id}_r & T \end{bmatrix}^\top \right\|_2 \\ &= \sqrt{\lambda_{\max}(\mathrm{Id}_r^2 + T^2)} = \sqrt{1 + \lambda_{\max}(T^2)} \\ &= \sqrt{1 + \|T\|_2^2} \leq \sqrt{1 + \|T\|_{\mathrm{F}}^2} \leq \sqrt{1 + r(n-r)\mu^2}. \end{aligned}$$

one candidate exchange



$$\frac{\mathrm{vol}(B_{\mathrm{new}})}{\mathrm{vol}(B)} = |t_{\alpha,i}|$$



$$= UU(I,:)^{-1} = \begin{bmatrix} \mathrm{Id}_r \\ T \end{bmatrix}$$

maxvol algorithm

The maxvol algorithm with threshold $\mu \geq 1$

Input: $U \in \mathbb{R}^{n \times r}$, threshold $\mu \geq 1$.
Output: row index set I.
Apply LUPP on U (or QRCP on $U^\top$) to get I.
$I_c \leftarrow \{1, \ldots, n\} \setminus I$.
repeat
  $B \leftarrow U(I,:)$, $T \leftarrow U(I_c,:)B^{-1}$.
  $(\alpha, i) = \arg\max_{\alpha,i} |T(\alpha, i)|$.
  if $|T(\alpha, i)| \leq \mu$ **then**
    stop.
  else
    $\alpha \leftarrow I_c(\alpha)$, $I(i) \leftarrow \alpha$.
  end if
until stopped

Theorem (Goreinov et al.'10)

Let I be the index set returned by maxvol with threshold μ. Then

$$\|U(I,:)^{-1}\|_2 \leq \sqrt{1 + \mu^2 r(n-r)}.$$

Remark

- Each entry of T is a one-row volume ratio.
- maxvol searches for an exchange that increases the volume by at least μ.

Goreinov et al.'10 = Goreinov, S. A., Oseledets, I. V., Savostyanov, D. V., Tyrtyshnikov, E. E., & Zamarashkin, N. L. (2010): How to find a good submatrix. In Matrix Methods: Theory, Algorithms and Applications, 247–256.

`maxvol` is closely related to sRRQR

Apply `maxvol` to U with threshold μ we get

$$P^\top U = \begin{bmatrix} B \\ U_c \end{bmatrix} \text{ s.t. } \|T\|_{\max} = \|U_c B^{-1}\|_{\max} \leq \mu.$$

Apply sRRQR ([Gu/Eisenstat'96]) to $U^\top$

$$U^\top \Pi = Q \begin{bmatrix} R_{11} & R_{12} \end{bmatrix} \text{ s.t. } \|R_{11}^{-1} R_{12}\|_{\max} \leq \mu.$$

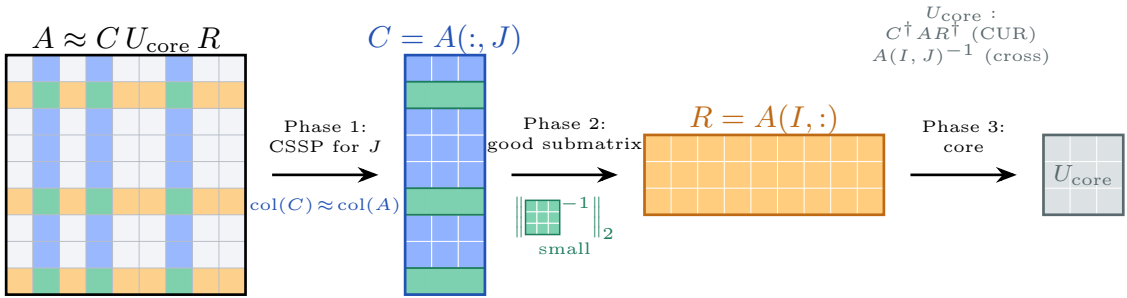
- `maxvol` controls $\|T\|_{\max} \leq \mu$.
- sRRQR controls $\|R_{11}^{-1} R_{12}\|_{\max} \leq \mu$.
- One can show: $R_{11}^{-1} R_{12} = T^\top$.
- `maxvol` usually swaps using the largest entry of T.
- Gu–Eisenstat sRRQR only needs to find a violating entry with $|(R_{11}^{-1} R_{12})_{ij}| > \mu$.



