Randomized methods for finding interpolative decompositions of very large matrices

Gunnar Martinsson
The University of Colorado at Boulder

Students:

Adrianna Gillman (now at Dartmouth)

Nathan Halko

Sijia Hao

Patrick Young (now at GeoEye, Inc.)

Collaborators:

Edo Liberty (Yahoo research)

Vladimir Rokhlin (Yale)

Yoel Shkolnisky (Tel Aviv University)

Joel Tropp (Caltech)

Mark Tygert (Courant)

Franco Woolfe (Goldman Sachs)

Outline:

1. Brief review of the "Nyström method" for approximation of low-rank matrices.

2. Some comments on rank-revealing QR factorizations.

3. Constructing Nyström factorizations via randomized sampling.

4. Approximating structured matrices ($\mathcal{H}$-matrices, HSS-matrices, etc).

Notation:

$\mathbf{A}$ is a matrix of size $m \times n$. Think of m and n as large.

k is the (numerical) rank of $\mathbf{A}$. Think $k \ll \min(m, n)$.

Nyström approximation: A basic identity

Let $\mathbf{A}$ be an $m \times n$ matrix of rank k. Let $\mathbf{A}_{11}$ denote the leading $k \times k$ block:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix}$$

Lemma: If $\mathbf{A}_{11}$ is non-singular, then $\mathbf{A}_{22} = \mathbf{A}_{21}\mathbf{A}_{11}^{-1}\mathbf{A}_{12}$.

As a consequence, we obtain a factorization

$$\begin{array}{ccccc} \mathbf{A} & = & \mathbf{C} & & \mathbf{X} \\ m \times n & & m \times k & & k \times n \end{array}$$

where $\mathbf{C} = \begin{bmatrix} \mathbf{A}_{11} \\ \mathbf{A}_{21} \end{bmatrix}$ consists of the first k columns of $\mathbf{A}$, and

$$\mathbf{X} = [\mathbf{I} \quad \mathbf{A}_{11}^{-1}\mathbf{A}_{12}].$$

Nyström approximation: Cheap SVD

Recall: $\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix}$ has rank k and the $k \times k$ matrix $\mathbf{A}_{11}$ is non-singular.

Set $\mathbf{C} = \begin{bmatrix} \mathbf{A}_{11} \\ \mathbf{A}_{21} \end{bmatrix}$ and $\mathbf{X} = [\mathbf{I}\ \mathbf{A}_{11}^{-1}\mathbf{A}_{12}]$. Then $\mathbf{A} = \mathbf{CX}$.

The SVD of $\mathbf{A}$ can now be computed via four operations:

$$\underset{m \times n}{\mathbf{A}} = \overbrace{\underset{m \times k}{\mathbf{C}}}^{=\mathbf{QR}} \underset{k \times n}{\mathbf{X}} = \mathbf{Q}\overbrace{\mathbf{RX}}^{=\hat{\mathbf{U}}\mathbf{\Sigma}\mathbf{V}^*} = \overbrace{\mathbf{Q}\hat{\mathbf{U}}}^{=\mathbf{U}}\mathbf{\Sigma}\mathbf{V}^* = \underset{m \times k}{\mathbf{U}}\ \underset{k \times k}{\mathbf{\Sigma}}\ \underset{k \times n}{\mathbf{V}^*}$$

Note: Each step involves matrices with at most k rows/columns.
The total cost is $O(k^2(m+n))$.

Note: Forming $\mathbf{RX}$ is cheap since $\mathbf{RX} = \mathbf{R}[\mathbf{I} \quad \mathbf{A}_{11}^{-1}\mathbf{A}_{12}] = [\mathbf{R} \quad \mathbf{Q}_1^{-1}\mathbf{A}_{12}]$.

Nyström approximation: Complications ...

Recall: $\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix}$ has rank k and the $k \times k$ matrix $\mathbf{A}_{11}$ is non-singular.

Complication 1: $\mathbf{A}_{11}$ could be ill-conditioned (or even singular). Then computing $\mathbf{A}_{11}^{-1}$ *is* problematic, and computing $\mathbf{A}_{11}^{-1}\mathbf{A}_{12}$ or $\mathbf{A}_{21}\mathbf{A}_{11}^{-1}$ *could be* problematic.

This is *in principle* easy to fix via pivoting.

Remedy: Find index vectors $I_1 \subset \mathbb{Z}_m^k$ and $J_1 \subset \mathbb{Z}_n^k$ such that

$$\det(\mathbf{A}(I_1, J_1))$$

is maximized. Set $I = [I_1, \mathbb{Z}_m \backslash I_1]$ and $J = [J_1, \mathbb{Z}_n \backslash J_1]$, partition

$$\mathbf{A}(I, J) = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix},$$

and proceed as before. Then all elements of the matrices $\mathbf{A}_{11}^{-1}\mathbf{A}_{12}$ or $\mathbf{A}_{21}\mathbf{A}_{11}^{-1}$ are bounded in modulus by one. (A direct consequence of Cramer's rule.)

Caveat: Solving the maximization problem is combinatorially hard.

Nyström approximation: Complications ...

Recall: $\mathbf{A}(I,J) = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix}$ has rank k and the $k \times k$ matrix $\mathbf{A}_{11}$ is non-singular.

Complication 2: The rank of $\mathbf{A}$ is typically not *precisely* k.

Example: Suppose ϵ is a small number and consider the $2k \times 2k$ matrix

$$\mathbf{A} = \begin{bmatrix} \mathbf{I}_k & \mathbf{0} \\ \mathbf{0} & \epsilon \mathbf{I}_k \end{bmatrix}$$

With $\mathbf{C} = \begin{bmatrix} \mathbf{I}_k \\ \mathbf{0} \end{bmatrix}$, the Nyström approximant is $\mathbf{A}^{(\text{nystrom})} = \mathbf{C}\mathbf{C}^{\dagger}\mathbf{A} = \begin{bmatrix} \mathbf{I}_k & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}$. Good.

With $\mathbf{C} = \begin{bmatrix} \mathbf{0} \\ \epsilon \mathbf{I}_k \end{bmatrix}$, the Nyström approximant is $\mathbf{A}^{(\text{nystrom})} = \mathbf{C}\mathbf{C}^{\dagger}\mathbf{A} = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \epsilon \mathbf{I}_k \end{bmatrix}$. Bad.

Nyström approximation: Complications ...

Recall: $\mathbf{A}(I,J) = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix}$ has rank k and the $k \times k$ matrix $\mathbf{A}_{11}$ is non-singular.

Complication 2: The rank of $\mathbf{A}$ is typically not *precisely* k.

Remedy: Find index vectors $I_1 \subset \mathbb{Z}_m^k$ and $J_1 \subset \mathbb{Z}_n^k$ such that

$$\det(\mathbf{A}(I_1, J_1))$$

is maximized.

Nyström approximation: Finding spanning rows/columns can be intrinsically valuable

Recall: $\mathbf{A}(I,J) = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix}$ has rank k and the $k \times k$ matrix $\mathbf{A}_{11}$ is non-singular.

Data interpretation: Collect statistical data in a large matrix. By finding a set of spanning columns/rows, you can identify some variables that "explain" the data. (Say a small collection of genes among a set of recorded genomes, or a small number of stocks in a portfolio.)

Preserve data structure: For example, if $\mathbf{A}$ is sparse, then the large factors in a Nyström decomposition are also sparse.

Nyström approximation: Different factorizations

Recall: $\mathbf{A}(I,J) = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix}$ has rank k and the $k \times k$ matrix $\mathbf{A}_{11}$ is non-singular.

Set $\mathbf{C} = \begin{bmatrix} \mathbf{A}_{11} \\ \mathbf{A}_{21} \end{bmatrix}$ and $\mathbf{R} = [\mathbf{A}_{11}\ \mathbf{A}_{12}]$. Then (with factors in red submatrices of $\mathbf{A}$):

- $\mathbf{A}(I,J) = \mathbf{CX}$ with $\mathbf{X} = [\mathbf{I} \quad \mathbf{A}_{11}^{-1}\mathbf{A}_{12}]$
- $\mathbf{A}(I,J) = \mathbf{YR}$ with $\mathbf{Y} = \begin{bmatrix} \mathbf{I} \\ \mathbf{A}_{21}\mathbf{A}_{11}^{-1} \end{bmatrix}$
- $\mathbf{A}(I,J) = \mathbf{YA}_{11}\mathbf{X}$ with $\mathbf{X} = [\mathbf{I} \quad \mathbf{A}_{11}^{-1}\mathbf{A}_{12}]$ and $\mathbf{Y} = \begin{bmatrix} \mathbf{I} \\ \mathbf{A}_{21}\mathbf{A}_{11}^{-1} \end{bmatrix}$
- $\mathbf{A}(I,J) = \mathbf{CUR}$ with $\mathbf{U} = \mathbf{A}_{11}^{-1}$.

Note: All factorizations are stable if I and J are chosen properly; except CUR.

Nyström approximation: Summary

Let $\mathbf{A}$ be an $m \times n$ matrix, let k be a specified rank, let $J_1 \subset \mathbb{Z}_n^k$ be an index vector, let $\mathbf{C}$ denote the corresponding collection of columns

$$\mathbf{C} = \mathbf{A}(:, J_1)$$

and define the corresponding *Nyström approximant* via

$$\mathbf{A}^{(\text{nystrom})} = \mathbf{C}\mathbf{C}^{\dagger}\mathbf{A}.$$

Question: What is the minimal error $||\mathbf{A} - \mathbf{A}^{(\text{nystrom})}||$?

Question: How do you find a "good" index vector J_1?

Recall that the singular values $\{\sigma_j\}_{j=1}^{\min(m,n)}$ of $\mathbf{A}$ satisfy

$$\sigma_{j+1} = \min\{||\mathbf{A} - \mathbf{B}|| : \mathbf{B} \text{ has rank } j\}.$$

Since the Nyström minimization problem is further constrained, we must have

$$||\mathbf{A} - \mathbf{A}^{(\text{nystrom})}|| \geq \sigma_{k+1}.$$

The standard framework for analyzing questions of this type is in terms of *rank-revealing QR factorizations* (RRQR).

$$\mathbf{A}(:, J) = \mathbf{QR} = [\mathbf{Q}_1 \mid \mathbf{Q}_2] \left[\begin{array}{c|c} \mathbf{R}_{11} & \mathbf{R}_{12} \\ \hline \mathbf{0} & \mathbf{R}_{22} \end{array}\right]$$

Desirable properties:

- The singular values of $\mathbf{R}_{11}$ should approximate the largest singular values of $\mathbf{A}$.
  $\sigma_j(\mathbf{R}_{11}) \approx \sigma_j(\mathbf{A})$ for $1 \leq j \leq k$.

- The "mass" in $\mathbf{R}_{22}$ is the Nyström error and should be as small as possible.
  $||\mathbf{R}_{22}|| \approx \sigma_{k+1}(\mathbf{A})$.

