ADAPTIVE ALGEBRAIC MULTIGRID WITH GRAPH MODULARITY MATCHING

1. Introduction

This library implements some tools to explore techniques of adaptive AMG. In classical AMG, the performance of the solver depends on the construction of the relaxation and coarse-grid correction process (cite M. BREZINA). Convergence of the method slows when the residual approaches the rhs of the system (the iterate is in the 'near null space' of the system). This happens when the current iterate is an approximation of a minimal eigenvalue of the matrix. Using this near null component and utilizing graph modularity, we can construct a new relaxation and coarse-grid correction. This provides the adaptivity when convergence suffers. Combining multiple of these adaptive relaxation steps can provide a composite adaptive multigrid method.

2. Library Details

The library is broken into 3 main modules: partitioner, preconditioner, and solver.

2.1. **Partitioner module.** Here is implemented an algorithm described in (quiring 2019) which is graph pairwise matching algorithm that only merges vertices where the change in modularity is positive. The only requirement for this algorithm is that the matrix has positive row sums. To convert our system to have positive rowsums, we construct a new system with a matching sparsity pattern using a 'near null' component. The modularity matching algorithm applied to this new system then gives a hierarchy of interpolation matrices that can be applied the the original system to create the coarse problem.

2.2. **Preconditioner module.** This module is called preconditioner but maybe a better name could be smoothers or methods. In this module there are functions that take a system and provide the inverse action of some method than can be used in an iterative solver. Currently, one can build L1 smoother, symmetric/forward/backward Gauss-Seidel, multilevel L1, and multilevel Gauss-Seidel. All of these can be used as methods for the stationary solver but only the symmetric methods can be used as a preconditioner in the conjugate gradient method.

2.3. **Solver module.** This module implements different iterative solvers that accept linear systems, a method to base the solver on, and some parameters for the solver. Currently, only the classical stationary iterative method and preconditioned conjugate gradient is implemented. Soon I will be adding an adaptive solver to this module.

3. Example Usage

The library is currently packaged with a CLI binary that shows how the library can be used to test different solvers on your systems.

4. Goals

Classical AMG often breaks down on highly anisotropic problems often found in finite element methods. To address this failure, sometimes specific knowledge of the system can be used to carefully construct a preconidtioner. The goal of this research is to test some adaptive methods that don't require any knowledge about the system and measure the computational complexity and memory demands of these methods on systems where classical AMG performs poorly.

5. Algorithm

Let $A = (a_{ij})_{i,j=1}^n$ be an $n \times n$ s.p.d. sparse matrix. We begin by searching for an algebraically smooth error vector, $\mathbf{w} = (w_j) \in \mathbb{R}^n$, such that $A\mathbf{w} \approx 0$. To obtain $\mathbf{w}$:

- Begin with a random starting vector and some smoother, either L1 or Gauss-Seidel, and perform a few iterations of relaxation.
- Construct the weighted matrix $\overline{A} = (\overline{a}_{ij})_{i,j=1}^n$ where $\overline{a}_{ij} = -w_i a_{ij} w_j$ if $i \neq j$ and $\overline{a}_{ii} = 0$. Note that its row sums are positive if $\mathbf{w}$ is a good enough approximation to the homogeneous system,

$$r_i \equiv \sum_j \overline{a}_{ij} \approx a_{ii} w_i^2 \geq 0 \text{ since } a_{ii} > 0.$$

  If the rowsums (r_i) are not all positive, go back to smoothing but starting with $\mathbf{w}$ instead of a new random vector for a few more iterations.

Once we have a suitable $\mathbf{w}$ that produces an $\overline{A}$ with positive rowsums, we now consider the modularity matrix $B = (b_{ij})$ associated with $\overline{A}$: Let $\mathbf{r} = (r_i)_{i=1}^n$ and $T = \sum_{i=1}^n r_i$.

$$B = \overline{A} - \frac{1}{T}\mathbf{r}\mathbf{r}^T.$$

That is, the entries of B are $b_{ij} = \overline{a}_{ij} - \frac{1}{T} r_i r_j$. We do not form B explicitly (since it will destroy the sparsity), i.e., we keep its sparse component $\overline{A}$ and the vector $\mathbf{r}$ separately.

Using B, we construct a hierarchy of interpolation matrices $[P_k]_{k=0}^l$ which define the hierarchy of coarse grid matrices used for AMG. $A_0 = A$ and $A_{k+1} = P_k^T A_k P^k$. To obtain one level of coarsening:

- Apply Luby's matching algorithm to B by merging locally maximal edges, only considering positive values. This gives an intermediate pairwise interpolation matrix, P.
- Obtain B_c by applying the interpolation to $\overline{A}$ and $\mathbf{r}$. Go back to step one to apply Luby's to B_c. After each recursive step, accumulate the new interpolation matrix into the old one by $P := PP_{new}$. Stop once P provides the desired coarsening factor and append it to the hierarchy.

Repeat the above steps until B no longer has any positive entries or the coarsest problem is small enough to solve directly.

This hierarchy of coarse problems with a chosen smoother defines a V-cycle AMG operator B_V and its inverse.

Adding B_V to a sequence of AMG operators defines a composite method. This composite method can now be used in place of the original smoother to search for another smooth error $\mathbf{w}$ vector to repeat all of the steps of the algorithm above. This allows the adaptivity of the solver. After each time adding a new multilevel B_V to the composite operator, we can test its convergence rate on the homogeneous problem with a random starting vector. We stop adding components when the convergence rate is satisfactory.