Printed Mar. 2014

ML's SemiCoarsening Feature

Ray S. Tuminaro
Sandia National Laboratories

Abstract

An addition to the standard **ML** manual describing a new semicoarsening feature.

1 What

A new **ML** interpolation strategy. The scheme coarsens in the z direction only taking advantage of extruded or structured meshes in the z direction. The algorithm uses some rather old multigrid ideas (termed operator dependent interpolation) for structured grids. Recursive semicoarsening can be followed by further standard **ML** coarsening. It should be noted that semicoarsening can be applied for systems with multiple degrees-of-freedom per node. The strategy's original driver was an anisotropic/stretched mesh problem for a thin structure. However, this semicoarsening may also be beneficial on isotropic problems.

2 Requirements

1. every (x,y) mesh pair must have the same number of z-nodes.
2. mesh must be partitioned so entire z-lines reside within a single core. Karen D. provided a new option. You must build seacas with an up-to-date repository and then use

```
decomp --rcb_ignore_z --processors=42 MyMesh.exo
nem_slice -l rcb,ignore_z  ...  MyMesh.exo
```

3. stencil must not span more than the immediate north and south neighboring layers. That is, stencils reaching across 4^+ vertical layers are not allowed.

Note: Code will go into **MueLu**. Also, not hard to extend to 2D (i.e., y coarsening) if there is interest. Finally, I build seacas via

```
NETCDF=/data2/TPL/netcdf-4.2.1.1/install
PREFIX=/data2/TPL/zoltsea-install
rm -f CMakeCache.txt
cmake -D CMAKE_INSTALL_PREFIX:PATH=$PREFIX \
      -D TPL_ENABLE_MPI:BOOL=ON \
      -D Trilinos_ENABLE_ALL_PACKAGES:BOOL=OFF \
      -D Trilinos_ENABLE_ALL_OPTIONAL_PACKAGES:BOOL=OFF \
      -D Trilinos_ENABLE_TESTS:BOOL=OFF \
      -D Trilinos_ENABLE_EXAMPLES:BOOL=OFF \
      -D Trilinos_ENABLE_SEACAS:BOOL=ON \
      -D Trilinos_ENABLE_SEACASAprepro:BOOL=ON \
      -D Trilinos_ENABLE_SEACASConjoin:BOOL=ON \
      -D Trilinos_ENABLE_SEACASEjoin:BOOL=ON \
      -D Trilinos_ENABLE_SEACASEpu:BOOL=ON \
      -D Trilinos_ENABLE_SEACASExodiff:BOOL=ON \
      -D Trilinos_ENABLE_SEACASNemslice:BOOL=ON \
      -D Trilinos_ENABLE_Zoltan:BOOL=ON \
      -D Trilinos_ENABLE_SEACASNemspread:BOOL=ON \
      -D TPL_ENABLE_Netcdf:BOOL=ON \
      -D TPL_Netcdf_INCLUDE_DIRS:PATH=$NETCDF/include \
      -D Netcdf_LIBRARY_DIRS:PATH=$NETCDF/lib ${TRILINOS_HOME}
```

3 ML parameters

```
<Parameter name="semicoarsen: number of levels" type="int" value="3"/>
```

⇒ determines the maximum number of times semicoarsening is applied. If "max levels" is larger than "semicoarsen: number of levels"+1, then **ML** switches to normal coarsening. However, **ML** also switches to normal coarsening if there is only one z-layer.

example

```
<Parameter name="semicoarsen: number of levels" type="int" value="2"/>
<Parameter name="max levels" type="int" value="4"/>
```

yields a 4-level method with 2 created by semicoarsening and the coarsest created by regular coarsening.

```
<Parameter name="semicoarsen: coarsen rate" type="int" value="3"/>
```

⇒ determines aggressiveness of semicoarsening. Smoothed aggregation is similar to a "3" while geometric or Ruge-Stuben multigrid resembles a "2". **ML** automatically chooses the coarse layers and the number of fine layers need not be evenly divisible by any magic number. If one wants immediate coarsening to a single z-layer, set this to any value greater than the number of fine z-layers.

```
<Parameter name="smoother: type" type="string" value="line Jacobi"/>
<Parameter name="smoother: type" type="string" value="line Gauss-Seidel"/>
```