- The columns in $\mathbf{Q}_1\mathbf{R}_{11} = \mathbf{A}(:, J(1:k))$ should form a good basis for ran($\mathbf{A}$).
  $\mathbf{R}_{11}^{-1}\mathbf{R}_{12}$ should have elements of moderate size.
  (Note that $\mathbf{A}_{11}^{-1}\mathbf{A}_{12} = \mathbf{R}_{11}^{-1}\mathbf{R}_{12}$.)

The standard framework for analyzing questions of this type is in terms of *rank-revealing QR factorizations* (RRQR).

$$\mathbf{A}(:, J) = \mathbf{QR} = [\mathbf{Q}_1 \mid \mathbf{Q}_2] \left[\begin{array}{c|c} \mathbf{R}_{11} & \mathbf{R}_{12} \\ \hline \mathbf{0} & \mathbf{R}_{22} \end{array}\right]$$

Desirable properties:

- The singular values of $\mathbf{R}_{11}$ should approximate the largest singular values of $\mathbf{A}$.
  $\sigma_j(\mathbf{R}_{11}) \approx \sigma_j(\mathbf{A})$ for $1 \leq j \leq k$.
  $\sigma_j(\mathbf{R}_{11}) \leq \sigma_j(\mathbf{A})$ *holds automatically for any* j *and* J.

- The "mass" in $\mathbf{R}_{22}$ should be as small as possible.
  $||\mathbf{R}_{22}|| \approx \sigma_{k+1}(\mathbf{A})$.
  $\sigma_j(\mathbf{R}_{22}) \geq \sigma_{k+j}(\mathbf{A})$ *holds automatically for any* j *and* J.

- The columns in $\mathbf{Q}_1\mathbf{R}_{11} = \mathbf{A}(:, J(1:k))$ should form a good basis for $\text{ran}(\mathbf{A})$.
  $\mathbf{R}_{11}^{-1}\mathbf{R}_{12}$ should have elements of moderate size.
  (Note that $\mathbf{A}_{11}^{-1}\mathbf{A}_{12} = \mathbf{R}_{11}^{-1}\mathbf{R}_{12}$.)

The standard framework for analyzing questions of this type is in terms of *rank-revealing QR factorizations* (RRQR).

$$\mathbf{A}(:, J) = \mathbf{QR} = [\mathbf{Q}_1 \mid \mathbf{Q}_2] \left[\begin{array}{c|c} \mathbf{R}_{11} & \mathbf{R}_{12} \\ \hline \mathbf{0} & \mathbf{R}_{22} \end{array}\right]$$

Desirable properties:

- The singular values of $\mathbf{R}_{11}$ should approximate the largest singular values of $\mathbf{A}$.
  $\sigma_j(\mathbf{R}_{11}) \approx \sigma_j(\mathbf{A})$ for $1 \leq j \leq k$.
  $\sigma_j(\mathbf{R}_{11}) \leq \sigma_j(\mathbf{A})$ *holds automatically for any* j *and* J.
  We seek J *such that* $\sigma_j(\mathbf{R}_{11}) \geq \frac{1}{p(k,n)}\sigma_j(\mathbf{A})$ *for some modest* $p(k, n)$.
- The "mass" in $\mathbf{R}_{22}$ should be as small as possible.
  $||\mathbf{R}_{22}|| \approx \sigma_{k+1}(\mathbf{A})$.
  $\sigma_j(\mathbf{R}_{22}) \geq \sigma_{k+j}(\mathbf{A})$ *holds automatically for any* j *and* J.
  We seek J *such that* $||\mathbf{R}_{22}|| \leq p(k, n)\, \sigma_{k+1}(\mathbf{A})$ *for some modest* $p(k, n)$.
- The columns in $\mathbf{Q}_1\mathbf{R}_{11} = \mathbf{A}(:, J(1 : k))$ should form a good basis for ran($\mathbf{A}$).
  $\mathbf{R}_{11}^{-1}\mathbf{R}_{12}$ should have elements of moderate size.
  (Note that $\mathbf{A}_{11}^{-1}\mathbf{A}_{12} = \mathbf{R}_{11}^{-1}\mathbf{R}_{12}$.)

$$\mathbf{A}(:, J) = \mathbf{QR} = [\mathbf{Q}_1 \mid \mathbf{Q}_2] \left[\begin{array}{c|c} \mathbf{R}_{11} & \mathbf{R}_{12} \\ \hline \mathbf{0} & \mathbf{R}_{22} \end{array}\right]$$

The following bounds *can* be achieved:

$$\begin{aligned} \frac{1}{\sqrt{1+k(n-k)}}\sigma_j(\mathbf{A}) \leq & \sigma_j(\mathbf{R}_{11}) \leq \sigma_j(\mathbf{A}) \\ ||\mathbf{R}_{22}|| \leq & \sqrt{1+k(n-k)}\sigma_{k+1}(\mathbf{A}) \\ |[\mathbf{R}_{11}^{-1}\mathbf{R}_{12}](i,j)| \leq & 1. \end{aligned}$$

Column-pivoted Gram-Schmidt factorization does not necessarily get close to the above. (In practice, it usually does, however.) Famous counter-example by Kahan.

A more sophisticated algorithm by Eisenstat and Gu guarantees bounds close to the achievable (optimal?) bounds reported above. It can require $O(mn^2)$ run time, but "typically" executes faster.

Question: In practical applications, do you really need an RRQR?

The set-up for RRQR is that we are given a rank k, and seek something close to the very best rank-k Nyström approximant.

Relaxed formulation suitable for many applications:
Given a matrix $\mathbf{A}$, and a tolerance ε, find an index vector J_1 such that

$$||\mathbf{A} - \mathbf{A}(:, J_1)\,\mathbf{X}|| \leq \varepsilon$$

where $\mathbf{X}$ is some matrix whose entries are "small," and the length of J_1 is "comparable" to the ε-rank of $\mathbf{A}$.

Observations:

- Overshooting the rank a bit is typically fine.
  (With k denoting the ε-rank, $\#J_1 = k + 10$ or even $\#J_1 = 2k$ is often OK.)
- If you overshoot in the first run, postprocessing (such as computing an SVD) will reveal the true rank up to precision ε.

Question: In practical applications, do you really need an RRQR?

The set-up for RRQR is that we are given a rank k, and seek something close to the very best rank-k Nyström approximant.

Relaxed formulation suitable for many applications:
Given a matrix $\mathbf{A}$, and a tolerance ε, find an index vector J_1 such that

$$||\mathbf{A} - \mathbf{A}(:, J_1)\,\mathbf{X}|| \leq \varepsilon$$

where $\mathbf{X}$ is some matrix whose entries are "small," and the length of J_1 is "comparable" to the ε-rank of $\mathbf{A}$.

Solution for small and moderate size problems:
Column pivoted Gram-Schmidt works well most of the time.
(Enforcing orthogonality religiously is crucial, however.)

Question: In practical applications, do you really need an RRQR?

The set-up for RRQR is that we are given a rank k, and seek something close to the very best rank-k Nyström approximant.

Relaxed formulation suitable for many applications:
Given a matrix $\mathbf{A}$, and a tolerance ε, find an index vector J_1 such that

$$||\mathbf{A} - \mathbf{A}(:, J_1)\,\mathbf{X}|| \leq \varepsilon$$

where $\mathbf{X}$ is some matrix whose entries are "small," and the length of J_1 is "comparable" to the ε-rank of $\mathbf{A}$.

Solution for large size problems:

1. Find (by any means) matrices $\mathbf{E}$ and $\mathbf{F}$ such that

$$||\mathbf{A} - \mathbf{EF}|| \leq \frac{1}{\sqrt{1 + 4k(n-k)}}\varepsilon.$$

2. Find a vector J_1 and a matrix $\mathbf{X}$ such that

$$\mathbf{F} = \mathbf{F}(:, J_1)\mathbf{X}.$$

3. Do nothing! Your J_1 and $\mathbf{X}$ will automatically work for $\mathbf{A}$.

Lemma: Suppose that

$$\mathbf{A} = \mathbf{EF}$$

and that

(1) $$\mathbf{F} = \mathbf{F}(:, J_1)\mathbf{X}.$$

Then

$$\mathbf{A} = \mathbf{A}(:, J_1)\mathbf{X}.$$

Proof: Multiply (1) by $\mathbf{E}$ from the left:

$$\underbrace{\mathbf{EF}}_{=\mathbf{A}} = \underbrace{\mathbf{EF}(:, J_1)}_{=\mathbf{A}(:,J_1)}\mathbf{X}.$$

Lemma: Suppose that

(2) $$\mathbf{F} = \mathbf{F}(:, J_1)\mathbf{X}.$$

and that

$$||\mathbf{A} - \mathbf{EF}|| \leq \frac{1}{1 + ||\mathbf{X}||}\,\varepsilon.$$

Then

$$||\mathbf{A} - \mathbf{A}(:, J_1)\mathbf{X}|| \leq \varepsilon.$$

Proof: Set

$$\mathbf{B} = \mathbf{EF}.$$

Then

$$\begin{aligned} ||\mathbf{A} - \mathbf{A}(:, J_1)\mathbf{X}|| &\leq ||\mathbf{A} - \underbrace{\mathbf{B}(:, J_1)\mathbf{X}}_{=\mathbf{B}}|| + ||\mathbf{B}(:, J_1)\mathbf{X} - \mathbf{A}(:, J_1)\mathbf{X}|| \\ &\leq ||\mathbf{A} - \mathbf{B}|| + ||\mathbf{B}(:, J_1) - \mathbf{A}(:, J_1)||\,||\mathbf{X}|| \leq \frac{1}{1 + ||\mathbf{X}||}\varepsilon + \frac{1}{1 + ||\mathbf{X}||}\varepsilon||\mathbf{X}|| = \varepsilon \end{aligned}$$

Note: If $\mathbf{X}$ is constructed via RRQR then $||\mathbf{X}|| \leq \sqrt{1 + 4k(n-k)}$.

Note: The loss of accuracy is clearly visible in most applications.

Recall: An algorithm for finding $\mathbf{X}$ and J_1 such that $\mathbf{A} \approx \mathbf{A}(:, J_1)\mathbf{X}$:

1. Find (by any means) matrices $\mathbf{E}$ and $\mathbf{F}$ such that $\mathbf{A} \approx \mathbf{EF}$.
2. Find a vector J_1 and a matrix $\mathbf{X}$ such that $\mathbf{F} = \mathbf{F}(:, J_1)\mathbf{X}$.
3. Do nothing! Your J_1 and $\mathbf{X}$ will automatically work for $\mathbf{A}$.

Observation: The matrix $\mathbf{E}$ is not *used*. It only needs to exist.
All we need is a collection of vectors that span the row space of $\mathbf{A}$.

This problem is ideally suited for randomized sampling!

