A Comparison of Algebraic Multigrid Preconditioning Approaches for Sampling-Based Uncertainty Propagation on Advanced Computing Architectures

SIAM Annual Meeting, July 7-11, 2014

Jonathan Hu, Eric Phipps

SAND2014-15836PE

Outline

- Background
- Multigrid preconditioning
- Numerical Experiments
- Concluding remarks

Motivation

- Forward uncertainty propagation is key for many UQ tasks
- Modern architectures hold potential for large speedups (MS29, Phipps)
- This talk: preconditioning for systems arising from PCE approaches with ensembles
- Context for this talk:
  - Steady-state finite dimensional model problem:

$$\text{Find } u(\xi) \text{ such that } f(u,\xi) = 0,\ \xi : \Omega \to \Gamma \subset R^M,\ \text{density } \rho$$

  - (Global) Polynomial Chaos approximation

$$u(\xi) \approx \hat{u}(\xi) = \sum_{i=0}^{P} u_i \psi_i(\xi), \quad \langle \psi_i \psi_j \rangle \equiv \int_\Gamma \psi_i(y)\psi_j(y)\rho(y)dy = \delta_{ij}\langle \psi_i^2 \rangle$$

  - Non-intrusive polynomial chaos (NIPC)

$$u_i = \frac{1}{\langle \psi_i^2 \rangle} \int_\Gamma \hat{u}(y)\psi_i(y)\rho(y)dy \approx \frac{1}{\langle \psi_i^2 \rangle} \sum_{k=0}^{Q} w_k u^k \psi_i(y^k), \quad f(u^k, y^k) = 0$$

Simultaneous ensemble propagation

- PDE:

$$f(u, \xi) = 0$$

- Propagating *m* samples – block diagonal (nonlinear) system:

$$\boldsymbol{F}(\boldsymbol{U}, \boldsymbol{Y}) = \boldsymbol{0}, \quad \boldsymbol{U} = \sum_{i=1}^{m} \boldsymbol{e}_i \otimes \boldsymbol{u}_i, \quad \boldsymbol{Y} = \sum_{i=1}^{m} \boldsymbol{e}_i \otimes \boldsymbol{y}_i, \quad \boldsymbol{F} = \sum_{i=1}^{m} \boldsymbol{e}_i \otimes \boldsymbol{f}(\boldsymbol{u}_i, \boldsymbol{y}_i)$$

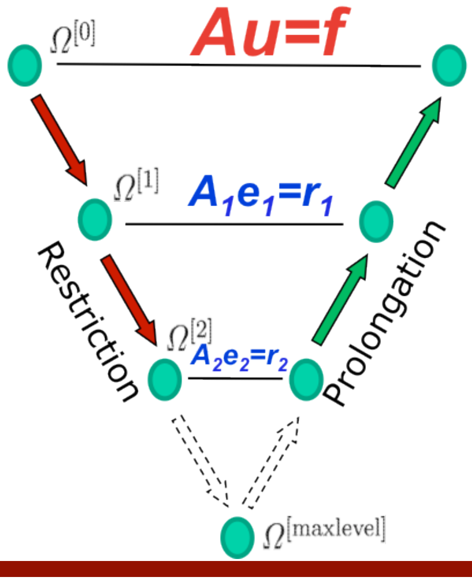
- Commute Kronecker products (just a reordering of DoFs):

$$\boldsymbol{F}_c(\boldsymbol{U}_c, \boldsymbol{Y}_c) = \boldsymbol{0}, \quad \boldsymbol{U}_c = \sum_{i=1}^{m} \boldsymbol{u}_i \otimes \boldsymbol{e}_i, \quad \boldsymbol{Y}_c = \sum_{i=1}^{m} \boldsymbol{y}_i \otimes \boldsymbol{e}_i, \quad \boldsymbol{F}_c = \sum_{i=1}^{m} \boldsymbol{f}(\boldsymbol{u}_i, \boldsymbol{y}_i) \otimes \boldsymbol{e}_i$$

- Each sample-dependent scalar replaced by length-*m* array
  - Automatically reuse non-sample dependent data
  - Sparse accesses amortized across ensemble
  - Math on ensemble naturally maps to vector arithmetic

