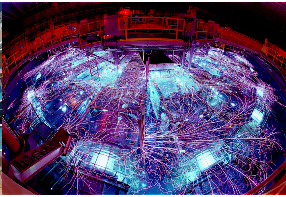
SAND2023-12753C

Exceptional service in the national interest



Algebraic multigrid solver for nonlocal equations

Christian Glusa

CCR, Sandia National Laboratories
February 27, 2023

Elliptic nonlocal operators

Let $\delta \in (0, \infty]$ be the *horizon*, $\Omega \subset \mathbb{R}^d$ a bounded open domain, define the *interaction domain*

$$\Omega_I := \{\mathbf{y} \in \mathbb{R}^d \setminus \Omega : |\mathbf{x} - \mathbf{y}| \leq \delta, \quad \text{for } \mathbf{x} \in \Omega\}.$$

We want to numerically solve equations involving the nonlocal operator

$$\mathcal{L}u(\mathbf{x}) = \text{p.v.} \int_{\Omega \cup \Omega_I} (u(\mathbf{y}) - u(\mathbf{x}))\gamma(\mathbf{x}, \mathbf{y}) d\mathbf{y}, \qquad \mathbf{x} \in \Omega,$$

with

$$\gamma(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x}, \mathbf{y}) \, |\mathbf{x} - \mathbf{y}|^{-\beta(\mathbf{x}, \mathbf{y})} \, \mathcal{X}_{|\mathbf{x}-\mathbf{y}| \leq \delta}, \qquad \mathbf{x}, \mathbf{y} \in \Omega \cup \Omega_I,$$
$$\phi(\mathbf{x}, \mathbf{y}) > 0.$$

- Examples:
  - Integral fractional Laplacian: $\phi \sim \text{const}$, $\beta = d + 2s$, $s \in (0, 1)$, $\delta = \infty$
  - Tempered fractional Laplacian: $\phi(\mathbf{x}, \mathbf{y}) \sim \exp(-\lambda |\mathbf{x} - \mathbf{y}|)$
  - Truncated fractional Laplacian: δ finite
  - Variable order fractional Laplacians with varying coefficient: $\beta(\mathbf{x}, \mathbf{y}) = d + 2s(\mathbf{x}, \mathbf{y})$, $\phi(\mathbf{x}, \mathbf{y}) > 0$
  - Integrable kernels: constant kernel ($\beta = 0$), "peridynamic" kernel ($\beta = 1$)
- Assumptions:
  - γ is symmetric.
  - Interaction domain is defined wrt ℓ_2-norm.

After FEM discretization:

$$A\mathbf{x} = \mathbf{b}, \qquad\qquad A \in \mathbb{R}^{n \times n}$$

Depending on δ and h:

- Straightforward discretization can lead to a fully dense matrix.
- Assembly and solve would have at least $\mathcal{O}(n^2)$ complexity and memory requirement.

Better approach

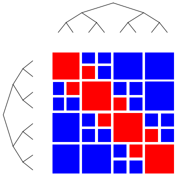
Panel clustering / Fast Multipole Method / hierarchical matrix approximation

Operator targeted for this talk:

$$(-\Delta)^s u(\mathbf{x}) = \text{p. v.} \int_{\mathbb{R}^d} d\mathbf{y}\; (u(\mathbf{x}) - u(\mathbf{y}))\, \gamma(\mathbf{x}, \mathbf{y}), \quad \mathbf{x} \in \Omega \subset \mathbb{R}^d$$

with kernel $\gamma(\mathbf{x}, \mathbf{y}) \sim 1/\left|\mathbf{x} - \mathbf{y}\right|^{d+2s}$, $\delta = \infty$ and homogeneous Dirichlet boundary conditions.

