SAND2022-9435C

Sandia National Laboratories

Exceptional service in the national interest

A Matrix-Free Approach to Smoothed Aggregation Algebraic Multigrid

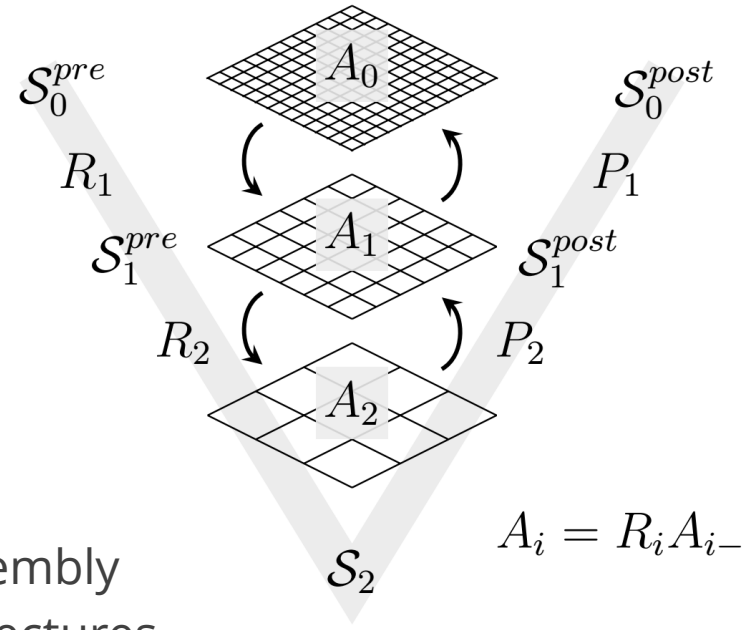
Graham Harper, 1442
Center for Computing Research
Sandia National Laboratories

8/2/2022

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Background



$$A_i = R_i A_{i-1} P_i$$

- What is multigrid?
  - One of the most efficient solvers for discretized PDE systems
  - Main idea: reduce high frequency error, then transfer to a coarser grid
- What is matrix-free?
  - Solving linear systems where a matrix is replaced by its action
  - Matrix apply operator recomputes expensive finite element assembly
  - Enables applications to push extreme scales on advanced architectures
  - Preconditioning/acceleration is critical!
- Matrix-free geometric multigrid (GMG) studied extensively
  - Kronbichler, Ljungkvist, *Multigrid for matrix-free high-order finite element computations on graphics processors*, ACM TOPC (2019)
  - Davydov, Pelteret, Arndt, et al, *A matrix-free approach for finite-strain hyperelastic problems using geometric multigrid,* IJNME (2020)
  - Brown, Barra, Beams, et al, *Performance Portable Solid Mechanics via Matrix-Free p-Multigrid,* arXiv (2022)
- GMG often requires geometric structure
- Algebraic multigrid (AMG) robustly handles grids with less geometric structure
  - Several challenges have prevented matrix-free AMG

Algebraic Multigrid

- AMG constructs coarse grids by looking at algebraic properties of an operator
  - A_{ij} represents correlation between DOF i and j

1. Aggregates constructed from operator information
   - Root node injected to coarse grid
2. Prolongator, restrictor constructed using aggregates
3. Error smoothed on the fine grid

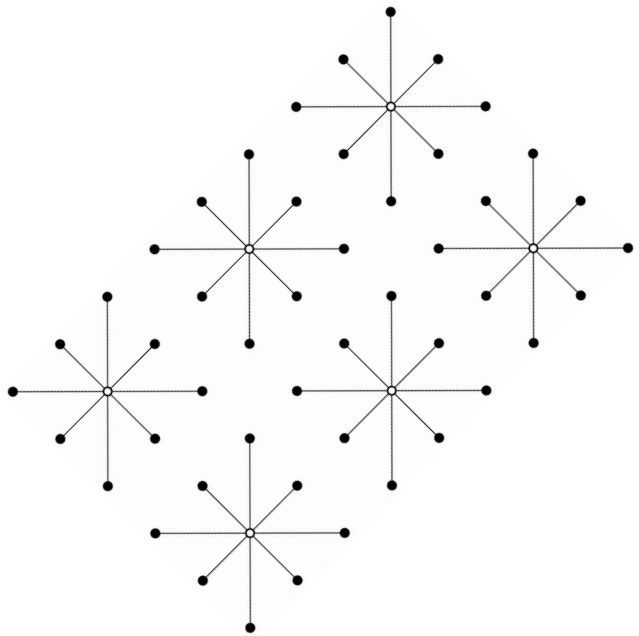
$$x \leftarrow x + \omega D^{-1}(f - Ax)$$