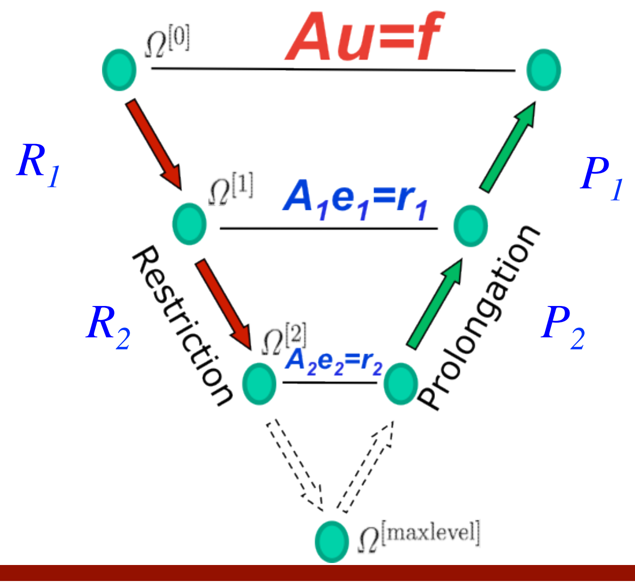
Algebraic multigrid (AMG)

- Scalable solution method for elliptic PDEs
- Typically used as preconditioner to Krylov method
- Idea: capture error at multiple resolutions:
  - **Smoothing** reduces oscillatory error (high energy)
  - **Coarse grid correction** reduces smooth error (low energy)



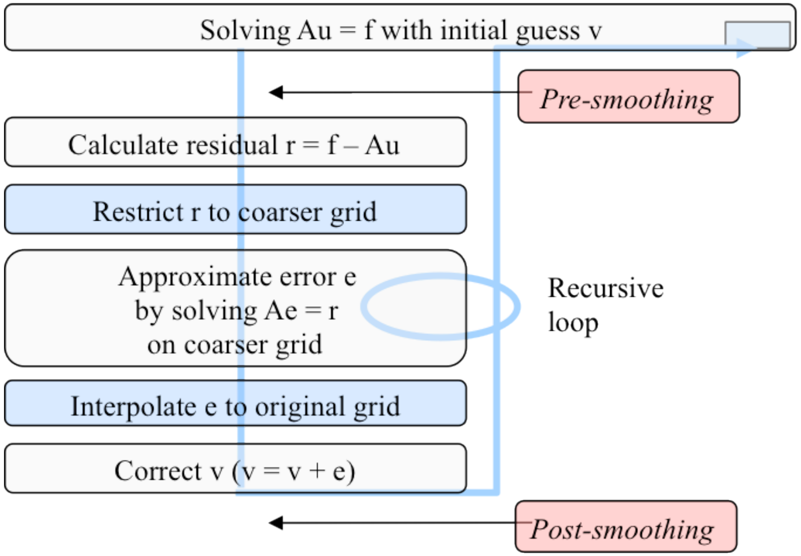
Algebraic multigrid (AMG)

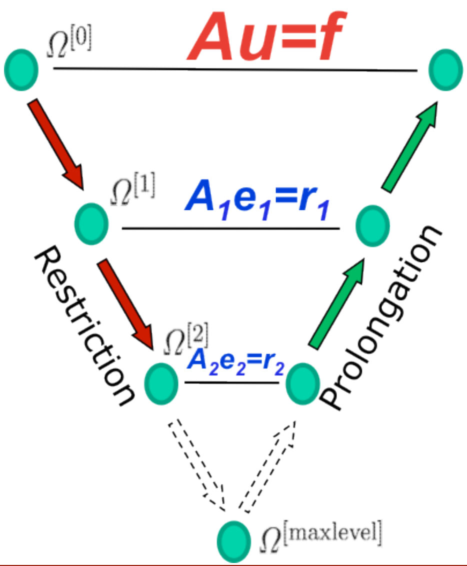
- Scalable solution method for elliptic PDEs
- Typically used as preconditioner to Krylov method
- Idea: capture error at multiple resolutions:
  - **Smoothing** reduces oscillatory error (high energy)
  - **Coarse grid correction** reduces smooth error (low energy)
- R_i's, P_i's and A_i's generated by AMG algorithm
  - $R_i = P_i^T$ for symmetric problems
  - $A_i = R_i A_{i-1} P_i$



Algebraic multigrid (AMG)

- Scalable solution method for elliptic PDEs
- Typically used as preconditioner to Krylov method
- Idea: capture error at multiple resolutions:
  - **Smoothing** reduces oscillatory error (high energy)
  - **Coarse grid correction** reduces smooth error (low energy)
- R_i's, P_i's and A_i's generated by AMG algorithm

