# Technical Report Multigrid

E. Ramona Stefanescu

Department of Mechanical Engineering, University of Buffalo,

ers32@buffalo.edu, http://www.acsu.buffalo.edu/~ers32/

**Abstract**

This report presents a succinct introduction into the multigrid framework.

**Key words.** Geometric multigrid, algebraic multigrid, smooth aggregation multigrid

# 1 Review

**Matrix Norm** – Let $A$ be a $n \times n$ matrix with elements $a_{ij}$. Consider the vector norm $\| x \|_p$ defined by

$$\| x \|_p = \left( \sum_{i=1}^{n} |x_i|^p \right)^{1/p}, \ 1 \leq p \leq \infty \tag{1.1}$$

$$\| x \|_\infty = \sup_{1 \leq i \leq n} |x_i|. \tag{1.2}$$

The matrix norm induced by the norm vector $\| \cdot \|_p$ is defined by

$$\| A \|_p = \sup_{x \neq 0} \frac{\| Ax \|_p}{\| x \|_p}. \tag{1.3}$$

For some particular values of $p$ the matrix norm induced can be written as:

$$\| A \|_1 = \max_j \sum_{i=1}^{n} |a_{ij}| \ (maximum \ column \ sum), \tag{1.4}$$

$$\| A \|_\infty = \max_i \sum_{i=1}^{n} |a_{ij}| \ (maximum \ row \ sum), \tag{1.5}$$

$$\| A \|_2 = \sqrt{spectral \ radius \ of \ A^T A} \tag{1.6}$$

The spectral radius of a matrix is given by:

$$\rho(A) = max|\lambda(A)|, \tag{1.7}$$

where $\lambda(A)$ denotes the eigenvalues of A. For symmetric matrices:

$$\| A \|_2 = \sqrt{(A^T A)} = \sqrt{(A^2)} = \rho(A) \tag{1.8}$$

**Singular M-matrices** - are important in the formulation of aggregation multigrid.

**Definition 1** *$A \in \mathcal{R}^{n\times n}$ is a singular M-matrix $\Leftrightarrow \exists\ B \in \mathcal{R}^{n\times n}, b_{ij} \geq 0, \forall$ i,j : $A = \rho(B)I$ - B.*

**Theorem 1 (Perron-Frobenius)** *Let $B \in \mathcal{R}^{n\times n}$, $b_{ij} \geq 0, \forall$ i,j. The following hold:*

- *$\rho(B)$ is an eigenvalue of B.*
- *$\exists x \in \mathcal{R}^n$, $x_i \geq 0\ \forall$ i: $Bx = \rho(B)x$ and $\exists y \in \mathcal{R}^n$, $y_i \geq 0\ \forall i : y^T B = \rho(B)y^T$.*
- *if B is irreducible, then the eigenvectors x and y in (2) are unique up to scaling, and the inequalities in (2) are strict.*
- *if B has a left or right eigenvector with strictly positive components, then this eigenvector has $\rho(B)$ as its eigenvalue.*

**Theorem 2** - *Some properties of M-matrices:*

- *Irreducible singular M-matrices have a unique solution to the problem $Ax = 0$, up to scaling. All components of x have strictly the same sign (i.e., scaling can be chosen s.t. $x_i > 0 \forall i$).*
- *An equivalent definition for singular M-matrices is: $A \in \mathcal{R}^{n\times n}$ is a singular M-matrix $\Leftrightarrow$ A is singular and all elements of $(A + \alpha I)^{-1}$ are nonnegative, $\forall \alpha > 0$.*
- *Irreducible singular M-matrices have nonpositive off-diagonal elements, and strictly positive diagonal elements ($n > 1$).*
- *If A has a strictly positive vector in its left or right nullspace and the off-diagonal elements of A are nonpositive, then A is a singular M-matrix.*

# 2 Introduction

Many phenomena in science and engineering are modeled mathematically with partial differential equations (PDEs). Solving a PDE analytical is often intractable, and alternative methods, such as numerical approximation, are needed. A discretization method, such as finite difference of finite elements, is used to approximate the original problem at unknowns, typically on a mesh [**?**, 2, 5]. This new problem takes the form of a linear system:

$$Au = f, \tag{2.1}$$

where A is an $n \times n$ matrix. This system is typically large ( millions of degrees of freedom (DOF)), sparse and ill-conditioned ( condition number approaches infinity as problem size increases).

