Integral Equation Methods for Numerical Solutions of Partial Differential Equations

Zewen Shen

University of Toronto

August, 2020

Learning Objectives

By the end of this talk, participants should be able to:

1. Be more familiar with the development of numerical methods for PDEs.
2. Understand the workflow of integral equation methods.
3. Know the advantages and disadvantages of integral equation methods, and use it appropriately in future research.
4. Know a subset of active problems arising from this field.

Introduction

What are the requirements of numerical methods for PDEs?

Introduction

What are the requirements of numerical methods for PDEs?

- Accuracy

Introduction

What are the requirements of numerical methods for PDEs?

- Accuracy
- Speed

Introduction

What are the requirements of numerical methods for PDEs?

- Accuracy
- Speed
- Memory

Introduction

What are the requirements of numerical methods for PDEs?

- Accuracy
- Speed
- Memory

These requirements don't hold universally.

Integral equation formulations of PDEs

- Finding a good integral equation formulation requires work.

Integral equation formulations of PDEs

- Finding a good integral equation formulation requires work.
- There exist different integral equation formulations for a same PDE.

Integral equation formulations of PDEs

- Finding a good integral equation formulation requires work.
- There exist different integral equation formulations for a same PDE.

What's the advantage of the formulations?

Example: Laplace's equation

Definition

Laplace's equation with Dirichlet boundary condition

$$-\nabla^2 u(x) = 0 \quad \text{for } x \in \Omega \tag{1}$$

$$u(x) = f(x) \quad \text{for } x \in \partial\Omega \tag{2}$$

We want to find a solution to it that takes the form

$$u(x) = \int_{\partial\Omega} d(x,y)\sigma(y)ds(y), \quad x \in \Omega,$$

where $d(x,y)$ is the normal derivative to the Green's function of Laplace's equation at the boundary point y
($d(x,y) = n(y) \cdot \nabla_y \phi(x-y) = \frac{n(y)\cdot(x-y)}{2\pi|x-y|^2}$ and $n(y)$ is the unit length inwards pointing normal to $\partial\Omega$ at $y \in \partial\Omega$).

Integral equation formulations of Laplace's equation

We want to find a solution to it that takes the form

$$u(x) = \int_{\partial\Omega} d(x,y)\sigma(y)ds(y), \quad x \in \Omega$$

Questions

Integral equation formulations of Laplace's equation

We want to find a solution to it that takes the form

$$u(x) = \int_{\partial\Omega} d(x,y)\sigma(y)ds(y), \quad x \in \Omega$$

Questions

- Q: Why can we assume that the solution is written in this weird integral form?

Integral equation formulations of Laplace's equation

We want to find a solution to it that takes the form

$$u(x) = \int_{\partial\Omega} d(x,y)\sigma(y)ds(y), \quad x \in \Omega$$

Questions

- Q: Why can we assume that the solution is written in this weird integral form?
- Q: Why do we pick this specific kernel function $d(x,y)$ instead of others?

Integral equation formulations of Laplace's equation

We want to find a solution to it that takes the form

$$u(x) = \int_{\partial\Omega} d(x, y)\sigma(y)ds(y), \quad x \in \Omega$$

Summary

1. Compute the correct σ such that the integral above matches the boundary condition.

Integral equation formulations of Laplace's equation

We want to find a solution to it that takes the form

$$u(x) = \int_{\partial\Omega} d(x,y)\sigma(y)ds(y), \quad x \in \Omega$$

Summary

1. Compute the correct σ such that the integral above matches the boundary condition.
2. Once σ is known, evaluate the PDE solution $u(x)$ everywhere inside the domain by computing the line integral.

Integral equation formulations of Laplace's equation

We want to find a solution to it that takes the form

$$u(x) = \int_{\partial\Omega} d(x,y)\sigma(y)ds(y), \quad x \in \Omega$$

Summary

1. Compute the correct σ such that the integral above matches the boundary condition.
2. Once σ is known, evaluate the PDE solution $u(x)$ everywhere inside the domain by computing the line integral.

Integral equation formulations of Laplace's equation

How to solve the density function $\sigma(y)$ defined on the boundary?

Apply the boundary condition: $u(x) = f(x) \quad \text{for } x \in \partial\Omega,$

Integral equation formulations of Laplace's equation

How to solve the density function $\sigma(y)$ defined on the boundary?

Apply the boundary condition: $u(x) = f(x) \quad \text{for } x \in \partial\Omega,$
or more rigorously: $\lim_{x' \to x, x' \in \Omega} u(x') = f(x) \quad \text{for } x \in \partial\Omega$

Integral equation formulations of Laplace's equation

How to solve the density function $\sigma(y)$ defined on the boundary?