Hierarchical matrices: Admissible sub-blocks



1. Flag sub-blocks for compression
2. Construct low-rank approximations

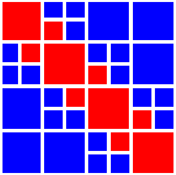
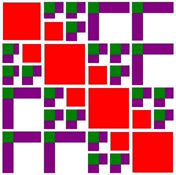
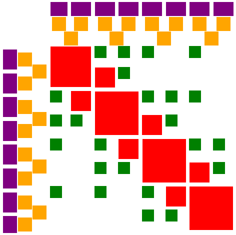
Build tree of clusters of DoFs.

- root contains all unknowns
- subdivision based on coordinates
- distributed computations: first level given by MPI distribution of unknowns

Criterion:

- Cluster pairs (P, Q) that are sufficiently separated compared to their sizes are *admissible* for compression.
- Matrix entries that are not part of any admissible blocks are assembled directly into a sparse near-field matrix A_{near}.

Hierarchical matrices: low-rank approximation

- Splitting of operator into sub-blocks based on admissibility

$$A = A_{\text{near}} + A_{\text{far}} = A_{\text{near}} + \sum_{\text{blocks}(P,Q)} A_{P,Q}$$

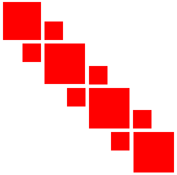
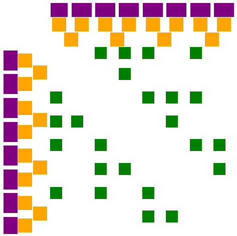
- $\mathcal{H}$-matrix approximation

$$A_{P,Q} \approx U_P \Gamma_{P,Q} U_Q^T \qquad \text{(low-rank approximation)}$$

- $\mathcal{H}^2$-matrices
  Using hierarchical nestedness of clusters, can express

$$U_P = \sum_{Q \text{ child of } P} U_Q T_{Q,P}$$

Matrix-vector product with an $\mathcal{H}^2$-matrix

$A =$  $+$ 

Steps:

- Matvec with sparse near-field matrix
- Upward recursion
- Cluster-cluster interaction
- Downward recursion

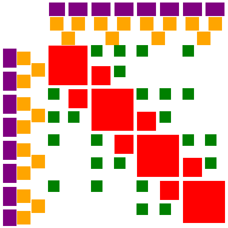
$\mathcal{H}^2$-matrix approximation

Matrix-vector product (and FE assembly) in $\mathcal{O}\left(n \log^{2d} n\right)$ operations & memory.

Representation using sparse matrices

Recast hierarchical matrices in terms of sparse matrices

- No special purpose code
- Leverage well-optimized distributed sparse linear algebra



Reindexing of far-field leads to

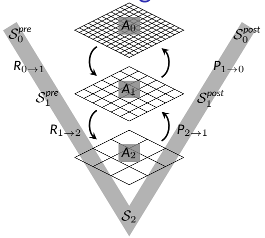
$$A \approx A_{\text{near}} + B\left[(I+T_K)^T \cdots (I+T_0)^T\right] \Gamma \left[(I+T_0)\cdots(I+T_K)\right] B^T,$$

A_{near} and Γ involve MPI communication, all other matrices are block diagonal

Solvers

- Dense direct solvers
  $\mathcal{O}(n^3)$ complexity, $\mathcal{O}(n^2)$ memory
- Hierarchical direct solvers $\mathcal{O}(n \log n)$ scaling, but often very large constant and nontrivial implementation
- Iterative solvers
  - $\mathcal{O}(n \log n)$ for single matvec
  - need preconditioners to achieve small number of iterations
  - Scalable options for elliptic PDEs:
    - Domain decomposition
    - **Multigrid**

Geometric multigrid



User specifies:

- Operators A_ℓ, assembled on hierarchy of nested meshes
- Transfer operators: prolongations $P_{\ell+1\to\ell}$, restrictions $R_{\ell\to\ell+1} = P^T_{\ell+1\to\ell}$,
- Smoothers $\mathcal{S}_\ell^{\text{pre/post}}$ (e.g. Jacobi)
- Coarse solver $\mathcal{S}_L$

How does multigrid work?