Organization

1. Motivation and notation
2. CUR/cross prototype algorithm
3. Good submatrix selection
4. Practical CUR/cross approximation algorithms
5. Connections to MOR with DEIM and tensor cross

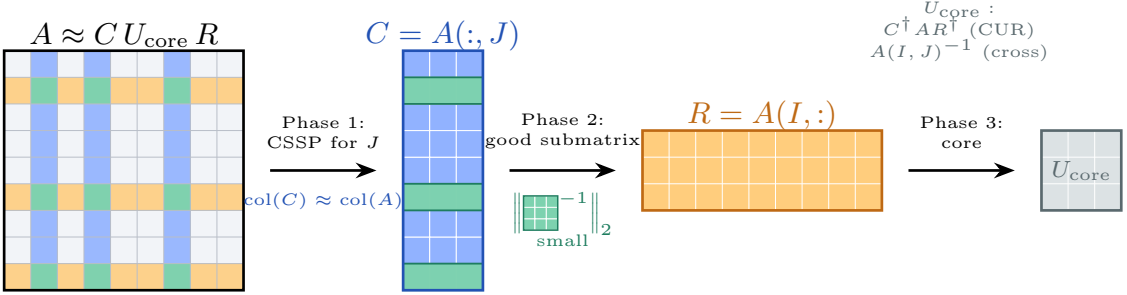
CUR/cross approximation algorithm



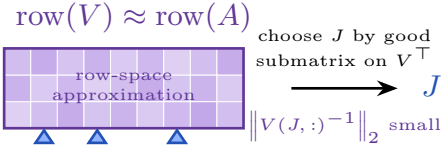
A typical prototype CUR/cross approximation algorithm for $A \in \mathbb{R}^{m \times n}$

1: Phase 1: choose columns J and get $C = A(:, J)$ (**CSSP step**)
2: Phase 2: choose rows I from $C = A(:, J)$ and get $R = A(I, :)$ (**Good submatrix step**)
3: Phase 3: choose a core U_{core}, e.g., $C^\dagger A R^\dagger$ for CUR or $A(I, J)^{-1}$ for cross
4: **return** I, J, U_{core}

Phase 1: CSSP through a row-space basis



CSSP can also be done using good submatrix selection:



1. choose a row-space approximator:
   a. $[\sim, \sim, V^\top] =$ `truncated_svd`(A);
   b. lazily select rows $V = A(I', :)$.
2. choose a good submatrix selector:
   LUPP / QRCP / maxvol

Practical algorithm 1: basis plus selector

$2\times$ basis plus selector

1: **Input:** matrix $A \in \mathbb{R}^{m\times n}$, target rank r, and a `good-submatrix` selector.
2: Compute column-space and row-space bases $U \in \mathbb{R}^{m\times r}$ and $V \in \mathbb{R}^{r\times n}$.
3: E.g., use $A \approx U\Sigma V^\top$ from a truncated or randomized SVD.
4: $J \leftarrow$ `good-submatrix`$(V^\top)$.
5: $I \leftarrow$ `good-submatrix`(U).
6: $C \leftarrow A(:, J)$ and $R \leftarrow A(I, :)$.
7: Choose U_{core}, e.g., $C^\dagger A R^\dagger$ for CUR or $A(I, J)^{-1}$ for cross.

`good-submatrix` can be LUPP, QRCP, `maxvol`, or sRRQR.

Sorensen, D. C., & Embree, M. (2016): A DEIM induced CUR factorization. SIAM Journal on Scientific Computing, 38(3), A1454–A1482.

Practical algorithm 2: sketch then pivot

$1\times$ basis plus selector + good-submatrix on selected cols

1: **Input:** matrix $A \in \mathbb{R}^{m\times n}$, target rank r, and a `good-submatrix` selector.
2: Compute a row-space basis $V \in \mathbb{R}^{r\times n}$.
3: E.g., use $A \approx U\Sigma V^\top$ from a truncated or randomized SVD.
4: More easily, $V \leftarrow \Omega A$ with $\Omega \in \mathbb{R}^{r\times m}$.
5: $J \leftarrow$ `good-submatrix`$(V^\top)$.
6: $C \leftarrow A(:, J)$.
7: $I \leftarrow$ `good-submatrix`(C).
8: $R \leftarrow A(I, :)$
9: Choose U_{core}, e.g., $C^\dagger A R^\dagger$ for CUR or $A(I, J)^{-1}$ for cross.

