Application of Algebraic Multigrid

Charlotte Gils, ETH Zurich, January 2004

Contents

AMG Part 1: Setup (*AMGSetup*)

- Choice of coarse variables
- Determination of interpolation matrix P ($\Longrightarrow$ restriction and coarse-grid matrix)
- Determination of smoothing matrix M
- Input: matrix A, options
- Output: structures L (results) and times (timing information)

AMG Part 2: V-Cycle ($AMGVcycle$)

- Multigrid algorithm
- Pre-Smoothing
- Coarse grid correction
- recursive call of V-Cycle algorithm on next level
- Restriction back to fine level
- Post-Smoothing

Data structures

- Store data in structures
- Initialize: *structname.fieldname*=.. ,
  Access: *structname.fieldname*
- Advantage: easier data management
- Options on parameters and algorithms used in a matlab structure (see *AMGDefaultOptions*)
- Structure L{l} contains entire data of AMG processes, with l being the level number, so that there exists a structure L{l} for each level

Choice of coarse variables (*AMGSelectCoarseGrid*)

- Starting point: matrix A whose elements $a_{ij} \neq 0$ represent the existence and strength of a coupling between variables i and j
- Splitting of all variables in Ω according to strength of coupling
- Definition:
  A variable i is strongly n-coupled to another variable, j $\neq$ i, if

$$-a_{ij} \geq \theta \max_{a_{ik}<0} |a_{ik}| \qquad \textit{with fixed } 0 < \theta < 1 \tag{1}$$

(Note that all positive connections are weak).

- Create a matrix A^{str} containing only those a_{ij} with i being strongly n-coupled to j, all other elements of A are set to 0

- The set of all strongly negatively couplings of variable i is denoted as S_i:

$$S_i = \{j \in \Omega : i \; strongly \; negatively \; coupled \; to \; j\} \tag{2}$$

whereas

$$S_i^T = \{j \in \Omega : i \in S_j\} \tag{3}$$

consists of all variables j which are strongly n-coupled to i.

- C/F splitting in principle: choose i $\in$ C, all j $\in S_i^T$ become F-variables, define new k $\in$ C, ...
- But: Try to avoid randomly distributed C/F -patches
- Define a measure of importance, λ_i, of any undecided variable i to become the next C-variable,

$$\lambda_i = |S_i^T \cap U| + 2|S_i^T \cap F| \qquad (i \in U) \qquad (4)$$

i.e. the more strong n-couplings in U or F a variable i has, and the more of them have already been assigned to F, the bigger λ_i is.

The C/F splitting algorithm

- C/F Splitting

```
{
    C = $\emptyset$; F = $\emptyset$; U = $\Omega$;
    while ($U \neq \emptyset$)
    {
        get i $\in$ U with maximum $\lambda_i$;
        C = C $\cup\{i\}$; U = U $\setminus\{i\}$;
        for (j $\in S_i^T \cap U$)
        {
            F = F $\cup\{j\}$; U = U $\setminus\{j\}$;
            for (k $\in S_j \cap U$)        $\lambda_k = \lambda_k + 1$;
        }
        for (j $\in S_i \cap U$)        $\lambda_j = \lambda_j - 1$;
    }
}
```

Modification of the λ_i within the algorithm

- Initialize all λ_i in a vector of length n (number of elements in Ω, i.e. length of A) by setting all $a_{ji} \neq 0$ in A^{str}, setting all other elements to 1 and summing up over all those a_{ki} in a column i
- Direct modification of the λ_i in a vector: too much time needed for search of maximum λ_i, what to do with already assigned variables...?
- Linked List: easy to remove elements
- Clever modification of two matrices, so that finding maximum λ needs less time

Coarsening algorithm, continuation

- Initialize a vector of length n with all entries set to U (undecided)
- C/F splitting changes all elements of this vector to either C or F
- The set of equations can now be permuted and written in block form:

$$A\,u = \begin{pmatrix} A_{CC} & A_{CF} \\ A_{FC} & A_{FF} \end{pmatrix} \begin{pmatrix} u_C \\ u_F \end{pmatrix} = \begin{pmatrix} f_F \\ f_C \end{pmatrix} = f. \quad (5)$$