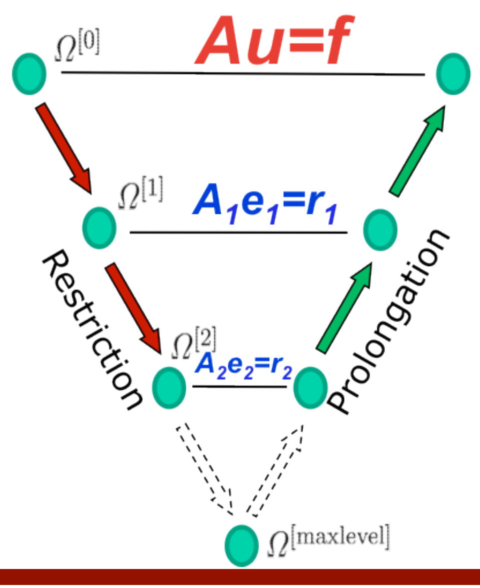




Algebraic multigrid (AMG)

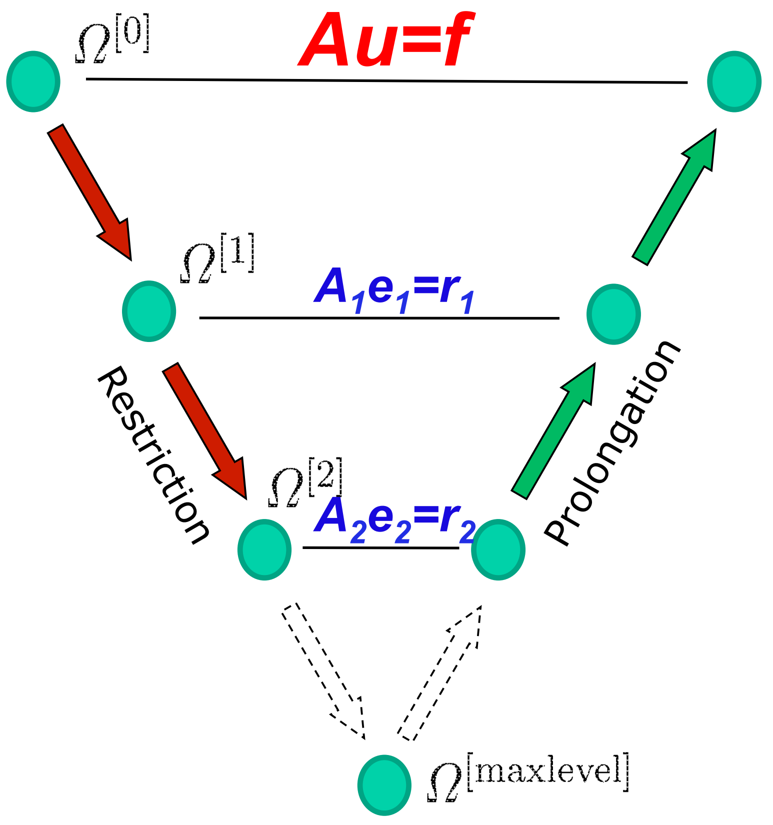
- Scalable solution method for elliptic PDEs
- Typically used as preconditioner to Krylov method
- Idea: capture error at multiple resolutions:
  - **Smoothing** reduces oscillatory error (high energy)
  - **Coarse grid correction** reduces smooth error (low energy)
- R_i's, P_i's and A_i's generated by AMG algorithm

- Two main variants
- Classical (Ruge-Stuben) AMG
  - Coarse grid DOFs are subset of fine DOFs
- Smoothed aggregation ←
  - Coarse grid DOFs are groups of fine DOFs



Smoothed Aggregation – Main Kernels

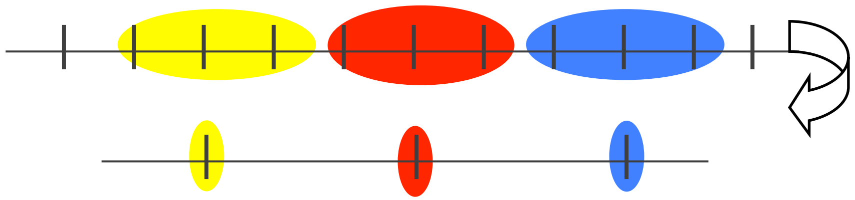
- Setup
  - Form coarse unknowns (aggregation)
  - Prolongator creation
    - $P=(I - \omega D^{-1}A)P^{(tent)}$
  - Matrix matrix multiply
    - $A_k = R\, A_{k-1}\, P$
  - Load balancing of A_k's
  - Smoother initialization
- Apply
  - Matrix-vector multiply