Both algorithms still requires the full matrix, i.e., **linear complexity** $\mathcal{O}(mn)$ ($\mathcal{O}(n^d)$ in tensors).

Dong, Y., & Martinsson, P. G. (2023). Simpler is better: a comparative study of randomized pivoting algorithms for CUR and interpolative decompositions. Advances in Computational Mathematics, 49(4), 66.

Practical algorithm 3: matrix cross

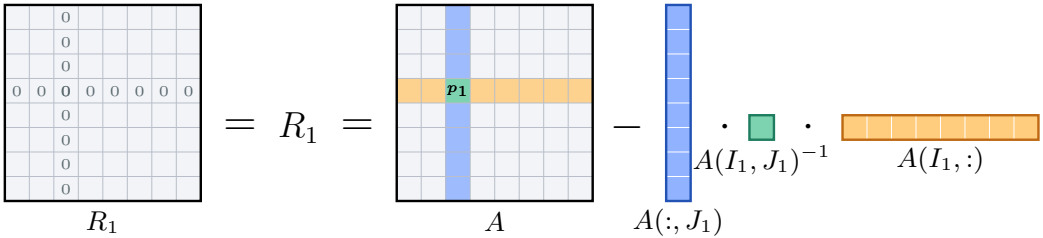
Alternating good-submatrix refinement

Input: matrix $A \in \mathbb{R}^{m \times n}$, target rank r, and an initial set J (usually random or just $\{1, \cdots, r\}$).
repeat
    Form $C \leftarrow A(:, J)$.
    Compute an orthornormal basis $Q_C = \mathrm{orth}(C)$.
    $I \leftarrow \texttt{good-submatrix}(Q_C)$.
    Form $R \leftarrow A(I, :)$.
    Compute an orthornormal basis $Q_C = \mathrm{orth}(R^\top)$.
    $J \leftarrow \texttt{good-submatrix}(Q_R)$.
until the indices or the approximation stop changing
Choose $U_{\mathrm{core}} = A(I, J)^{-1}$ or a stabilized core $A(I, J)^\dagger$.

This is the matrix algorithm behind many tensor-cross routines, as it only requires $\mathcal{O}((m+n)r)$ entries of the matrix ($\mathcal{O}(dn\,\mathrm{poly}(r))$ in tensors), i.e., **sublinear complexity**.

[Oseledets/Tyrtyshnikov'10], [Savostyanov'14], [Chiu/Demanet'13], [Li/Yang/Martin/Ho/Ying'15], [Cortinovis/Ying'26].

Practical algorithm 4: ACA



$(i_1, j_1) \in \arg\max_{i,j} |A(i,j)|$

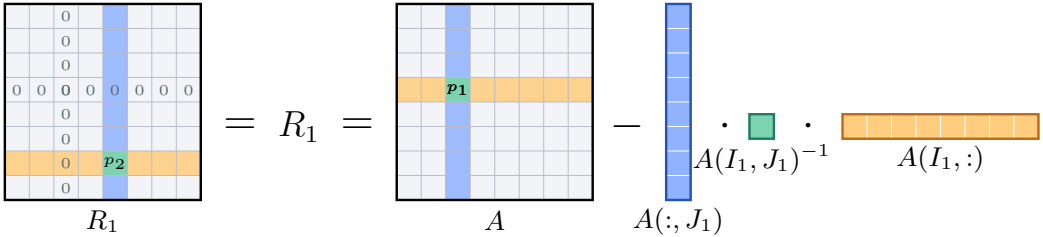
Adaptive cross approximation ([Bebendorf'00], [Bebendorf'11])

1: **Input:** entry oracle for $A \in \mathbb{R}^{m \times n}$ and tolerance ε.
2: $R_0 \leftarrow A$, $I \leftarrow \emptyset$, and $J \leftarrow \emptyset$.
3: **for** $k = 1, 2, \ldots$ **do**
4: $\quad (i_k, j_k) \in \arg\max_{i,j} |R_{k-1}(i,j)|$; $p_k \leftarrow R_{k-1}(i_k, j_k)$.
5: $\quad$ **if** $|p_k| \leq \varepsilon$ **then** stop.
6: $\quad I \leftarrow I \cup \{i_k\}$ and $J \leftarrow J \cup \{j_k\}$; $R_k \leftarrow R_{k-1} - R_{k-1}(:, j_k) p_k^{-1} R_{k-1}(i_k, :)^\top$.
7: **end for**
8: **return** $A(:, J)$, $A(I, J)^{-1}$, and $A(I, :)$.

