Discovering Hierarchical Matrix Structure Through Recursive Tensor Decomposition

Xiaozhe Hu[1], Misha Kilmer[1], Arvind Saibaba[2], Mitchell Scott[1]

[1]Tufts University, Department of Mathematics
[2] North Caroline State University, Department of Mathematics

January 4, 2023

Work Partially Supported by NSF DMS Grant #1821148, Tufts GSC Presenter Grant

Fractional PDEs are useful in Scientific applications

- Fractional Partial Differential Equations (fPDEs) are used in modeling turbulence, financial markets, anomalous diffusion.

Definition (Fractional PDE)

For a fractional index $\alpha \in (1, 2)$ and function $f \in L^2[b, c]$, the initial value problem we are trying to solve is:

$$\mathcal{D}_x^\alpha u(x) = f(x), \quad x \in (b, c)$$
$$u(b), u(c) = 0$$

where the Riesz fractional derivative is:

$$\mathcal{D}_x^\alpha f(x) = \frac{-1}{2\cos(\alpha\pi/2)\Gamma(2-\alpha)} \frac{\mathrm{d}^2}{\mathrm{d}x^2} \int_b^c |x - \xi|^{1-\alpha} f(\xi) \mathrm{d}\xi$$

Discretizing the fPDE

- We use the weak formulation of the fPDE and the Galerkin method to get a finite element discretization.
- We can take the discrete solution to get a vector $\vec{u}$, which is the PDE solution at discrete locations.
- This means we get a linear system: $\mathbf{A}\vec{u} = \vec{f}$

X. Zhao *et.al.* "Adaptive finite element method for fractional differential equations using hierarchical matrices," Comput. Methods Appl. Mech. Engrg.325, pp. 56-76, (2017)

Problems with Adaptive Grid on discretized fPDEs

- The stiffness matrices require $\mathcal{O}(n^2)$ storage and $\mathcal{O}(n^3)$ flops to solve exactly.
  - This motivates a need for a storage efficient approximation.
- On a uniform mesh, this matrix has Toeplitz structure, which has known algorithms for efficient storage and fast mat-vecs.
  - Uniform meshes can still have issues with singularities around the boundary even with smooth inputs.
- If we use an adaptive mesh, we could minimize the singularities and get hierarchical structure, leading to low rank off-diagonal blocks. But this is still an appoximation.
- To form a better approximation, we propose a novel tensor based method to construct a matrix that uses less memory to give a better approximation.

Matrix Properties

Definition (Kronecker Product)

Let $\mathbf{A} \in \mathbb{R}^{m \times p}$, $\mathbf{B} \in \mathbb{R}^{n \times l}$. Then the Kronecker Product $\mathbf{A} \otimes \mathbf{B} \in \mathbb{R}^{(mn)\times(pl)}$ is denoted as

$$\mathbf{A} \otimes \mathbf{B} = \begin{pmatrix} a_{11}\mathbf{B} & a_{12}\mathbf{B} & \cdots & a_{1p}\mathbf{B} \\ a_{21}\mathbf{B} & a_{22}\mathbf{B} & \cdots & a_{2p}\mathbf{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}\mathbf{B} & a_{m2}\mathbf{B} & \cdots & a_{mp}\mathbf{B} \end{pmatrix} \tag{1}$$

Definition (Relative Error)

Let $\hat{\mathcal{A}}$ be an approximation to $\mathcal{A}$. The relative error is the norm of the difference between $\mathcal{A} - \hat{\mathcal{A}}$ divided by the norm of the original, namely $\frac{\|\mathcal{A}-\hat{\mathcal{A}}\|}{\|\mathcal{A}\|}$. We use the Frobenius norm for this project.

What is a Tensor?

Definition (Tensor)

A tensor is a multidimensional array of numbers.

Example

- A scalar, $c \in \mathbb{R}$ is a zeroth order tensor.
- A vector, $\vec{v} \in \mathbb{R}^n$ is a first order tensor.
- A matrix, $\mathbf{A} \in \mathbb{R}^{m \times n}$ is a second order tensor.
- A tensor, $\mathcal{T} \in \mathbb{R}^{m \times p \times n}$ is a third order tensor.

Definition (Tensor Order)

The tensor $\mathcal{T} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ is a d^{th}-order tensor, sometimes also read as d-way tensor. The order corresponds to the dimension of tensor.

