Fast algorithms and numerical methods for the solution of Boundary Element Methods

Session 4: Low rank approximations

Stéphanie Chaillat-Loseille

Laboratoire POEMS (CNRS-INRIA-ENSTA)
stephanie.chaillat@ensta.fr

MS-04
2024/2025

Outline of the boundary element method

Illustration with the EFIE with Dirichlet Boundary Condition

Step 1: Solve the boundary integral equation

$$\int_\Gamma G(\boldsymbol{x}-\boldsymbol{y})p(\boldsymbol{y})dS_{\boldsymbol{y}} = -u^{inc}(\boldsymbol{x}), \quad \boldsymbol{x} \in \Gamma$$

- Linear system to solve
- Unknowns only on the boundary

Step 2: Invoke the boundary integral representation for the evaluation of the quantities at interior points (boundary excluded)

$$u^+(\boldsymbol{x}) = \int_\Gamma G(\boldsymbol{x}-\boldsymbol{y})p(\boldsymbol{y})dS_{\boldsymbol{y}}, \quad \boldsymbol{x} \in \Omega^+/\Gamma.$$

- Cost reduced matrix-vector product: $p(\boldsymbol{y})$ already known on Γ

How to reduce the costs of the BEM?

BIE to solve (EFIE): $\int_\Gamma G(\boldsymbol{x}-\boldsymbol{y})p(\boldsymbol{y})dS_{\boldsymbol{y}} = -u^{inc}(\boldsymbol{x}), \quad \boldsymbol{x} \in \Gamma$

BEM discretization ⇒ fully-populated system $\mathbb{A}\boldsymbol{p} = \boldsymbol{b}$

Assembly of the matrix system and matrix-vector product: $O(N^2)$

How can we reduce the costs?

- Not possible to speed-up the solution of the initial system
- But it is possible for an approximate system

$$\mathbf{A} := \begin{bmatrix} -2 & 4 & 6 & -3 \\ 4 & -8 & -12 & 6 \\ -6 & 12 & 18 & -9 \\ -8 & 16 & 24 & -12 \end{bmatrix} = \begin{bmatrix} 1 \\ -2 \\ 3 \\ 4 \end{bmatrix} \begin{bmatrix} -2 & 4 & 6 & -3 \end{bmatrix}$$

- BEM system is not low-rank but it can be approximated by a low-rank system: reduction of storage and solution time

Algebraic fast BEM

Example with 1 Gauss point per element and a $\mathbb{P}^0$ interpolation

$$\begin{bmatrix} b_1 \\ \vdots \\ b_N \end{bmatrix} = \begin{bmatrix} w_{\Gamma_1} & & \\ & \ddots & \\ & & w_{\Gamma_N} \end{bmatrix} \begin{bmatrix} G(\mathbf{x}_i, \mathbf{y}_j) \end{bmatrix} \begin{bmatrix} w_{\Gamma_1} & & \\ & \ddots & \\ & & w_{\Gamma_N} \end{bmatrix} \begin{bmatrix} p_1 \\ \vdots \\ p_N \end{bmatrix}$$

If we can find $\mathbb{U}$ ($N \times r$ with $r \ll N$) and $\mathbb{V}$ ($N \times r$ with $r \ll N$) such that $\mathbb{G} \simeq \mathbb{U}\mathbb{V}^T$, it follows a similar approximation for $\mathbb{A} \simeq \mathbb{U}_A \mathbb{V}_A^T$.

$$\begin{bmatrix} b_1 \\ \vdots \\ b_N \end{bmatrix} = \underbrace{\begin{bmatrix} w_{\Gamma_1} & & \\ & \ddots & \\ & & w_{\Gamma_N} \end{bmatrix} \mathbb{U}}_{\mathbb{U}_A \text{ of size } N \times r} \underbrace{\mathbb{V}^T \begin{bmatrix} w_{\Gamma_1} & & \\ & \ddots & \\ & & w_{\Gamma_N} \end{bmatrix}}_{\mathbb{V}_A^T \text{ of size } r \times N} \begin{bmatrix} p_1 \\ \vdots \\ p_N \end{bmatrix}$$

Advantages of this algebraic approach?

- Factorization of matrix-vector product $\mathbb{A}\boldsymbol{p} = \mathbb{U}_A(\mathbb{V}_A^T \boldsymbol{p})$
- Possibility to combine with a direct solver

Singular Value Decomposition

Rank of a matrix:

Column rank: max # of linearly independent column vectors
Column and row ranks are equal $\Rightarrow$ rank of the matrix

Definition (Singular Value Decomposition)
$\mathbb{M} \in \mathbb{C}^{m \times n}$ with $\text{rank}(\mathbb{M}) = r$. The Singular Value Decomposition (SVD) of $\mathbb{M}$ is the choice of two orthogonal basis

- $\mathbf{v}_1, \ldots, \mathbf{v}_r$ of row space of $\mathbb{M}$ (right singular vectors)
- and $\mathbf{u}_1, \ldots, \mathbf{u}_r$ of column space of $\mathbb{M}$ (left singular vectors)
- such that $\mathbb{M}\boldsymbol{v}_i = \sigma_i \mathbf{u}_i$, $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r \geq 0$ (singular values)

