Enrichment for Multigrid in PETSc

Matthew Knepley

Computer Science and Engineering & CDSE
University at Buffalo

SIAM Computational Science and Engineering,
Spokane, WA February 27, 2019

Truism

Multigrid can solve

hard, complex problems

using basis enrichment.

Main Idea

Basis enrichment

consists of pieces, that

should be composable.

Outline

Homogenization

Overlapping multicell eigenproblems

- Variational multiscale
- Stuff from Todd Arbogast
- Localized orthogonal decomposition (LOD)

LOD defines localization as *k*-cell neighborhood

Efficient implementation of the LOD method
Engwer, Henning, Målqvist, Peterseim, 1602.01658, 2016.

Domain Decomposition

Energy-minimizing extensions to dof support

- Wirebasket (PCEXOTIC)
- Generalized Dryja Smith Widlund (GDSW)
- Need to know nullspace

Extensions are linear solves, not eigensolves

A Parallel Impl. . . . with Energy-Minimizing Coarse Space . . .
Heinlein, Klawonn, Rheinbach, SISC 38(6), 2016.

Multigrid

Generalized eigenmodes of the operator

- Adaptive AMG (αAMG)
- AMGe
- Bootstrap AMG

Get *near null* modes

Bootstrap AMG
Brandt, Brannick, Kahl, Livshits, SISC 33(2), 2011.

Multilevel Eigensolver (MEPS)

Based on a simple relation for some matrix W,

$$\langle x_l, x_l \rangle_{W_l} = \langle P_l x_l, P_L x_l \rangle_W$$

where prolongator P_l defines

$$W_l = P_l^\dagger W P_l.$$

Multilevel Eigensolver (MEPS)

Suppose that

$$A_l x_l = \lambda_l \tilde{M}_l x_l.$$

Then

$$\begin{aligned}
\lambda_l &= \frac{\langle x_l, x_l \rangle_{A_l}}{\langle x_l, x_l \rangle_{\tilde{M}_l}} \\
&= \frac{\langle P_l x_l, P_l x_l \rangle_{A}}{\langle P_l x_l, P_l x_l \rangle_{\tilde{M}}} \\
&= RQ(P_l x_l)
\end{aligned}$$

provides a way to connect levels.

Multilevel Eigensolver (MEPS)

First solve

$$A_l x_l = \lambda_l \tilde{M}_l x_l.$$

Then guess

$$\begin{aligned} \lambda_{l+1} &= \lambda_l, \\ x_{l+1} &= P_l^{l+1} x_l, \end{aligned}$$

smooth

$$\left(A_{l+1} - \lambda_{l+1} \tilde{M}_{l+1}\right) x_{l+1} = 0.$$

and update

$$\lambda_{l+1} = RQ(x_{l+1}).$$

Multilevel Eigensolver (MEPS)

This relies on $\tilde{M}$ satisfying

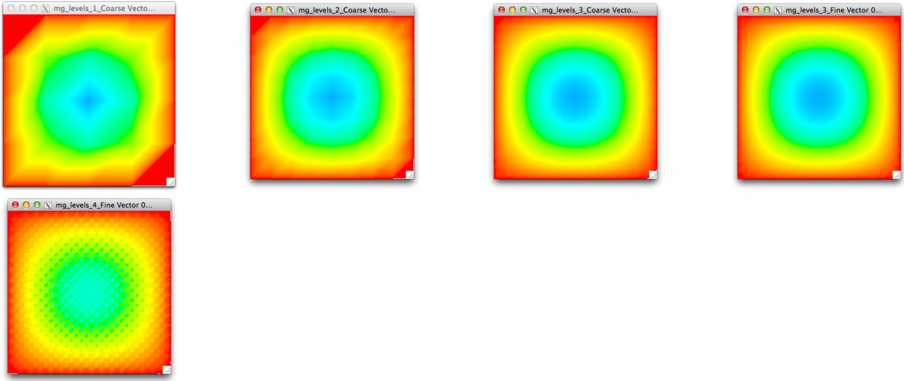
$$\tilde{M}_l \approx P_l^\dagger \tilde{M} P_l.$$

Also, I needed to orthogonalize coarse vectors during the iteration on each level.