Prolongator Construction

$Au=f$

- Group fine unknowns into *aggregates* to form coarse unknowns
  - Based on matrix stencil coefficients
  - DOFs at node are aggregated together

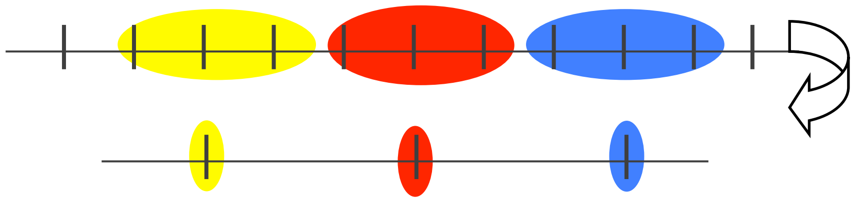


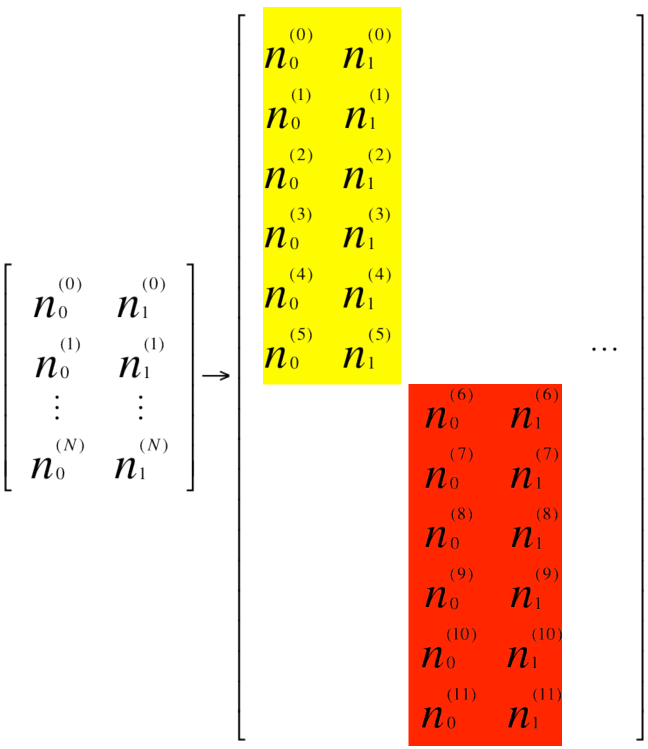
Prolongator Construction

$Au=f$

- Group fine unknowns into *aggregates* to form coarse unknowns
  - Based on matrix stencil coefficients
  - DOFs at node are aggregated together

- Partition given (near) nullspace across aggregates to have local support. Call it $N_{(h)}$.



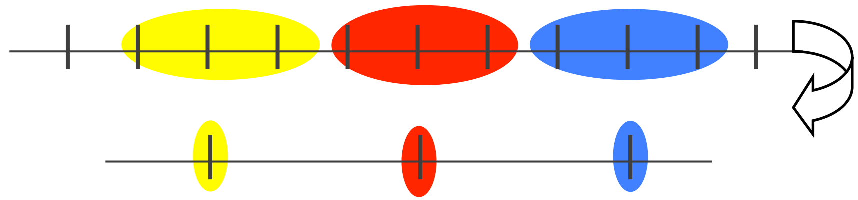


$$\begin{bmatrix} n_0^{(0)} & n_1^{(0)} \\ n_0^{(1)} & n_1^{(1)} \\ \vdots & \vdots \\ n_0^{(N)} & n_1^{(N)} \end{bmatrix} \rightarrow \begin{bmatrix} n_0^{(0)} & n_1^{(0)} & & & \\ n_0^{(1)} & n_1^{(1)} & & & \\ n_0^{(2)} & n_1^{(2)} & & & \\ n_0^{(3)} & n_1^{(3)} & & & \\ n_0^{(4)} & n_1^{(4)} & & & \\ n_0^{(5)} & n_1^{(5)} & & & \cdots \\ & & n_0^{(6)} & n_1^{(6)} & \\ & & n_0^{(7)} & n_1^{(7)} & \\ & & n_0^{(8)} & n_1^{(8)} & \\ & & n_0^{(9)} & n_1^{(9)} & \\ & & n_0^{(10)} & n_1^{(10)} & \\ & & n_0^{(11)} & n_1^{(11)} & \end{bmatrix} \quad N_{(h)}$$