- On each level: smoother reduces high frequency error, low frequency error is transferred to coarser levels
- High/low frequency splitting depends on mesh

Drawbacks:

- Need hierarchy of nested meshes, complications for locally refined meshes
- Assembly on every level, tight coupling between assembly and solve

Smoothed Aggregation Algebraic multigrid (SA-AMG)

User specifies:

- A_0, DoF coordinates **c**, near-nullspace (constant, rigid body modes, etc)

AMG setup

- "aggregation": construction of transfer operators $P_{\ell+1\to\ell}$ using only algebraic information (e.g. matrix graph, strength of connection)
- Galerkin projection $A_{\ell+1} = P^T_{\ell+1\to\ell} A_\ell P_{\ell+1\to\ell}$

Issues for nonlocal problems:

- Usual graph algorithms used for AMG construction cannot be applied directly to $\mathcal{H}$-matrices
- Inefficient for operators that are too dense
- Hierarchical information contain in $\mathcal{H}$-matrix does not translate well into a multigrid hierarchy.

Auxiliary operator multigrid

Idea

- Construct multigrid transfer operators $P_{\ell+1,\ell}$ wrt an auxiliary matrix $\widetilde{A}_0$.
- Then construct preconditioner via Galerkin projections $A_{\ell+1} = P^T_{\ell+1\to\ell} A_\ell P_{\ell+1\to\ell}$.

Requirements for auxiliary operator:

- sparse
- contains sufficient information about nonlocal problem

Possible auxiliary operators:

- PDE Laplacian on the same mesh
- distance Laplacian on graph G of filtered near-field matrix

$$L_{ij} = \begin{cases} -1/\left|\mathbf{c}_i - \mathbf{c}_j\right| & \text{if } (i,j) \in G, i \neq j, \\ -\sum_{k\neq i} L_{ik} & \text{if } i = j, \end{cases} \quad , \mathbf{c}_i \text{ DoF coordinates}$$

- lumped and re-scaled near-field matrix

Additional operations on $\mathcal{H}^2$-matrices

- Galerkin projection:
  If

$$A \approx A_{\text{near}} + B\left[(I+T_K)^T \cdots (I+T_0)^T\right]\Gamma\left[(I+T_0)\cdots(I+T_K)\right]B^T,$$

  then

$$P^TAP \approx \underbrace{P^TA_{\text{near}}P}_{\text{multiplied out}} + \underbrace{(P^TB)}_{\text{multiplied out}}\left[(I+T_K)^T \cdots (I+T_0)^T\right]\Gamma\left[(I+T_0)\cdots(I+T_K)\right](P^TB)^T.$$

  - This is reusing the same compression of the off-rank matrix blocks.
  - Low-rank representation of small sub-blocks might not be efficient anymore.
- Recompression:
  Drop one (or more) levels of the cluster tree:

$$A \approx \left[A_{\text{near}} + B(I+T_K)^T\Gamma_K(I+T_K)B^T\right]$$
$$+ \left[B(I+T_K)^T\right]\left[(I+T_{K-1})^T \cdots (I+T_0)^T\right]\left[\Gamma-\Gamma_K\right]\left[(I+T_0)\cdots(I+T_{K-1})\right]\left[B(I+\right.$$

- Conversion to dense format: multiply it all out

$\Rightarrow$ All operations required for AMG setup use sparse matrix-matrix addition & multiplication.

Implementation details

Components:

- PyNucleus[1] for assembly of nonlocal operators
- Trilinos/Tpetra[2] for distributed sparse linear algebra
- Trilinos/Belos[2] for Krylov solvers
- Trilinos/MueLu[2] for Algebraic Multigrid
- Kokkos[3] programming model for performance portability

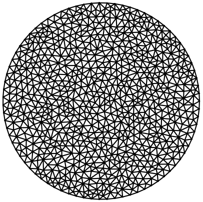
Features:

- $\mathcal{H}$- and $\mathcal{H}^2$-matrices, reader for hierarchical operators
- Krylov solvers, AMG preconditioner
- MPI distributed
- Compute architectures supported by Kokkos:
  CPU (Serial, OpenMP), GPU (Cuda, HIP, ...), ...