Tensor Properties

Definition (k^{th}-mode Tensor Unfolding)

Let $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ be a d-way tensor. Then the k^{th}-mode unfolding is defined as

$$\mathbf{A}_{(k)} \in \mathbb{R}^{n_k \times n_1 n_2 \cdots n_{k-1} n_{k+1} \cdots n_d} \tag{2}$$

Definition (mode-k product)

The mode-k product is a way of denoting a tensor-matrix product, where the tensor is unfolded in the k^{th} mode and left multiplied by a matrix, assuming matrix dimensions match. Mathematically,

$$\mathcal{A} \times_k \mathbf{U} := \mathbf{U}\mathbf{A}_{(k)} \tag{3}$$

Tensor Examples

Let $\mathcal{A} \in \mathbb{R}^{2\times2\times2}$, where the frontal slices of the tensor are:

$$\mathcal{A}_{::1} = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \quad \mathcal{A}_{::2} = \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix} \tag{4}$$

Then the following are the k^{th}-mode unfoldings ("matricizations")

$$\mathbf{A}_{(1)} = \begin{pmatrix} 1 & 2 & 5 & 6 \\ 3 & 4 & 7 & 8 \end{pmatrix} \tag{5}$$

$$\mathbf{A}_{(2)} = \begin{pmatrix} 1 & 3 & 5 & 7 \\ 2 & 4 & 6 & 8 \end{pmatrix} \tag{6}$$

$$\mathbf{A}_{(3)} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{pmatrix} \tag{7}$$

Tensor Decompositions

Definition (Higher Order Singular Value Decomposition (HOSVD))

We perform an SVD on each unfolding, keeping the left singular vectors, denoted $\mathbf{U}, \mathbf{V}, \mathbf{W}$ respectively. Then the core tensor $\mathcal{G}$ is computed by

$$\mathcal{G} := \mathcal{A} \times_1 \mathbf{U}^T \times_2 \mathbf{V}^T \times_3 \mathbf{W}^T \tag{8}$$

Once we have the core tensor, we can truncated it in any mode possible to get a tensor approximation, namely

$$\hat{\mathcal{A}} \approx \mathcal{G} \times_1 \mathbf{U} \times_2 \mathbf{V} \times_3 \mathbf{W} \tag{9}$$

Matrix to Tensor Bijective Mapping

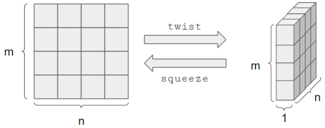


Figure: The bijective mapping between an $m \times n$ matrix and an $m \times 1 \times n$ tensor.

Theorem

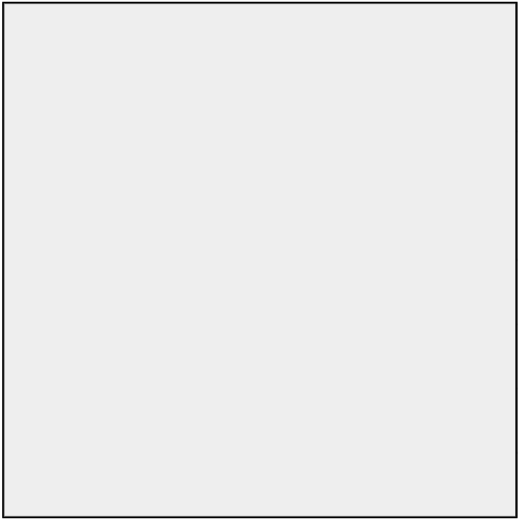
The error between the original matrix $\mathbf{A}$ *and the matrix form of the tensor approximation* $\hat{\mathbf{A}}$ *is the same error as the tensor* $\mathcal{T}$ *and tensor approximation* $\hat{\mathcal{T}}$ *in the Frobenious norm.*[a]

[a] M. Kilmer and A. Saibaba, "Structured Matrix Approximations via Tensor Decompositions," arXiv:2105.01170 [math.NA], May 2021

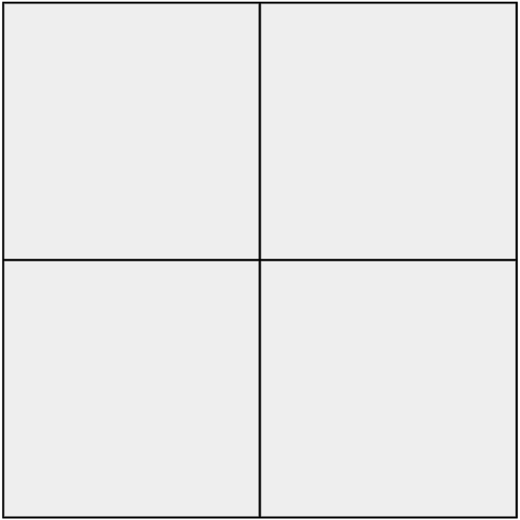
Algorithmic approach to our Proposed Method