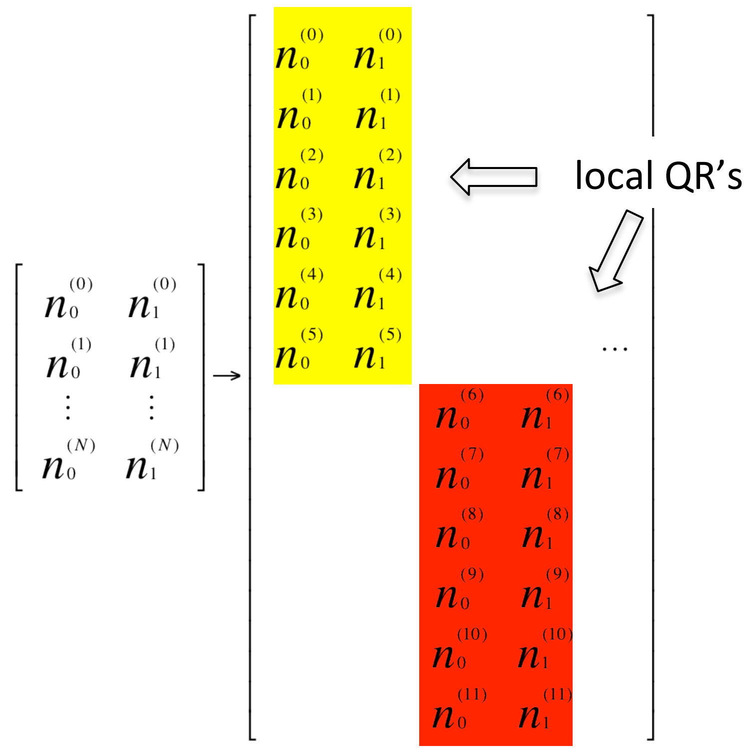
Prolongator Construction

$Au=f$

- Group fine unknowns into *aggregates* to form coarse unknowns
  - Based on matrix stencil coefficients
  - DOFs at node are aggregated together



- Partition given (near) nullspace across aggregates to have local support. Call it $N_{(h)}$.

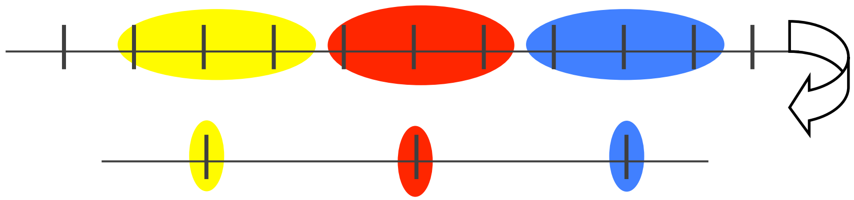


- Calculate $QR=N_{(h)}$.
- Set $P^{tent}=Q$ and $N_{(c)}=R$.