To solve 2, two groups of linear solution methods exist: *direct* and *iterative*. Direct method include Guassian Elimination methods such as LU factorization, Cholesky factorization, etc. They have the advantage that solution time is insensitive to the conditioning

of the matrix. These methods, however, scale more poorly than iterative methods and are difficult to parallelize.

Iterative methods produce a sequence of approximate solutions:

$$u^{(k+1)} = u^{(k)} + \alpha^{(k)} d^{(k)}, \tag{2.2}$$

where $u^{(k)}$ is the approximate solution in the $k$th iteration and $\alpha^{(k)} d^{(k)}$ is an update vector that removes components of the error from $u^{(k)}$.

Types of iterative solvers:

- relaxation method: Jacobi, Gauss-Seidel
- Krylov methods: Conjugate Gradient, Generalized Minimal Residual
- Multilevel/ multigrid methods

The **relaxation techniques** use the residual $r$ to compute a correction to the current approximation:

$$r^{(k)} = f - Au^{(k)}, \quad e^{(k)} = u - u^{(k)} \tag{2.3}$$

$$r^{(k)} = f - Au^{(k)} = Au - Au^{(k)} = A(u - u^{(k)}) = Ae^{(k)} \tag{2.4}$$

The residual equation says that the error satisfies the same set of equations as the unknown $u$ when $f$ is replaced by the residual $r$.

$$Ae^{(k)} = r^{(k)} \tag{2.5}$$

The basic idea for the relaxation methods is that instead of solving for $Ae^{(k)} = r^{(k)}$ which is difficult, solve for a sparse linear system

$$B\hat{e}^{(k)} = r^{(k)}, \ where \ B \approx A \tag{2.6}$$

$$u^{(k+1)} = u^{(k)} + \hat{e}^{(k)} \tag{2.7}$$

The matrix $A$ is split in the form:

$$A = D - L - U \tag{2.8}$$

where $D$ is the diagonal $A$, and $-L$ and $-U$ are the strictly lower and upper triangular parts of $A$, respectively. $Au = f$ becomes $(D - L - U)u = f$.

*Jacobi method* is produced by solving the $j$th equation for the $j$th unknown and using the current approximation for the $(j-1)$st and $(j+1)$st unknowns. In matrix form we isolate the diagonal terms of $A$:

$$Du = (L + U)u + f \ \rightarrow \ u = D^{-1}(L + U)u + D^{-1}f. \tag{2.9}$$

The Jacobi iteration matrix is defined by:

$$R_J = D^{-1}(L + U) \tag{2.10}$$

while the weighted Jacobi iteration matrix is given by:

$$R_\omega = (1-\omega)I + \omega R_J. \tag{2.11}$$

which gives the updated approximation of the solution at the next step:

$$u^{(k+1)} = [(1-\omega)I + \omega R_J]u^{(k)} + \omega D^{-1} f \tag{2.12}$$

*Gauss-Seidel method* corresponds to solving the $j$th equation for $u_j$ and using new approximations for components $1, 2, \ldots, j-1$. The Gauss-Seidel iteration matrix is defined by:

$$R_G = (D-L)^{-1}U \tag{2.13}$$

resulting in the iteration:

$$u^{(k+1)} = R_G u^{(k)} + (D-L)^{-1} f. \tag{2.14}$$

**Convergence of relaxation schemes** Recalling that $e = u - u^{(k)}$ and $Ae^{(k)} = r^{(k)}$, we have

$$u^{(1)} - u^{(0)} = A^{-1} r^{(0)} \rightarrow u^{(1)} - u^{(0)} = Br^{(0)} \tag{2.15}$$

where $B$ is an approximation to $A^{-1}$. The iteration is form as followed:

$$u^{(1)} = u^{(0)} + Br^{(0)} = u^{(0)} + B(f - Au^{(0)}) = (I - BA)u^{(0)} + Bf = Ru^{(0)} + Bf \tag{2.16}$$

The general iteration matrix is written as $R = I - BA$. It can be shown that $m$ sweeps of this iteration result in

$$u^{(m)} = R^m u^{(0)} + C(f) \tag{2.17}$$

where $C(f)$ represents a series of operations on $f$ [1].