4. Residual transferred to the coarse grid

$$r_c = R(f - Ax)$$

5. Coarse matrix computed by RAP, solve/recurse
6. Correction from the coarse grid is applied

$$x \leftarrow x + Px_c$$



- Challenge: construct a robust AMG solver without explicitly forming A

Smoothed Aggregation

- Smoothed Aggregation AMG (SA–AMG) constructs grid transfers by **smoothed** aggregation

1. Construct tentative prolongator on aggregates
2. Compute eigenvalue estimate for $D^{-1}A$

$$\lambda_{\ell,m} = \rho(D_\ell^{-1} A_\ell)$$

3. Apply prolongator smoother to tentative prolongator

$$P_\ell = \left(I - \omega D_\ell^{-1} A_\ell\right) P_\ell^{(t)}$$

where

$$\omega = \frac{4}{3\lambda_{\ell,m}}$$

4. For symmetric problems,

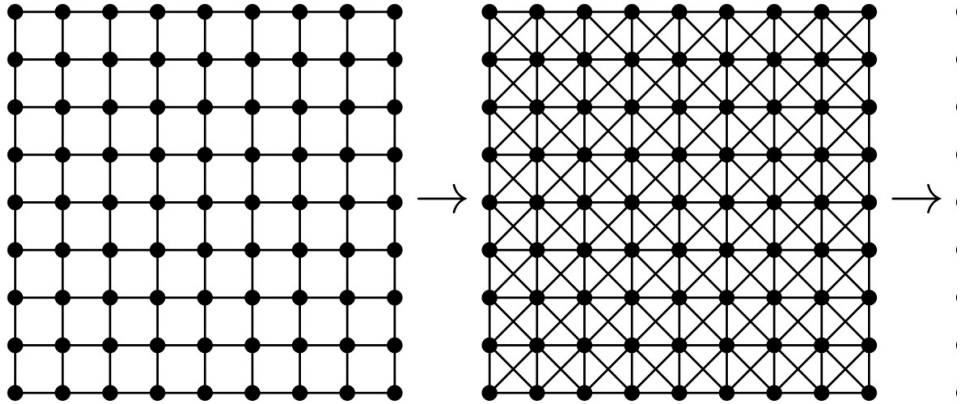
$$R_\ell = P_\ell^T$$

$$P_\ell^{(t)} = \begin{pmatrix} 1 & & \cdot & \\ 1 & & \cdot & \\ 1 & & \cdot & \\ & 1 & \cdot & \\ & 1 & \cdot & \\ & 1 & \cdot & \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ & & \cdot & 1 \\ & & \cdot & 1 \\ & & \cdot & 1 \end{pmatrix}$$