A very cheap but sometimes unreliable randomized algorithm:

Objective: Given $\mathbf{A}$, find $\mathbf{X}$ and J_1 such that $\mathbf{A} \approx \mathbf{A}(:, J_1)\mathbf{X}$:

Algorithm:

1. Find a matrix $\mathbf{F}$ such that $\mathbf{A} \approx \mathbf{E}\mathbf{F}$ for some matrix $\mathbf{E}$.
   (a) Form $\mathbf{F}$ by drawing ℓ rows of $\mathbf{A}$ "at random."
2. Find via RRQR a vector J_1 and a matrix $\mathbf{X}$ such that $\mathbf{F} = \mathbf{F}(:, J_1)\mathbf{X}$.
3. Do nothing. Your J_1 and $\mathbf{X}$ will automatically work for $\mathbf{A}$.

Procedures of this type can work well for specific classes of matrices.

However, for a general matrix $\mathbf{A}$, you cannot be assured of good performance.
The number of rows needed to attain precision ε can vastly exceed the ε-rank of $\mathbf{A}$.

A reliable randomized algorithm:

Objective: Given $\mathbf{A}$, find $\mathbf{X}$ and J_1 such that $\mathbf{A} \approx \mathbf{A}(:, J_1)\mathbf{X}$:

Algorithm:

1. Find a matrix $\mathbf{F}$ such that $\mathbf{A} \approx \mathbf{E}\mathbf{F}$ for some matrix $\mathbf{E}$.
   (a) Draw a Gaussian matrix $\mathbf{\Omega}$ of size $\ell \times m$. (Think $\ell = k + 10$ or $\ell = 2k$.)
   (b) Form a sample matrix $\mathbf{F} = \mathbf{\Omega}\mathbf{A}$.
2. Find via RRQR a vector J_1 and a matrix $\mathbf{X}$ such that $\mathbf{F} = \mathbf{F}(:, J_1)\mathbf{X}$.
3. Do nothing. Your J_1 and $\mathbf{X}$ will automatically work for $\mathbf{A}$.

Cost: ℓ matvecs. Dense operations on matrices with ℓ columns or rows.

Some comments on errors (details later):

- Tight theory exists (see survey paper in June 2011 issue of SIAM Review).
- If the singular values of $\mathbf{A}$ decay rapidly, the errors are close to minimal.
- Cheap error estimators can be implemented for the "given precision" case.

A reliable and highly accurate randomized algorithm:

Objective: Given $\mathbf{A}$, find $\mathbf{X}$ and J_1 such that $\mathbf{A} \approx \mathbf{A}(:, J_1)\mathbf{X}$:

Algorithm:

1. Find a matrix $\mathbf{F}$ such that $\mathbf{A} \approx \mathbf{E}\mathbf{F}$ for some matrix $\mathbf{E}$.
   (a) Draw a Gaussian matrix $\mathbf{\Omega}$ of size $\ell \times m$. (Think $\ell = k + 10$ or $\ell = 2k$.)
   (b) Form a sample matrix $\mathbf{F} = \mathbf{\Omega}\mathbf{A}(\mathbf{A}^*\mathbf{A})^q$ where q is a small integer.
   Note: Rounding errors can derail the procedure. Remedies exist.
2. Find via RRQR a vector J_1 and a matrix $\mathbf{X}$ such that $\mathbf{F} = \mathbf{F}(:, J_1)\mathbf{X}$.
3. Do nothing. Your J_1 and $\mathbf{X}$ will automatically work for $\mathbf{A}$.

Cost: $(2q + 1)\ell$ matrix-vector multiplications. Dense operations on matrices with ℓ columns or rows.

Errors: Tight theory exists. The errors can be made arbitrarily close to optimal. Error estimators can be deployed.

A reliable and fast randomized algorithm:

Objective: Given $\mathbf{A}$, find $\mathbf{X}$ and J_1 such that $\mathbf{A} \approx \mathbf{A}(:, J_1)\mathbf{X}$:

Algorithm:

1. Find a matrix $\mathbf{F}$ such that $\mathbf{A} \approx \mathbf{E}\mathbf{F}$ for some matrix $\mathbf{E}$.
   (a) Draw an "SRFT" matrix $\mathbf{\Omega}$ of size $\ell \times m$. (Think $\ell = 2k$.)
   (b) Form a sample matrix $\mathbf{F} = \mathbf{\Omega}\mathbf{A}$.
2. Find via RRQR a vector J_1 and a matrix $\mathbf{X}$ such that $\mathbf{F} = \mathbf{F}(:, J_1)\mathbf{X}$.
3. Do nothing. Your J_1 and $\mathbf{X}$ will automatically work for $\mathbf{A}$.

Cost: $O(mn \log(\ell))$!

Errors: Errors are "typically" similar to the Gaussian case but can in principle be much worse. Adaptive error estimation is a little dicier.

Question: What is the "SRFT" matrix on the previous slide?

SRFT stands for *subsampled random Fourier Transform:*

$$\begin{array}{ccccc} \mathbf{\Omega} & = & \mathbf{S} & \mathbf{F} & \mathbf{D} \\ \ell \times m & & \ell \times m & m \times m & m \times m \end{array}$$

where,

- $\mathbf{D}$ is a diagonal matrix whose entries are i.i.d. random variables drawn from a uniform distribution on the unit circle in $\mathbb{C}$.
- $\mathbf{F}$ is the discrete Fourier transform, $\mathbf{F}_{jk} = \frac{1}{\sqrt{m}} e^{-2\pi i (j-1)(k-1)/m}$.
- $\mathbf{S}$ is a matrix whose entries are all zeros except for a single, randomly placed 1 in each row. (In other words, the action of $\mathbf{S}$ is to draw ℓ rows at random from $\mathbf{D}\,\mathbf{F}$.)

References: Ailon and Chazelle (2006); Liberty, Rokhlin, Tygert, and Woolfe (2006).

The algorithms presented are supported by rigorous theory. For instance:

Theorem: *[Halko, Martinsson, Tropp 2009] Fix a real $m \times n$ matrix $\mathbf{A}$ with singular values $\sigma_1, \sigma_2, \sigma_3, \ldots$. Choose integers $k \geq 1$ and $p \geq 2$, and draw a $(k+p) \times m$ standard Gaussian random matrix $\mathbf{\Omega}$. Construct the sample matrix $\mathbf{F} = \mathbf{\Omega A}$. Then*

$$\mathbb{E}||\mathbf{A} - \mathbf{A}\mathbf{F}^{\dagger}\mathbf{F}||_{\text{Frob}} \leq \left(1 + \frac{k}{p-1}\right)^{1/2} \left(\sum_{j=k+1}^{\min(m,n)} \sigma_j^2\right)^{1/2}.$$

Moreover,

$$\mathbb{E}||\mathbf{A} - \mathbf{A}\mathbf{F}^{\dagger}\mathbf{F}|| \leq \left(1 + \sqrt{\frac{k}{p-1}}\right) \sigma_{k+1} + \frac{e\sqrt{k+p}}{p} \left(\sum_{j=k+1}^{\min(m,n)} \sigma_j^2\right)^{1/2}.$$

For a proof, see

N. Halko, P.G. Martinsson, J. Tropp, "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions." *SIAM Review*, **53**(2), pp. 217-288, 2011.

Theorem: *[Halko, Martinsson, Tropp 2009] Fix a* real $m \times n$ *matrix* $\mathbf{A}$ *with singular values* $\sigma_1, \sigma_2, \sigma_3, \ldots$. *Choose integers* $k \geq 1$ *and* $p \geq 4$, *and draw an* $(k+p) \times m$ *standard Gaussian random matrix* $\mathbf{\Omega}$. *Construct the sample matrix* $\mathbf{F} = \mathbf{\Omega A}$. *Then for all* $u, t \geq 1$,

$$||\mathbf{A} - \mathbf{A}\mathbf{F}^{\dagger}\mathbf{F}|| \leq \left[\left(1 + t \cdot \sqrt{\frac{3k}{p+1}} \right) \sigma_{k+1} + t \cdot \frac{e\sqrt{k+p}}{p+1} \left(\sum\nolimits_{j>k} \sigma_j^2 \right)^{1/2} \right] + ut \cdot \frac{e\sqrt{k+p}}{p+1} \sigma_{k+1},$$

with failure probability at most $2t^{-p} + e^{-u^2/2}$.

The theorem can be simplified by by choosing t and u appropriately. For instance,

$$||\mathbf{A} - \mathbf{A}\mathbf{F}^{\dagger}\mathbf{F}|| \leq \left[1 + 9\sqrt{k+p} \cdot \sqrt{\min\{m, n\}} \right] \sigma_{k+1}$$

with failure probability at most $3 \cdot p^{-p}$.

Note: There are two sources of error:

(1) Error in randomized sampling:

$$||\mathbf{A} - \mathbf{EF}|| \sim \sqrt{k(m-k)}\,\sigma_{k+1}$$

(2) Error resulting from Nyström approximation:

$$||\mathbf{A} - \mathbf{A}(:, J_1)\mathbf{X}|| \leq \sqrt{1 + 4k(n-k)}\,||\mathbf{A} - \mathbf{EF}||.$$

Combined: $||\mathbf{A} - \mathbf{A}(:, J_1)\mathbf{X}|| \sim k\sqrt{mn}\,\sigma_{k+1}$.

Note: The error estimates above are typically pessimistic and should not be used to guide practical computations. The recommended approach is to construct J_1 and $\mathbf{X}$ via randomized sampling, and then estimate $||\mathbf{A} - \mathbf{A}(:, J_1)\mathbf{X}||$ via another randomized procedure. If the error is too large, then simply add more samples and compute a new set of prospective spanning columns to $\mathbf{C}$. Repeat as necessary.

Note: The extra error from "Nyström approximation" can be avoided if you do not insist on using columns of $\mathbf{A}$ as the basis.

An algorithm for computing the SVD of a matrix (no Nyström):

1. Draw an $\ell \times m$ Gaussian random matrix $\mathbf{\Omega}$.

2. Form an $\ell \times m$ sample matrix $\mathbf{F} = \mathbf{\Omega A}$.

3. Form a QR factorization $\mathbf{F}^* = \mathbf{QR}$.
   (In other words, the columns of $\mathbf{Q}$ form an ON basis for the rows of $\mathbf{F}$).

4. Form the $\ell \times m$ matrix $\mathbf{B} = \mathbf{Q}^*\mathbf{A}$ (so that $\mathbf{A} \approx \mathbf{QQ}^*\mathbf{A} = \mathbf{QB}$).