Following the same steps as above, the error after $m$ relaxation sweeps is given by

$$e^{(m)} = R^m e^{(0)} \tag{2.18}$$

For a particular vector norm, it is possible to bound the error after $m$ iterations

$$\| e^{(m)} \| \leq \| R \|^m \| e^{(0)} \| \tag{2.19}$$

This leads to the conclusion that if $\| R \| \leq 1$, the error is forced to zero as the iteration proceeds. It was showed in the literature that

$$\lim_{m \to \infty} R^m = 0 \; if \; and \; only \; if \; \rho(R) < 1. \tag{2.20}$$

$\rho(R)$ called the *asymptotic convergence factor* tells us approximately how many iterations are required to reduce the error by a factor of $10^{-d}$ equivalent with $\frac{\|e^{(m)}\|}{\|e^{(0)}\|} \leq 10^{-d}$. This condition will be approximately satisfied if

$$[\rho(R)]^m \leq 10^{-d}. \tag{2.21}$$

Solving for $m$, we have:

$$m \geq -\frac{d}{\log_{10}[\rho(R)]} \tag{2.22}$$

These iterative schemes work very well for the first several iterations, after that the convergence is slow and the entire scheme appears to stall. The explanation for this phenomenon is that the rapid decrease in error during the early iterations is due to the efficient elimination of the oscillatory modes of that error; but once the oscillatory modes have been removed, the iteration is much less effective in reducing the remaining smooth components [4]. This *smooth error* is a serious limitation of the relaxation methods. However, this limitation can be overcome using multigrid.

**Smooth error** - When a relaxation scheme begins to stall, the error that remains is considered *smooth error*. Starting from $A = (D + L + U)$, the iteration is developed:

$$\begin{aligned} Au &= f \\ (D - L)u &= f + Uu \\ u^{(k+1)} &= (D - L)^{-1} f + (D - L)^{-1} U u^{(k)} \end{aligned} \tag{2.23}$$

Add and subtract $(D - L)^{-1}(D - L)u^{(k)}$ to get:

$$u^{(k+1)} = u^{(k)} + (D - L)^{-1} r^{(k)} \tag{2.24}$$

where $r^{(k)} = f - Au^{(k)} = f - (D - L - U)u^{(k)}$. Subtracting 2 on both sides from the exact solution we get:

$$\begin{aligned} u - u^{(k+1)} &= u - (u^{(k)} + (D - L)^{-1} r^{(k)} \\ e^{(k+1)} &= e^{(k)} + (D - L)^{-1} r^{(k)} \end{aligned} \tag{2.25}$$

Using $r = Ae$, 2 can be written as:

$$e^{(k+1)} = [I + (D - L)^{-1} A] e^{(k)} \tag{2.26}$$

Smooth error, therefore, satisfies $e \approx [I + (D - L)^{-1} A]e$. This expression is true when $[I + (D - L)^{-1} A]e \approx 0$.

For methods such as weighted Jacobi or Gauss-Seidel the smooth error expression is reduced to

$$Ae \approx 0, \tag{2.27}$$

implying that smooth error is composed of eigenvectors whose eigenvalues are close to zero (i.e. are near-nullspace). Additionally, it states that smooth error has a small residual.

