FRIEDRICH-ALEXANDER-UNIVERSITÄT ERLANGEN-NÜRNBERG

INSTITUT FÜR INFORMATIK (MATHEMATISCHE MASCHINEN UND DATENVERARBEITUNG)

Lehrstuhl für Informatik 10 (Systemsimulation)

Multigrid Conjugate Gradient Method

Sudarsan N.S Acharya

Master Thesis

Multigrid Conjugate Gradient Method

Sudarsan N.S Acharya

Master Thesis

Aufgabensteller:	Prof. Dr. Christoph Pflaum
Betreuer:	Prof. Dr. Christoph Pflaum
Bearbeitungszeitraum:	1.04.2006 – 30.09.2006

Erklärung:

Ich versichere, daß ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und daß die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Erlangen, den 2. Oktober 2006 .....

Abstract

The multigrid conjugate gradient method is an optimal, fast converging and easy to implement numerical method, based on the multigrid and conjugate gradient methods, for ellipic PDEs with large jumping coefficients. This approach constructs new correction directions from correction spaces, that are constructed using multilevel restricted residual vectors. Further improvements in constructing the correction spaces, to achieve better convergence rates, using smoothened versions of the restricted residual vectors and using V-cycle type restricted residual vectors, are presented. Several combinations of the correction spaces were numerically tested to obtain the best possible convergence rates. Numerical results for a Poisson's equation with large jumping coefficients are presented. The numerical results are a strong indication that, correction spaces constructed using the normally restricted and V-cycle type residual vectors lead to a faster convergence rate.

To Thatha, Patti, Appa, Amma, Vikki, Vasu and all my teachers

Acknowledgements

Working with the multigrid and conjugate gradient methods was a wonderful experience and I'm very glad that there is more to come. I'm extremely grateful to my professor, mentor and guide Dr. Christoph Pflaum, not just for the 6 months of exciting thesis work, but also for his motivating lectures and discussions, that have immensely helped me achieve my scientific goals, so far. Thanks to Dr. Norman Uhlmann of the Fraunhofer Institut for his personal support. My gratitude goes to all my teachers and friends in Erlangen, and to our wonderful computing centre - the CIP pool, that has provided shelter for many souls including me, more than any apartment in Erlangen!

Contents

1 Introduction **12**

2 Multigrid Gradient based Methods **15**

2.1 Theoretical Background 15

2.2 Multigrid Gradient Method (MGgM) 18

2.2.1 Variants of the MGgM 26

2.2.2 MGgM with Rough and Smooth Correction Directions (MGgM-3) 27

2.3 Multigrid Conjugate Gradient Method (MGgM) 34

2.3.1 Variants of the MGcgM 40

2.3.2 MGcgM with Smooth and Rough Correction Directions (MGcgM-3) 41

2.4 Computational Complexity of the MGgM and MGcgM 49

3 Multilevel V-cycle based Space Correction Methods **50**

3.1 Multilevel V-Cycle based Space Correction Method (MLV-SCOM) 50

3.2 Multilevel V-Cycle based Conjugate Space Correction Methods (MLV-CSCOM) 53

3.2.1 Variants of the MLV-CSCOM 54

3.2.2 MLV-CSCOM-3A 54

3.2.3 MLV-CSCOM-3B 54

4 Numerical Results and Observations **61**

Bibliography **76**

List of Figures

4.1 Jumping coefficient and its contour, for $\alpha = 10^6$ in Example 1. 63

4.2 Jumping coefficient and its contour, for $\alpha = 10^6$ in Example 2. 63

4.3 **Example 1**, improved MGgM convergence with smooth correction directions constructed using $\mathcal{R}^k_{smooth}$, with $\alpha = 10^3$, $N \approx 2^{18}$. 65

4.4 **Example 2**, improved MGgM convergence with smooth correction directions constructed using $\mathcal{R}^k_{smooth}$, with $\alpha = 10^3$, $N \approx 2^{18}$. 66

4.5 **Example 1**, improved convergence with partial A-orthonormalisation of successive correction spaces, with $\alpha = 10^5$. 67

4.6 **Example 2**, improved convergence with partial A-orthonormalisation of successive correction spaces, with $\alpha = 10^5$. 67

4.7 **Example 1**, Comparison of V-cycle multigrid method with and without postsmoothing, with $\alpha = 10^3$. 68