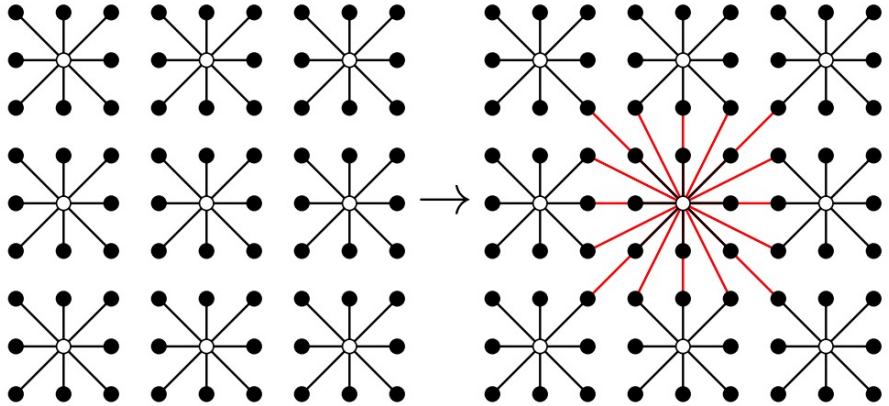
*scale columns

Matrix-Free Smoothed Aggregation

- Aggregate construction problems:
  - Often need matrix entries
  - Can cross processor boundaries
- Simple case solution:
  - Interpret mesh nodes as a graph
  - Connect nodes on same elements



- Prolongator construction problems:
  - Matrix-matrix multiply
  - Diagonal construction
  - Tentative prolongator is simple, but prolongator smoother extends basis support



- Solution:
  - Distance-2 coloring of aggregates to compress
  - Take few power method iterations, multiply by factor

$$P_\ell = \left(I - \omega D_\ell^{-1} A_\ell\right) P_\ell^{(t)}$$

Coarse Problem Construction

- Goal: form a coarse problem without performing many matrix-vector multiplies
- Problem: coarse problem takes the form

$$A_c = R_\ell A_\ell P_\ell = (P_\ell^{(t)})^T (I - \omega A_\ell^T D^{-1}) A_\ell (I - \omega D^{-1} A_\ell)(P_\ell^{(t)})^T$$

1. Form the coarse grid matrix and solve
   - not necessarily in-line with low-memory goals
   - mitigated by more aggressive coarsening
2. Solve the coarse grid iteratively
   - apply A at least 3 times per coarse grid iteration

$$P_\ell^{(t)} = \begin{pmatrix} 1 & & \cdot & \\ 1 & & \cdot & \\ 1 & & \cdot & \\ & 1 & \cdot & \\ & 1 & \cdot & \\ & 1 & \cdot & \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ & & \cdot & 1 \\ & & \cdot & 1 \\ & & \cdot & 1 \end{pmatrix}$$

*scale columns

- Graph colorings on prolongator drastically reduce required effort
- Form the coarse grid matrix by N_{colors} matrix-vector multiplies
  - N_{colors} affected by coarsening rate
- Option 1 works well with unsmoothed aggregation

2-Level Multigrid Algorithm

Matrix-free smoothed aggregation algorithm for nodal DOFs

- One-time costs:
  - Aggregation on locally owned mesh nodes
  - Color aggregates
  - Eigenvalue estimate (10 fine-grid apply)

1. Construct grid transfers
2. Apply pre-smoother (1 fine-grid apply)
3. Construct coarse matrix (N_{colors} fine-grid apply)
4. Direct solve residual on coarse grid (1 fine-grid apply to transfer)
5. Apply coarse grid correction (1 fine-grid apply to transfer)
6. Apply post-smoother (1 fine-grid apply)

Matrix-free smoothed aggregation algorithm for high-order discretizations

- Coarsen to nodal case via p-multigrid

Fine-scale apply scales based on coloring of the problem
Fine-scale apply reduced greatly for unsmoothed aggregation… but less scalable