Apply the boundary condition: $u(x) = f(x) \quad$ for $x \in \partial\Omega$,
or more rigorously: $\lim_{x' \to x, x' \in \Omega} u(x') = f(x) \quad$ for $x \in \partial\Omega$

Remarks

Our kernel function $d(x, y)$ has a singularity as $x \to y$, which leads to an extra term as we take the limit

$$\lim_{x' \to x, x' \in \Omega} u(x') = -\frac{1}{2}\sigma(x) + \int_{\partial\Omega} d(x, y)\sigma(y)ds(y).$$

Integral equation formulations of Laplace's equation

How to solve the density function $\sigma(y)$ defined on the boundary?

Apply the boundary condition: $u(x) = f(x) \quad$ for $x \in \partial\Omega$,
or more rigorously: $\lim_{x' \to x, x' \in \Omega} u(x') = f(x) \quad$ for $x \in \partial\Omega$

Remarks

Our kernel function $d(x, y)$ has a singularity as $x \to y$, which leads to an extra term as we take the limit

$$\lim_{x' \to x, x' \in \Omega} u(x') = -\frac{1}{2}\sigma(x) + \int_{\partial\Omega} d(x, y)\sigma(y) ds(y).$$

Therefore, we got the integral equation that $\sigma(x)$ satisfies:

$$-\frac{1}{2}\sigma(x) + \int_{\partial\Omega} d(x, y)\sigma(y) ds(y) = f(x), \quad x \in \partial\Omega.$$

Integral equation formulations of Laplace's equation

References

- J. Levandosky, Notes on potential theory, Stanford University, 2003.
- P.G. Martinsson, Direct Solvers for Integral Equations, CBMS/NSF Conference on Fast Direct Solvers, 2014.
- P.G. Martinsson, Fast Direct Solvers for Elliptic PDEs, SIAM, 2019

Discretization of the integral equation

The density function $\sigma(x)$ satisfies a Fredholm integral equation of second kind:

$$-\frac{1}{2}\sigma(x) + \int_{\partial\Omega} d(x,y)\sigma(y)ds(y) = f(x), \quad x \in \partial\Omega.$$

We discretize the integral equation by replacing the integral with a representative weighted sum, i.e., a selected quadrature rule:

$$\int_{\partial\Omega} \phi(y)ds(y) = \sum_{j=1}^{N} w_j \phi(x_j), \quad \text{for a certain class of } \phi.$$

Discretization of the integral equation

The density function $\sigma(x)$ satisfies a Fredholm integral equation of second kind:

$$-\frac{1}{2}\sigma(x) + \int_{\partial\Omega} d(x,y)\sigma(y)ds(y) = f(x), \quad x \in \partial\Omega.$$

We discretize the integral equation by replacing the integral with a representative weighted sum, i.e., a selected quadrature rule:

$$\int_{\partial\Omega} \phi(y)ds(y) = \sum_{j=1}^{N} w_j \phi(x_j), \quad \text{for a certain class of } \phi.$$

Then the discretized integral equation becomes a $N \times N$ linear system

$$-\frac{1}{2}\sigma(x_i) + \sum_{j=1}^{N} w_j d(x_i,x_j)\sigma(x_j) = f(x_i), \quad i = 1,2,\ldots,N.$$

Discretization of the integral equation

Examples of common quadrature rules

- Trapezoidal rule: Converge super-algebraically when the boundary condition f is C^∞ and periodic, plus the boundary of the domain is smooth. Easy to code.
- Composite Gaussian quadrature: Customized convergence rate depending on the selected order of quadrature. Allows for local refinement, which is good for domains with corners and edges.
- Singular integral quadrature: Necessary when we need to evaluate the integral of a singular kernel. We don't need it in our example, but it's required for more sophisticated problems, e.g., Helmholtz equation.

Discretization of the integral equation

To write the discretized integral equation

$$-\frac{1}{2}\sigma(x_i) + \sum_{j=1}^{N} w_j d(x_i, x_j)\sigma(x_j) = f(x_i)$$

as a $N \times N$ linear system, we define vectors $\sigma, f \in R^n$ via

$$f(i) = \sqrt{w_i} f(x_i), \quad \sigma(i) = \sqrt{w_i}\sigma(x_i),$$

and define matrix $A \in R^{N \times N}$ via

$$A(i,j) = \sqrt{w_i} d(x_i, x_j)\sqrt{w_j} - \frac{1}{2}\delta_{i,j}. \tag{3}$$

Finally, we get a linear system $A\sigma = f$ to solve.

Why do we need the $\sqrt{w}$ weighting scheme?

Such weighting scheme allows the matrix to capture the L^2 behaviour of an integral operator rather than its pointwise behavior, which leads to a better-conditioned linear system due to some intrinsic property of our integral equation.