5. Form the SVD $\mathbf{B} = \hat{\mathbf{U}}\mathbf{\Sigma}\mathbf{V}^*$.

6. Form $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$.

The end result are factors $\mathbf{U}$, $\mathbf{\Sigma}$, $\mathbf{V}$ such that

$$\mathbb{E}||\mathbf{A} - \mathbf{U\Sigma V}^*|| \leq \left(1 + \sqrt{\frac{k}{p-1}}\right)\sigma_{k+1} + \frac{e\sqrt{k+p}}{p}\left(\sum_{j=k+1}^{\min(m,n)} \sigma_j^2\right)^{1/2}.$$

Possible improvements: Use powers of $\mathbf{A}$, use an SRFT instead of a Gaussian, etc.

Variation of the algorithm when $\mathbf{A}$ *is symmetric positive definite (spd):*

Given an orthonormal matrix $\mathbf{Q}$ whose columns form an approximate basis for the range of $\mathbf{A}$, a very useful approximation is:

$$\mathbf{A} \approx (\mathbf{AQ}) \left(\mathbf{Q}^*\mathbf{AQ}\right)^{-1} (\mathbf{AQ})^* = \left[(\mathbf{AQ}) \left(\mathbf{Q}^*\mathbf{AQ}\right)^{-1/2}\right] \left[(\mathbf{AQ}) \left(\mathbf{Q}^*\mathbf{AQ}\right)^{-1/2}\right]^* = \mathbf{FF}^*,$$

Algorithm for computing the approximate Cholesky factor $\mathbf{F}$:

1. Draw a random matrix $\mathbf{\Omega}$ and form $\mathbf{Y} = \mathbf{A\Omega}$.
2. Orthonormalize the columns of $\mathbf{Y}$ to form $\mathbf{Q}$.
3. Form the matrices $\mathbf{B}_1 = \mathbf{AQ}$ and $\mathbf{B}_2 = \mathbf{Q}^*\mathbf{B}_1$.
4. Perform a Cholesky factorization $\mathbf{B}_2 = \mathbf{C}^*\mathbf{C}$.
5. Form $\mathbf{F} = \mathbf{B}_1\mathbf{C}^{-1}$ using a triangular solve.

An optional final step is to compute the SVD $\mathbf{F} = \mathbf{U\Sigma V}^*$. Then $\mathbf{A} \approx \mathbf{U}(\mathbf{\Sigma}^2)\mathbf{U}^*$.

Application to structured matrix computations

So far, we have discussed the situation where a matrix $\mathbf{A}$ has low rank.

Next, we will describe how randomized sampling can be used to approximate a matrix $\mathbf{A}$ whose *off-diagonal blocks* have low rank.

In what follows, we make several assumptions on $\mathbf{A}$:

- $\mathbf{A}$ and $\mathbf{A}^*$ can rapidly be applied to vectors.
- The cost of evaluating an individual entry of $\mathbf{A}$ is small.
- $\mathbf{A}$ has off-diagonal blocks of low (numerical) rank.
  (The precise sense will be specified in the next several slides.)

Motivating example: $\mathbf{A}$ approximates a boundary integral operator.

Block-separable matrices

Consider a matrix $\mathbf{A}$ consisting of $p \times p$ blocks of size $n \times n$:

$$\mathbf{A} = \begin{bmatrix} \mathbf{D}_{11} & \mathbf{A}_{12} & \mathbf{A}_{13} & \mathbf{A}_{14} \\ \mathbf{A}_{21} & \mathbf{D}_{22} & \mathbf{A}_{23} & \mathbf{A}_{24} \\ \mathbf{A}_{31} & \mathbf{A}_{32} & \mathbf{D}_{33} & \mathbf{A}_{34} \\ \mathbf{A}_{41} & \mathbf{A}_{42} & \mathbf{A}_{43} & \mathbf{D}_{44} \end{bmatrix}. \qquad \text{(Shown for } p = 4.\text{)}$$

Core assumption: Each off-diagonal block $\mathbf{A}_{ij}$ admits the factorization

$$\begin{array}{ccccc} \mathbf{A}_{ij} & = & \mathbf{U}_i & \tilde{\mathbf{A}}_{ij} & \mathbf{V}_j^* \\ n \times n & & n \times k & k \times k & k \times n \end{array}$$

where the rank k is significantly smaller than the block size n. (Say $k \approx n/2$.)

The critical part of the assumption is that all off-diagonal blocks in the i'th row use the same basis matrices $\mathbf{U}_i$ for their column spaces (and analogously all blocks in the j'th column use the same basis matrices $\mathbf{V}_j$ for their row spaces).

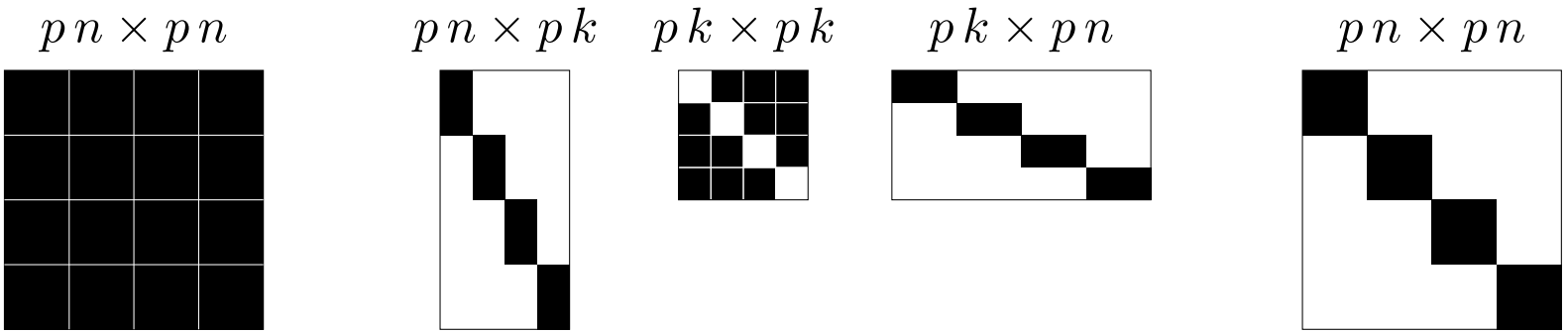
We get $\mathbf{A} = \begin{bmatrix} \mathbf{D}_{11} & \mathbf{U}_1 \tilde{\mathbf{A}}_{12} \mathbf{V}_2^* & \mathbf{U}_1 \tilde{\mathbf{A}}_{13} \mathbf{V}_3^* & \mathbf{U}_1 \tilde{\mathbf{A}}_{14} \mathbf{V}_4^* \\ \mathbf{U}_2 \tilde{\mathbf{A}}_{21} \mathbf{V}_1^* & \mathbf{D}_{22} & \mathbf{U}_2 \tilde{\mathbf{A}}_{23} \mathbf{V}_3^* & \mathbf{U}_2 \tilde{\mathbf{A}}_{24} \mathbf{V}_4^* \\ \mathbf{U}_3 \tilde{\mathbf{A}}_{31} \mathbf{V}_1^* & \mathbf{U}_3 \tilde{\mathbf{A}}_{32} \mathbf{V}_2^* & \mathbf{D}_{33} & \mathbf{U}_3 \tilde{\mathbf{A}}_{34} \mathbf{V}_4^* \\ \mathbf{U}_4 \tilde{\mathbf{A}}_{41} \mathbf{V}_1^* & \mathbf{U}_4 \tilde{\mathbf{A}}_{42} \mathbf{V}_2^* & \mathbf{U}_4 \tilde{\mathbf{A}}_{43} \mathbf{V}_3^* & \mathbf{D}_{44} \end{bmatrix}$.

Then $\mathbf{A}$ admits the factorization:

$$\mathbf{A} = \underbrace{\begin{bmatrix} \mathbf{U}_1 & & & \\ & \mathbf{U}_2 & & \\ & & \mathbf{U}_3 & \\ & & & \mathbf{U}_4 \end{bmatrix}}_{=\mathbf{U}} \underbrace{\begin{bmatrix} \mathbf{0} & \tilde{\mathbf{A}}_{12} & \tilde{\mathbf{A}}_{13} & \tilde{\mathbf{A}}_{14} \\ \tilde{\mathbf{A}}_{21} & \mathbf{0} & \tilde{\mathbf{A}}_{23} & \tilde{\mathbf{A}}_{24} \\ \tilde{\mathbf{A}}_{31} & \tilde{\mathbf{A}}_{32} & \mathbf{0} & \tilde{\mathbf{A}}_{34} \\ \tilde{\mathbf{A}}_{41} & \tilde{\mathbf{A}}_{42} & \tilde{\mathbf{A}}_{43} & \mathbf{0} \end{bmatrix}}_{=\tilde{\mathbf{A}}} \underbrace{\begin{bmatrix} \mathbf{V}_1^* & & & \\ & \mathbf{V}_2^* & & \\ & & \mathbf{V}_3^* & \\ & & & \mathbf{V}_4^* \end{bmatrix}}_{=\mathbf{V}^*} + \underbrace{\begin{bmatrix} \mathbf{D}_1 & & & \\ & \mathbf{D}_2 & & \\ & & \mathbf{D}_3 & \\ & & & \mathbf{D}_4 \end{bmatrix}}_{=\mathbf{D}}$$

or

$$\mathbf{A} = \mathbf{U}\ \tilde{\mathbf{A}}\ \mathbf{V}^* + \mathbf{D},$$

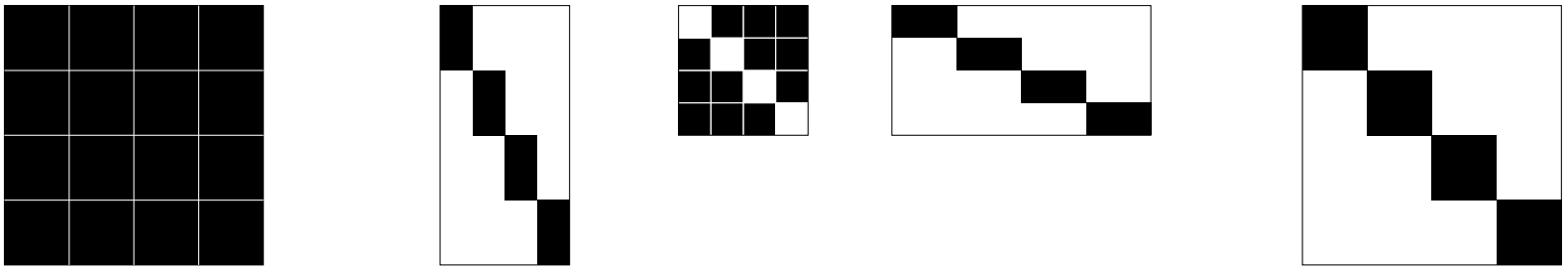


How to perform a fast matrix-vector product is obvious. We can also invert $\mathbf{A}$ ...