Remark

ACA with complete pivoting (partial pivoting) is just GECP (GEPP) directly applied to A.

Practical algorithm 4: ACA



$(i_2, j_2) \in \arg\max_{i,j} |R_1(i, j)|$

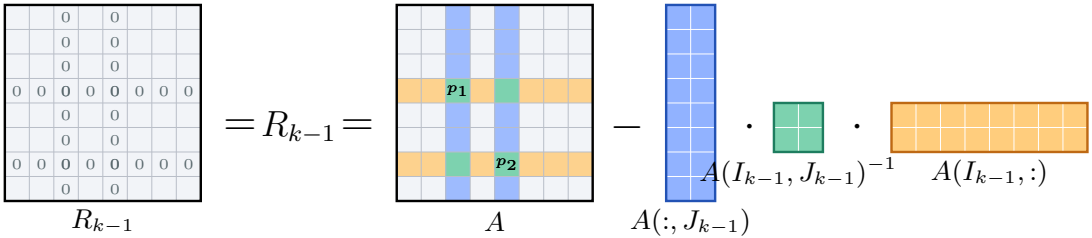
Adaptive cross approximation ([Bebendorf'00], [Bebendorf'11])

1: **Input:** entry oracle for $A \in \mathbb{R}^{m \times n}$ and tolerance ε.
2: $R_0 \leftarrow A$, $I \leftarrow \emptyset$, and $J \leftarrow \emptyset$.
3: **for** $k = 1, 2, \ldots$ **do**
4: $\quad (i_k, j_k) \in \arg\max_{i,j} |R_{k-1}(i, j)|$; $p_k \leftarrow R_{k-1}(i_k, j_k)$.
5: $\quad$ **if** $|p_k| \leq \varepsilon$ **then** stop.
6: $\quad I \leftarrow I \cup \{i_k\}$ and $J \leftarrow J \cup \{j_k\}$; $R_k \leftarrow R_{k-1} - R_{k-1}(:, j_k) p_k^{-1} R_{k-1}(i_k, :)^\top$.
7: **end for**
8: **return** $A(:, J)$, $A(I, J)^{-1}$, and $A(I, :)$.

Remark

ACA with complete pivoting (partial pivoting) is just GECP (GEPP) directly applied to A.

Practical algorithm 4: ACA



$$R_{k-1} = A - A(:, J_{k-1})A(I_{k-1}, J_{k-1})^{-1}A(I_{k-1}, :)$$

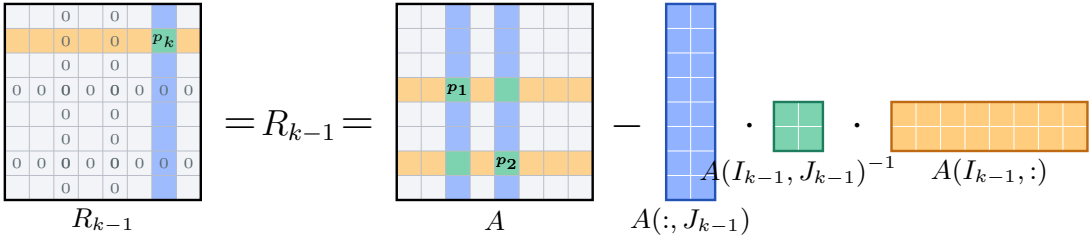
Adaptive cross approximation ([Bebendorf'00], [Bebendorf'11])

1: **Input:** entry oracle for $A \in \mathbb{R}^{m \times n}$ and tolerance ε.
2: $R_0 \leftarrow A$, $I \leftarrow \emptyset$, and $J \leftarrow \emptyset$.
3: **for** $k = 1, 2, \ldots$ **do**
4: $\quad (i_k, j_k) \in \arg\max_{i,j} |R_{k-1}(i, j)|$; $p_k \leftarrow R_{k-1}(i_k, j_k)$.
5: $\quad$ **if** $|p_k| \leq \varepsilon$ **then** stop.
6: $\quad I \leftarrow I \cup \{i_k\}$ and $J \leftarrow J \cup \{j_k\}$; $R_k \leftarrow R_{k-1} - R_{k-1}(:, j_k)p_k^{-1}R_{k-1}(i_k, :)^\top$.
7: **end for**
8: **return** $A(:, J)$, $A(I, J)^{-1}$, and $A(I, :)$.