Discretization of the integral equation

References

- J Bremer, Z Gimbutas, V Rokhlin, A nonlinear optimization procedure for generalized Gaussian quadratures, SIAM Journal on Scientific Computing, 2010
- S Hao, AH Barnett, PG Martinsson, P Young, High-order accurate methods for Nyström discretization of integral equations on smooth curves in the plane, Advances in Computational Mathematics, 2014
- J. Bremer, A fast direct solver for the integral equations of scattering theory on planar curves with corners, Journal of Computational Physics, 2012
- J. Bremer, A Nyström method for weakly singular integral operators on surfaces, Journal of Computational Physics, 2012

Fast algorithms for rank-structured matrices

Let's take a close look at the matrix A where

$$A(i,j) = \sqrt{w_i}\, d(x_i, x_j) \sqrt{w_j} - \frac{1}{2}\delta_{i,j}. \tag{4}$$

Unlike the usual sparse matrices arising from the common numerical PDE methods, our matrix is dense and rank-structured! This tells us...

Fast algorithms for rank-structured matrices

Let's take a close look at the matrix A where

$$A(i,j) = \sqrt{w_i}\, d(x_i, x_j) \sqrt{w_j} - \frac{1}{2}\delta_{i,j}. \tag{4}$$

Unlike the usual sparse matrices arising from the common numerical PDE methods, our matrix is dense and rank-structured! This tells us...

- Conventional matrix solvers will be slow.

Fast algorithms for rank-structured matrices

Let's take a close look at the matrix A where

$$A(i,j) = \sqrt{w_i}\, d(x_i, x_j) \sqrt{w_j} - \frac{1}{2}\delta_{i,j}. \tag{4}$$

Unlike the usual sparse matrices arising from the common numerical PDE methods, our matrix is dense and rank-structured! This tells us...

- Conventional matrix solvers will be slow.
- The matrix requires more storage as its columns can't be stored sparsely.

Rank-structured matrix

What's special about our matrix

$$A(i,j) = \sqrt{w_i}\, d(x_i, x_j) \sqrt{w_j} - \frac{1}{2}\delta_{i,j} \quad ?$$

Rank-structured matrix

What's special about our matrix

$$A(i,j) = \sqrt{w_i}\, d(x_i, x_j) \sqrt{w_j} - \frac{1}{2}\delta_{i,j} \quad ?$$

- Each entry represents an interaction between point x_i and x_j.

Rank-structured matrix

What's special about our matrix

$$A(i,j) = \sqrt{w_i}\, d(x_i, x_j) \sqrt{w_j} - \frac{1}{2}\delta_{i,j} \quad ?$$

- Each entry represents an interaction between point x_i and x_j.
- Such interactions decay in magnitude over distances...

Rank-structured matrix

What's special about our matrix

$$A(i,j) = \sqrt{w_i}\, d(x_i, x_j) \sqrt{w_j} - \frac{1}{2}\delta_{i,j} \quad ?$$

- Each entry represents an interaction between point x_i and x_j.
- Such interactions decay in magnitude over distances...
- Maybe we can set the interaction to 0 if two points are far from each other enough?

Rank-structured matrix

What's special about our matrix

$$A(i,j) = \sqrt{w_i}\, d(x_i, x_j) \sqrt{w_j} - \frac{1}{2}\delta_{i,j} \quad ?$$

- Each entry represents an interaction between point x_i and x_j.
- Such interactions decay in magnitude over distances...
- Maybe we can set the interaction to 0 if two points are far from each other enough?

However, while the magnitudes of these interactions decay only slowly, the amount of information that they carry decays enormously fast. We call such interaction low-rank.

Rank-structured matrix

While the magnitudes of these fields decay only slowly, the amount of information that they carry decays enormously fast. We call such interaction low-rank.

Definition: Given an error tolerance ϵ, if A is a $m \times n$ low-rank matrix with numerical rank k, then we have $U \in R^{m \times k}$ and $V \in R^{k \times n}$

$$A \approx UV,$$

and the numerical rank k is the smallest integer such that $|A - UV|_2 < \epsilon$:

$$\text{numrank}(A, \epsilon) = \min\{k : \exists U \in R^{m \times k}, V \in R^{k \times n}, s.t., |A - UV|_2 < \epsilon\}.$$

Rank-structured matrix

While the magnitudes of these fields decay only slowly, the amount of information that they carry decays enormously fast. We call such interaction low-rank.

Definition: Given an error tolerance ϵ, if A is a $m \times n$ low-rank matrix with numerical rank k, then we have $U \in R^{m \times k}$ and $V \in R^{k \times n}$

$$A \approx UV,$$

and the numerical rank k is the smallest integer such that $|A - UV|_2 < \epsilon$:

$$\text{numrank}(A, \epsilon) = \min\{k : \exists U \in R^{m \times k}, V \in R^{k \times n}, s.t., |A - UV|_2 < \epsilon\}.$$

Remarks

The numerical rank of a low rank matrix is usually far lower than the actual rank, i.e., $k << \min(m, n)$.
What happens to the storage?

Rank-structured matrix

While the magnitudes of these fields decay only slowly, the amount of information that they carry decays enormously fast. We call such interaction low-rank.