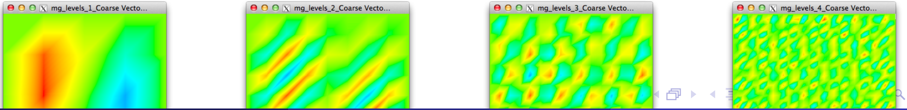
- Used BV from SLEPc

Multilevel Eigensolver (MEPS)

Mode 0 ($\lambda = 0.0014$)

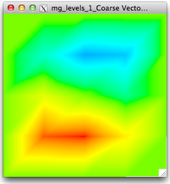


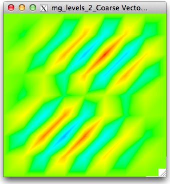
Mode 1 ($\lambda = 1.195$)

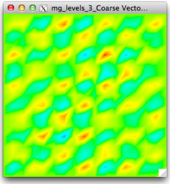


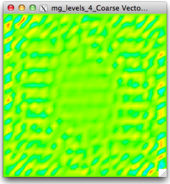
Multilevel Eigensolver (MEPS)

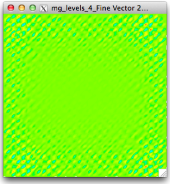
Mode 2 ($\lambda = 1.43$)



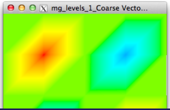


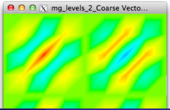


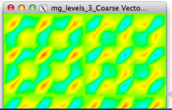


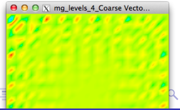


Mode 3 ($\lambda = 1.15$)



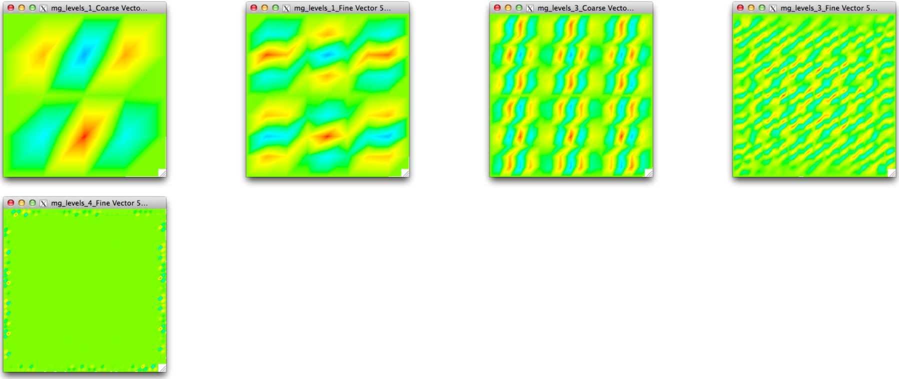




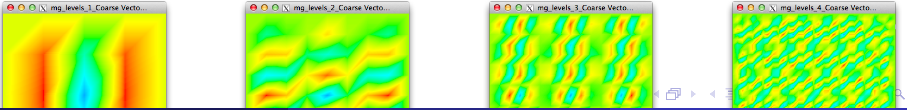


Multilevel Eigensolver (MEPS)

Mode 5 ($\lambda = 1.80$)



Mode 7 ($\lambda = 1.80$)



Why do I think it works?

We can measure the eigen-residual

$$\|M^{-1}Ax - \lambda x\|_2$$

for each coarse basis vector.

We see 10^{-3}–10^{-5}.

Its unclear (to me) how accurate this basis needs to be for fast convergence.