Lemma: [Variation of Woodbury] If an $N \times N$ matrix $\mathbf{A}$ admits the factorization

$$\mathbf{A} = \mathbf{U}\ \tilde{\mathbf{A}}\ \mathbf{V}^*\ + \mathbf{D},$$

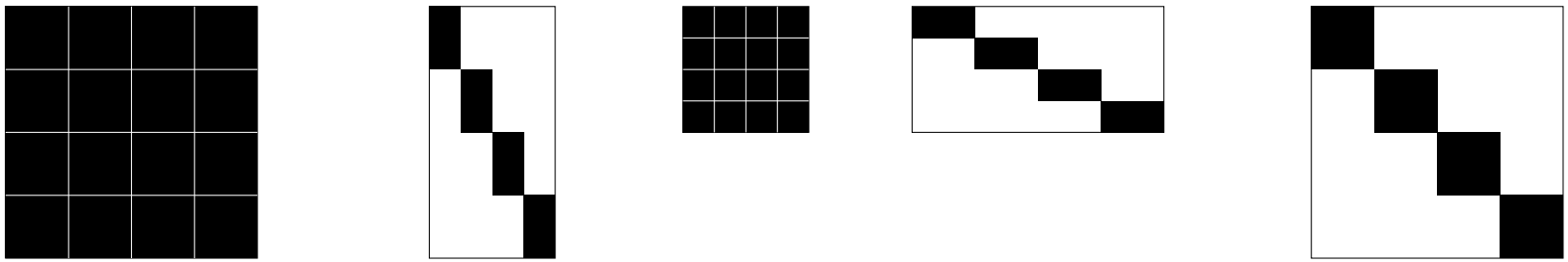
$p\,n \times p\,n$ $\quad$ $p\,n \times p\,k$ $\quad$ $p\,k \times p\,k$ $\quad$ $p\,k \times p\,n$ $\quad$ $p\,n \times p\,n$



then

$$\mathbf{A}^{-1} = \mathbf{E}\ (\tilde{\mathbf{A}} + \hat{\mathbf{D}})^{-1}\ \mathbf{F}^*\ + \mathbf{G},$$

$p\,n \times p\,n$ $\quad$ $p\,n \times p\,k$ $\quad$ $p\,k \times p\,k$ $\quad$ $p\,k \times p\,n$ $\quad$ $p\,n \times p\,n$



where (provided all intermediate matrices are invertible)

$$\hat{\mathbf{D}} = \left(\mathbf{V}^*\,\mathbf{D}^{-1}\,\mathbf{U}\right)^{-1}, \quad \mathbf{E} = \mathbf{D}^{-1}\,\mathbf{U}\,\hat{\mathbf{D}}, \quad \mathbf{F} = (\hat{\mathbf{D}}\,\mathbf{V}^*\,\mathbf{D}^{-1})^*, \quad \mathbf{G} = \mathbf{D}^{-1} - \mathbf{D}^{-1}\,\mathbf{U}\,\hat{\mathbf{D}}\,\mathbf{V}^*\,\mathbf{D}^{-1}.$$

Note: All matrices set in blue are block diagonal.

What is the role of the basis matrices $\mathbf{U}_\tau$ *and* $\mathbf{V}_\tau$*?*

To answer this question, we introduce some notation.
Let $\{I_\tau\}_{\tau=1}^4$ and $\{J_\tau\}_{\tau=1}^4$ be partitions of the index vector

$$[1,\, 2,\, 3,\, \ldots,\, N] = [I_1,\, I_2,\, I_3,\, I_4] = [J_1,\, J_2,\, J_3,\, J_4]$$

so that in the factorization

$$\mathbf{A} = \begin{bmatrix} \mathbf{D}_{11} & \mathbf{A}_{12} & \mathbf{A}_{13} & \mathbf{A}_{14} \\ \mathbf{A}_{21} & \mathbf{D}_{22} & \mathbf{A}_{23} & \mathbf{A}_{24} \\ \mathbf{A}_{31} & \mathbf{A}_{32} & \mathbf{D}_{33} & \mathbf{A}_{34} \\ \mathbf{A}_{41} & \mathbf{A}_{42} & \mathbf{A}_{43} & \mathbf{D}_{44} \end{bmatrix}$$

we have

$$\mathbf{D}_\tau = \mathbf{A}(I_\tau, J_\tau)$$

and

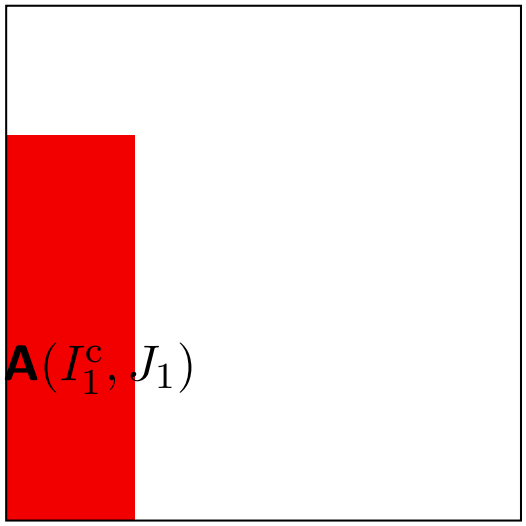
$$\mathbf{A}_{\sigma,\tau} = \mathbf{A}(I_\sigma, J_\tau).$$

(We typically have $I_\tau = J_\tau$.)

What is the role of the basis matrices $\mathbf{U}_\tau$ *and* $\mathbf{V}_\tau$*?*

Recall our toy example: $\mathbf{A} = \begin{bmatrix} \mathbf{D}_{11} & \mathbf{U}_1 \tilde{\mathbf{A}}_{12} \mathbf{V}_2^* & \mathbf{U}_1 \tilde{\mathbf{A}}_{13} \mathbf{V}_3^* & \mathbf{U}_1 \tilde{\mathbf{A}}_{14} \mathbf{V}_4^* \\ \mathbf{U}_2 \tilde{\mathbf{A}}_{21} \mathbf{V}_1^* & \mathbf{D}_{22} & \mathbf{U}_2 \tilde{\mathbf{A}}_{23} \mathbf{V}_3^* & \mathbf{U}_2 \tilde{\mathbf{A}}_{24} \mathbf{V}_4^* \\ \mathbf{U}_3 \tilde{\mathbf{A}}_{31} \mathbf{V}_1^* & \mathbf{U}_3 \tilde{\mathbf{A}}_{32} \mathbf{V}_2^* & \mathbf{D}_{33} & \mathbf{U}_3 \tilde{\mathbf{A}}_{34} \mathbf{V}_4^* \\ \mathbf{U}_4 \tilde{\mathbf{A}}_{41} \mathbf{V}_1^* & \mathbf{U}_4 \tilde{\mathbf{A}}_{42} \mathbf{V}_2^* & \mathbf{U}_4 \tilde{\mathbf{A}}_{43} \mathbf{V}_3^* & \mathbf{D}_{44} \end{bmatrix}$.

We see that the columns of $\mathbf{V}_1$ must span the row space of the matrix $\mathbf{A}(I_1^c, J_1)$ where I_1 and J_1 are the index vectors for the first block and $I_1^c = I \backslash I_1$.



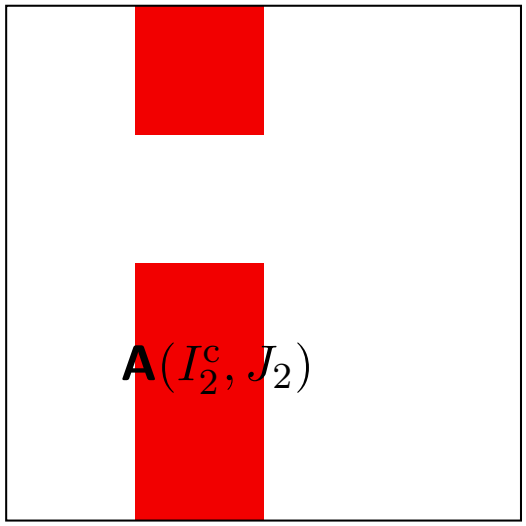
The matrix $\mathbf{A}$

What is the role of the basis matrices $\mathbf{U}_\tau$ *and* $\mathbf{V}_\tau$*?*

Recall our toy example: $\mathbf{A} =$

$$\begin{bmatrix} \mathbf{D}_{11} & \mathbf{U}_1 \tilde{\mathbf{A}}_{12} \mathbf{V}_2^* & \mathbf{U}_1 \tilde{\mathbf{A}}_{13} \mathbf{V}_3^* & \mathbf{U}_1 \tilde{\mathbf{A}}_{14} \mathbf{V}_4^* \\ \mathbf{U}_2 \tilde{\mathbf{A}}_{21} \mathbf{V}_1^* & \mathbf{D}_{22} & \mathbf{U}_2 \tilde{\mathbf{A}}_{23} \mathbf{V}_3^* & \mathbf{U}_2 \tilde{\mathbf{A}}_{24} \mathbf{V}_4^* \\ \mathbf{U}_3 \tilde{\mathbf{A}}_{31} \mathbf{V}_1^* & \mathbf{U}_3 \tilde{\mathbf{A}}_{32} \mathbf{V}_2^* & \mathbf{D}_{33} & \mathbf{U}_3 \tilde{\mathbf{A}}_{34} \mathbf{V}_4^* \\ \mathbf{U}_4 \tilde{\mathbf{A}}_{41} \mathbf{V}_1^* & \mathbf{U}_4 \tilde{\mathbf{A}}_{42} \mathbf{V}_2^* & \mathbf{U}_4 \tilde{\mathbf{A}}_{43} \mathbf{V}_3^* & \mathbf{D}_{44} \end{bmatrix}.$$

We see that the columns of $\mathbf{V}_2$ must span the row space of the matrix $\mathbf{A}(I_2^c, J_2)$ where I_2 and J_2 are the index vectors for the first block and $I_2^c = I \backslash I_2$.



The matrix $\mathbf{A}$

Recall: The block separable structure relies on factorizations such as (for $k < n$)

$$\begin{array}{ccccc} \mathbf{A}_{\sigma,\tau} & = & \mathbf{U}_\sigma & \tilde{\mathbf{A}}_{\sigma,\tau} & \mathbf{V}_\tau^* \\ n \times n & & n \times k & k \times k & k \times n \end{array}$$

For the representation to be numerically stable, it is critical that the basis matrices $\mathbf{U}_\tau$ and $\mathbf{V}_\tau$ be well-conditioned.

Standard practice is to have $\mathbf{U}_\tau$ and $\mathbf{V}_\tau$ be orthonormal (i.e. $\mathbf{U}_\tau^*\mathbf{U}_\tau = \mathbf{V}_\tau^*\mathbf{V}_\tau = \mathbf{I}_k$). This maximizes stability.

We have decided to instead use *interpolatory decompositions* in which:

1. $\mathbf{U}_\tau$ and $\mathbf{V}_\tau$ each contain the $k \times k$ identity matrix as a submatrix.
2. $\mathbf{U}_\tau$ and $\mathbf{V}_\tau$ are "reasonably" well-conditioned.
3. $\tilde{\mathbf{A}}_{\sigma,\tau}$ is a submatrix of $\mathbf{A}$ for all σ, τ.