Definition: Given an error tolerance ϵ, if A is a $m \times n$ low-rank matrix with numerical rank k, then we have $U \in R^{m \times k}$ and $V \in R^{k \times n}$

$$A \approx UV,$$

and the numerical rank k is the smallest integer such that $|A - UV|_2 < \epsilon$:

$$\text{numrank}(A, \epsilon) = \min\{k : \exists U \in R^{m \times k}, V \in R^{k \times n}, s.t., |A - UV|_2 < \epsilon\}.$$

Remarks

The numerical rank of a low rank matrix is usually far lower than the actual rank, i.e., $k << \min(m, n)$.

What happens to the storage? What happens to the matvec operation?

Rank-structured matrix

While the magnitudes of these fields decay only slowly, the amount of information that they carry decays enormously fast. We call such interaction low-rank.

Definition: Given an error tolerance ϵ, if A is a $m \times n$ low-rank matrix with numerical rank k, then we have $U \in R^{m \times k}$ and $V \in R^{k \times n}$

$$A \approx UV,$$

and the numerical rank k is the smallest integer such that $|A - UV|_2 < \epsilon$:

$$\text{numrank}(A, \epsilon) = \min\{k : \exists U \in R^{m \times k}, V \in R^{k \times n}, s.t., |A - UV|_2 < \epsilon\}.$$

Remarks

The numerical rank of a low rank matrix is usually far lower than the actual rank, i.e., $k << \min(m, n)$.

What happens to the storage? What happens to the matvec operation?

Low-rank approximation

How do we compute a low-rank approximation of the matrix?

Examples

- SVD: Optimal! (by Eckart-Young-Mirsky Theorem) But...
- Randomized range finder: Fast. Receiving more and more attentions nowadays. However, its advantage is not obvious compared to the next example in our application...
- Interpolatory decomposition: Widely used by the integral equation method researchers! Nearly optimal, reasonably fast, no need to construct the whole matrix A when integrated into our algorithm (by exploiting the analytical property of our kernel...)

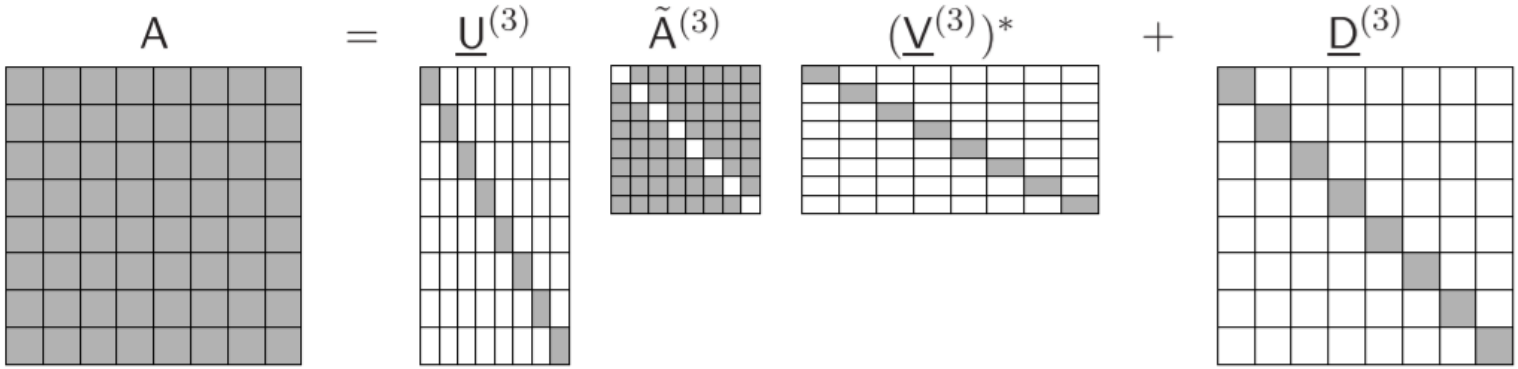
Rank-structured matrix and low-rank approximation

References

- A. Kloeckner, Tools for Low-Rank Linear Algebra, Fast Algorithms and Integral Equation Methods, Fall 2019
- P.G. Martinsson, Fast Direct Solvers for Elliptic PDEs, SIAM, 2019

Low-rank approximation of our matrix

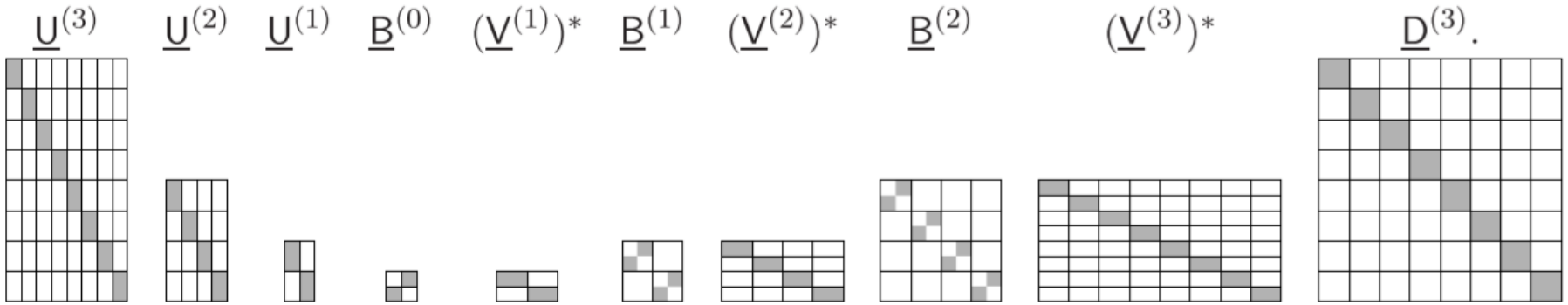
Low-rank approximation of the first level (cited from [Gillman, et al])