[1] `https://github.com/sandialabs/PyNucleus`
[2] `https://github.com/trilinos/Trilinos`
[3] `https://github.com/kokkos/kokkos`

Numerical results - CPU

Solo, SNL, Broadwell CPUs



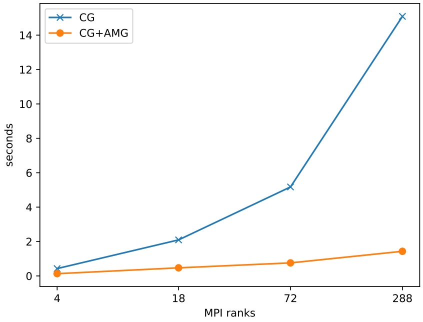
- Quasi-uniform mesh, P1 elements
- 2 Jacobi sweeps of pre-/post-smoothing
- LAPACK coarse solve

		memory (finest level)		iterations (time)	
DoFs	ranks	dense	$\mathcal{H}^2$	PDE Δ	distance Δ
12,173	4	1.1 GB	0.1 GB	8 (0.15s)	8 (0.14s)
49,139	18	18 GB	0.55 GB	8 (0.47s)	9 (0.54s)
197,565	72	291 GB	3 GB	9 (0.73s)	10 (0.84s)
792,548	288	4,680 GB	19.7 GB	9 (1.43s)	10 (1.56s)
n	n	n^2	$n \log^4 n$	constant ($\log^4 n$)	

Table: 2D fractional Poisson problem on unit disk, $s = 0.75$, $\delta = \infty$

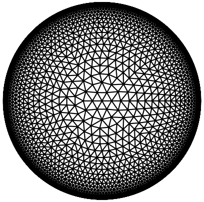
- Dense matrices only for comparison.
- Only the first two dense problems would actually fit in memory.

Numerical results - Comparison with unpreconditioned CG



- Both solvers use a $\mathcal{H}^2$-matrix.
- AMG preconditioned solve is scalable, Krylov by itself is not.

Numerical results - graded meshes



- Motivation: resolution of low regularity near domain boundary improves convergence of discretization error
- Weak scaling of solve time needs work (load balancing).

			memory (finest level)		iterations (time)
DoFs	$h_{\max}/h_{\min}$	ranks	dense	$\mathcal{H}^2$	CG+SA-AMG
15,852	105	4	1.87 GB	0.33 GB	7 (0.37s)
78,674	218	18	46.1 GB	2.4 GB	7 (1.74s)
363,472	439	72	984.3 GB	16.6 GB	8 (3.73s)

Table: 2D fractional Poisson problem on graded unit disk, $s = 0.75$, $\delta = \infty$

Numerical results - GPU

Lassen, LLNL, V100 GPUs

		memory (finest level)		iterations (time)
DoFs	ranks	dense	$\mathcal{H}^2$	CG+SA-AMG
49,139	4	18 GB	0.6 GB	9 (0.12s)
197,565	16	291 GB	2.9 GB	11 (0.29s)
792,548	64	4,680 GB	14.7 GB	12 (0.62s)
3,175,042	256	75,109 GB	61.9 GB	12 (1.79s)

Table: 2D fractional Poisson problem on unit disk, $s = 0.75$, $\delta = \infty$

- 1000x reduction in memory
- Weak scaling behavior can be improved
  (no AMG parameter tuning for GPU so far)

Conclusion:

- Algebraic multigrid for nonlocal elliptic equations in hierarchical format: scalable in iterations, memory, complexity
- Sparse matrix representation of hierarchical matrices allows to leverage a lot of existing code.

Outlook:

- Coefficients variations (AMG should be good for that!)
- Application to nonlocal operators in sparse format (finite kernel interaction horizon, peridynamics)
- Application to boundary integral equations

Thank you for your attention!