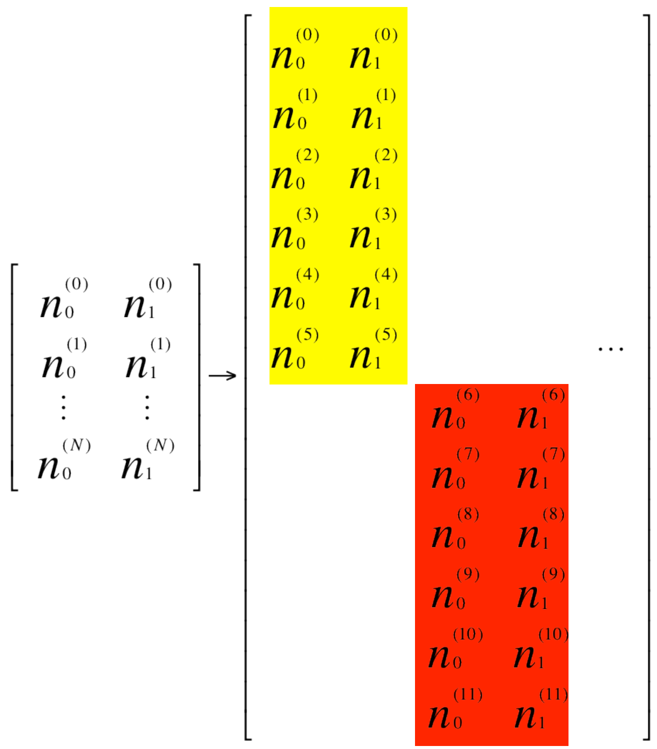
Prolongator Construction

$Au=f$

- Group fine unknowns into *aggregates* to form coarse unknowns
  - Based on matrix stencil coefficients
  - DOFs at node are aggregated together



- Partition given (near) nullspace across aggregates to have local support. Call it $N_{(h)}$.



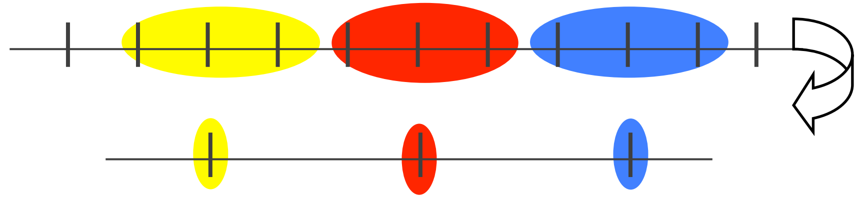
- Calculate $QR=N_{(h)}$.
- Set $P^{tent}=Q$ and $N_{(c)}=R$.

- Form final prolongator $P = (I - \omega D^{-1}A)P^{tent}$

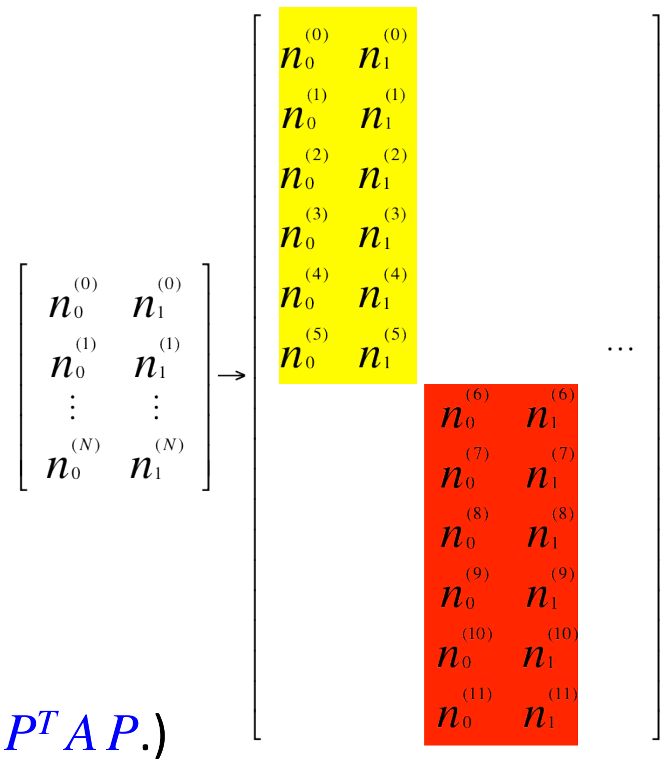
Prolongator Construction

$Au=f$

- Group fine unknowns into *aggregates* to form coarse unknowns
  - Based on matrix stencil coefficients
  - DOFs at node are aggregated together



- Partition given (near) nullspace across aggregates to have local support. Call it $N_{(h)}$.



- Calculate $QR=N_{(h)}$.
- Set $P^{tent}=Q$ and $N_{(c)}=R$.
- Form final prolongator $P = (I - \omega D^{-1}A)P^{tent}$
  - By construction, $N_{(h)} = PN_{(c)}$ and nullspace of A_c is $N_{(c)}$. (Recall $A_c = P^T A P$.)