Final Low-rank approximation (cited from [Gillman, Young, Martinsson])

$$\underline{A} = \underline{U}^{(3)}\left(\underline{U}^{(2)}\left(\underline{U}^{(1)}\,\underline{B}^{(0)}\,(\underline{V}^{(1)})^* + \underline{B}^{(1)}\right)(\underline{V}^{(2)})^* + \underline{B}^{(2)}\right)(\underline{V}^{(3)})^* + \underline{D}^{(3)}. \tag{4.4}$$

The block structure of the right hand side of (4.4) is:

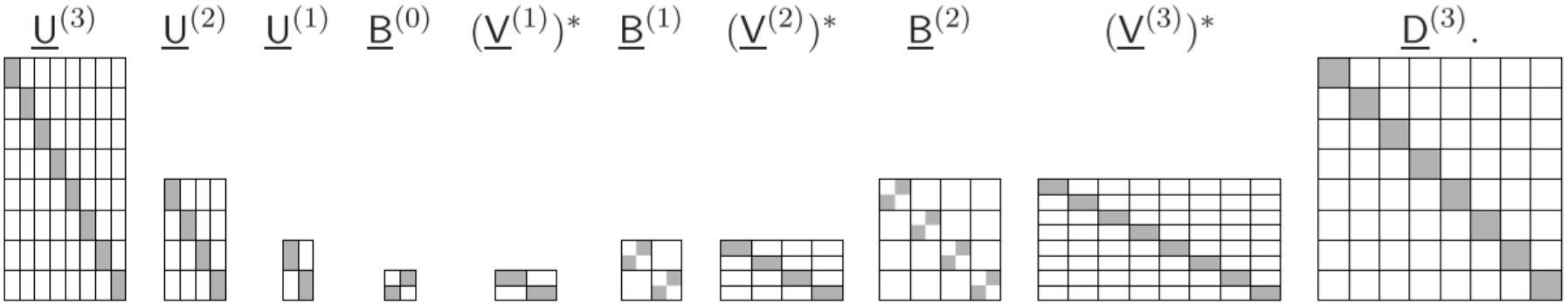


Low-rank approximation of our matrix

Final Low-rank approximation (cited from [Gillman, Young, Martinsson])

$$\mathsf{A} = \underline{\mathsf{U}}^{(3)}\left(\underline{\mathsf{U}}^{(2)}\left(\underline{\mathsf{U}}^{(1)}\,\underline{\mathsf{B}}^{(0)}\,(\underline{\mathsf{V}}^{(1)})^{*} + \underline{\mathsf{B}}^{(1)}\right)(\underline{\mathsf{V}}^{(2)})^{*} + \underline{\mathsf{B}}^{(2)}\right)(\underline{\mathsf{V}}^{(3)})^{*} + \underline{\mathsf{D}}^{(3)}. \tag{4.4}$$

The block structure of the right hand side of (4.4) is:



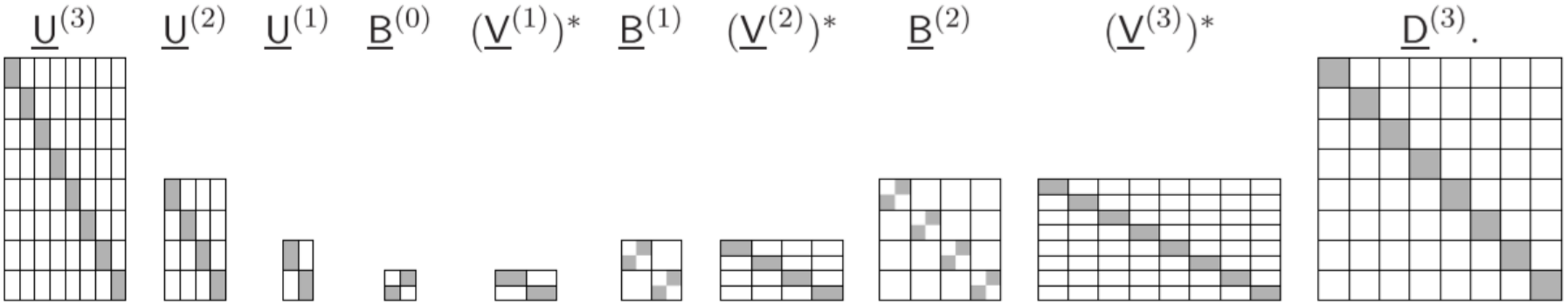
- Time complexity of the factorization: $O(Nk^2)$. No need for first constructing the whole matrix A.