Link with the eigendecomposition

The left singular vectors of $\mathbb{M}$ are eigenvectors of $\mathbb{M}\mathbb{M}^*$
The right-singular vectors of $\mathbb{M}$ are eigenvectors of $\mathbb{M}^*\mathbb{M}$
The non-zero singular values of $\mathbb{M}$ are the square roots of the non-zero eigenvalues of $\mathbb{M}\mathbb{M}^*$ and $\mathbb{M}^*\mathbb{M}$

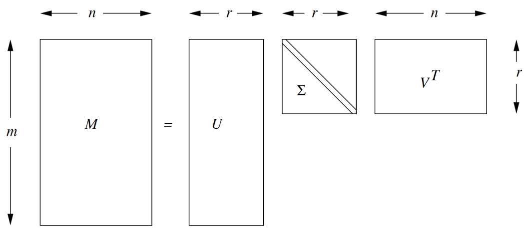
Singular Value Decomposition: matrix form

Theorem (Singular Value Decomposition)
$\mathbb{M} \in \mathbb{C}^{m \times n}$, *there exists a factorization of* $\mathbb{M}$ *of the form* $\mathbb{M} = \mathbb{U}\Sigma\mathbb{V}^*$

- $\mathbb{U}$ *and* $\mathbb{V}$ *are unitary matrices:* $\mathbb{U}^*\mathbb{U} = \mathbb{I}_m$ *and* $\mathbb{V}^*\mathbb{V} = \mathbb{I}_n$
- Σ *is a diagonal matrix (singular values)*

The storage is reduced to $O(mr + r + nr)$

G.H. Golub and C.F. Van Loan. *Matrix computations.* JHU Press, 2012.



SVD and low-rank approximations

The SVD does not give an approximation but only a factorization

Definition (Truncated SVD)
$\mathbb{M}_r$ is the SVD of $\mathbb{M}$ truncated to the r largest singular values

$$\mathbb{M}_r = \sum_{i=1}^{r} \mathbb{U}_i \Sigma_{ii} \mathbb{V}_i^*$$

The numerical rank depends on the used norm

$$k(\varepsilon) := \min\{r \quad | \quad ||\mathbb{M} - \mathbb{M}_r|| \leq \varepsilon ||\mathbb{M}||\}$$

Unitary invariant norm $||\mathbb{U}\mathbb{M}\mathbb{V}|| = ||\mathbb{M}||$ for all unitary matrices $\mathbb{U}$ and $\mathbb{V}$

- Frobenius norm: $||\mathbb{M}||^2{}_F = \sum_{i,j} |\mathbb{M}_{ij}|^2$
  - Easy to compute (if $\mathbb{M}$ is known)
- Spectral or 2-norm: $||\mathbb{M}||_2 = \sigma_1$ (σ_1 largest singular value)
  - Need to compute the SVD
- Frobenius norm is always at least as large as the spectral radius

$$||\mathbb{M}||_2 \leq ||\mathbb{M}||_F \leq \sqrt{r}||\mathbb{M}||_2$$

Best low-rank approximation

Theorem (Eckart-Young, Best low rank approximation)
$\mathbb{M} \in \mathbb{C}^{m \times n}$ *with* $m \geq n$, *and* $||.||$ *a unitary invariant norm. The best rank-k approximation* $\mathbb{M}_r$ *of* $\mathbb{M}$ *defined such that*

$$\mathbb{M}_r := min\left\{||\mathbb{M} - \mathbb{R}|| \quad | \quad \mathbb{R} \in \mathbb{C}^{m \times n}, rank(\mathbb{R}) \leq r\right\} = ||\mathbb{M} - \mathbb{M}_r||$$

$$\textit{is} \quad \mathbb{M}_r = \sum_{i=1}^{r} \mathbb{U}_i \Sigma_{ii} \mathbb{V}_i^*, \quad \textit{with} \quad \mathbb{M} = \mathbb{U}\Sigma\mathbb{V}^*.$$

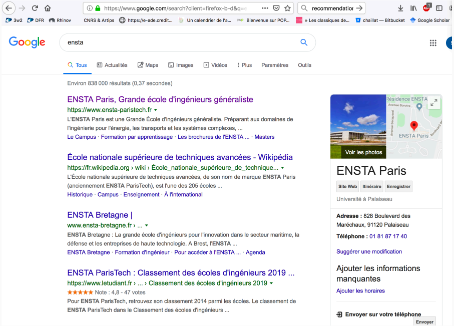
In addition, $||\mathbb{M} - \mathbb{M}_r||_F^2 = \sum_{i=r+1}^{n} \sigma_i^2$ *and* $||\mathbb{M} - \mathbb{M}_r||_2 = \sigma_{r+1}$.

Truncated SVD is the best low-rank approximation for L^2-norm.