The set of **Krylov methods** include the Conjugate Gradient (CG) method for symmetric positive definite systems and the Generalized Minimal Residual (GMRES) method for non-symmetric systems. The CG method minimizes a functional $f(u) = \frac{1}{2}(r, A^{-1}r)$, where $r$ is the residual vector $r = f - Au$ and $(\cdot, \cdot)$ is an inner product. Minimizing this functional is equivalent to solving the system. The solution is improved iteratively. At each iteration a search direction, $d^{(k)}$ is chosen, along with a scalar $\tau^{(k)}$ designed to optimize the movement of the solution along that search direction. The solution is the improved by $u^{(k+1)} = u^{(k)} + \tau^{(k)}d^{(k)}$. Each iteration of the solver requires a new search direction. The first direction is chosen to be the residual vector and each of the following directions is chosen to be orthogonal to all of the previous ones through an efficient two-term recurrence relation. The iterations of this solver are therefore equivalent to the steps of building up the Krylov space associated with $A$ and $r^{(0)}$.

# 3 Geometric multigrid

Multigrid methods are fast iterative methods for the solution of large sparse systems arising from discretization of partial differential equations. The main idea of multigrid methods is to complement the local exchange of information in point-wise iterative methods by utilizing several related systems, called coarse levels, with a smaller number of variables. The coarse levels interact with the original system to propagate information globally. A multigrid method always consists of four ingredients namely

- the coarsening process,
- the transfer operators,
- the smoothing operators, and
- the coarse grid operators.

As it was establish above, many standard iterative methods possess the smoothing property, which makes these methods very effective at eliminating the high-frequency or oscillatory components of the error, while leaving the low-frequency or smooth components relative unchanged. It was observed that smooth modes on a finer grid look less smooth on a coarse grid, which suggested that when relaxation begins to stall, it is advise to move to a coarser grid. There, the smooth error modes appear more oscillatory and relaxation will be more effective.

All geometric multigrid approaches operate on pre-defined grid hierarchies. That is, the coarsening process itself is fixed and kept as simple as possible. Fixing the hierarchy, however, puts particular requirements on the smoothing properties of the smoother used in order to ensure an efficient interplay between smoothing and coarse-grid correction. Each level has

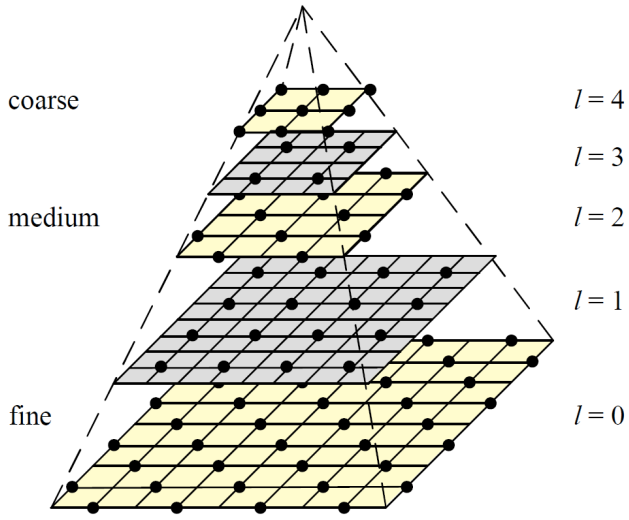



Figure 3.1: Multigrid levels

an associated grid, thus $P_{k+1}^k$ and $R_k^{k+1}$ can be constructed using geometric information of two consecutive meshes.

We want to construct a general multigrid method for a set of linear equations

$$A_h u = f_h, \tag{3.1}$$

the index indicates the level, h is the finest level, H will be the coarsest.

**Pre- and Post- Smoothing**

The pre- and post-smooths are sweeps of the iterative solver taken before and after the coarse grid correction. The pre-smoothing produces an initial, fine-grid, approximation to the solution. In this way high frequency error has been eliminated from this approximation using fine grid smooths. The post-smoothing is simply to re-adjust the solution after the corse grid correction (removing any high frequency components that the correction may be introduced). The number of pre- and post-smooths, $\nu_1$ and $\nu_2$ respectively, depends on the application but will typically range from zero to three.