Remark

ACA with complete pivoting (partial pivoting) is just GECP (GEPP) directly applied to A.

Practical algorithm 4: ACA



$$(i_k, j_k) \in \arg\max_{i,j} |R_{k-1}(i,j)|$$

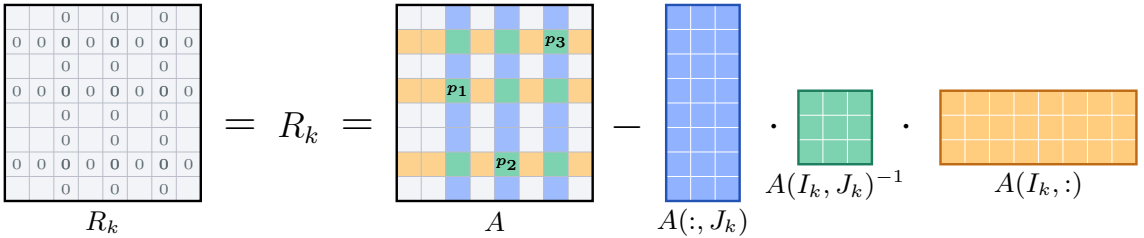
Adaptive cross approximation ([Bebendorf'00], [Bebendorf'11])

1: **Input:** entry oracle for $A \in \mathbb{R}^{m \times n}$ and tolerance ε.
2: $R_0 \leftarrow A$, $I \leftarrow \emptyset$, and $J \leftarrow \emptyset$.
3: **for** $k = 1, 2, \ldots$ **do**
4: $\quad (i_k, j_k) \in \arg\max_{i,j} |R_{k-1}(i,j)|$; $p_k \leftarrow R_{k-1}(i_k, j_k)$.
5: $\quad$ **if** $|p_k| \leq \varepsilon$ **then** stop.
6: $\quad I \leftarrow I \cup \{i_k\}$ and $J \leftarrow J \cup \{j_k\}$; $R_k \leftarrow R_{k-1} - R_{k-1}(:, j_k) p_k^{-1} R_{k-1}(i_k, :)^\top$.
7: **end for**
8: **return** $A(:, J)$, $A(I, J)^{-1}$, and $A(I, :)$.

Remark

ACA with complete pivoting (partial pivoting) is just GECP (GEPP) directly applied to A.

Practical algorithm 4: ACA



$$R_k = A - A(:, J_k)A(I_k, J_k)^{-1}A(I_k, :)$$

Adaptive cross approximation ([Bebendorf'00], [Bebendorf'11])

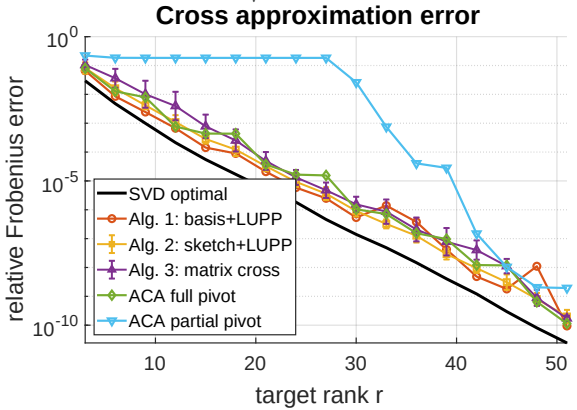
1: **Input:** entry oracle for $A \in \mathbb{R}^{m \times n}$ and tolerance ε.
2: $R_0 \leftarrow A$, $I \leftarrow \emptyset$, and $J \leftarrow \emptyset$.
3: **for** $k = 1, 2, \ldots$ **do**
4: $\quad (i_k, j_k) \in \arg\max_{i,j} |R_{k-1}(i, j)|$; $p_k \leftarrow R_{k-1}(i_k, j_k)$.
5: $\quad$ **if** $|p_k| \leq \varepsilon$ **then** stop.
6: $\quad I \leftarrow I \cup \{i_k\}$ and $J \leftarrow J \cup \{j_k\}$; $R_k \leftarrow R_{k-1} - R_{k-1}(:, j_k)p_k^{-1}R_{k-1}(i_k, :)^\top$.
7: **end for**
8: **return** $A(:, J)$, $A(I, J)^{-1}$, and $A(I, :)$.

