Parallel Recursive Skeletonization Solver for Dense Linear Systems on GPU-Accelerated Computers

Molena Huynh

North Carolina State University

Table of Contents

Table of Contents

Motivation

Given a data set $\{\mathbf{x}_i\}_{i=1}^{N}$, where $\mathbf{x}_i \in \mathbb{R}^d$, and a kernel function $\mathscr{K} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, the associated kernel matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$ is defined as

$$\mathbf{A}_{i,j} = \mathscr{K}(x_i, x_j), \qquad \forall i,j = 1,2,\ldots,N.$$

Interested kernel functions:

- From kernel methods (using in support vector machines) in machine learning:
  For example: The Gaussian kernel for some $\sigma \in \mathbb{R}$,

$$\mathbf{A}_{i,j} = \mathscr{K}(x_i, x_j) = e^{-\frac{\|x_i - x_j\|_2^2}{2\sigma^2}}$$

- From discretization of boundary integral equations and elliptical PDEs:
  For example: The kernel derived from the Green's function for the Laplace equation:

$$\mathbf{A}_{i,j} = \mathscr{K}(x_i, x_j) = \frac{-1}{2\pi} \ln(\|x_i - x_j\|_2)$$

  where x_i and x_j are points on the boundary of the domain.

Table of Contents

Problem Statement

Every kernel matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$ associated with the listed interested kernel functions employs a multilevel structure of low-rank off-diagonal blocks [1][2]

The first goal is to

- factorize matrix $\mathbf{A}$ within the optimal $\mathscr{O}(N)$ complexity in 1D, and $\mathscr{O}(N^{3(1-\frac{1}{d})})$ complexity in d dimensions (which surpasses the cubic $\mathscr{O}(N^3)$ complexity of traditional matrix factorization, for example, LU factorization)

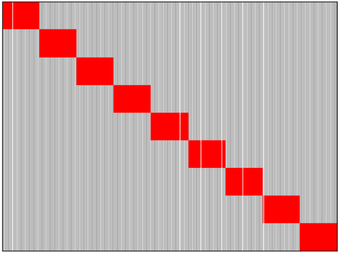

Figure 1: A hierarchical off-diagonal low-rank matrix with nine 64×64 diagonal blocks

Problem Statement

Consider the large dense linear system of equations $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{b} \in \mathbb{R}^N$ is given, and $\mathbf{A} \in \mathbb{R}^{N \times N}$ is a given kernel matrix from the interested kernel functions.

The second goal is to

- use the decomposition of $\mathbf{A}$ to solve for the unknown $\mathbf{x} \in \mathbb{R}^N$ quickly through a sequence of matrix-vector multiplication.

Table of Contents

Column-pivoted QR Decomposition

Consider the column-pivoted QR factorization of a low-rank matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$

$$\mathbf{AP} = \mathbf{QR} = \begin{bmatrix} \\ \mathbf{Q} \\ \\ \end{bmatrix} \begin{bmatrix} \mathbf{R}_1 & \mathbf{R}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{QR}_1 & \mathbf{QR}_2 \end{bmatrix} \tag{1}$$

- where $\mathbf{Q} \in \mathbb{R}^{N\times k}$ has orthonormal columns, and $\mathbf{R} \in \mathbb{R}^{k\times N}$ is upper triangular,
- $\mathbf{P} \in \mathbb{R}^{N\times N}$ is a chosen permutation matrix so that $\mathbf{R}_1 \in \mathbb{R}^{k\times k}$ is nonsingular.

$$\mathbf{QR}_2 = (\mathbf{QR}_1)(\mathbf{R}_1^{-1}\mathbf{R}_2) \tag{2}$$

- $\mathbf{QR}_1$: the skeleton columns of $\mathbf{A}$
- $\mathbf{QR}_2$: the redundant columns of $\mathbf{A}$
- $\mathbf{T} = \mathbf{R}_1^{-1}\mathbf{R}_2$

Table of Contents

Low-rank Compression

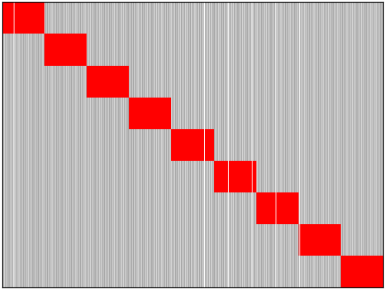

Figure 2: A matrix with nine diagonal blocks

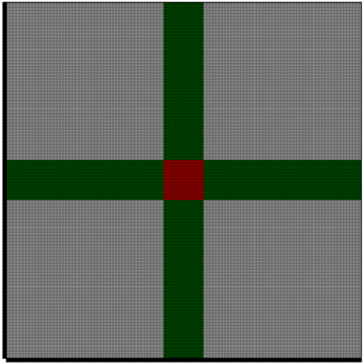

Figure 3: Low-rank compression for one diagonal block

Low-rank compression for $\begin{bmatrix} \text{green\_column} \\ \text{green\_row.T} \end{bmatrix}$ to get $\mathbf{T}$.