**Intergrid transfer operations**

A first class of intergrid transfer operations involves moving vectors from a coarse grid to a fine grid. They are generally called *prolongation or interpolation* operators and are denoted by $P_2^1 : \mathcal{R}^{n_2} \to \mathcal{R}^{n_1}$ and $n_2 < n_1$. Many prolongation operators can be used from to linear and bilinear interpolation to some more complicated. We also need a *restriction* operator which moves vectors from a fine grid to a coarse one, $R_1^2 : \mathcal{R}^{n_1} \to \mathcal{R}^{n_2}$, for which we use $R_1^2 = (P_1^2)^T$. Now we can build the Galerkin projected matrix:

$$A_2 = R_1^2 A_1 P_2^1 \tag{3.2}$$

Repeating this step we end up with a set of prolongation matrices $P_{k+1}^k, k = 1, \ldots, L-1$, where $P_{k+1}^k : \mathcal{R}^{n_{k+1}} \to \mathcal{R}^{n_k}, n1 > n_2 > \cdots > n_L$, a set of restriction matrices $R_k^{k+1}$, and a set

of coarse level matrices $K_k$ and auxiliary matrices $H_k$ with

$$A_{k+1} = R_k^{k+1} A_k P_{k+1}^k \tag{3.3}$$

The above methods are easily specified in a recursive manner as shown in Algorithm 1. In this case each grid is pre-smoothed and the residual is restricted to the next, coarser, grid until the coarsest grid is reached. On the coarsest grid the problem is solved exactly. Then each grid receives a correction, performs post-smooths and passes the improved correction to the next, finer, grid.

**Algorithm 1:** $u_{MG}^{(k)} = MG_k(u_0^{(k)}, f^{(k)}, \nu_1, \nu_2, \mu)$

Relax $\nu_1$ times $A^{(k)}u^{(k)} = f^{(k)}$, with $u_0^{(k)}$ as the initial guess, to produce the approximation $u_1^{(k)}$
Form the residual $r^{(k)} = f^{(k)} - A^{(k)}u_1(k)$
Restrict the residual $f^{(k+1)} = R_k^{k+1} r^{(k)}$
**if** $k + 1 \neq L$ **then**
    $u_0^{(k+1)} = 0$;
    **for** $j \leftarrow 1$ **to** $\mu$ **do**
        $u_j^{(k+1)} = MG_{k+1}(u_{j-1}^{(k+1)}, f^{(k+1)}, \nu_1, \nu_2, \mu)$;
    **end**
    $u^{(k+1)} = u_\mu^{(k+1)}$
**else**
    solve the level $L$ system directly to obtain $u^{(L)}$;
**end**
Interpolate and add the correction $u_2^{(k)} = u_1^{(k)} + P_{k+1}^k u^{(k+1)}$
Relax $\nu_2$ times on $A^{(k)}u^{(k)} = f^{(k)}$, with $u_2(k)$ as the initial guess, to produce the approximation $u_{MG}^{(k)}$

While for large $\nu_1$, $\nu_2$ and $\mu$ the cycle might be viewed as a direct solver, it is much more practical to consider small $\nu_1$, $\nu_2$ and $\mu$ and analyze it as an iterative method. A multigrid with $\mu = 1$ is often referred as a $V$-cycle, while the $\mu = 2$ is name a $W$-cycle.

**Convergence analysis** - The most common way of proofing the convergences is the splitting of the error components in two classes. One that can be reproduced on coarser levels/ grids and therefore can be reduced by the coarse grid correction and one that has to be reduced by the smoother. For the geometric multigrid the first group consists typically of low frequency parts, the second of high frequency parts of the error. Thus, the two-grid convergence proofing is done by splitting the two-grid iteration operator

$$\mathcal{M}_k^{k+1} = (I - P_{k+1}^k A_{k+1}^{-1} R_k^{k+1} A_k)\mathcal{S}_k^\nu \tag{3.4}$$

where $\nu$ is the number of smoothing steps (i.e. $\nu = \nu_1$ in Algorithm 1) and $\mathcal{S}_k$ the iteration matrix of the smoother. The convergence is shown by splitting the sum. Here the projections