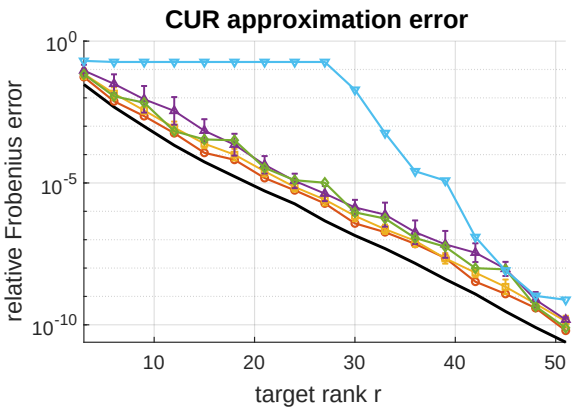
Remark

ACA with complete pivoting (partial pivoting) is just GECP (GEPP) directly applied to A.

Numerical experiment: cross/CUR approximation

$$A_{ij} = \exp\left(-15\sqrt{x_j^2 + y_i^2}\right) + \exp\left(-75\sqrt{(x_j-1)^2 + (y_i-1)^2}\right), \quad x_j \in [0,1], \quad y_i \sim \mathrm{Unif}(0,1).$$





This test follows the setup of [Cortinovis/Kressner'26, Fig. 3(b)] with $n = 2000$, and the randomized methods use 20 independent runs.

Organization

1. Motivation and notation
2. CUR/cross prototype algorithm
3. Good submatrix selection
4. Practical CUR/cross approximation algorithms
5. Connections to MOR with DEIM and tensor cross

Why should we care?

Model reduction

- DEIM selects entries of nonlinear terms.
- The online cost depends on selected entries, not on the full state dimension.

H-matrices

- ACA queries expensive matrix entries adaptively.
- Low-rank blocks are built from sampled rows and columns.

Tensor cross

- Fibers replace rows and columns.
- Good-submatrix ideas reappear on unfoldings and interfaces.

The same index-selection question reappears in many computational settings.

Vector approximation by interpolation

From orthogonal projection to oblique interpolation

Let $f \in \mathbb{R}^n$ be a high-dimensional vector; let $U \in \mathbb{R}^{n \times r}$ have orthonormal columns.
Goal: want to approximate f in the subspace $\operatorname{range} U$.
Best approximation: orthogonal projection $\Pi_U f = UU^\top f$. **Disadvantage**: Requires all entries of f.
For selected indices $I = \{i_1, \ldots, i_r\}$, define the sampling matrix

$$S_I = [e_{i_1}, e_{i_2}, \ldots, e_{i_r}].$$

DEIM replaces it by the oblique projection $\Pi_I f = U(S_I^\top U)^{-1} S_I^\top f = UU(I,:)^{-1} S_I^\top f$.

$$f = \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix} \in \mathbb{R}^n \quad \text{v.s.} \quad S_I^\top f = \begin{bmatrix} f_{i_1} \\ \vdots \\ f_{i_r} \end{bmatrix} \in \mathbb{R}^r$$

Vector approximation by interpolation

From orthogonal projection to oblique interpolation

Let $f \in \mathbb{R}^n$ be a high-dimensional vector; let $U \in \mathbb{R}^{n \times r}$ have orthonormal columns.
Goal: want to approximate f in the subspace $\operatorname{range} U$.
Best approximation: orthogonal projection $\Pi_U f = UU^\top f$. **Disadvantage**: Requires all entries of f.
For selected indices $I = \{i_1, \ldots, i_r\}$, define the sampling matrix

$$S_I = [e_{i_1}, e_{i_2}, \ldots, e_{i_r}].$$

DEIM replaces it by the oblique projection $\Pi_I f = U(S_I^\top U)^{-1} S_I^\top f = UU(I,:)^{-1} S_I^\top f$.

Error amplification

$$\|f - \Pi_I f\|_2 \leq \|U(I,:)^{-1}\|_2 \cdot \|f - UU^\top f\|_2.$$

Recall: LUPP gives $\left\|U(I,:)^{-1}\right\|_2 \leq \sqrt{nr}\,2^{r-1}$, QRCP gives $\left\|U(I,:)^{-1}\right\|_2 \leq \dfrac{\sqrt{n-r+1}}{3}\sqrt{4^r + 6r - 1}$,
`maxvol` (sRRQR) gives $\left\|U(I,:)^{-1}\right\|_2 \leq \sqrt{1 + \mu^2 r(n-r)}$.