Outline

Alternatives

Homogenization

Solve full problem in new basis

Domain Decomposition

Add in solution from coarse problem

Linear Algebra

Augment Krylov basis (DGMRES)

Multigrid

Optimize the prolongator P_l^{l+1}

- Reproduce coarse modes accurately
- Minimize energy of interpolant

Solve L_2 least squares problem for each row

- BAMG: coarse-fine point division
- BGMG: one function discretized on both levels

$$\min_{P_{ij}} \sum_k w_k \| f_i^{F,k} - \sum_j P_{ij} f_j^{C,k} \|_2$$

Adaptation API

Adapting the Prolongator:

```
DMAdaptInterpolator(DM dmc, DM dmf, Mat In, KSP smoother,
  PetscInt Nc, Vec vf[], Vec vc[], Mat *InAdapt, void *user);
DMCheckInterpolator(DM dmf, Mat In,
  PetscInt Nc, Vec vc[], Vec vf[], PetscReal tol);
```

Multilevel Eigensolver in PCMG:

```
PCMGComputeCoarseSpace(PC pc, PetscInt l, PCMGCoarseSpaceType cstype,
  PetscInt Nc, Vec cspace[], Vec *space[]);
PCMGAdaptInterpolator(PC pc, PetscInt l, KSP csmooth, KSP fsmooth,
  PetscInt Nc, Vec cspace[], Vec fspace[]);
PCMGRecomputeLevelOperators(PC pc, PetscInt l);
```

Adaptation Commandline

PCMG adaptation:

```
-pc_mg_adapt_interp
-pc_mg_adapt_interp_coarse_space
  <polynomial,harmonic,eigenvector>
-pc_mg_adapt_interp_n <k>
```

Multilevel Eigensolve:

```
-pc_mg_mesp_ksp_type richardson
-pc_mg_mesp_ksp_richardson_self_scale
-pc_mg_mesp_ksp_max_it 100
-pc_mg_mesp_pc_type <none, jacobi>
```

Why do I think it works?

We can measure

$$\|f^F - Pf^C\|_\infty \text{ and } \|f^F - Pf^C\|_2$$

for each coarse basis vector.

We cannot just use max-norm since the *interpolator sparsity pattern* near boundaries can be very restricted.

Outline

Checkerboard Example

This example comes from

Optimal Interpolation & Compatible Relaxation in Classical AMG
Brannick, Cao, Kahl, Falgout, Hu, SISC 40(3), 2018.

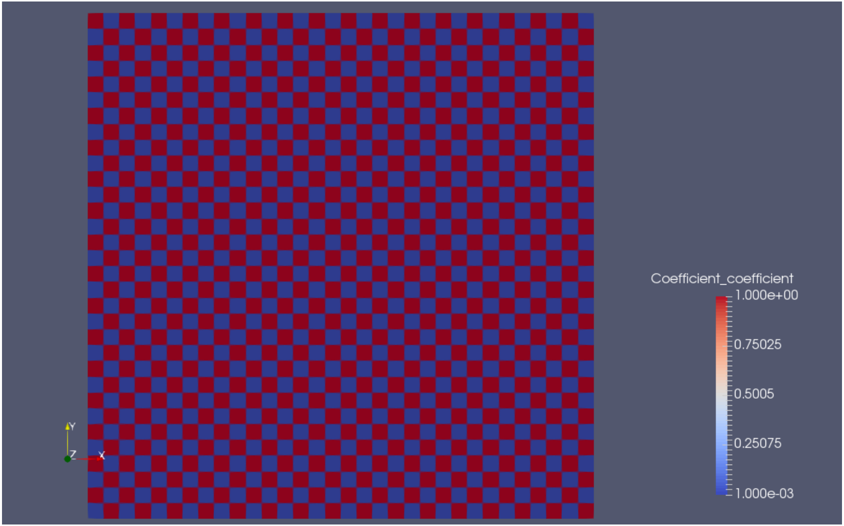
It solves

$$-\nabla \cdot \nu(\vec{x})\nabla u - 20e^{-\left|\vec{x}-\vec{x}_0\right|^2} = 0,$$

where $\nu \in \left[10^{-k}, 1\right]$ in a checkerboard pattern.