$P^{low}$ and $P^{high}$ (which project on the subspace spanned by the low and high frequency eigenvectors of the system matrix) are introduced and the identity, decomposed into $I = P^{low} \oplus P^{high}$, is inserted into $\mathcal{M}_k^{k+1}$ (left of $\mathcal{S}_k^\nu$) to get the estimate

$$\begin{aligned} \| \mathcal{M}_k^{k+1} \| \leq & \| (I - P_{k+1}^k A_{k+1}^{-1} R_k^{k+1} A_k) P^{low} \| \| \mathcal{S}_k \|^\nu \\ & + \| (I - P_{k+1}^l A_{k+1}^{-1} R_k^{k+1} A_k) \| \| P^{high} \mathcal{S}_l^\nu \| \end{aligned} \tag{3.5}$$

Two-grid convergence is then proven by showing that the term $\| (I - P_{k+1}^k A_{k+1}^{-1} R_k^{k+1} A_k) P^{low} \|$ is small and that $\| P^{high} \mathcal{S}_l^\nu \|$ is arbitrary small for sufficiently many smoothing steps $\nu$.

While geometric multigrid can be quite effective, its performance is quite problem dependent, and, in many situations, this performance breaks down. The bilinear interpolation used is appropriate in the case when we expect small changes in the fine-grid solution, between coarse and fine grids. For some problems, particularly those with discontinuous coefficients, this is not necessarily the case. A breakdown occurs in the case of strong anisotropy or convection. In these cases, interpolation that ignores the directionality of the operator cannot properly approximate error components whose smoothness shares this directionality. Accurate interpolation is also difficult when the geometry is not regular. In this case a simple bilinear interpolation based on the assumption of equal mesh spacing can significantly over- or under- estimate the appropriate correction factor.

# 4 Algebraic multigrid (AMG)

AMG gained popularity due to its efficiency in solving certain classes of large, sparse linear systems and has been shown to converge independently of problem size for a number of problems. The major differences between AMG and geometric multigrid methods are in the choice of the coarse grids and intergrid transfer operators [3]. While geometric multigrid makes use of additional information, AMG makes these choices solely on the matrix, A. Since the original theory of AMG was developed for elliptic differential equations, matrix A is assumed to be a symmetric $M - matrix$: it is symmetric ($A^T = A$) and positive-definite ($u^T A u > 0$ for all $u \neq 0$) and has positive diagonal entries and nonpositive off-diagonal entries. These assumptions are not necessary for AMG to work.

In AMG, smooth error does not necessarily satisfy the same conditions of smoothness. The smooth error in both cases is, however, the result of the relaxation scheme stalling. One of the concepts in AMG is that *smooth error varies slowly in the direction of relatively large coefficients of the matrix*, which can be written as:

$$\sum_{i<j} (-a_{ij})(e_i - e_j)^2 \ll 1 \; for \; -a_{ij} > 0. \tag{4.1}$$

Algebraic multigrid algorithms execute in two phases: the *setup phase* and the *solve phase*. The AMG setup phase is responsible for selecting coarse grids, building prolongation and restriction operators, and constructing the coarse-level operators. The solve phase uses these products and implements a multigrid cycle using relaxation, restriction and prolongation.

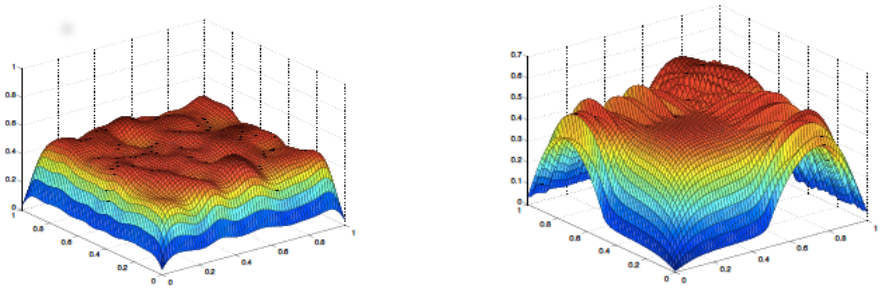


Figure 4.1: a) Geometrically smooth error b) Algebraically smooth error

**Coarse-grid selection** - All AMG algorithms select coarse grids to accurately represent smooth error, to be suitable for accurately interpolating vectors from the coarse grid to the fine grid, and to be significantly smaller than the fine grid. The coarsening procedure partitions the grid into $C$-points (points on the coarse grid) and $F$-points (points not on the coarse grid) - $C/F$ splitting.