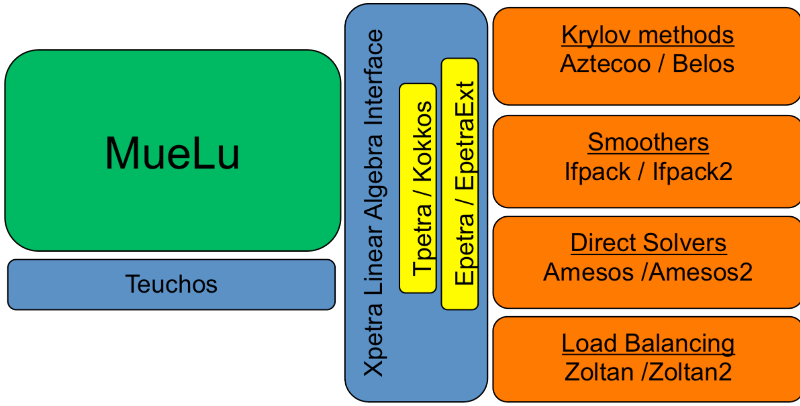
Multigrid for Ensembles

- 1) AMG templated on ensemble scalar type
  - Single AMG preconditioner per ensemble
  - Ensemble scalar type propagated through stack to multigrid solver
  - From multigrid perspective, ensemble can be viewed as vector of scalars → system of PDEs

- 2) AMG based on mean-based system
  - Single AMG setup
  - AMG coarsens matrix that is mean of the ensemble matrix
  - From AMG perspective, system can be viewed as scalar PDE

MueLu Multigrid Library

- C++ framework for implementing multigrid methods
  - Can explicitly use Tpetra (sparse linear algebra)
  - Can implicitly use Kokkos (node-level parallelism)
  - Templated on ordinal, scalar, node types



- Aggregation
  - Uncoupled, coupled
  - Matrix-based dropping, distance-based dropping
  - Design permits alternative AMG and GMG implementations

- SOR, Chebyshev, ℓ1 Gauss-Seidel, incomplete factorizations, additive Schwarz, line smoothing* (Ifpack2)
- SuperLU direct solver

- Algorithms for Poisson, elasticity, Helmholtz, convection-diffusion, Maxwell (eddy current)

Numerical Experiments

- Sandia Linux test bed "Shannon"
  - two 8-core Sandy Bridge Xeon E5-2670s per node
  - 128 GB per node
  - OpenMPI, Intel 13.1, OpenMP

- Problem description
  - Nonlinear diffusion equation

$$-\kappa \nabla^2 u + u^2 = 0$$

    - 3-D, linear FEM discretization
    - Cubic domain, 64^3 mesh
    - KL-like random field model for diffusion coefficient

- Single-node performance

CG iterations, no preconditioning

cl	ensemble size	UQ dim (#random var.)		
		3	5	7
10	1	853	871	869
	16	1350	1160	1110
	32	1830	1180	1120

cl	ensemble size	UQ dim		
		3	5	7
1.0	1	852	853	845
	16	1390	1130	1010
	32	1910	1180	1050

cl	ensemble size	UQ dim		
		3	5	7
0.1	1	800	800	800
	16	1460	977	867
	32	2060	1100	913

cl=correlation length

CG iterations, ensemble-based AMG

cl	ensemble size	UQ dim		
		3	5	7
10	1	45.1	44.9	44.9
	16	55.8	48.2	46.2
	32	73.9	48.3	46.4

cl	ensemble size	UQ dim		
		3	5	7
1.0	1	45.1	44.9	44.9
	16	55.4	48.1	46.1
	32	74.7	48.3	46.4

cl	ensemble size	UQ dim		
		3	5	7
0.1	1	45.0	44.9	44.9
	16	55.8	48.1	46.1
	32	73.9	48.3	46.5

CG iterations, mean-based AMG

cl	ensemble size	UQ dim		
		3	5	7
10	16	58.3	50.7	48.4
	32	78.8	51.1	48.9

cl	ensemble size	UQ dim		
		3	5	7
1.0	16	59.5	49.8	47.0
	32	81.2	50.9	47.7

cl	ensemble size	UQ dim		
		3	5	7
0.1	16	62.8	49.7	46.7
	32	88.6	51.5	47.5

Conclusions

- Future work
  - Optimization of mean-based preconditioner
  - Further investigate space of AMG options
  - Introduce Kokkos kernels into multigrid setup
  - Reuse across ensembles
- MueLu public release in fall of 2014
- Trilinos: www.trilinos.org

Acknowledgements

- DOE ASCR Equinox project
- DOE ASC program
- NNSA ASC Sandia Advanced Systems Technology Test Beds
  - www.sandia.gov/asc/computational_systems/HAAPS.html
- Collaborators
  - Carter Edwards (SNL)
  - Sarah Osborn, Vicki Howle (Texas Tech U.)