4.8 **Example 2**, Comparison of V-cycle multigrid method with and without postsmoothing, with $\alpha = 10^3$. 69

4.9 **Example 1**, Comparison of V-cycle multigrid method with and without postsmoothing, with $\alpha = 10^5$. 69

4.10 **Example 2**, Comparison of V-cycle multigrid method with and without postsmoothing, with $\alpha = 10^5$. 70

4.11 **Example 1**, improved convergence with V-cycle based smooth correction directions, with $\alpha = 10^4$. 71

4.12 **Example 1**, improved convergence with V-cycle based smooth correction directions, with $\alpha = 10^5$. 72

4.13 **Example 1**, improved convergence with V-cycle based smooth correction directions, with $\alpha = 10^6$. 72

4.14 **Example 2**, improved convergence with V-cycle based smooth correction directions, with $\alpha = 10^4$. 73
4.15 **Example 2**, improved convergence with V-cycle based smooth correction directions, with $\alpha = 10^5$. 73
4.16 **Example 2**, improved convergence with V-cycle based smooth correction directions, with $\alpha = 10^6$. 74

List of Tables

1 **Example 1**, number of iterations to achieve $\| b - Ax^k \|_2 < 10^{-8}$, with $N \approx 2^{18}$. 64

2 **Example 2**, number of iterations to achieve $\| x^k - \hat{x} \|_2 \approx 10^{-5}$, with $N \approx 2^{18}$. 64

3 **Example 1**, optimal convergence rate ($\| b - Ax^k \|_2 < 10^{-12}$) independent of number of unknowns N, with $\alpha = 10^3$. 65

4 **Example 1**, MGcgM-1+Mg(10) - number of iterations required to achieve $\| b - Ax^k \|_2 < 10^{-9}$ for different values of the parameter α, with $s = 10$. . 74

5 **Example 1**, number of iterations required to achieve $\| b - Ax^k \|_2 < 10^{-9}$ for different values of the parameters α and s. 75

List of Algorithms

1 Multigrid Gradient Algorithm (MGgM) 25
2 Multigrid Gradient Algorithm with Rough and Smooth Correction Directions (MGgM-3) 32
3 Multigrid Conjugate Gradient Algorithm (MGcgM) 38
4 Multigrid Conjugate Gradient Algorithm with Rough and Smooth Correction Directions (MGcgM-3) 45
5 Multilevel V-cycle based Space Correction Method (MLV-SCOM) 53
6 Multilevel V-cycle based Space Correction Method with Rough and Smooth Correction Directions Partially A-orthonormalised (MLV-CSCOM-3A) . . 55
7 Multilevel V-cycle based Space Correction Method with Rough Correction Directions Partially A-orthonormalised (MLV-CSCOM-3B) 57

Chapter 1

Introduction

Numerical solution techniques for solving linear systems of equations arising from the discretisation of elliptic partial differential equations (PDEs) with large jumping coefficients is an active research area. Such coefficients typically represent material properties such as conductivity and permeability in heterogeneous media. Such a situation arises, for example, in simulation of heat and electricity conduction in composite materials [6], oil reservoir simulation, and simulation of subsurface fluid flow and contaminant migration [1]. A PDE of fundamental importance that models the above mentioned phenomena is,

$$\frac{\partial u}{\partial t} - \nabla \,.\, a(\mathbf{x}) \nabla u = f$$

and in steady state

$$-\nabla \,.\, a(\mathbf{x}) \nabla u = f. \tag{1.1}$$

Equation (1.1) represents many different physical phenomena. For example, it represents the equation of steady state heat (electric) conduction through composite materials, with $a(\mathbf{x})$ representing the thermal (electric) conductivity tensor and u representing the temperature (electric potential). It also represents the pressure equation of single phase steady flow through porous media, with $a(\mathbf{x}) = k/\mu$, where k is the permeability tensor and μ is the fluid viscosity, and u representing the pressure. To add one more, it also represents the equation of electrostatics in heterogeneous media, with $a(\mathbf{x})$ representing

the dielectric constant and u representing the electric potential.