1. Divide the Matrix
2. Form Tensors at different levels
3. Compress with Higher Order SVD
4. Map back to a Matrix

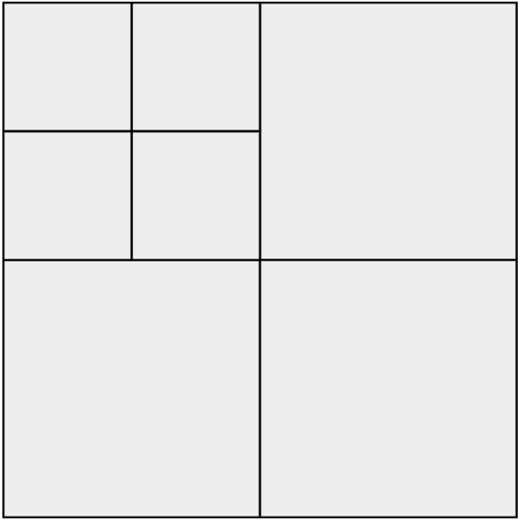
Divide the Matrix (Diagonal + Low Rank)



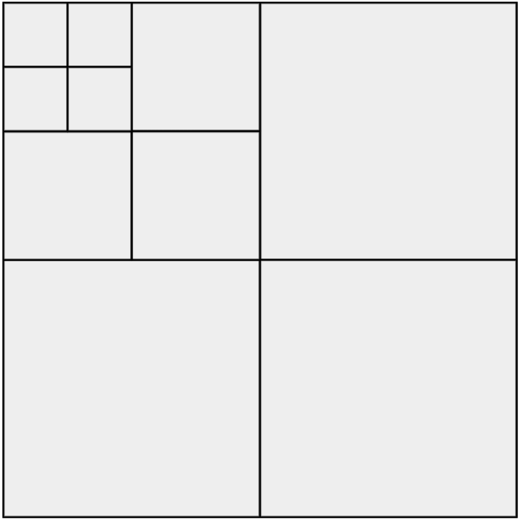
Divide the Matrix (Diagonal + Low Rank)



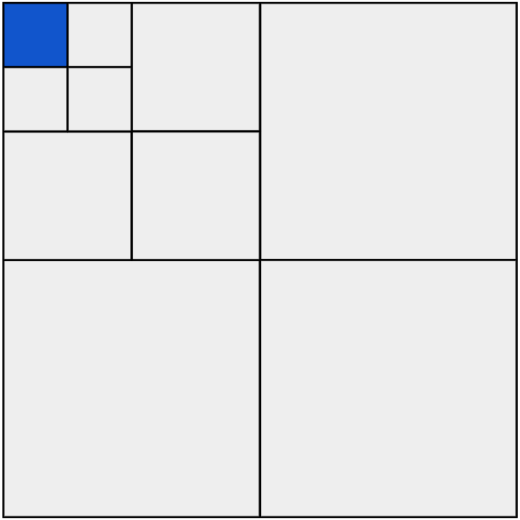
Divide the Matrix (Diagonal + Low Rank)



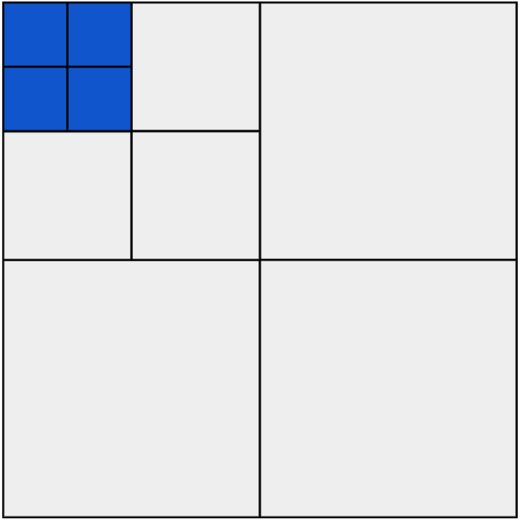
Divide the Matrix (Diagonal + Low Rank)



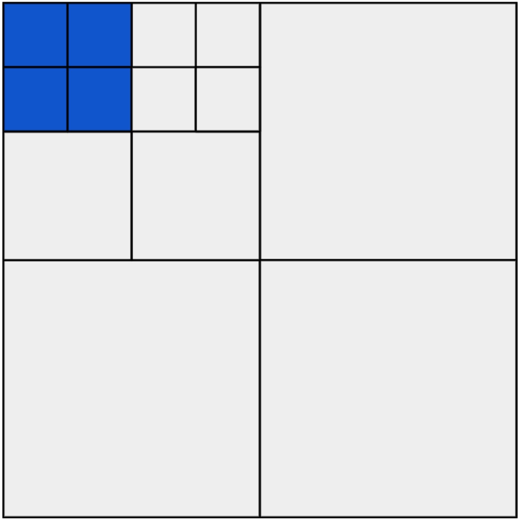
Divide the Matrix (Diagonal + Low Rank)