Low-rank approximations: finding the main information

Representing concepts hidden in massive datasets: matrices are used to

- Evaluate the importance of Web pages: Pagerank algorithm (number of occurrences is easy to fool, add the links between pages)
- Community detection: social networks, protein interaction network
- Recommendation systems: Amazon, Netflix



Finding concepts underlying movies

	M1	M2	M3	M4	M5
Jill	3	1	1	3	1
Jane	1	2	4	1	3
Joe	3	1	1	3	1
Jack	4	3	5	4	4

Finding concepts underlying movies

	M1	M2	M3	M4	M5
Jill	3	1	1	3	1
Jane	1	2	4	1	3
Joe	3	1	1	3	1
Jack	4	3	5	4	4

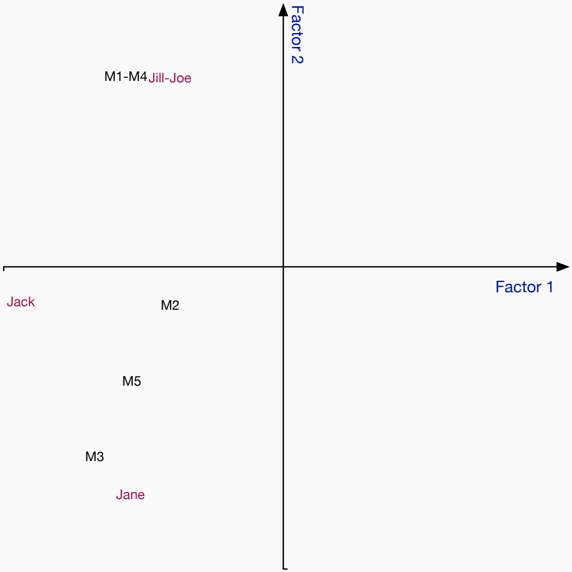
= U S V' =

-0.3460	0.5294
-0.4190	-0.6515
-0.3460	0.5294
-0.7649	-0.1221

11.822	0
0	3.9039

-0.4698	-0.3235	-0.5238	-0.4698	-0.4237
0.5217	-0.1564	-0.5527	0.5217	-0.3545

SVD Maps Users and Items Into Latent Space

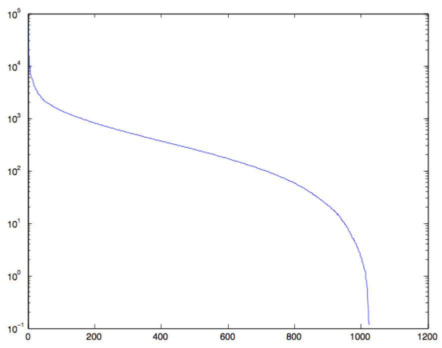


Low-rank approximations: finding the main information

Image Compression: the goal is to reduce the storage

- Images represented as matrices of size n times m pixels
- Gray scale images: 1 number per pixel
- Color images: 3 numbers per pixel (red, green and blue)

SVD: form the best rank-r approximations for the matrix

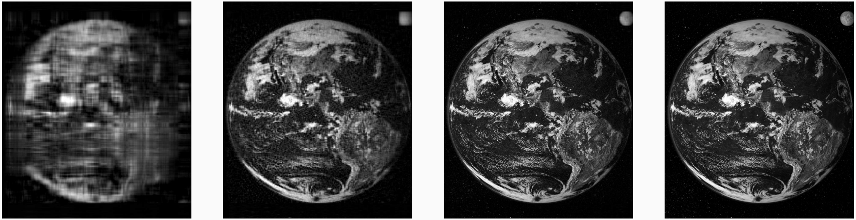


Low-rank approximations: finding the main information

Image Compression: the goal is to reduce the storage

- Images represented as matrices of size n times m pixels
- Gray scale images: 1 number per pixel
- Color images: 3 numbers per pixel (red, green and blue)

SVD: form the best rank-r approximations for the matrix



$r = 10$ $r = 50$ $k = 200$ $r = 1024$

Truncated SVD to remove the redundant information

Low-rank matrices

If we have a low-rank representation of the matrix: $\mathbb{M} = \mathbb{A}\mathbb{B}^T$

- with $\mathbb{A} \in \mathbb{R}^{m \times r}$ and $\mathbb{B} \in \mathbb{R}^{n \times r}$
- Then the storage is reduced from mn to $r(m+n)$

Acceleration of the matrix-vector multiplication: $\mathbb{M}\boldsymbol{x} = \mathbf{y}$

Low-rank matrices

If we have a low-rank representation of the matrix: $\mathbb{M} = \mathbb{A}\mathbb{B}^T$

- with $\mathbb{A} \in \mathbb{R}^{m \times r}$ and $\mathbb{B} \in \mathbb{R}^{n \times r}$
- Then the storage is reduced from mn to $r(m+n)$

Acceleration of the matrix-vector multiplication: $\mathbb{M}\boldsymbol{x} = \mathbf{y}$

- Step 1: $\mathbf{w} \leftarrow \mathbb{B}^T\mathbf{x}$
- Step 2: $\mathbf{y} \leftarrow \mathbb{A}\mathbf{w}$
- The number of operations is reduced from $O(mn)$ to $O(r(m+n))$