Considerations on efficiency

- Overall efficiency determined by

  a) the speed of convergence

  - depends from approximation of algebraically smooth error by interpolation
  - the stronger the F-to-C connectivity, the better the interpolation is ($\Longleftrightarrow$ uniform C/F-splittings)
  - strong F-to-C connectivity also via strongly coupled neighboring F-variables ($\longrightarrow$ Aggressive coarsening)

  b) the amount of work needed per cycle

  - directly related to the total memory requirement
  - better with fewer C-variables

- Goal is to fulfil both of these requirements

Interpolation (*AMGMakeInterpolation*)

- Interpolation

$$e_i^l = (Pe^{l+1})_i = \begin{cases} e_i^{l+1} & i \in C^l \\ \sum_{k \in P_i} w_{ik} e_k & i \in F^l \end{cases} \tag{6}$$

  with the interpolatory points k $\in P_i = C \cap S_i$

- Ruge-Stueben interpolation characterized by the approximation

$$e_i a_{ii} \approx - \sum_{j \in N_i} a_{ij} e_j \tag{7}$$

  where $N_i = \{j \in \Omega : j \neq i, \, a_{ij} \neq 0\}$ indicate all couplings of a variable i $\in \Omega$.

- $N_i = S_i \cup W_i$, where W_i are the variables j which are weakly connected to i
- a_{ij} with j $\in S_i$ contribute most to e_j
- $\Longrightarrow$ Lumping:

$$\sum_{j \in W_i} a_{ij} e_j \approx (\sum_{j \in W_i} a_{ij}) e_i \tag{8}$$

Eq. (7) now is

$$(a_{ii} + \sum_{j \in W_i} a_{ij}) e_i = - \sum_{j \in S_i} a_{ij} e_j \tag{9}$$

- Lumping (left side in (10)) implemented as

$$\sum_{j \in W_i} a_{ij} := (D_{FF})_{ii}$$
$$= \sum_{m,n=1}^{n_F,n_C} [(A_{FF})_{im} + (A_{FC})_{in}] - \sum_{m,n=1}^{n_F,n_C} [(A_{FF}^{str})_{im} + (A_{FC}^{str})_{in}] \tag{10}$$

- Include strong F-F connections into interpolation, i.e. eliminate all e_j where j $\in F_i^S$ ($F_i^S = F \cap S_i$)
- $\Longrightarrow$ replace e_j (where j $\in S_i$) with weighted average of those coarse grid errors, which are strongly n-coupled to both e_i and e_j:

$$e_j = \frac{\sum\limits_{k \in C_i \cap C_j} a_{jk} e_k}{\sum\limits_{k \in C_i \cap C_j} a_{jk}} \qquad for \;\; j \in F_i \tag{11}$$

- The resulting interpolation formula is:

$$(a_{ii}+\sum_{j\in W_i} a_{ij})e_i = -\sum_{k\in S_i\cap C} a_{ik}e_k - \sum_{j\in S_i\cap F} a_{ij}\frac{\sum\limits_{k\in C_i\cap C_j} a_{jk}e_k}{\sum\limits_{k\in C_i\cap C_j} a_{jk}} \tag{12}$$

$(C_i = \mathrm{C} \cap S_i)$

- Can also be written as:

$$(D_{FF})_{ii}e_i = -(A^{str}_{FC})_{ik}e_k - \frac{\sum\limits_{j\in S_i\cap F} (A^{str}_{FF})_{ij}(A^{str}_{FC})_{jk}}{\sum\limits_{k\in C_i\cap C_j} (A^{str}_{FC})_{ik}} e_k \tag{13}$$

- $\Longrightarrow$ Prolongation operator:

$$P = -\frac{\hat{A}_{FC}^{str}}{D_{FF}} \tag{14}$$

  where $\hat{A}_{FC}^{str}$ is the modified A_{FC} according to eq. 14.

- Note that D_{FF} is easy to invert
- Restriction R = P^T, if A is an M-matrix
- Coarse grid matrix now equals to

$$A^{l+1} = R^l\, A^l\, P^l \tag{15}$$

- Setup done for all levels l, until a minimum number of coarse variables or a maximum level is reached
- Call of V-Cycle

References

- Matlab Software "AMGToolbox", M. Verbeek, J. Cullum, W. Joubert, University of California at Los Alamos Laboratory, Office of Science, Mathematical, Information, and Computational Sciences, 1999.
- Appendix A, K. Stueben, in "Algebraic Multigrid", U. Trottenberg, K. Oosterlee, A. Schueller, Academic Press, 1999.
- "A Review of Algebraic Multigrid", K. Stueben, GMD Report, Sankt Augustion, 1999.