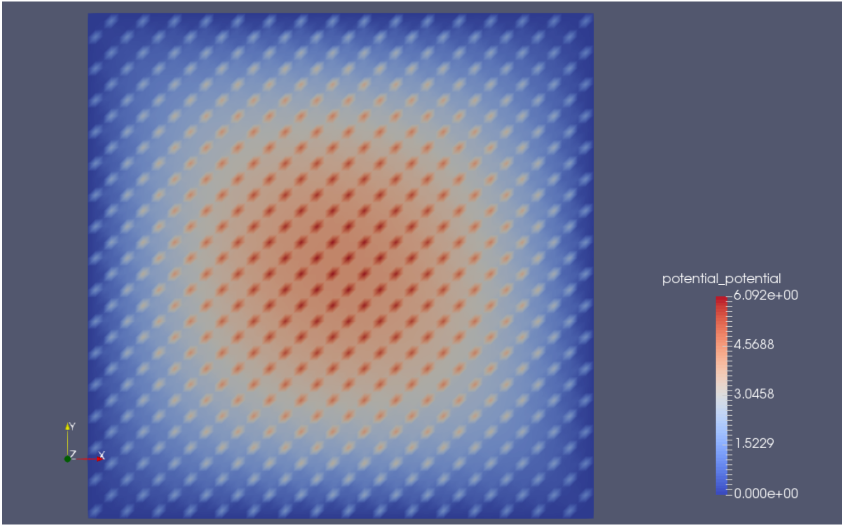
Checkerboard Example

Coefficient 64x64 $k = 3$



Checkerboard Example

Solution 64x64 $k = 3$



Checkerboard Example

64x64 $k = 3$

Standard PETSc GMG:

```
 0 KSP Residual norm 2.643944129967e-01
 1 KSP Residual norm 2.573911048750e-01
 2 KSP Residual norm 2.573256470533e-01
 3 KSP Residual norm 2.571837465231e-01
 4 KSP Residual norm 2.562369395483e-01
 5 KSP Residual norm 2.562233942117e-01
 6 KSP Residual norm 2.561877374398e-01
 ⋮
81 KSP Residual norm 4.333658836503e-09
82 KSP Residual norm 2.870042453496e-09
83 KSP Residual norm 1.724444567606e-09
```

Checkerboard Example

64x64 $k = 3$

Adaptive GMG with 8 eigenvectors:

```
 0 KSP Residual norm 2.643944129967e-01
 1 KSP Residual norm 2.622984060861e-01
 2 KSP Residual norm 2.242690218384e-01
 3 KSP Residual norm 1.853298561871e-01
 4 KSP Residual norm 1.482379261196e-01
 5 KSP Residual norm 1.039149927776e-01
 6 KSP Residual norm 6.282001523842e-02
 ⋮
33 KSP Residual norm 7.334589478602e-09
34 KSP Residual norm 4.163311731389e-09
35 KSP Residual norm 2.338748316520e-09
```

Checkerboard Example

64x64 $k = 3$

Adaptive GMG with 1 eigenvector:

```
0 KSP Residual norm 2.643944129967e-01
1 KSP Residual norm 1.368739283743e-01
2 KSP Residual norm 2.229521556344e-02
3 KSP Residual norm 1.673518835746e-03
4 KSP Residual norm 1.403981092990e-04
5 KSP Residual norm 1.147445564476e-05
6 KSP Residual norm 8.831252126121e-07
7 KSP Residual norm 7.332391283986e-08
8 KSP Residual norm 5.999730555945e-09
9 KSP Residual norm 4.943868744122e-10
```

Checkerboard Example

64x64 $k = 3$

Adaptive GMG with 2 eigenvectors:

```
 0 KSP Residual norm 2.643944129967e-01
 1 KSP Residual norm 1.419407097359e-01
 2 KSP Residual norm 9.135393298863e-02
 3 KSP Residual norm 3.020106692060e-02
 4 KSP Residual norm 1.210952999077e-02
 5 KSP Residual norm 4.286622379522e-03
 6 KSP Residual norm 1.913486044360e-03
 ⋮
20 KSP Residual norm 1.485802738171e-08
21 KSP Residual norm 5.589085527836e-09
22 KSP Residual norm 2.201671896250e-09
```