Most of AMG rests on two fundamental concepts: *smooth error* and *strength of connection*. The classical strength of connection measure is based in the magnitude of off-diagonal entries in A. The set of unknown $i$ strongly depends upon is defined as:

$$S_i = \{j : i \neq j \; and \; |a_{ij}| \geq \theta \max_{k \neq i} |a_{ik}|\}, \tag{4.2}$$

where $a_{ij}$ is the entry in row $i$, column $j$ of matrix $A$ and $0 < \theta \leq 1$. Here, $\theta$ is a predetermined threshold is 0.25. The set of unknowns that $i$ strongly influences, denoted $S_i^T$, is defined as the set of unknowns that strongly depend on $i$:

$$S_i^T = \{j : i \in S_j\}. \tag{4.3}$$

The set of coarse-grid variables used to interpolate the value of the fine-grid variable $u_i$, called the coarse interpolatory set for $i$, is denoted by $C_i$.

There are two criteria for choosing the coarse grid points:

- **F - F dependence** For each ı $\in F$, each point $j \in S_i$ should either be in $C$ itself or should depend on at least one point in $C_i$.
- **Maximal Subset** $C$ should be a maximal subset with the property that no $C$-point depends on another.

These criteria cannot generally be satisfied simultaneously. The first criteria is required by the classical AMG interpolation scheme, so it must be satisfied. The second criteria, on the other hand, is used to guide the selection of coarse grids with few $C$-points.

**Transfer Operator Construction** The prolongation operators $P_{k+1}^k$ transfers vectors from coarse levels to finer levels $P_{k+1}^k e^{(k+1)} = e^{(k)}$ and is based on entries in $A$:

$$P_{e_i} = \begin{cases} e_i & \text{if } i \in C, \\ \sum_{j \in C_i} \omega_{ij} e_j & \text{if } i \in F. \end{cases} \tag{4.4}$$

where $w_{ij}$ is defined as:

$$w_{ij} = \frac{a_{ij} + \sum_{m \in D_i^s} \left( \frac{a_{im} a_{mj}}{\sum_{k \in C_i} a_{mk}} \right)}{a_{ii} + \sum_{n \in D_i^\omega} a_{in}} \tag{4.5}$$

$D_i^s$ denotes the neighboring fine-grid points that strongly influence $i$, while $D_i^\omega$ is the set of points that do not strongly influence $i$ - the set of *weakly connected neighbors*.

Once we have chosen the coarse points, $C$, and interpolation operator, $P$, we must still choose a restriction operator $R$, and a coarse level operator, $A_C$. Assuming that $A$ is a symmetric positive-definite matrix, it is natural to define these operators by the Galerkin conditions: $R = P^T$ and $A_C = RAP$.

The Setup AMG stages is presented in Algorithm 2. The Solve Phase is similar to the one shown in Algorithm 1.

**Algorithm 2:** AMG Setup Phase

$k \leftarrow 0$
**while** $k \neq L$ **do**
  Select grid points
  Build prolongation operator $P_{k+1}^k$ from level $k+1$ to $k$
  $R_k^{k+1} \leftarrow (P_{k+1}^k)^T$
  $A_{k+1} \leftarrow R_k^{k+1} A_k P_{k+1}^k$
  $k \leftarrow k+1$
**end**
$m \leftarrow k$ store the number of levels in hierarchy

AMG is an efficient solver for many problems, including those involving discretization on irregular grids, or discretization of many problems with anisotropic or variable coefficients. There are, however, still many problems for which AMG is not entirely effective, including problems with highly anisotropic or highly variable coefficients, or those coming from the discretization of certain systems of PDEs such as Stokes or nearly incompressible linear elasticity.