Low-rank approximation of our matrix

Final Low-rank approximation (cited from [Gillman, Young, Martinsson])

$$\mathsf{A} = \underline{\mathsf{U}}^{(3)}\left(\underline{\mathsf{U}}^{(2)}\left(\underline{\mathsf{U}}^{(1)}\,\underline{\mathsf{B}}^{(0)}\,(\underline{\mathsf{V}}^{(1)})^{*} + \underline{\mathsf{B}}^{(1)}\right)(\underline{\mathsf{V}}^{(2)})^{*} + \underline{\mathsf{B}}^{(2)}\right)(\underline{\mathsf{V}}^{(3)})^{*} + \underline{\mathsf{D}}^{(3)}. \tag{4.4}$$

The block structure of the right hand side of (4.4) is:



- Time complexity of the factorization: $O(Nk^2)$. No need for first constructing the whole matrix A.
- Time complexity of matvec: $O(kN)$.

Iterative methods for rank-structured matrices

The most classical approach that people used to compute Ax in an iterative solver is the fast multipole method (one of the top 10 algorithms in 20th century), which solves N-body problems in $O(N)$ time.

Iterative methods for rank-structured matrices

The most classical approach that people used to compute Ax in an iterative solver is the fast multipole method (one of the top 10 algorithms in 20th century), which solves N-body problems in $O(N)$ time.

$$A(i,j) = \sqrt{w_i}\, d(x_i, x_j) \sqrt{w_j} - \frac{1}{2}\delta_{i,j}$$

Iterative methods for rank-structured matrices

The most classical approach that people used to compute Ax in an iterative solver is the fast multipole method (one of the top 10 algorithms in 20th century), which solves N-body problems in $O(N)$ time.

$$A(i,j) = \sqrt{w_i}\, d(x_i, x_j) \sqrt{w_j} - \frac{1}{2}\delta_{i,j}$$

- Analytical expansion vs matrix factorization.

Iterative methods for rank-structured matrices

The most classical approach that people used to compute Ax in an iterative solver is the fast multipole method (one of the top 10 algorithms in 20th century), which solves N-body problems in $O(N)$ time.

$$A(i,j) = \sqrt{w_i}\, d(x_i, x_j) \sqrt{w_j} - \frac{1}{2}\delta_{i,j}$$

- Analytical expansion vs matrix factorization.
- Strong admissibility vs weak admissibility.

Iterative methods for rank-structured matrices

References

- A. Gillman, P. Young, P.G. Martinsson, A direct solver with O(N) complexity for integral equations on one-dimensional domains, Front. Math. China, 7 (2012), pp. 217–247.
- R. Beatson, L. Greengard, A short course on fast multipole methods
- L. Greengard, V. Rokhlin, A fast algorithm for particle simulations, Journal of Computational Physics, 1987

Direct solver for rank-structured matrices

Why do we need a direct solver?

Direct solver for rank-structured matrices

Why do we need a direct solver?

- Multiple RHS.

Direct solver for rank-structured matrices

Why do we need a direct solver?

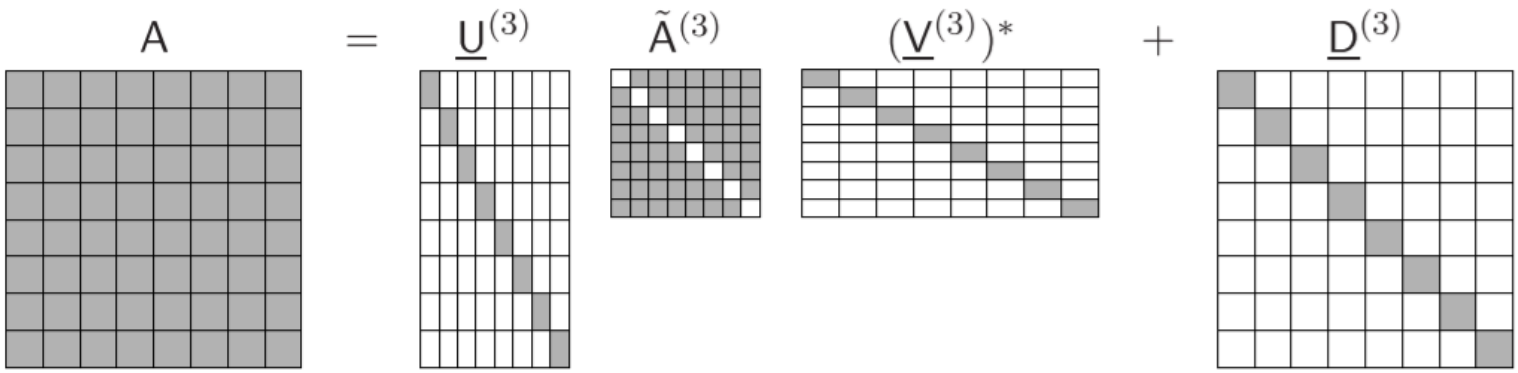
- Multiple RHS.
- Independent of condition number.