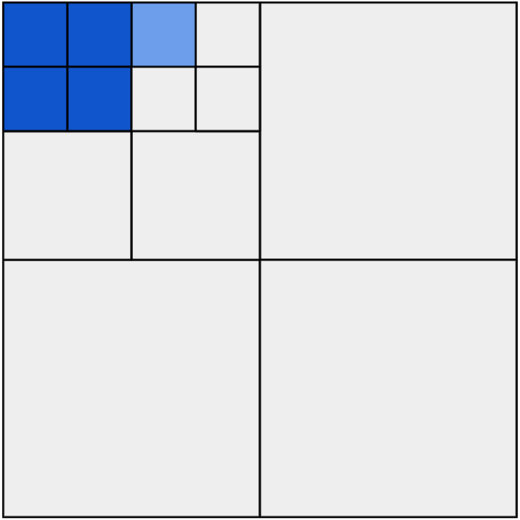
Divide the Matrix (Diagonal + Low Rank)



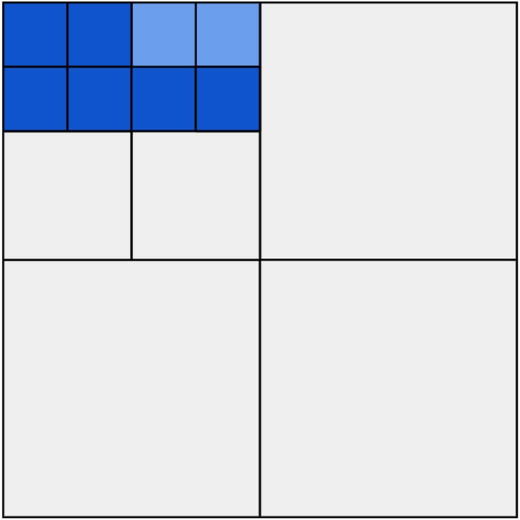
Divide the Matrix (Diagonal + Low Rank)



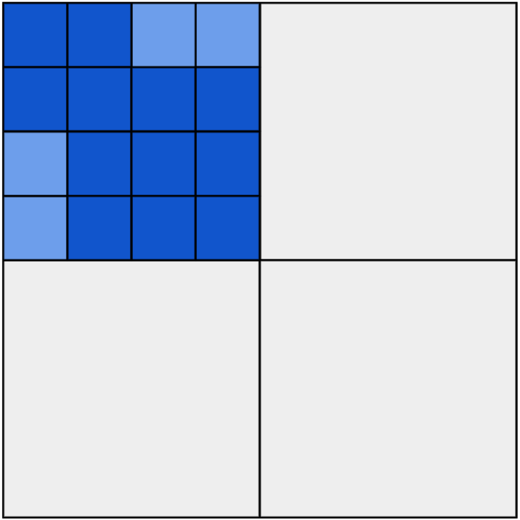
Divide the Matrix (Diagonal + Low Rank)



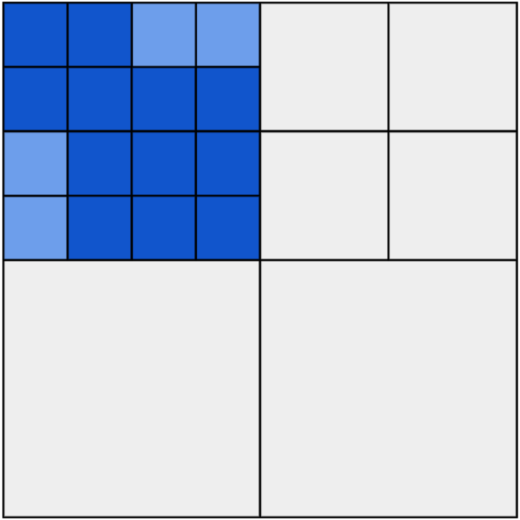
Divide the Matrix (Diagonal + Low Rank)