Definition (Low-rank matrices)
$\mathbb{M} \in \mathbb{R}^{m \times n}$ of rank r is called low-rank if

$$r(m+n) \ll m.n$$

We will always use this representation for low-rank matrices

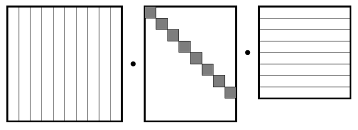
Computing a low-rank approximation

The truncated SVD gives the best low-rank approximation
But computing SVD too expensive: $O(rN^3)$ (r: rank of approximation)

- If we know the SVD: $\mathbb{M} = \mathbb{U}\Sigma\mathbb{V}^*$
  - Direct solver: compute the pseudo-inverse $\mathbb{M}^+ = \mathbb{V}\Sigma^+\mathbb{U}^*$
  - Iterative solver: compute the approximation $\mathbb{M} = \mathbb{A}\mathbb{B}^* = \mathbb{U}\Sigma\mathbb{V}^*$ to accelerate the matrix-vector product

Is the SVD the only way to compute a low-rank approximation?

Need of a factorization but not of all the properties of the SVD
SVD requires all entries of a matrix to construct low-rank approx.



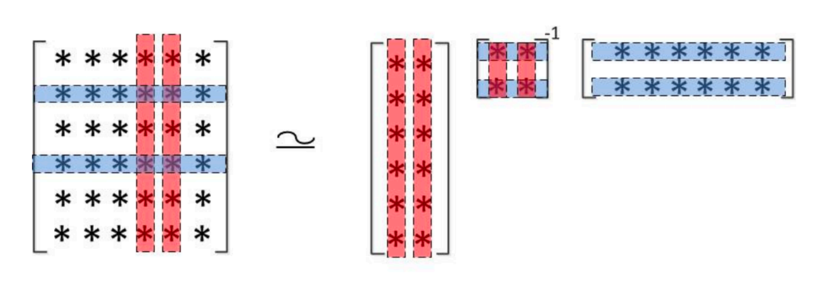
Savings if we use only a small part of the entries

We use the Adaptative Cross Approximation

Skeleton decomposition

Definition (Skeleton decomposition)
$\mathbb{A} \in \mathbb{R}^{m\times n}$, rank $\mathbb{A} = r$. There exists a non-singular submatrix $\hat{\mathbb{A}} \in \mathbb{R}^{r\times r}$ $\hat{\mathbb{A}} = \mathbb{A}(\hat{I}, \hat{J})$ with $\mathbb{A} = \mathbb{C}\hat{\mathbb{A}}^{-1}\mathbb{R}$, $\mathbb{C} = \mathbb{A}(I, \hat{J})$, $\mathbb{R} = \mathbb{A}(\hat{I}, J)$

Goreinov, Tyrtyshnikov and Zamarashkin. *A Theory of Pseudoskeleton Approximations.* Linear Algebra and its Applications, 1997.



Skeleton decomposition

Sketch of the proof

- By definition of the rank, since $\mathbb{A}$ is of rank r there exists an invertible submatrix of $\mathbb{A}$, $\hat{\mathbb{A}}$ of size $r \times r$
- It follows the definition of $\hat{I}$, $\hat{J}$, $\mathbb{R}$ and $\mathbb{C}$

Noting $\mathbb{A} = \begin{bmatrix} \alpha\mathbb{A}_2 & \mathbb{A}_2 & \beta\mathbb{A}_2 \\ \alpha\hat{\mathbb{A}} & \hat{\mathbb{A}} & \beta\hat{\mathbb{A}} \\ \alpha\mathbb{A}_7 & \mathbb{A}_7 & \beta\mathbb{A}_7 \end{bmatrix}$, it follows $\mathbb{C}\hat{\mathbb{A}}^{-1}\mathbb{R} = \begin{bmatrix} \mathbb{A}_2\hat{\mathbb{A}}^{-1}\alpha\hat{\mathbb{A}} & \mathbb{A}_2 & \mathbb{A}_2\hat{\mathbb{A}}^{-1}\beta\hat{\mathbb{A}} \\ \hat{\mathbb{A}}\hat{\mathbb{A}}^{-1}\alpha\hat{\mathbb{A}} & \hat{\mathbb{A}} & \hat{\mathbb{A}}\hat{\mathbb{A}}^{-1}\beta\hat{\mathbb{A}} \\ \mathbb{A}_7\hat{\mathbb{A}}^{-1}\alpha\hat{\mathbb{A}} & \mathbb{A}_7 & \mathbb{A}_7\hat{\mathbb{A}}^{-1}\beta\hat{\mathbb{A}} \end{bmatrix}$

- Finally, use the fact that rows and columns are linear combinations of the rows and columns of $\hat{\mathbb{A}}$

Verify that it is correct on a small matrix of rang 3

Fully-pivoted Cross Approximation

Starting point: Every rank r matrix is the sum of r matrices of rang 1

Principle: iteratively removing a row and a column of the matrix

- Successive approximations applied to the remainder

$$\mathbb{A} = \mathbb{A}_k + \mathbb{R}_k, \quad \mathbb{R}_k = \mathbb{A} - \sum_{\ell=1}^{k} \mathbf{u}_\ell \mathbf{v}_\ell^T$$