Implementation Details

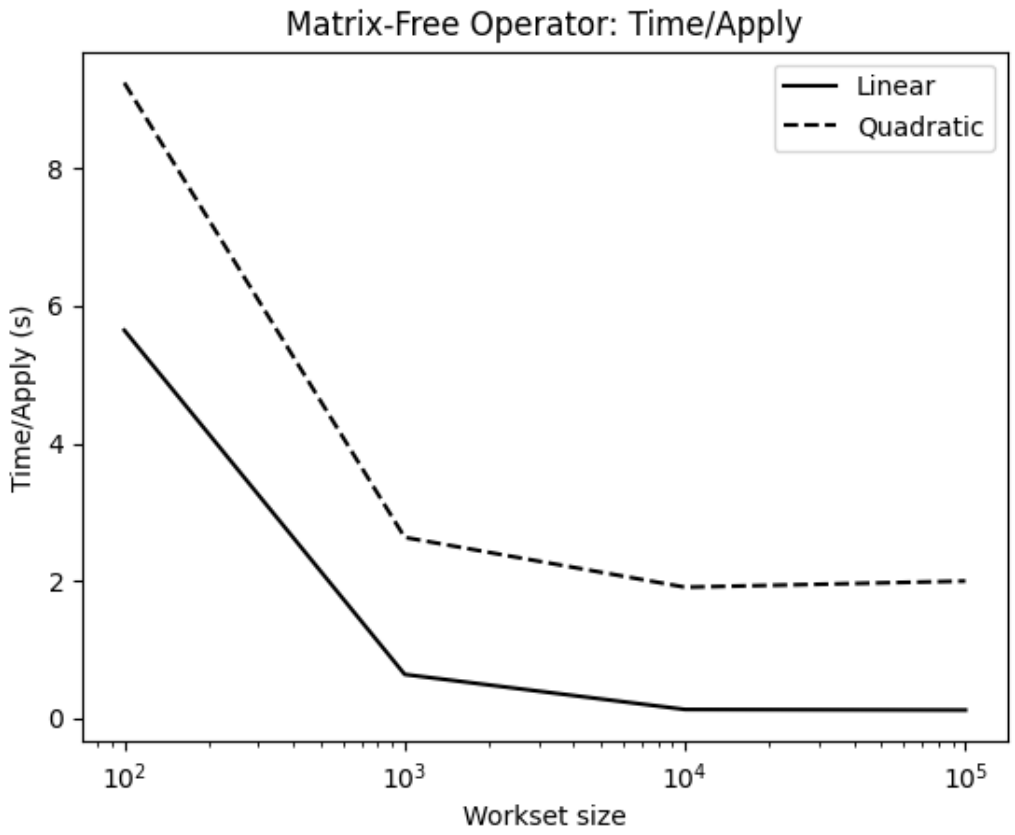
- Tests are run on Sandia's Weaver machine
  - Dual IBM Power9 with Dual Nvidia Tesla V100s. 319GB RAM per node

- Implementation details
  - Panzer: DOF manager, handle STK mesh under the hood
  - Intrepid2: Finite element evaluation and integration, sum factorization
  - Tpetra: Templated linear algebra
  - Belos: Linear solvers
  - MueLu: Multigrid
  - Kokkos: Performance portability

- Matrix-free finite elements written from scratch, utilizing above tools
- Matrix-free multigrid hierarchy written from scratch, utilizing MueLu for aggregation

Results – Matrix-Free Operator Evaluation on GPU

- Hgrad basis
- 100x100x100 structured mesh
- Reduce memory load via hierarchical parallelism, i.e. “worksets”
- Degree, workset size varied
- Cubic and higher crash on GPU due to insufficient shared Cuda memory
- Balance problem size and solve speed; see diminishing returns
- Sum factorization helps reduce iterations as p grows



Extensions

- Computational benefits not realized until high-order discretizations
  - Sum factorizations slower for linears, lower spatial dimensions
- Use p-multigrid to reduce to the linear case
- Explore more advanced smoothing options
- Use auxiliary operators
  - Low-order refined operators
  - Distance Laplacian for collocated DOF problems
    - Captures mesh irregularities
- Explore more aggressive coarsening
- Explore face-, edge-based elements
- Region-based multigrid, hybrid hierarchical grids
- Deeper hierarchies

$$L_{ij} = \begin{cases} -1/dist(i,j) & \text{if } i \neq j \\ -\sum_{k \neq i} L_{ik} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

Conclusion

- Still room for improvement!
- Matrix-free capabilities still being added to other Trilinos packages
- Many configurations/choices to explore
  - Smoothed vs unsmoothed aggregation
  - Choice of coarse grid approach
  - Approaches for high-order solvers

- Current priorities:
  - Add ability for aggregation directly on meshes in MueLu
  - Add customizable matrix-free operators to Trilinos proper
  - Identify issues with shared memory allocations on device

Thank You!

Questions?

gbharpe@sandia.gov