Our choice leads to some loss of accuracy, but enables the construction of (relatively) simple compression algorithms.

Constructing $\{\mathbf{U}_\tau\}_\tau$ and $\{\mathbf{V}_\tau\}_\tau$ via randomized sampling

$$\text{Set } \mathbf{B} = \mathbf{A} - \mathbf{D} = \begin{bmatrix} \mathbf{0} & \mathbf{A}_{12} & \mathbf{A}_{13} & \mathbf{A}_{14} \\ \mathbf{A}_{21} & \mathbf{0} & \mathbf{A}_{23} & \mathbf{A}_{24} \\ \mathbf{A}_{31} & \mathbf{A}_{32} & \mathbf{0} & \mathbf{A}_{34} \\ \mathbf{A}_{41} & \mathbf{A}_{42} & \mathbf{A}_{43} & \mathbf{0} \end{bmatrix}.$$

Then the rows of $\mathbf{V}_\tau^*$ must span the row space of $\mathbf{B}(:, J_\tau)$.

We use randomized sampling to find a spanning set for $\mathbf{B}(:, J_\tau)$:

1. Draw an $\ell \times N$ Gaussian random matrix $\mathbf{\Omega}$. (Think $\ell = k + 10$.)
2. Form a sample matrix $\mathbf{S} = \mathbf{\Omega B} = \mathbf{\Omega A} - \mathbf{\Omega D}$.
3. For each τ, form $\mathbf{V}_\tau^*$ by performing RRQR on the columns of $\mathbf{S}(:, J_\tau)$:

$$\mathbf{S}(:, J_\tau) = \mathbf{S}(:, \tilde{J}_\tau)\mathbf{V}_\tau^*.$$

If $\{\mathbf{U}_\tau\}_\tau$ and $\{\tilde{I}_\tau\}_\tau$ are constructed analogously, then

$$\mathbf{A}(I_\sigma, J_\tau) = \mathbf{U}_\sigma \underbrace{\mathbf{A}(\tilde{I}_\sigma, \tilde{J}_\tau)}_{=\tilde{\mathbf{A}}_{\sigma,\tau}} \mathbf{V}_\tau^* \qquad \text{when } \sigma \neq \tau.$$

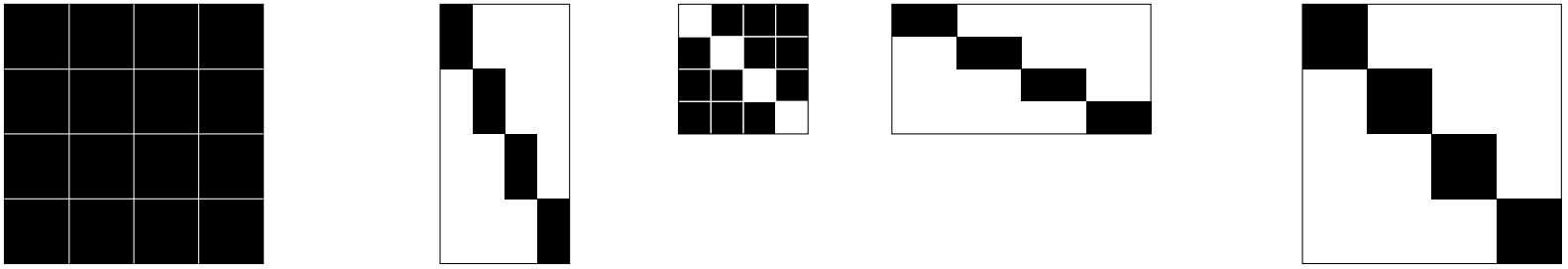
We have now obtained a factorization

$$\mathbf{A} = \underbrace{\begin{bmatrix} \mathbf{U}_1 & & & \\ & \mathbf{U}_2 & & \\ & & \mathbf{U}_3 & \\ & & & \mathbf{U}_4 \end{bmatrix}}_{=\mathbf{U}} \underbrace{\begin{bmatrix} \mathbf{0} & \tilde{\mathbf{A}}_{12} & \tilde{\mathbf{A}}_{13} & \tilde{\mathbf{A}}_{14} \\ \tilde{\mathbf{A}}_{21} & \mathbf{0} & \tilde{\mathbf{A}}_{23} & \tilde{\mathbf{A}}_{24} \\ \tilde{\mathbf{A}}_{31} & \tilde{\mathbf{A}}_{32} & \mathbf{0} & \tilde{\mathbf{A}}_{34} \\ \tilde{\mathbf{A}}_{41} & \tilde{\mathbf{A}}_{42} & \tilde{\mathbf{A}}_{43} & \mathbf{0} \end{bmatrix}}_{=\tilde{\mathbf{A}}} \underbrace{\begin{bmatrix} \mathbf{V}_1^* & & & \\ & \mathbf{V}_2^* & & \\ & & \mathbf{V}_3^* & \\ & & & \mathbf{V}_4^* \end{bmatrix}}_{=\mathbf{V}^*} + \underbrace{\begin{bmatrix} \mathbf{D}_1 & & & \\ & \mathbf{D}_2 & & \\ & & \mathbf{D}_3 & \\ & & & \mathbf{D}_4 \end{bmatrix}}_{=\mathbf{D}}$$

or

$$\mathbf{A} = \mathbf{U}\ \tilde{\mathbf{A}}\ \mathbf{V}^* + \mathbf{D},$$

$p\,n \times p\,n$ $\quad$ $p\,n \times p\,k$ $\quad$ $p\,k \times p\,k$ $\quad$ $p\,k \times p\,n$ $\quad$ $p\,n \times p\,n$



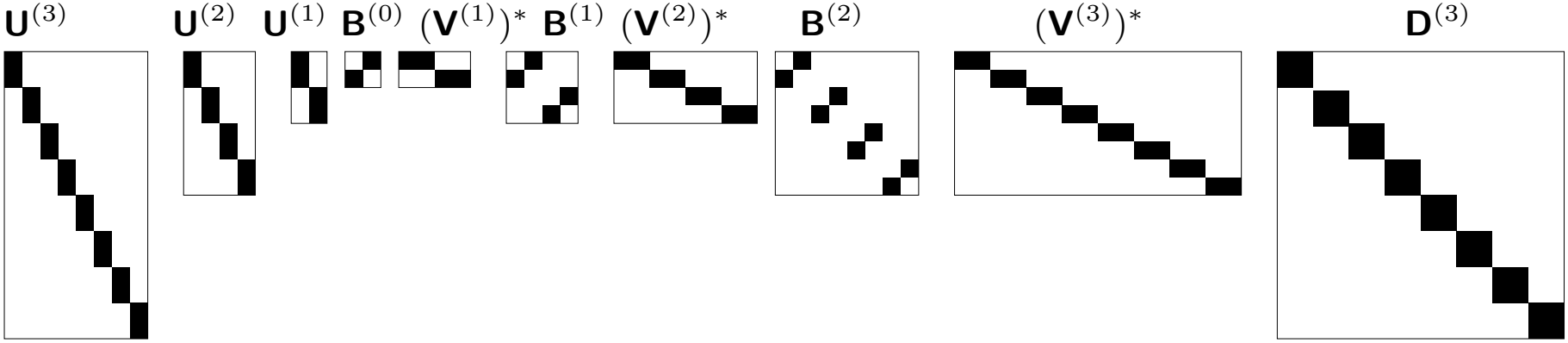
Important: Since we use interpolative factorizations, $\tilde{\mathbf{A}}_{\sigma,\tau} = \mathbf{A}(\tilde{I}_\sigma, \tilde{J}_\tau)$, so $\tilde{\mathbf{A}}$ is a submatrix of $\mathbf{A} - \mathbf{D}$.

We can recurse!

Recursion results in a telescoping factorization of $\mathbf{A}$:

$$\mathbf{A} = \mathbf{U}^{(3)}\big(\mathbf{U}^{(2)}\big(\mathbf{U}^{(1)}\,\mathbf{B}^{(0)}\,\mathbf{V}^{(1)}\big)^* + \mathbf{B}^{(1)}\big)\big(\mathbf{V}^{(2)}\big)^* + \mathbf{B}^{(2)}\big)\big(\mathbf{V}^{(3)}\big)^* + \mathbf{D}^{(3)},$$

with the block structure:



All matrices are now block diagonal except $\mathbf{B}^{(0)}$*, which is small.*

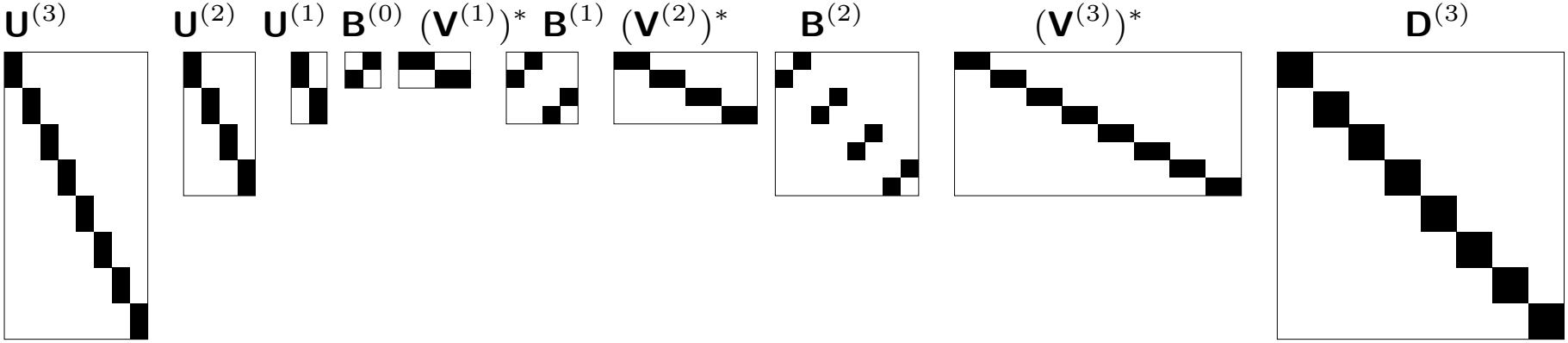
Using a telescoping factorization of $\mathbf{A}$

$$\mathbf{A} = \mathbf{U}^{(3)}\big(\mathbf{U}^{(2)}\big(\mathbf{U}^{(1)}\,\mathbf{B}^{(0)}\,\mathbf{V}^{(1)}\big)^* + \mathbf{B}^{(1)}\big)\big(\mathbf{V}^{(2)}\big)^* + \mathbf{B}^{(2)}\big)\big(\mathbf{V}^{(3)}\big)^* + \mathbf{D}^{(3)},$$

we have a telescoping inversion formula

$$\mathbf{A}^{-1} = \mathbf{E}^{(3)}\big(\mathbf{E}^{(2)}\big(\mathbf{E}^{(1)}\,\hat{\mathbf{D}}^{(0)}\,\mathbf{F}^{(1)}\big)^* + \hat{\mathbf{D}}^{(1)}\big)\big(\mathbf{F}^{(2)}\big)^* + \hat{\mathbf{D}}^{(2)}\big)\big(\mathbf{V}^{(3)}\big)^* + \hat{\mathbf{D}}^{(3)}.$$