- Similarly to the Gaussian elimination, the pivot is the largest entry of the matrix (to define a stable algorithm)
- At each iteration, we nullify in the remainder the rows and columns dependent from the pivot row and column

$$\mathbb{A} = \begin{bmatrix} a_{11} & a_{12} & \alpha a_{11} \\ a_{21} & a_{22} & \alpha a_{21} \\ a_{31} & a_{32} & \alpha a_{31} \end{bmatrix} \quad \mathbb{R}_1 = \mathbb{A} - \mathbb{A}(:,1)\mathbb{A}(2,:)/a_{21} = \begin{bmatrix} 0 & r_{12} & 0 \\ 0 & 0 & 0 \\ 0 & r_{32} & 0 \end{bmatrix}$$

At iteration k:

- Find the pivot (i^*, j^*) such that $(i^*, j^*) = \text{argmax}_{ij} |(\mathbb{R}_k)_{ij}|$
- Compute vectors: $\mathbf{u}_{k+1} := \frac{(\mathbb{R}_k)_{ij^*}}{(\mathbb{R}_k)_{i^*j^*}}$, $\mathbf{v}_{k+1} := (\mathbb{R}_k)_{i^*j}$
- Update the approximation: $\mathbb{A}_{k+1} = \mathbb{A}_k + \mathbf{u}_{k+1}\mathbf{v}_{k+1}^T$

Example

$$R_0 = \begin{array}{cc} v_0 & \begin{bmatrix} 0.431 & 0.354 & 0.582 & 0.417 & 0.455 \\ 0.491 & 0.396 & 0.674 & 0.449 & 0.427 \\ 0.446 & 0.358 & 0.583 & 0.413 & 0.441 \\ 0.380 & 0.328 & 0.557 & 0.372 & 0.349 \\ 0.412 & 0.340 & 0.516 & 0.375 & 0.370 \end{bmatrix} \\ & u_0 \end{array}$$

Example

$$R_0 = \begin{array}{c} \\ v_0 \\ \\ \\ \\ \\ \end{array}\begin{array}{c} \begin{bmatrix} 0.431 & 0.354 & 0.582 & 0.417 & 0.455 \\ 0.491 & 0.396 & 0.674 & 0.449 & 0.427 \\ 0.446 & 0.358 & 0.583 & 0.413 & 0.441 \\ 0.380 & 0.328 & 0.557 & 0.372 & 0.349 \\ 0.412 & 0.340 & 0.516 & 0.375 & 0.370 \end{bmatrix} \\ \begin{array}{ccccc} & & u_0 & & \end{array} \end{array}$$

$$R_1 = R_0 - u_0 v_0$$

$$= \begin{array}{c} v_1 \\ \\ \\ \\ \\ \\ \end{array}\begin{array}{c} \begin{bmatrix} 0.0070 & 0.0121 & 0 & 0.0293 & 0.0863 \\ 0 & 0 & 0 & 0 & 0 \\ 0.0213 & 0.0155 & 0 & 0.0246 & 0.0717 \\ -0.0258 & 0.0007 & 0 & 0.0009 & -0.0039 \\ 0.0361 & 0.0368 & 0 & 0.0313 & 0.0431 \end{bmatrix} \\ \begin{array}{ccccc} & & & & u_1 \end{array} \end{array}$$

Example

$$
R_0 = \begin{array}{c c}
\begin{matrix} \\ v_0 \\ \\ \\ \\ \end{matrix} &
\begin{bmatrix}
0.431 & 0.354 & 0.582 & 0.417 & 0.455 \\
0.491 & 0.396 & 0.674 & 0.449 & 0.427 \\
0.446 & 0.358 & 0.583 & 0.413 & 0.441 \\
0.380 & 0.328 & 0.557 & 0.372 & 0.349 \\
0.412 & 0.340 & 0.516 & 0.375 & 0.370
\end{bmatrix} \\
& \begin{matrix} & & u_0 & & \end{matrix}
\end{array}
$$

$$
R_2 = R_1 - u_1 v_1
$$

$$
= \begin{bmatrix}
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0.0155 & 0.0055 & 0 & 0.0003 & 0 \\
-0.0255 & 0.0013 & 0 & 0.0023 & 0 \\
-0.0326 & 0.0308 & 0 & 0.0166 & 0
\end{bmatrix}
$$

Example

$$R_0 = \begin{array}{cc} \begin{array}{c} \\ v_0 \\ \\ \\ \end{array} & \begin{bmatrix} 0.431 & 0.354 & 0.582 & 0.417 & 0.455 \\ 0.491 & 0.396 & 0.674 & 0.449 & 0.427 \\ 0.446 & 0.358 & 0.583 & 0.413 & 0.441 \\ 0.380 & 0.328 & 0.557 & 0.372 & 0.349 \\ 0.412 & 0.340 & 0.516 & 0.375 & 0.370 \end{bmatrix} \\ & \begin{array}{ccccc} & & u_0 & & \end{array} \end{array}$$

...