Checkerboard Example

64x64 $k = 4$

Standard PETSc GMG:

```
  0 KSP Residual norm 2.643944129967e-01
  1 KSP Residual norm 2.574526515771e-01
  2 KSP Residual norm 2.574284005602e-01
  3 KSP Residual norm 2.573988430822e-01
  4 KSP Residual norm 2.570713945704e-01
  5 KSP Residual norm 2.569879721799e-01
  6 KSP Residual norm 2.567320802002e-01
  ⋮
260 KSP Residual norm 4.486735550171e-09
261 KSP Residual norm 3.311768847514e-09
262 KSP Residual norm 2.312191750445e-09
```

Checkerboard Example

64x64 $k = 4$

Adaptive GMG with 1 eigenvector:

```
0 KSP Residual norm 2.643944129967e-01
1 KSP Residual norm 1.299664043058e-01
2 KSP Residual norm 2.192319438963e-02
3 KSP Residual norm 1.651881252886e-03
4 KSP Residual norm 1.347033005332e-04
5 KSP Residual norm 1.063898499377e-05
6 KSP Residual norm 8.034146403034e-07
7 KSP Residual norm 6.704449807355e-08
8 KSP Residual norm 5.531374943735e-09
9 KSP Residual norm 4.471792062324e-10
```

Checkerboard Example

64x64 $k = 4$

Standard PETSc GMG on quads:

```
  0 KSP Residual norm 2.643871530208e-01
  1 KSP Residual norm 2.584172081575e-01
  2 KSP Residual norm 2.572661193862e-01
  3 KSP Residual norm 2.550802805550e-01
  4 KSP Residual norm 2.529879158256e-01
  5 KSP Residual norm 2.522691968307e-01
  6 KSP Residual norm 2.514243101722e-01
  ⋮
207 KSP Residual norm 2.969084590440e-09
208 KSP Residual norm 2.673859078427e-09
209 KSP Residual norm 2.432348693080e-09
```

Checkerboard Example

64x64 $k = 4$

Adaptive GMG with 1 eigenvector on quads:

```
0 KSP Residual norm 2.643871530208e-01
1 KSP Residual norm 7.741923747475e-02
2 KSP Residual norm 1.537706401791e-02
3 KSP Residual norm 1.613987533605e-03
4 KSP Residual norm 8.169056466595e-05
5 KSP Residual norm 9.163477739245e-06
6 KSP Residual norm 7.982598339429e-07
7 KSP Residual norm 6.736115781708e-08
8 KSP Residual norm 7.180920762406e-09
9 KSP Residual norm 7.742861190126e-10
```

Checkerboard Example

Quads $k = 4$

Adaptive GMG with 1 eigenvector on 128x128:

```
 0 KSP Residual norm 1.329466728347e-01
 1 KSP Residual norm 7.237500958468e-02
 2 KSP Residual norm 1.438549155985e-02
 3 KSP Residual norm 1.891895494543e-03
 4 KSP Residual norm 6.980593794529e-05
 5 KSP Residual norm 1.131139401771e-05
 6 KSP Residual norm 1.058946413598e-06
 7 KSP Residual norm 9.961020640464e-08
 8 KSP Residual norm 1.188564376601e-08
 9 KSP Residual norm 1.388263057429e-09
10 KSP Residual norm 1.380589117211e-10
```

Checkerboard Example

Quads $k = 4$

Adaptive GMG with 1 eigenvector on 256x256:

```
 0 KSP Residual norm 6.666002610424e-02
 1 KSP Residual norm 5.245837884989e-02
 2 KSP Residual norm 1.186051192857e-02
 3 KSP Residual norm 1.484602815289e-03
 4 KSP Residual norm 8.401165001048e-05
 5 KSP Residual norm 1.156096091669e-05
 6 KSP Residual norm 1.014700764074e-06
 7 KSP Residual norm 1.128389370980e-07
 8 KSP Residual norm 1.135388821859e-08
 9 KSP Residual norm 1.219154510549e-09
10 KSP Residual norm 1.663780386661e-10
```