Block structure of factorization:

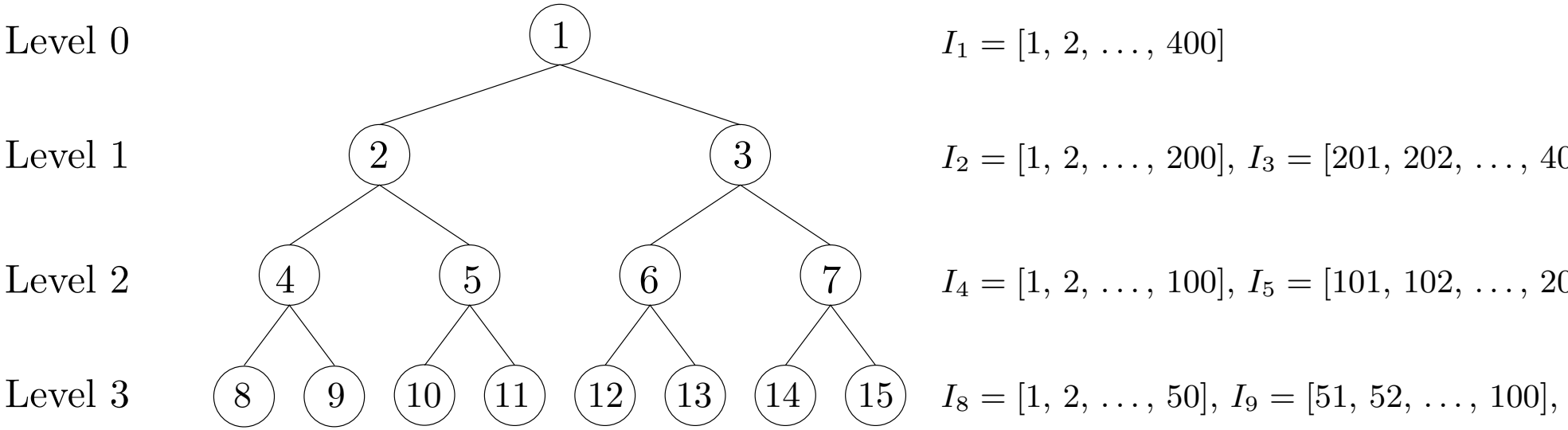


All matrices are now block diagonal except $\hat{\mathbf{D}}^{(0)}$*, which is small.*

Formal definition of a *Hierarchically Block Separable (HBS)* matrix

Suppose $\mathcal{T}$ is a binary tree on the index vector $I = [1, 2, 3, \ldots, N]$.

For a node τ in the tree, let I_τ denote the corresponding index vector.



Numbering of nodes in a fully populated binary tree with $L = 3$ levels.
The root is the original index vector $I = I_1 = [1, 2, \ldots, 400]$.

Formal definition of a *Hierarchically Block Separable (HBS)* matrix

Suppose $\mathcal{T}$ is a binary tree.

For a node τ in the tree, let I_τ denote the corresponding index vector.

For leaves σ and τ, set $\mathbf{A}_{\sigma,\tau} = \mathbf{A}(I_\sigma, I_\tau)$ and suppose that all off-diagonal blocks satisfy

$$\begin{array}{ccccccc} \mathbf{A}_{\sigma,\tau} & = & \mathbf{U}_\sigma & \tilde{\mathbf{A}}_{\sigma,\tau} & \mathbf{V}_\tau^* & & \sigma \neq \tau \\ n \times n & & n \times k & k \times k & k \times n & & \end{array}$$

For non-leaves σ and τ, let $\{\sigma_1, \sigma_2\}$ denote the children of σ, and let $\{\tau_1, \tau_2\}$ denote the children of τ. Set

$$\mathbf{A}_{\sigma,\tau} = \begin{bmatrix} \tilde{\mathbf{A}}_{\sigma_1,\tau_1} & \tilde{\mathbf{A}}_{\sigma_1,\tau_2} \\ \tilde{\mathbf{A}}_{\sigma_2,\tau_1} & \tilde{\mathbf{A}}_{\sigma_2,\tau_2} \end{bmatrix}$$

Then suppose that the off-diagonal blocks satisfy

$$\begin{array}{ccccccc} \mathbf{A}_{\sigma,\tau} & = & \mathbf{U}_\sigma & \tilde{\mathbf{A}}_{\sigma,\tau} & \mathbf{V}_\tau^* & & \sigma \neq \tau \\ 2k \times 2k & & 2k \times k & k \times k & k \times 2k & & \end{array}$$

	Name:	Size:	Function:
For each leaf node τ:	$\mathbf{D}_\tau$	$n \times n$	The diagonal block $\mathbf{A}(I_\tau, I_\tau)$.
	$\mathbf{U}_\tau$	$n \times k$	Basis for the columns in the blocks in row τ.
	$\mathbf{V}_\tau$	$n \times k$	Basis for the rows in the blocks in column τ.
For each parent node τ:	$\mathbf{B}_\tau$	$2k \times 2k$	Interactions between the children of τ.
	$\mathbf{U}_\tau$	$2k \times k$	Basis for the columns in the (reduced) blocks in row τ.
	$\mathbf{V}_\tau$	$2k \times k$	Basis for the rows in the (reduced) blocks in column τ.

An HBS matrix $\mathbf{A}$ *with a tree* $\mathcal{T}$ *is fully specified if the factors listed above are provided.*

Generate an $N \times \ell$ Gaussian random matrix $\mathbf{\Omega}$. (think $\ell = k + 10$)

Evaluate $\mathbf{S} = \mathbf{A}\,\mathbf{\Omega}$ using the fast matrix-vector multiplier.

loop over levels, finer to coarser, $\ell = L,\, L-1,\, \ldots,\, 2,\, 1$

loop over all nodes τ on level ℓ

if τ is a leaf node then

$I_{\text{loc}} = I_\tau$

$\mathbf{\Omega}_{\text{loc}} = \mathbf{\Omega}(I_\tau, :)$

$\mathbf{S}_{\text{loc}} = \mathbf{S}(I_\tau, :) - \mathbf{A}(I_\tau, I_\tau)\,\mathbf{\Omega}_{\text{loc}}$

else

Let ν_1 and ν_2 be the two children of τ.

$I_{\text{loc}} = [\tilde{I}_{\nu_1},\, \tilde{I}_{\nu_2}]$

$$\mathbf{\Omega}_{\text{loc}} = \begin{bmatrix} \mathbf{\Omega}_{\nu_1} \\ \mathbf{\Omega}_{\nu_2} \end{bmatrix}$$

$$\mathbf{S}_{\text{loc}} = \begin{bmatrix} \mathbf{S}_{\nu_1} - \mathbf{A}(\tilde{I}_{\nu_1}, \tilde{I}_{\nu_2})\,\mathbf{\Omega}_{\nu_2} \\ \mathbf{S}_{\nu_2} - \mathbf{A}(\tilde{I}_{\nu_2}, \tilde{I}_{\nu_1})\,\mathbf{\Omega}_{\nu_1} \end{bmatrix}$$

end if

$[\mathbf{U}_\tau, J_\tau] = \texttt{interpolate}(\mathbf{S}^*_{\text{loc}})$ (i.e. perform RRQR and form $\mathbf{R}_{11}^{-1}\mathbf{R}_{12}$)

$\mathbf{\Omega}_\tau = \mathbf{U}_\tau^*\,\mathbf{\Omega}_{\text{loc}}$

$\mathbf{S}_\tau = \mathbf{S}_{\text{loc}}(J_\tau, :)$

$\tilde{I}_\tau = I_{\text{loc}}(J_\tau)$

end loop

end loop

For all leaf nodes τ, set $\mathbf{D}_\tau = \mathbf{A}(I_\tau,\, I_\tau)$.

For all sibling pairs $\{\nu_1,\, \nu_2\}$ set $B_{\nu_1,\nu_2} = \mathbf{A}(\tilde{I}_{\nu_1},\, \tilde{I}_{\nu_2})$.

The asymptotic cost of applying the algorithm to an $N \times N$ matrix $\mathbf{A}$ is

$$T_{\text{total}} \sim T_{\text{mult}} \times 2\,(k+10) + T_{\text{rand}} \times N\,(k+10) + T_{\text{entry}} \times 2\,N\,k + T_{\text{flop}} \times c\,N\,k^2,$$

where k is an upper bound for the rank of an off-diagonal block of $\mathbf{A}$, and where

T_{total}	is the total execution time
T_{mult}	is the cost of a matrix-vector multiply involving $\mathbf{A}$
T_{rand}	is the cost of "drawing" a Gaussian random number
T_{entry}	is the cost of evaluating an entry of $\mathbf{A}$
T_{flop}	is the cost of a flop

Important: The matrix vector multiplies can all be executed in parallel. Specifically, you only need to compute the two matrix-matrix products

$$\underset{N \times N}{\mathbf{A}} \quad \underset{N \times (k+10)}{\mathbf{\Omega}} \qquad \text{and} \qquad \underset{N \times N}{\mathbf{A}^*} \quad \underset{N \times (k+10)}{\mathbf{\Omega}}$$

Numerical example

Let **A** be a discrete approximations of the boundary integral operator

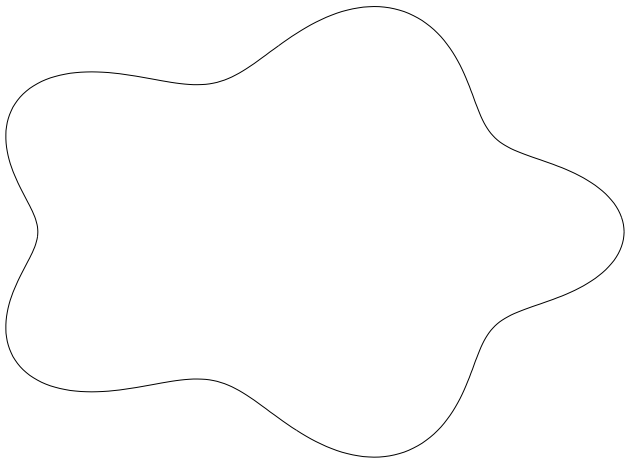
$$[Tu](x) = \alpha\, u(x) + \int_\Gamma K(x,y)\, u(y)\, ds(y), \qquad x \in \Gamma, \tag{3}$$

where Γ is the contour shown below, and α and K are chosen as either one of the following two options:

$$\alpha = 0 \quad \text{and} \quad K(x,y) = \log|x-y| \qquad \text{(the "single layer" kernel)} \tag{4}$$

$$\alpha = 1/2 \quad \text{and} \quad K(x,y) = \big(n(y)\cdot(x-y)\big)/|x-y|^2 \qquad \text{(the "double layer" kernel)} \tag{5}$$

For $y \in \Gamma$, $n(y)$ denotes the unit normal of Γ at y. The single layer operator was discretized via the trapezoidal rule with a Kapur-Rokhlin end-point modification of the 6^{th} order for handling the singularity in the kernel $k(x,y)$ as y approaches x, resulting in a symmetric coefficient matrix.