Direct solver for rank-structured matrices

Why do we need a direct solver?

- Multiple RHS.
- Independent of condition number.

Next question: How to compute the inverse of our factorized matrix?



There's a name for such factorization...

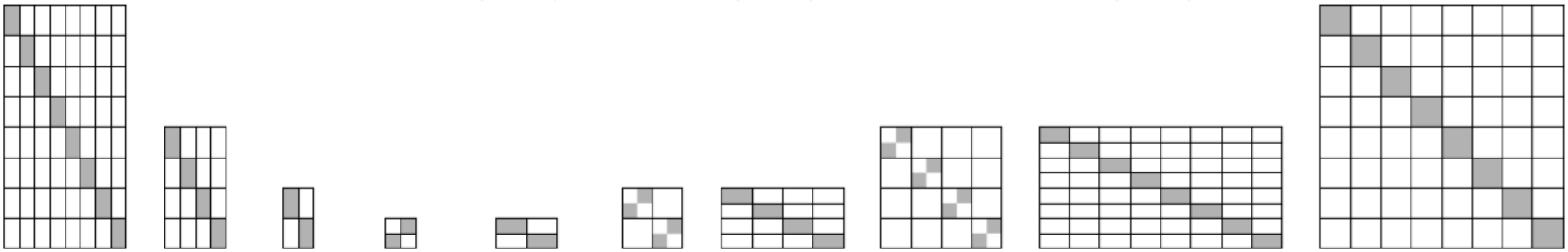
Direct solver for rank-structured matrices

Final Low-rank approximation (cited from [Gillman, Young, Martinsson])

$$\mathsf{A} = \underline{\mathsf{U}}^{(3)}\big(\underline{\mathsf{U}}^{(2)}\big(\underline{\mathsf{U}}^{(1)}\,\underline{\mathsf{B}}^{(0)}\,(\underline{\mathsf{V}}^{(1)})^* + \underline{\mathsf{B}}^{(1)}\big)(\underline{\mathsf{V}}^{(2)})^* + \underline{\mathsf{B}}^{(2)}\big)(\underline{\mathsf{V}}^{(3)})^* + \underline{\mathsf{D}}^{(3)}. \tag{4.4}$$

The block structure of the right hand side of (4.4) is:

$\underline{\mathsf{U}}^{(3)}$ $\underline{\mathsf{U}}^{(2)}$ $\underline{\mathsf{U}}^{(1)}$ $\underline{\mathsf{B}}^{(0)}$ $(\underline{\mathsf{V}}^{(1)})^*$ $\underline{\mathsf{B}}^{(1)}$ $(\underline{\mathsf{V}}^{(2)})^*$ $\underline{\mathsf{B}}^{(2)}$ $(\underline{\mathsf{V}}^{(3)})^*$ $\underline{\mathsf{D}}^{(3)}$.



- Sherman–Morrison-Woodbury formula

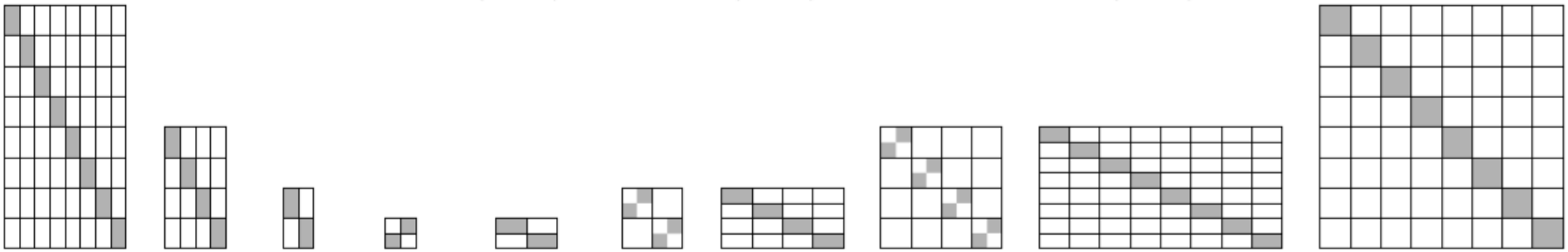
Direct solver for rank-structured matrices

Final Low-rank approximation (cited from [Gillman, Young, Martinsson])

$$A = \underline{U}^{(3)}\left(\underline{U}^{(2)}\left(\underline{U}^{(1)}\,\underline{B}^{(0)}\,(\underline{V}^{(1)})^* + \underline{B}^{(1)}\right)(\underline{V}^{(2)})^* + \underline{B}^{(2)}\right)(\underline{V}^{(3)})^* + \underline{D}^{(3)}. \tag{4.4}$$

The block structure of the right hand side of (4.4) is:

$\underline{U}^{(3)}$ $\underline{U}^{(2)}$ $\underline{U}^{(1)}$ $\underline{B}^{(0)}$ $(\underline{V}^{(1)})^*$ $\underline{B}^{(1)}$ $(\underline{V}^{(2)})^*$ $\underline{B}^{(2)}$ $(\underline{V}^{(3)})^*$ $\underline{D}^{(3)}$.