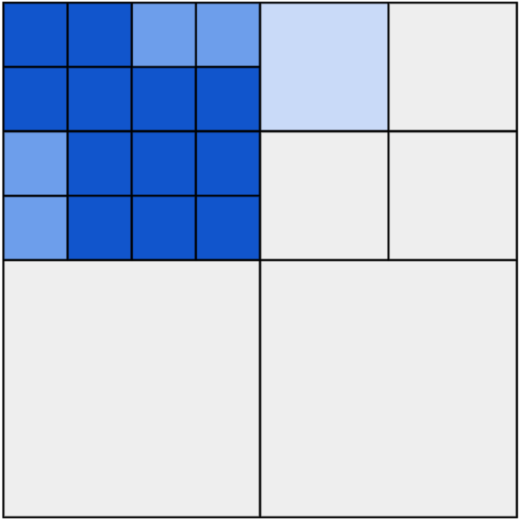
Divide the Matrix (Diagonal + Low Rank)



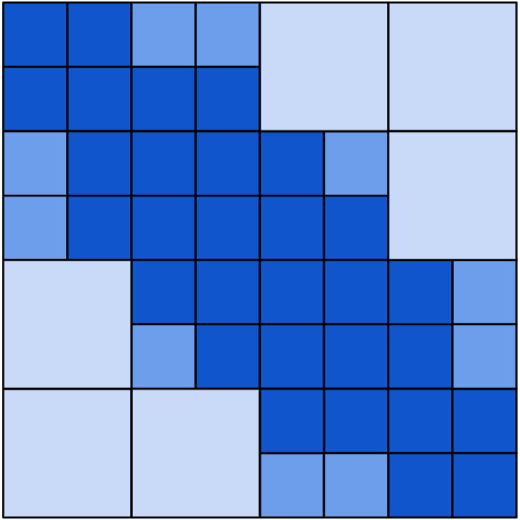
Divide the Matrix (Diagonal + Low Rank)



Divide the Matrix (Diagonal + Low Rank)



Divide the Matrix (Diagonal + Low Rank)

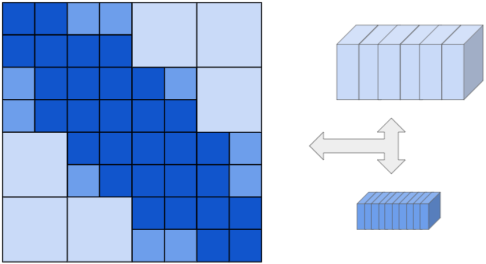


Algorithmic approach to our Proposed Method

1. Divide the Matrix
2. Form Tensors at different levels
3. Compress with Higher Order SVD
4. Map back to a Matrix

Form Tensors at different levels

- This hierarchical structured approximation of the true stiffness matrix allows us to identify candidate submatrices to twist into a tensor.
- These submatrices share more properties than just rank and size, and we plan to use these higher dimensional properties to make our lives easier.



Algorithmic approach to our Proposed Method

1. Divide the Matrix
2. Form Tensors at different levels
3. Compress with Higher Order SVD
4. Map back to a Matrix

Compress with Higher Order SVD

- Using either storage or approximation thresholds, we change the structure of the tensor approximations of these ill-conditioned adaptive meshes.

Algorithmic approach to our Proposed Method

1. Divide the Matrix
2. Form Tensors at different levels
3. Compress with Higher Order SVD
4. Map back to a Matrix

Map back to a Matrix

- This leads to a structured factorization.
- We can treat it as a sum of kronecker products.
  - Using kronecker properties, we can make this a cheaper representation.

Comparing our preliminary results with current literature

- Using a similar relative error/ compression benchmark, our method has better preconditioning, lower error, and better storage properties.

Method	**Literature**	**Proposed**
Rel Error	1.5905e-5	1.2217e-5
Compression	10.16%	52.67%

Method	**Literature**	**Proposed**
Compression	10.16%	11.13%
Rel Error	1.5905e-5	1.8462e-12

- This is due to exploiting the multidimensional structure of the data, considering it all at once, and not at individual blocks.

X. Zhao *et.al.* "Adaptive finite element method for fractional differential equations using hierarchical matrices," Comput. Methods Appl. Mech. Engrg.325, pp. 56-76, (2017)

Summary and Future Work

- Right now, our toy problem is small and easy to compute, so we are looking at kronecker based SMW methods to never form the inverse.
  - This leads us to use this method to produce $\hat{\mathbf{A}}$ as a preconditioner for the original $\mathbf{A}$.
- We are looking at using integer programming to determine the optimal truncation rank and weights across the matrix unfoldings and the two levels of tensors.
- We are also looking for techniques in constructing the tensor to ensure even if the adaptive mesh matrix is more general, our tensor-based approach still works.

Questions?

- Thank you organizers for this opportunity.
- Thank *you* for coming to my talk.
- Are there any questions?
- If you have any questions after this talk, you can reach me at
  - mitchell.scott@tufts.edu