Fully-pivoted Cross Approximation: pseudo-code

Initialization: $\mathbb{R}_0 := A$, $\mathscr{P}_r = \emptyset$; $\mathscr{P}_c = \emptyset$, $k = 0$

repeat

$k := k + 1$

Find the pivot $(i^*, j^*) := \text{argmax}_{i,j} |\mathbb{R}_{k-1}(i, j)|$

$\mathscr{P}_r = \mathscr{P}_r \cup \{i^*\}, \quad \mathscr{P}_c = \mathscr{P}_c \cup \{j^*\}$

$\delta_k := \mathbb{R}_{k-1}(i^*, j^*)$

$\mathbf{u}_k := \mathbb{R}_{k-1}(:, j^*)$

$\mathbf{v}_k := \mathbb{R}_{k-1}(i^*, :)/\delta_k$

$\mathbb{R}_k = \mathbb{R}_{k-1} - \mathbf{u}_k \mathbf{v}_k$

until $||\mathbb{R}_k||_F \leq \varepsilon ||\mathbb{A}||_F$

- It requires steps to generate an approximation of rank r
- It requires to compute the pivot indices

Fully-pivoted Cross Approximation: pseudo-code

Initialization: $\mathbb{R}_0 := A$, $\mathscr{P}_r = \emptyset$; $\mathscr{P}_c = \emptyset$, $k = 0$
repeat
  $k := k + 1$
  Find the pivot $(i^*, j^*) :=$ argmax$_{i,j} |\mathbb{R}_{k-1}(i,j)|$
  $\mathscr{P}_r = \mathscr{P}_r \cup \{i^*\}, \quad \mathscr{P}_c = \mathscr{P}_c \cup \{j^*\}$
  $\delta_k := \mathbb{R}_{k-1}(i^*, j^*)$
  $\mathbf{u}_k := \mathbb{R}_{k-1}(:, j^*)$
  $\mathbf{v}_k := \mathbb{R}_{k-1}(i^*, :)/\delta_k$
  $\mathbb{R}_k = \mathbb{R}_{k-1} - \mathbf{u}_k \mathbf{v}_k$
until $||\mathbb{R}_k||_F \leq \varepsilon ||\mathbb{A}||_F$

- It requires $O(rmn)$ steps to generate an approximation of rank r
- It requires all the entries of $\mathbf{A}$ to compute the pivot indices

Exact reproduction of rank r matrices

Lemma (Exact reproduction of rank r matrices)
Let $\mathbb{A}$ be matrix of rank exactly r. Then the matrix $\mathbb{A}_r$ is equal to $\mathbb{A}$.

$$\mathbb{A}_r := \sum_{\ell=1}^{r} \mathbf{u}_\ell \mathbf{v}_\ell^T$$

If rank($\mathbb{A}$)=r, the algorithm terminates in r steps.

Consistent with the Skeleton decomposition: $\mathbb{A} \in \mathbb{R}^{m \times n}$, rank $A = r$.
There exists a non-singular submatrix $\hat{\mathbb{A}} \in \mathbb{R}^{r \times r}$ $\hat{\mathbb{A}} = \mathbb{A}(\hat{I}, \hat{J})$ with
$\mathbb{A} = \mathbb{C}\hat{\mathbb{A}}^{-1}\mathbb{R}$, $\mathbb{C} = \mathbb{A}(I, \hat{J})$, $\mathbb{R} = \mathbb{A}(\hat{I}, J)$

How can we reduce the complexity?

Principle of the Partially-pivoted Cross Approximation

- Fully-pivoted: pivot is the largest entry in the residual
- Partially-pivoted: maximize only for 1 of the 2 indices (the other one is fixed) → only one row or one column is assembled

Partially-pivoted CA: pseudo-code

function $\textsc{AcaPartial}(f)$
  $k \leftarrow 1,\ \mathcal{P}_l \leftarrow \varnothing,\ \mathcal{P}_c \leftarrow \varnothing$
  $i^* \leftarrow 1$ ▷ La première ligne est choisie comme pivot
  repeat
    $\mathcal{P}_l \leftarrow \mathcal{P}_l \cup \{i^*\}$
    $b^k_j \leftarrow M_{i^*j} - \sum_{\nu=1}^{k-1} a^\nu_{i^*} b^\nu_j$ ▷ Calcul et mise à jour
    $j^* \leftarrow \arg\max_{j \in \tau - \mathcal{P}_c} |b^\nu|,\ \delta \leftarrow b^k_{j^*}$ ▷ Recherche de la colonne pivot
    if $\delta = 0$ **then**
      if $\sigma - \mathcal{P}_l = \varnothing$ **then** ▷ Il n'est plus possible de trouver une ligne pivot.
        return ▷ Pas de convergence
      end if
      $i^* = \min\{i \in \sigma \mid i \notin \mathcal{P}_l\}$ ▷ Ou tout autre choix dans $\sigma - \mathcal{P}_l$
    else ▷ Un pivot non nul est trouvé
      $\mathcal{P}_c \leftarrow \mathcal{P}_c \cup j^*$
      $b \leftarrow b/\delta$
      $a^k_i \leftarrow M_{ij^*} - \sum_{\nu=1}^{k-1} a^\nu_i b^\nu_{j^*}$ ▷ Calcul et mise à jour
      $i^* \leftarrow \arg\max_{i \in \sigma - \mathcal{P}_l} |a^\nu|$ ▷ Recherche de la ligne pivot
      $k \leftarrow k+1$
    end if
  until Convergence
  return $A := (a^\nu)_{\nu=1,\ldots,k} \quad B := (b^\nu)_{\nu=1,\ldots,k}$
end function