# 5 Smooth aggregation multigrid

While classic AMG is developed based on the null space properties of typical discretization of elliptic operators, the intergrid transfer operators used in smoothed aggregation (SA) are based on explicit knowledge of the near null space of the discretized operator. These methods are particularly effective for problems such as elasticity where the near null space is the span of several distinct vectors, all of which can be easily incorporated into the interpolation and restriction operators [**?**]. One if the most interesting aspects of SA is its approach for defining interpolation. In all the methods we have discussed so far, interpolation is viewed

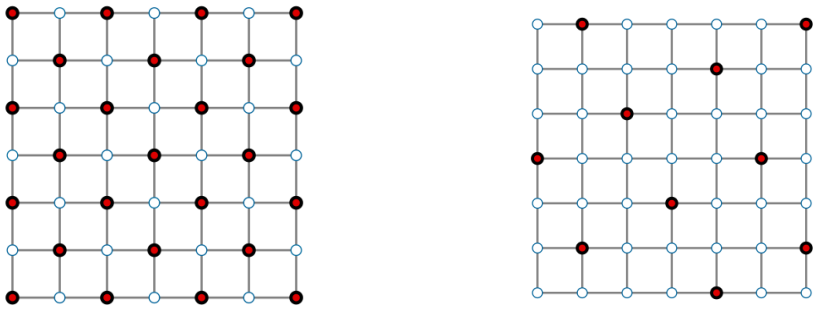
(and constructed) as rows of $P$, i.e. we think of interpolation as being "to point $i$". However, if $p_j$ are the columns of $P$, then interpolation of a coarse-grid vector $e_c$ can be written as:

$$Pe_c = \sum_j e_{c,j} p_j. \tag{5.1}$$

The interpolation can be viewed as a linear combination of basis functions $p_j$. The SA algorithm takes this view and builds a set of sparse (local) basis functions from a given small set of near-null space components.

In order to construct a multigrid hierarchy with smoothed aggregation, one requires a set of $K$ vectors that span near-null space of A. These vectors are partitioned into sub vectors over each aggregate. These local representations of the near null space are then orthogonalized via the $QR$ factorization. The $Q$ factor for a given aggregate spans the same local space of as the original $K$ vectors restricted to the aggregate.

The tentative prolongation operator, $P$, is then formed by assembling these local factors; the first $K$ columns contain the $Q$ from the first aggregate in the rows corresponding to that aggregate and zeros elsewhere.



(a) (b)

Figure 5.1: a) final AMG (25 points) b) final SA (10 points)

# 6 Multiscale segmentation

Clustering is obtained through a process of unsupervised learning in which the data is split into clusters, which reveals its inner cluster. Multiscale segmentation is applied to the case of image segmentation. Image segmentation is a process of grouping together neighboring pixels whose properties are coherent. Many of the image segmentation graphs construct a graph in which the nodes represent the pixels in the image and arcs represent affinities between nearby pixels. The image is segmented by minimizing a cost associated with cutting the graph into subgraphs. The algorithm is based on representing the minimization problem at different scales. Because of its multiscale nature, the algorithm provides a full hierarchical decomposition of the image into segments in just one pass.

Given a image, first a graph is constructed so that every pixel is a node in the graph and neighboring pixels are connected by an arc. A weight is associated with the arc reflecting the likelihood that the corresponding pixels are separated by an edge. To find the minimal cuts in the graph, we recursively coarsen the graph using a *weighted aggregation* procedure - borrowed from AMG, in which we repeatedly select smaller sets of representative pixels (blocks). The purpose of these coarsening steps is to produce smaller and smaller graphs that faithfully represent the same minimization problem.

# References


[1] W. L. Briggs, S. F. McCormick, et al., *A multigrid tutorial*, vol. 72, Siam, 2000.

[2] S. F. McCormick, *Multigrid methods*, vol. 3, siam, 1987.

[3] K. Stuben, *An introduction to algebraic multigrid*, Multigrid, (2001), pp. 413–532.

[4] P. Vaněk, J. Mandel, and M. Brezina, *Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems*, Computing, 56 (1996), pp. 179–196.

[5] P. Wesseling, *Introduction to multigrid methods.*, tech. rep., DTIC Document, 1995.