Proxy Trick for Skeletonization

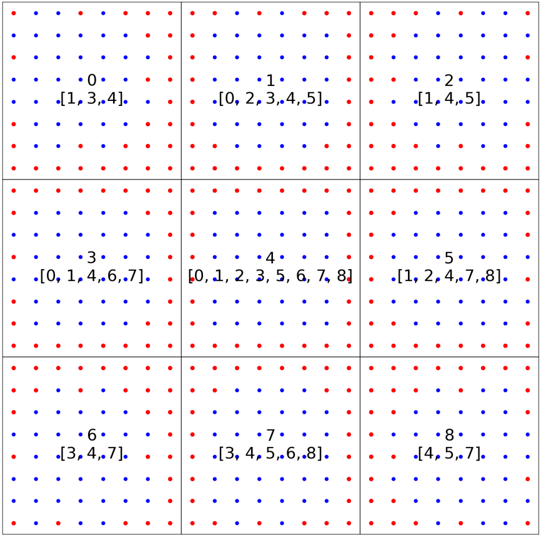

Figure 4: Proxy trick for all diagonal blocks

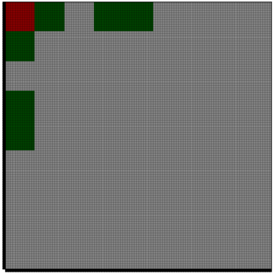

Figure 5: Proxy trick for one diagonal block

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{rr} & \mathbf{A}_{rs} & \blacksquare \\ \mathbf{A}_{sr} & \mathbf{A}_{ss} & \blacktriangleright \\ \blacksquare & \blacktriangleright & \square \end{bmatrix}, \tag{3}$$

- where $\begin{bmatrix} \blacksquare \\ \blacktriangleright \\ \begin{bmatrix} \blacksquare & \blacktriangleright \end{bmatrix}^T \end{bmatrix}$ is the input of the low-rank compression (in green in Figure 3),
- $\begin{bmatrix} \mathbf{A}_{rr} & \mathbf{A}_{rs} \\ \mathbf{A}_{sr} & \mathbf{A}_{ss} \end{bmatrix}$ is one diagonal block (in red in Figure 2 and Figure 3).
- $\square$ blocks remain unchanged,

Calculate

- $\mathbf{B}_{rr} = \mathbf{A}_{rr} - \mathbf{T}^*\mathbf{A}_{sr} - \mathbf{A}_{rs}\mathbf{T} + \mathbf{T}^*\mathbf{A}_{ss}\mathbf{T}$, where $\mathbf{T}$ is from low-rank compression for the black blocks [3].
- $\mathbf{D}_{rr}$ is from the LDU factorization $\mathrm{ldu}(\mathbf{B}_{rr})$.

The goal is to solve for $\mathbf{x} \in \mathbb{R}^N$ given the dense linear system given $\mathbf{Ax} = \mathbf{b}$.

$$\mathbf{A} = \mathbf{U} \begin{bmatrix} \mathbf{D}_{rr} & & \\ & \mathbf{B}_{ss} & \blacktriangleright \\ & \blacktriangleright & \square \end{bmatrix} \mathbf{V} \tag{4}$$

- where $\mathbf{D}_{rr}$ is from $\text{ldu}(\mathbf{B}_{rr})$.
- $\blacktriangleright$ are the skeleton columns.
- $\mathbf{B}_{ss} = \mathbf{A}_{ss} - (\mathbf{A}_{sr} - \mathbf{A}_{ss}\mathbf{T})\mathbf{B}_{rr}^{-1}(\mathbf{A}_{rs} - \mathbf{T}^*\mathbf{A}_{ss})$ is the associated Schur complement.
- $\mathbf{U}$ and $\mathbf{V}$ are triangular matrices from $\mathbf{T}$ and the LDU factorization $\text{ldu}(\mathbf{B}_{rr})$ [3].

Skeletonization

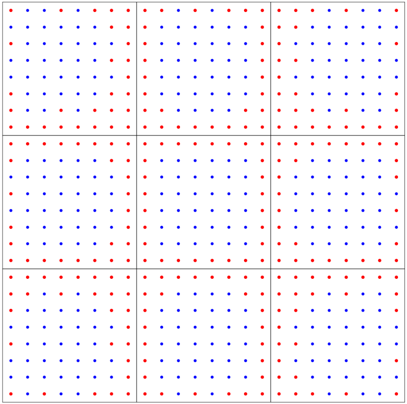

Figure 5: After each level, the algorithm returns skeleton (red) and redundant indices (blue).