From PhD of B. Lizé

Adaptive Cross Approximation (ACA)

$\mathbb{R}_k$ is never explicitly formed

$$\mathbb{R}_k(i,j) = \mathbb{A}(i,j) - \sum_{\ell=1}^{k} \mathbf{u}_\ell(i)\mathbf{v}_\ell(j)$$

Can we determine the rank k adaptively for a given approximation accuracy ε?

- Fully-pivoted ACA: $||\mathbb{A} - \mathbb{A}_k||_F \leq \varepsilon ||\mathbb{A}||_F$
- Partially-pivoted ACA: $\mathbb{A}$ is not formed, stagnation-based error estimator

$$||\mathbf{u}_k||_2 ||\mathbf{v}_k||_2 \leq \varepsilon ||\mathbb{A}_k||_F$$

- Optimal computation of the Frobenius norm

$$||\mathbf{A}_k||_F^2 = ||\mathbf{A}_{k-1}||_F^2 + 2\sum_{\ell=1}^{k-1} \mathbf{u}_k^T \mathbf{u}_\ell \mathbf{v}_\ell^T \mathbf{v}_k + ||\mathbf{u}_k||_2^2 ||\mathbf{v}_k||_2^2$$

Illustration with a rank 2 matrix

$$\mathbb{A} = \begin{bmatrix} 6.5 & 31 & -14 & -43 \\ 9.1 & -3 & 11 & 31 \\ 17.6 & -16 & 28 & 80 \\ 26.2 & 50 & -7 & -26 \end{bmatrix}$$

First iteration, we set $i^\star = 1$, $\mathscr{P}_r = \{1\}$

$$\mathbb{R}_0 = \begin{bmatrix} 6.5 & 31 & -14 & \mathbf{-43} \\ 9.1 & -3 & 11 & 31 \\ 17.6 & -16 & 28 & 80 \\ 26.2 & 50 & -7 & -26 \end{bmatrix}$$

Illustration with a rank 2 matrix

$$\mathbb{A} = \begin{bmatrix} 6.5 & 31 & -14 & -43 \\ 9.1 & -3 & 11 & 31 \\ 17.6 & -16 & 28 & 80 \\ 26.2 & 50 & -7 & -26 \end{bmatrix}$$

First iteration, we set $i^\star = 1$, $\mathscr{P}_r = \{1\}$ and find $j^\star = 4$, $\mathscr{P}_c = \{4\}$

$$\mathbb{R}_0 = \begin{bmatrix} 6.5 & 31 & -14 & \mathbf{-43} \\ 9.1 & -3 & 11 & 31 \\ 17.6 & -16 & 28 & 80 \\ 26.2 & 50 & -7 & -26 \end{bmatrix}$$

Illustration with a rank 2 matrix

$$\mathbb{A} = \begin{bmatrix} 6.5 & 31 & -14 & -43 \\ 9.1 & -3 & 11 & 31 \\ 17.6 & -16 & 28 & 80 \\ 26.2 & 50 & -7 & -26 \end{bmatrix}$$

First iteration, we set $i^\star = 1$, $\mathscr{P}_r = \{1\}$ and find $j^\star = 4$, $\mathscr{P}_c = \{4\}$

$$\mathbb{R}_0 = \begin{bmatrix} 6.5 & 31 & -14 & \mathbf{-43} \\ 9.1 & -3 & 11 & 31 \\ 17.6 & -16 & 28 & 80 \\ 26.2 & 50 & -7 & -26 \end{bmatrix}$$

We have $u_1 = [1 \ -0.7209 \ -1.8605 \ 0.6047]^T$ $\quad v_1 = [6.5 \ 31 \ -14 \ -43]$

Next pivot is $i^\star = 3$ ($i^\star = 1$ already used)

Illustration with a rank 2 matrix

$$\mathbb{A} = \begin{bmatrix} 6.5 & 31 & -14 & -43 \\ 9.1 & -3 & 11 & 31 \\ 17.6 & -16 & 28 & 80 \\ 26.2 & 50 & -7 & -26 \end{bmatrix}$$

First iteration, we set $i^\star = 1$, $\mathscr{P}_r = \{1\}$ and find $j^\star = 4$, $\mathscr{P}_c = \{4\}$

$$\mathbb{R}_0 = \begin{bmatrix} 6.5 & 31 & -14 & \mathbf{-43} \\ 9.1 & -3 & 11 & 31 \\ 17.6 & -16 & 28 & 80 \\ 26.2 & 50 & -7 & -26 \end{bmatrix}$$