⇒ use z-line relaxation, i.e. block Jacobi or block Gauss-Seidel where each block essentially corresponds to grouping nodes along each vertical mesh line (see "Fancy parameters" section for details). Line relaxation is useful for large coarsening rates, but not strictly needed for modest coarsening (e.g., ≤ 3). NOTE: damping factors can be set via "smoother: damping factor"

4 Fancy parameters

```
<Parameter name="smoother: line GS Type" type="string" value="symmetric"/>
```

⇒ "symmetric" is the default, in which case each line Gauss-Seidel invocation uses a forward and back sweep. "standard" implies that only a forward sweep is used. "efficient symmetric" employs a forward sweep in pre-smoothing while a backward sweep is used within post-smoothing.

```
<Parameter name="smoother: line group dofs" type="string" value="separate"/>
```

⇒ "group" implies all degrees-of-freedom associated with nodes along a vertical mesh line give rise to one block in the block relaxation scheme. However, the default is "separate" and this actually creates num_PDEs (number of degrees-of-freedom per node) blocks for each vertical mesh line and each block corresponds to ONLY ONE unknown type (e.g. u-velocities or pressures).

```
<Parameter name="repartition: start level" type="int" value="3"/>
<Parameter name="repartition: Zoltan dimensions" type="int" value="2"/>
```

⇒ If coarse level repartitioning is desired, it is important to tell **ML** not to repartition until semicoarsening is finished. That is, to start the repartitioning at a sufficiently coarse level so that only standard **ML** coarsening will be applied to the repartitioned matrix. At this point, the problem is effectively two dimensional and so **Zoltan** should be informed of this.

5 Parameters when coordinates are NOT supplied to ML

```
<Parameter name="semicoarsen: line direction nodes" type="int" value="6"/>
```

⇒ number of z-layers on finest level

```
<Parameter name="semicoarsen: line orientation" type="int" value="vertical"/>
```

⇒ "vertical" implies all nodes within a vertical line are numbered consecutively while "horizontal" assumes that all nodes within a horizontal layer are numbered consecutively and that node i+NumVerticalLines lies directly above node i where NumVerticalLines is the total number of vertical mesh lines.

NOTE: one needs the options "smoothing: line direction nodes" and "smoothing: line orientation" if line smoothing on the finest level without semicoarsening and without supplying coordinates.

6 Algorithm details

For each coarse node, i, solve a mini-problem

$$T\ b_i = e_i$$

where T is $k \times k$. In the single PDE case, T is tridiagonal, b_i is a vector corresponding to the nonzero entries in one interpolation basis function, e_i has only one nonzero entry equal to 1 (associated with the coarse node). To make things concrete consider computing one interpolation basis function for the middle c point in the one dimensional mesh:

$$f \quad f \quad c \quad f \quad f \quad f \quad c \quad f \quad f \quad f \quad f \quad c \quad f \quad f$$

Here, k is 8 corresponding to interpolation at $f \quad f \quad f \quad c \quad f \quad f \quad f \quad f$ and $e_i = (0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0)^T$. In 1D, the 7 f-point rows of T are simply copies of A's rows (except the first and last rows do not contain connections to the leftmost and rightmost c-points). T's c-point row has one nonzero equal to 1 on the matrix diagonal. Thus, the quantity $A\ \hat{b}_i$ is nonzero only at c-points, where $\hat{b}_i$ is b_i extended over the whole domain. One can think of $\hat{b}_i$ as Schur complement derived or low energy. For multi-dimensional PDEs, we coarsening only in the z direction, and T is obtained by collapsing the original PDE discretation matrix A to one dimension. For each row, this is done by categorizing $A(i,:)$'s nonzeros into 3 groups depending on the column's associated z-layer. A 3-point stencil is obtained by summing all member within each group. In this way, a 7 pt-Laplace operator is collapsed to the stencil $[-1 \quad 2 \quad -1]$. The algorithm basically does the same thing for multiple degrees-of-freedom per node with T as a block tridiagonal matrix, b_i as a multi-vector, and e_i as a multi-vector with an identity in the block associated with a coarse node.