The problem of solving $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A} \in \mathbb{R}^{N \times N}$, reducing the size nearly by one half, to become a smaller linear system $\mathbf{Cu} = \mathbf{v}$, where $\mathbf{C} \in \mathbb{R}^{\sqrt{N} \times \sqrt{N}}$.
The complexity of factorizing matrix $\mathbf{C}$ is then downsized to $O\left(N^{3/2}\right)$ from the original complexity $O\left(N^3\right)$ of solving the same problem using matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$.

$$\mathbf{A} = \mathscr{U}\tilde{\mathbf{A}}\mathscr{V} \tag{5}$$

where $\mathscr{V} \in \mathbb{R}^{N\times N}, \mathscr{U} \in \mathbb{R}^{N\times N}$ are block triangular.

$$\tilde{\mathbf{A}} = \begin{bmatrix} \mathbf{D} & \\ & \mathbf{B} \end{bmatrix} \tag{6}$$

where $\mathbf{D}$ is a diagonal block matrix with b diagonal blocks (each block is from one box), each $\mathbf{D}_i \in \mathbb{R}^{r_i\times r_i}$, where r_i is the number of redundant indices of each box, for $1 \leq i \leq b$.

$$\mathbf{D} = \text{diag}(\mathbf{D}_1, \mathbf{D}_2, \ldots, \mathbf{D}_b) \tag{7}$$

Moreover, $\mathbf{B}$ is a small dense matrix (for example, $\mathbf{B} \ll 1{,}024$ if $\mathbf{A} \gg 16,384$).

Table of Contents

Numerical Results

epsilon	N	|s_L|	lu_fact	t_f	m_f	t_a/s	e_s
1.0e-03	12544	28	2.1e+01	2.5e+00	2.1e-04	2.1e-02	5.9e-05
1.0e-06	12544	48	2.1e+01	2.6e+00	2.1e-04	1.9e-02	3.1e-09
1.0e-09	12544	56	2.2e+01	2.6e+00	2.1e-04	1.8e-02	8.7e-11
1.0e-12	12544	60	2.1e+01	2.5e+00	2.1e-04	1.7e-02	8.6e-14
1.0e-03	14400	28	2.9e+01	3.5e+00	2.4e-04	1.9e-02	9.3e-05
1.0e-06	14400	48	2.9e+01	3.3e+00	2.4e-04	2.0e-02	3.2e-09
1.0e-09	14400	56	2.9e+01	2.9e+00	2.4e-04	1.9e-02	8.9e-11
1.0e-12	14400	60	2.8e+01	3.2e+00	2.4e-04	1.9e-02	8.9e-14
1.0e-03	16384	28	4.6e+01	3.7e+00	2.7e-04	2.4e-02	9.7e-05
1.0e-06	16384	48	4.7e+01	3.7e+00	2.7e-04	2.1e-02	7.4e-09
1.0e-09	16384	56	4.8e+01	3.4e+00	2.7e-04	2.0e-02	9.1e-11
1.0e-12	16384	60	4.8e+01	3.6e+00	2.7e-04	2.0e-02	1.5e-13

Figure 6: Numerical Results PRSkel compared to traditional LU Factorization.

- epsilon: Desired accuracy/tolerance (from user's input) for low-rank compression
- $|s_L|$: Number of skeleton columns from each 64×64 diagonal block on leaf level
- lu_fact: Time (in seconds) for traditional LU Factorization
- t_f: Time (in seconds) for PRSkel Factorization
- m_f: Memory (in GB) for PRSkel Factorization
- t_a/s: Time (in seconds) for solving for $\mathbf{x} \in \mathbb{R}^N$ using PRSkel
- e_s: Accuracy from PRSkel

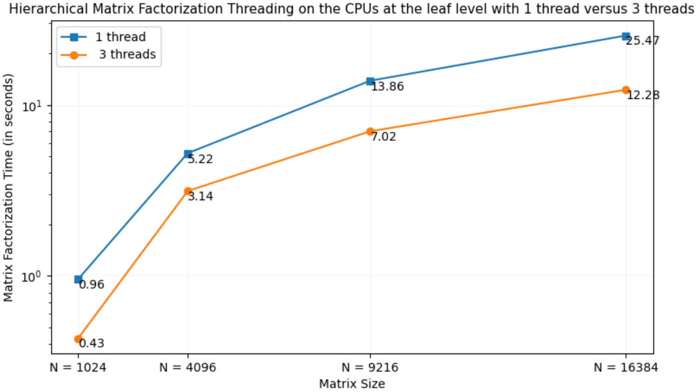

Figure 7: For each $N \in \{1024, 4096, 9216, 16384\}$, the figure shows the amount of time (in seconds) taken to factorize a matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$ in the linear system $\mathbf{Ax} = \mathbf{b}$.

[1] Sivaram Ambikasaran, Daniel Foreman-Mackey, Leslie Greengard, David W. Hogg, and Michael O'Neil. Fast direct methods for gaussian processes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 38(2):252–265, feb 2016.

[2] Chao Chen and Per-Gunnar Martinsson. Solving linear systems on a gpu with hierarchically off-diagonal low-rank approximations. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, SC '22. IEEE Press, 2022.

[3] Kenneth Ho and Lexing Ying. Hierarchical interpolative factorization for elliptic operators: Integral equations. *Communications on Pure and Applied Mathematics*, 69, 07 2013.