We have $u_1 = [1 \; -0.7209 \; -1.8605 \; 0.6047]^T$ $\quad v_1 = [6.5 \; 31 \; -14 \; -43]$

Next pivot is $i^\star = 3$ ($i^\star = 1$ already used)

If we compute the residual (not performed in practice):

$$\mathbb{R}_1 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 13.7860 & 19.3488 & 0.9070 & 0 \\ 29.6930 & 41.6744 & 1.9535 & 0 \\ 22.2698 & 31.25581 & 1.4651 & 0 \end{bmatrix}$$

$||\mathbb{A}_1||_F = 127.6636$, $||v_1||_2||u_1||_2$=127.6636, convergence not achieved

Illustration with a rank 2 matrix

$$\mathbb{A} = \begin{bmatrix} 6.5 & 31 & -14 & -43 \\ 9.1 & -3 & 11 & 31 \\ 17.6 & -16 & 28 & 80 \\ 26.2 & 50 & -7 & -26 \end{bmatrix} \quad \mathbb{R}_1 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 13.7860 & 19.3488 & 0.9070 & 0 \\ 29.6930 & 41.6744 & 1.9535 & 0 \\ 22.2698 & 31.25581 & 1.4651 & 0 \end{bmatrix}$$

$\mathscr{P}_r = \{1,\ 3\}$. New row $\mathbb{R}_1$: $[29.693\ 41.6744\ 1.9536\ 0]$

Illustration with a rank 2 matrix

$$\mathbb{A} = \begin{bmatrix} 6.5 & 31 & -14 & -43 \\ 9.1 & -3 & 11 & 31 \\ 17.6 & -16 & 28 & 80 \\ 26.2 & 50 & -7 & -26 \end{bmatrix} \quad \mathbb{R}_1 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 13.7860 & 19.3488 & 0.9070 & 0 \\ 29.6930 & \mathbf{41.6744} & 1.9535 & 0 \\ 22.2698 & 31.25581 & 1.4651 & 0 \end{bmatrix}$$

$\mathscr{P}_r = \{1,\ 3\}$. New row $\mathbb{R}_1$: $[29.693\ 41.6744\ 1.9536\ 0]$

We find $j^\star = 2$, $\mathscr{P}_c = \{4,\ 2\}$. New column $[0\ 19.3488\ 41.6744\ 31.2558]^T$

Illustration with a rank 2 matrix

$$\mathbb{A} = \begin{bmatrix} 6.5 & 31 & -14 & -43 \\ 9.1 & -3 & 11 & 31 \\ 17.6 & -16 & 28 & 80 \\ 26.2 & 50 & -7 & -26 \end{bmatrix} \quad \mathbb{R}_1 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 13.7860 & 19.3488 & 0.9070 & 0 \\ 29.6930 & \mathbf{41.6744} & 1.9535 & 0 \\ 22.2698 & 31.25581 & 1.4651 & 0 \end{bmatrix}$$

$\mathscr{P}_r = \{1,\ 3\}$. New row $\mathbb{R}_1$: $[29.693\ 41.6744\ 1.9536\ 0]$

We find $j^\star = 2$, $\mathscr{P}_c = \{4,\ 2\}$. New column $[0\ 19.3488\ 41.6744\ 31.2558]^T$

$u_2 = [0\ 0.4643\ 1\ 0.75]^T$ and $v_2 = [29.6930\ 41.6744\ 1.9535\ 0]$

Next pivot is $i^\star = 4$ ($i^\star = 1$ or 3 already used)

Illustration with a rank 2 matrix

$$\mathbb{A} = \begin{bmatrix} 6.5 & 31 & -14 & -43 \\ 9.1 & -3 & 11 & 31 \\ 17.6 & -16 & 28 & 80 \\ 26.2 & 50 & -7 & -26 \end{bmatrix} \quad \mathbb{R}_1 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 13.7860 & 19.3488 & 0.9070 & 0 \\ 29.6930 & \mathbf{41.6744} & 1.9535 & 0 \\ 22.2698 & 31.25581 & 1.4651 & 0 \end{bmatrix}$$

$\mathscr{P}_r = \{1,\ 3\}$. New row $\mathbb{R}_1$: $[29.693\ 41.6744\ 1.9536\ 0]$

We find $j^\star = 2$, $\mathscr{P}_c = \{4,\ 2\}$. New column $[0\ 19.3488\ 41.6744\ 31.2558]^T$

$u_2 = [0\ 0.4643\ 1\ 0.75]^T$ and $v_2 = [29.6930\ 41.6744\ 1.9535\ 0]$

Next pivot is $i^\star = 4$ ($i^\star = 1$ or 3 already used)

$||\mathbb{A}_2||_F = 113.7962$ et $||v_2||_2||u_2||_2$=68.2826

$$\mathbb{R}_2 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

End of the algorithm: We cannot find a new pivot