Checkerboard Example

Quads $k = 4$

Adaptive GMG with 1 eigenvector on 512x512:

```
 0 KSP Residual norm 3.337648678109e-02
 1 KSP Residual norm 3.111433698057e-02
 2 KSP Residual norm 1.065894924276e-02
 3 KSP Residual norm 9.640896517047e-04
 4 KSP Residual norm 9.820033295993e-05
 5 KSP Residual norm 8.393723404542e-06
 6 KSP Residual norm 1.022686427555e-06
 7 KSP Residual norm 1.113263119115e-07
 8 KSP Residual norm 1.030057125946e-08
 9 KSP Residual norm 1.385292895835e-09
10 KSP Residual norm 2.268336022704e-10
```

Checkerboard Example

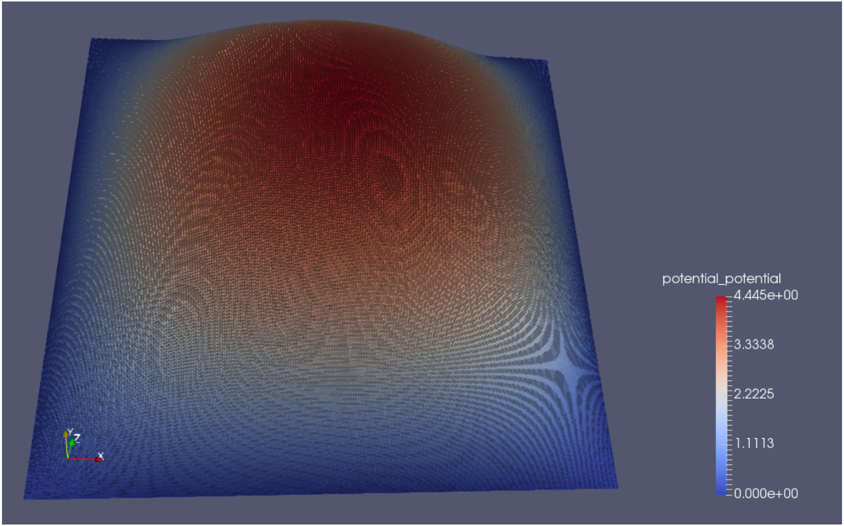
Quads $k = 4$

Adaptive GMG with 1 eigenvector on 1024x1024:

```
 0 KSP Residual norm 1.669983694677e-02
 1 KSP Residual norm 1.639873418320e-02
 2 KSP Residual norm 9.242836605258e-03
 3 KSP Residual norm 6.300719730305e-04
 4 KSP Residual norm 1.050428485267e-04
 5 KSP Residual norm 6.586618545239e-06
 6 KSP Residual norm 1.028698937734e-06
 7 KSP Residual norm 1.125558670293e-07
 8 KSP Residual norm 9.578610906761e-09
 9 KSP Residual norm 1.466361824205e-09
10 KSP Residual norm 2.232688949097e-10
11 KSP Residual norm 2.040775816394e-11
```

Checkerboard Example

Solution 1024x1024 $k = 4$



Outline

Can I Try It?

Repository:

https://bitbucket.org/petsc/petsc/

Branch:

knepley/feature-plex-adaptive-interpolation

Improvements

- More general interpolator sparsity pattern
  - Could adaptively find spartsity pattern like SPAI
- Smoothing in MESP should be FAS
  - Should use SNES

    Inverse, Shifted Inverse, and Rayleigh Quotient Iteration as Newton's Method

    Tapia, Dennis, Schäfermeyer, SIAM Review 60(1), 2018.
- Better examples
  - Stokes with variable coefficient using Braess-Sarazin/Vanka