- Sherman–Morrison-Woodbury formula
- Time complexity of computing inverse: $O(Nk^2)$

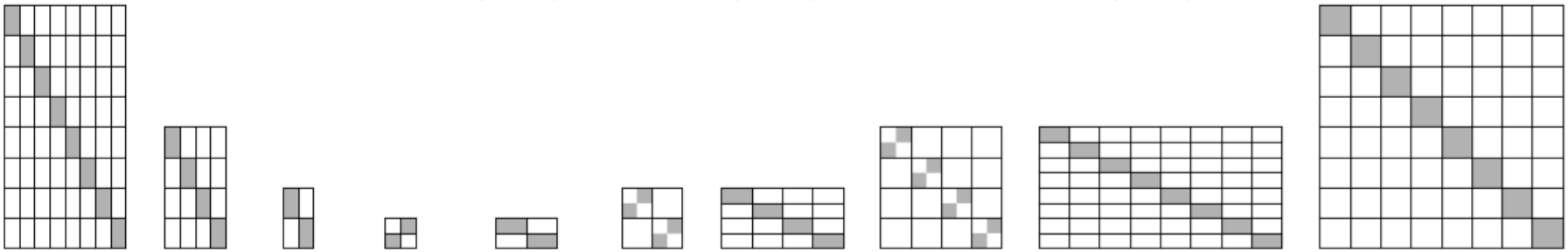
Direct solver for rank-structured matrices

Final Low-rank approximation (cited from [Gillman, Young, Martinsson])

$$\mathsf{A} = \underline{\mathsf{U}}^{(3)}\big(\underline{\mathsf{U}}^{(2)}\big(\underline{\mathsf{U}}^{(1)}\,\underline{\mathsf{B}}^{(0)}\,(\underline{\mathsf{V}}^{(1)})^* + \underline{\mathsf{B}}^{(1)}\big)(\underline{\mathsf{V}}^{(2)})^* + \underline{\mathsf{B}}^{(2)}\big)(\underline{\mathsf{V}}^{(3)})^* + \underline{\mathsf{D}}^{(3)}. \tag{4.4}$$

The block structure of the right hand side of (4.4) is:

$\underline{\mathsf{U}}^{(3)}$ $\underline{\mathsf{U}}^{(2)}$ $\underline{\mathsf{U}}^{(1)}$ $\underline{\mathsf{B}}^{(0)}$ $(\underline{\mathsf{V}}^{(1)})^*$ $\underline{\mathsf{B}}^{(1)}$ $(\underline{\mathsf{V}}^{(2)})^*$ $\underline{\mathsf{B}}^{(2)}$ $(\underline{\mathsf{V}}^{(3)})^*$ $\underline{\mathsf{D}}^{(3)}$.



- Sherman–Morrison-Woodbury formula
- Time complexity of computing inverse: $O(Nk^2)$
- Time complexity of applying the inverse: $O(Nk)$

Direct solver for rank-structured matrices

References

- J. Bremer, A. Gillman, P.G. Martinsson, A high-order accurate accelerated direct solver for acoustic scattering from surfaces, BIT Numerical Mathematics, 2015
- A. Gillman, P. Young, P.G. Martinsson, A direct solver with O(N) complexity for integral equations on one-dimensional domains, Front. Math. China, 7 (2012), pp. 217–247.
- P.G. Martinsson, Fast Direct Solvers for Elliptic PDEs, SIAM, 2019
- Z. Shen, lap2d: A Fast Direct Dense Solver for 2-D Laplace's Equation, 2020

Summary

Advantages

- High-order convergence rate plus reduction of dimensionality.

Summary

Advantages

- High-order convergence rate plus reduction of dimensionality.
- Be able to achieve machine precision accuracy.

Summary

Advantages

- High-order convergence rate plus reduction of dimensionality.
- Be able to achieve machine precision accuracy.
- Works well for exterior problems (acoustics, electromagnetic scattering).

Summary

Advantages

- High-order convergence rate plus reduction of dimensionality.
- Be able to achieve machine precision accuracy.
- Works well for exterior problems (acoustics, electromagnetic scattering).
- Capability of exploiting all kinds of analytical properties of PDEs. Nice combination between analysis and numerical linear algebra!

Summary

Disadvantages

- Harder to code.

Summary

Disadvantages

- Harder to code.
- Special considerations need to be done when the problem has body load, singular kernels, a domain with corners, mixed boundary condition, etc.

Questions?

Questions?

Thanks to prof. Kirill Serkh and Xinyue Lu for providing suggestions to the talk.