DEIM is interpolation at selected entries

Interpolation condition

The residual vanishes at the selected entries:

$$S_I^\top (I - \Pi_I) f = 0.$$

Hence $\Pi_I f$ matches f exactly on the index set I.

Same good-submatrix problem

1: The basis U is the space approximator.
2: The selected rows form $S_I^\top U = U(I, :)$.
3: The stability factor is $\|U(I, :)^{-1}\|_2$.
4: LUPP gives the original DEIM ([Chaturantabut/Sorensen'10])
5: QRCP gives Q-DEIM ([Drmac/Gugercin'16])
6: sRRQR (plus leverage score sampling) gives R-DEIM ([Drmac/Saibaba'18], [Saibaba'20]).

Continuous version: EIM ([Barrault/Maday/Nguyen/Patera'04]).

POD+DEIM

Consider a large ODE

$$\dot{x}(t) = Ax(t) + F(x(t)), \qquad A \in \mathbb{R}^{n\times n}.$$

Basic idea of Proper orthogonal decomposition (POD)

1. Simulate the ODE for one or more initial conditions and collect trajectories at discrete time points into snapshot matrix:
$$X = \big([x(t_1), \ldots, x(t_m)]\big) \in \mathbb{R}^{n\times m}.$$
2. Compute basis approximation $\mathrm{range}(U) \approx \mathrm{range}(X)$ with $U \in \mathbb{R}^{n\times r}$, $r \ll n$, e.g., $[U, \sim, \sim] = \texttt{truncated_svd}(X)$.
3. Approximate by orth. proj. $x(t) \approx UU^\top x(t) = Uy(t)$ with $y(t) = U^\top x(t)$. Galerkin projection gives a small ODE:
$$\dot{y}(t) = U^\top AUy(t) + U^\top F(Uy(t)).$$

Problem: The nonlinear term still asks for the full vector $U^\top F(Uy(t)) \in \mathbb{R}^n$.

Solution: Replace orth. proj. by oblique proj., i.e., DEIM: $U^\top F(Uy(t)) \approx S_I^\top U^{-1} S_I^\top F(Uy(t))$.

POD+DEIM

POD

$x(t) \approx UU^\top x(t) = Uy(t), \qquad y(t) = U^\top x(t).$

ROM takes the form:

$$\dot{y}(t) = U^\top AUy(t) + U^\top F(Uy(t)).$$

Nonlinear term still asks for the full vector

$$F(x) = \begin{bmatrix} f_1(x_1) \\ \vdots \\ f_n(x_n) \end{bmatrix}.$$

POD+DEIM

Using $U^\top F(Uy(t)) \approx S_I^\top U^{-1} S_I^\top F(Uy(t))$.

ROM takes the form:

$$\dot{y}(t) = U^\top AUy(t) + (S_I^\top U)^{-1} S_I^\top F(Uy(t)).$$

Only requires

$$S_I^\top F(x) = \begin{bmatrix} f_{i_1}(x_{i_1}) \\ \vdots \\ f_{i_r}(x_{i_r}) \end{bmatrix}.$$

DEIM needs to evaluate only r entries instead of n entries of the high-dimensional function F.

TT-cross: matrix cross on unfoldings

What changes for tensors?

1: A tensor entry is indexed by several modes, e.g., $\mathcal{A}(i_1, \ldots, i_d)$.
2: Rows and columns are replaced by fibers, slabs, and interface matrices.
3: A tensor-cross method never forms all tensor entries.
4: It repeatedly builds small matrix-cross problems on unfoldings or interfaces.
5: Good-submatrix selection chooses stable fibers and interface indices.
6: The result is a compressed tensor representation, for example in tensor-train form.

[Oseledets/Tyrtyshnikov'10], [Savostyanov'14], [Goreinov/Tyrtyshnikov/Zamarashkin'97].

Conclusion

1. CUR/cross approximation is easy once the index sets are fixed.
2. The prototype separates the task into a space or column subset selection step, a good-submatrix step, and a core formation step.
3. The hard stability question is the conditioning of the selected submatrix.
4. LUPP, QRCP, `maxvol` are practical volume-based selectors.
5. Practical algorithms differ mainly in how they obtain the space approximator before applying a selector.
6. MOR-DEIM and tensor cross reuse the same good-submatrix principle in different contexts.

Thank you!