The contour Γ

Numerical example: implementation details

$\mathbf{A}$ is a discrete approximation to a boundary integral operator.

The sample matrices $\mathbf{S} = \mathbf{A\Omega}$ and $\mathbf{S}' = \mathbf{A}^*\mathbf{\Omega}$ were evaluated via the Fast Multipole Method. Observe that *two calls* to the FMM is sufficient.

The randomized hierarchical procedure was used to compute an approximant $\mathbf{A}^{(\text{approx})}$ in a hierarchically block separable format.

Using a power iteration the following error metric was computed:

$$e_1 = \frac{||\mathbf{A} - \mathbf{A}^{(\text{approx})}||}{||\mathbf{A}||}$$

Hardware: A single processor 3.2GHz Pentium IV with 2GB of RAM.

Results: The double layer potential (a well-conditioned matrix)

$\epsilon = 10^{-5}$, $\ell = 50$							
N	$T_{\text{compression}}$	$T_{\text{inversion}}$	T_{matvec}	$\|\|\mathbf{A}^{(\text{approx})}\|\|$	$\|\|\mathbf{G}\|\|$	e_1	e_2
400	0.047	0.031	0.000	1.04	3.6	2.6e-6	5.5e-6
800	0.094	0.031	0.016	1.04	3.6	3.1e-6	6.5e-6
1600	0.219	0.094	0.000	1.04	3.5	2.9e-6	6.3e-6
3200	0.406	0.140	0.016	1.04	3.5	2.6e-6	5.4e-6
6400	0.844	0.297	0.031	1.04	3.5	3.4e-6	7.6e-6
12800	1.688	0.578	0.062	1.04	3.6	3.6e-6	7.8e-6
25600	3.344	1.156	0.141	1.04	3.3	3.4e-6	7.3e-6
$\epsilon = 10^{-10}$, $\ell = 100$							
N	$T_{\text{compression}}$	$T_{\text{inversion}}$	T_{matvec}	$\|\|\mathbf{A}^{(\text{approx})}\|\|$	$\|\|\mathbf{G}\|\|$	e_1	e_2
400	0.093	0.032	0.000	1.04	3.6	2.1e-11	4.5e-11
800	0.156	0.079	0.000	1.04	3.6	2.0e-11	4.4e-11
1600	0.297	0.109	0.016	1.04	3.6	1.5e-11	3.1e-11
3200	0.579	0.203	0.015	1.04	3.4	1.9e-11	4.0e-11
6400	1.094	0.344	0.047	1.04	3.6	2.5e-11	5.2e-11
12800	2.141	0.687	0.078	1.04	3.6	2.0e-11	4.2e-11
25600	4.093	1.266	0.141	1.04	3.6	3.4e-11	7.1e-11

Times in seconds. Recall $e_1 = \frac{||\mathbf{A}-\mathbf{A}^{(\text{approx})}||}{||\mathbf{A}||}$ *and* $e_2 = ||\mathbf{I} - \mathbf{AG}||$, *where* $\mathbf{G} \approx \mathbf{A}^{-1}$.

Observation: The FMM itself is much slower than the HBS algebra!

	$N = 800$	$N = 1\,600$	$N = 3\,200$	$N = 6\,400$	$N = 12\,800$	$N = 25\,600$
$N_{\text{vec}} = 1$	1.328	1.891	2.875	4.531	7.343	13.266
$N_{\text{vec}} = 50$	1.500	2.266	3.578	5.969	10.531	19.375
$N_{\text{vec}} = 100$	1.656	2.563	4.110	7.062	12.844	23.891

Time in seconds required by our implementation of the FMM to apply a matrix of size $N \times N$ to N_{vec} vectors simultaneously. The FMM uses multipole expansions of length 40, leading to about 15 accurate digits.

Observation:

Being able to compute the ℓ matvecs in parallel is highly advantageous.

Numerical example: Inversion of HBS-matrix

We performed an HBS inversion to compute

$$\mathbf{G} \approx \left(\mathbf{A}^{(\text{approx})}\right)^{-1}$$

and evaluated the error metric

$$e_2 = ||\mathbf{I} - \mathbf{AG}||.$$

Results: The double layer potential (a well-conditioned matrix)

$\epsilon = 10^{-5}$, $\ell = 50$							
N	$T_{\text{compression}}$	$T_{\text{inversion}}$	T_{matvec}	$\|\|\mathbf{A}^{(\text{approx})}\|\|$	$\|\|\mathbf{G}\|\|$	e_1	e_2
400	0.047	0.031	0.000	1.04	3.6	2.6e-6	5.5e-6
800	0.094	0.031	0.016	1.04	3.6	3.1e-6	6.5e-6
1600	0.219	0.094	0.000	1.04	3.5	2.9e-6	6.3e-6
3200	0.406	0.140	0.016	1.04	3.5	2.6e-6	5.4e-6
6400	0.844	0.297	0.031	1.04	3.5	3.4e-6	7.6e-6
12800	1.688	0.578	0.062	1.04	3.6	3.6e-6	7.8e-6
25600	3.344	1.156	0.141	1.04	3.3	3.4e-6	7.3e-6
$\epsilon = 10^{-10}$, $\ell = 100$							
N	$T_{\text{compression}}$	$T_{\text{inversion}}$	T_{matvec}	$\|\|\mathbf{A}^{(\text{approx})}\|\|$	$\|\|\mathbf{G}\|\|$	e_1	e_2
400	0.093	0.032	0.000	1.04	3.6	2.1e-11	4.5e-11
800	0.156	0.079	0.000	1.04	3.6	2.0e-11	4.4e-11
1600	0.297	0.109	0.016	1.04	3.6	1.5e-11	3.1e-11
3200	0.579	0.203	0.015	1.04	3.4	1.9e-11	4.0e-11
6400	1.094	0.344	0.047	1.04	3.6	2.5e-11	5.2e-11
12800	2.141	0.687	0.078	1.04	3.6	2.0e-11	4.2e-11
25600	4.093	1.266	0.141	1.04	3.6	3.4e-11	7.1e-11

Times in seconds. Recall $e_1 = \frac{||\mathbf{A}-\mathbf{A}^{(\text{approx})}||}{||\mathbf{A}||}$ *and* $e_2 = ||\mathbf{I} - \mathbf{AG}||$, *where* $\mathbf{G} \approx \mathbf{A}^{-1}$.

Results: The single layer potential (an ill-conditioned matrix)

$\epsilon = 10^{-5}$, $\ell = 50$							
N	$T_{\text{compression}}$	$T_{\text{inversion}}$	T_{matvec}	$\|\|\mathbf{A}^{(\text{approx})}\|\|$	$\|\|\mathbf{G}\|\|$	e_1	e_2
400	0.047	0.031	0.000	1.23	6.4e3	5.1e-6	2.9e-3
800	0.078	0.063	0.000	0.77	1.4e4	5.2e-6	2.4e-3
1600	0.140	0.141	0.016	0.57	1.6e5	1.1e-5	2.0e-2
3200	0.297	0.297	0.031	0.57	2.3e5	5.8e-6	1.2e-2
6400	0.625	0.625	0.062	0.57	1.1e6	2.9e-6	1.4e-2
12800	1.281	1.328	0.141	0.57	4.2e6	3.5e-6	8.0e-2
25600	2.625	2.875	0.265	0.57	5.6e6	6.5e-6	1.2e-1
$\epsilon = 10^{-10}$, $\ell = 100$							
N	$T_{\text{compression}}$	$T_{\text{inversion}}$	T_{matvec}	$\|\|\mathbf{A}^{(\text{approx})}\|\|$	$\|\|\mathbf{G}\|\|$	e_1	e_2
400	0.047	0.047	0.000	1.24	6.4e3	3.3e-11	1.5e-8
800	0.109	0.094	0.000	0.75	1.4e4	4.3e-11	2.0e-8
1600	0.203	0.203	0.032	0.57	1.6e5	4.3e-11	1.2e-7
3200	0.422	0.406	0.031	0.57	2.3e5	4.3e-11	1.2e-5
6400	0.843	0.844	0.078	0.57	1.1e6	4.4e-11	4.6e-5
12800	1.687	1.703	0.141	0.57	4.2e6	3.3e-11	2.2e-4
25600	3.407	3.547	0.266	0.57	5.6e6	2.6e-11	2.0e-5

Times in seconds. Recall $e_1 = \frac{||\mathbf{A}-\mathbf{A}^{(\text{approx})}||}{||\mathbf{A}||}$ *and* $e_2 = ||\mathbf{I} - \mathbf{AG}||$, *where* $\mathbf{G} \approx \mathbf{A}^{-1}$.

Key points:

1. Finding spanning rows and columns of a large matrix can be very useful.
   - Faster algorithms.
   - Data interpretation.
   - Factorizations that preserve structure (e.g. sparsity).

2. There are many different factorizations to choose from.
   Take care to not introduce instabilities unnecessarily.

3. Finding "tight" factorizations is a delicate but well-studied subject.
   More relaxed formulations admit very easy algorithms and are often adequate.

4. Randomized sampling + row/column selection is a powerful combination.
   - Approximation of *Hierarchically Block Separable* matrices.

References:

- N. Halko, P.G. Martinsson, J. Tropp: *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions* SIAM Review, 53(2), pp. 217–288, 2011.

- P.G. Martinsson: *Approximation of Structured Matrices via Randomized Sampling* (arXiv.org #0806.2339) To appear in the SIAM Journal on Matrix Analysis and applications.

- H. Cheng, Z. Gimbutas, P.G. Martinsson, V. Rokhlin, *On the compression of low rank matrices* SIAM Journal of Scientific Computing, 26(4), pp. 1389-1404, 2005

- M. Gu and S.C. Eisenstat: *Efficient algorithms for computing a strong rank-revealing QR factorization* SIAM J. Sci. Comput. 17 (1996), no. 4, 848869.

- Franco Woolfe, Edo Liberty, Vladimir Rokhlin, and Mark Tygert, *A fast randomized algorithm for the approximation of matrices* Applied and Computational Harmonic Analysis, 25 (3): 335-366, 2008.

- N. Ailon and B. Chazelle, *Approximate nearest neighbors and the fast JohnsonLindenstrauss transform* in Proceedings of the 38th Annual ACM Symposium on Theory of Computing (STOC 06), 2006, pp. 557563.