To numerically solve equations like (1.1), optimal iterative solvers such as the multigrid method and several variants exist. But the performance of the classical multigrid method with standard coarsening deteriorates for equations like (1.1) with coefficients that have large jumps, or are oscillatory, or are anisotropic in nature. Standard multigrid techniques like linear interpolation and full coarsening result in a slow convergence for such problems. Other strategies such as problem dependent prolongation and restriction operators [2] and jump-interface preserving coarsening [13] involve complicated implementation details.

The *multigrid conjugate gradient method* (MGcgM) is a new approach to combine the conjugate gradient and multigrid methods [8]. The MGcgM can tackle the problem of large jumping coefficients, is computationally optimal, and is easy to implement. Similar to the multigrid method, the MGcgM also requires the restriction of the residual vector to coarser grids. The space spanned by the restricted residual vectors is called the *multilevel residuum space* $\mathcal{R}$. The idea of making a correction in a particular correction direction as in the steepest descent (gradient) and conjugate gradient method is applied now. The initial correction direction is calculated from the space $\mathcal{R}$. After this, a new *multilevel correction space* $\mathcal{D}$ is calculated such that it is A-orthogonal to the old correction space. The new correction direction is now obtained from the space $\mathcal{D}$ and this process is iterated to obtain rapid convergence.

MGcgM is *computationally optimal* because the correction spaces are spanned by vectors that are stored not just on the finest level of the multigrid, but on different coarser levels. Operations like matrix-vector multiplication, involving those vectors happen at all the levels of the multigrid. MGcgM is *easy to implement* because it does not involve complicated problem specific variations such as the definition of problem dependent prolongation and restriction operators. Theoretical results and numerical experiments from Pflaum [8], reveal that a particular variant of the MGcgM results in a faster convergence compared to other combinations of the multigrid and gradient methods, such as the multigrid preconditioned conjugate gradient method, for systems of equations arising from a Poisson's and Stokes problem with large jumping coefficients.

In this work I describe two modified approaches for the MGcgM that lead to a significant increase in convergence rate for elliptic P.D.Es with large jumping coefficients. In

the first approach, the *multilevel residuum space* $\mathcal{R}$ is made bigger by using smoothened residual vectors. In the second approach, the *multilevel residuum space* $\mathcal{R}$ is made bigger by using V-cycle type residual vectors. The *multilevel correction space* $\mathcal{D}$ is then constructed from these enlarged *multilevel residuum spaces*. We shall in fact see that such (enlarged) correction spaces result in a drastic improvement in the convergence behaviour. I will explain these approaches in detail in the following chapters.

I performed numerical experiments for a Poisson's equation with large jumping coefficients and the results indicate that a bigger correction space leads to faster convergence. Two test problems were considered for the numerical experiments, to compare the performances of the two modified approaches. The numerical methods were implemented and tested C++ expression templates based library 2D-EXPDE [4].
I would like to present the reader a brief description of the chapters that follow.

Chapter 2 starts with the forerunner of the numerical methods described in the further chapters. This is the *multigrid gradient method* (MGgM). We will go through the necessary theoretical background and problem formulation. We take here a first look at correction with a bigger space for the MGgM and I describe the corresponding algorithms. After this, we discuss the MGcgM and its variants.

Chapter 3 presents the *multilevel V-cycle based space correction methods* (MLV-SCOM). We will discuss here the key idea of using V-cycle type residual vectors to construct a bigger correction space. The natural extension to the MLV-SCOM, the *multilevel V-cycle based conjugate space correction method* (MLV-CSCOM) and its variants will be discussed next.

Chapter 4 presents the numerical results and observations for a Poisson's problem with jumping coefficients. We will see here, which of the two modified approaches leads to improved convergence behaviour for such problems.

Chapter 2

Multigrid Gradient based Methods

2.1 Theoretical Background

Let us consider problems of the form:

▶ find $x \in \mathbb{R}^n$ such that:

$$Ax = b, \tag{2.1}$$

where $A \in \mathbb{R}^{n \times n}$ is a real, symmetric and positive definite matrix and $b \in \mathbb{R}^n$ is a known vector. For our purposes, let us assume that the matrix A arises from a finite element discretisation of a P.D.E. We can now define a scalar product in the space $\mathbb{R}^n$ as:

$$(x, y)_A = a(x, y) := x^T A y \quad \forall x, y \in \mathbb{R}^n.$$

This scalar product induces the energy norm:

$$\| x \|_A = \sqrt{x^T A x} \quad \forall x \in \mathbb{R}^n.$$

In the multigrid gradient based methods, we combine the core ideas of the gradient and the multigrid methods. First, let us consider that of the gradient method. That is, the solution to the equation (2.1) also satisfies the following *minimisation problem*:

▶ find $x \in \mathbb{R}^n$ such that $f(x)$ is minimum, where $f \colon \mathbb{R}^n \mapsto \mathbb{R}$, is a functional defined as:

$$f(x) = \frac{1}{2}\, a(x, x) - b^T x. \tag{2.2}$$

Next, we consider the *multilevel concept* of the multigrid method. Let us assume that there exists a sequence of finite dimensional coarse grids Ω_i $(i = 1, \ldots, l)$, each of size $n_i = |\Omega_i|$, such that:

$$n = n_l > n_{l-1} > n_{l-2} > \cdots > n_1.$$

Then we need *operators for inter-grid transfers*. These are the prolongation and restriction operators. The prolongation operators can be defined as:

$$I_i^{i+1} \colon \mathbb{R}^{n_i} \mapsto \mathbb{R}^{n_{i+1}}, \quad i = (l-1), \ldots, 1, \tag{2.3}$$

and the restriction operators can be defined as:

$$I_i^{i-1} \colon \mathbb{R}^{n_i} \mapsto \mathbb{R}^{n_{i-1}}, \quad i = l, \ldots, 2\,. \tag{2.4}$$

For certain finite element constructions on multilevels, the prolongation and restriction operators can be naturally defined through the relationship between the basis functions corresponding to different levels. Such finite element constructions share the property that the sequence of coarse grid finite element spaces are related as:

$$\Omega_1 \subset \Omega_2 \subset \cdots \subset \Omega_{l-1} \subset \Omega_l. \tag{2.5}$$

Examples of such finite element constructions are linear finite element spaces on triangulations and bilinear finite element spaces on rectangular grids. If $\psi_i^{k'}$ and ψ_{i+1}^{k} with $k' \in \Omega_i$ and $k \in \Omega_{i+1}$, are basis functions corresponding to the grid spaces Ω_i and Ω_{i+1} respectively, then according to (2.5), we have the following relationship between those basis functions:

$$\psi_i^{k'} = \sum_{k \in \Omega_{i+1}} \gamma_k^{k'} \psi_{i+1}^{k}, \quad \text{where } \gamma_k^{k'} \in \mathbb{R}. \tag{2.6}$$

Using equation (2.6), we can now construct the prolongation and restriction operators respectively, between the spaces Ω_i and Ω_{i+1} as follows:

$$I_i^{i+1} = \left[\gamma_k^{k'}\right]_{(k \in \Omega_{i+1}\,,\, k' \in \Omega_i)} \quad \text{and} \quad I_{i+1}^{i} = \left[\gamma_k^{k'}\right]_{(k' \in \Omega_i\,,\, k \in \Omega_{i+1})}.$$

That is the prolongation and restriction operators between any two levels of the multigrid are transposes of each other. Let us complete our discussion on inter-grid transfer operators, by defining I_i^i to be the unit matrix I corresponding to the grid space Ω_i.

The next step is to define the coarse grid versions A_i of the matrix A. We will do it using the *Galerkin coarse grid operator* approach. That is,

$$A_i = I_{i+1}^i \, A_{i+1} \, I_i^{i+1} \quad i = l-1, \ldots, 1 \, . \tag{2.7}$$

I would like to point here that the construction (2.7) is a stable approach to compute the coarse grid stiffness matrices for problems with large jumping coefficients [9]. We now can make the following abbreviations, if we consider each of (2.3), (2.4), and (2.7) respectively as a recursion:

$$\begin{aligned} I_l^m &= I_{m-1}^m \, I_{m-2}^{m-1} \cdots I_l^{l+1} \quad (l < m), \\ I_m^l &= I_{l+1}^l \, I_{l+2}^{l+1} \cdots I_m^{m-1} \quad (l < m), \\ A_i &= I_l^i \, A_l \, I_i^l. \end{aligned}$$

To complete our discussion with respect to the multigrid method, we define matrices M_i as:

$$M_i \colon \mathbb{R}^{n_i} \mapsto \mathbb{R}^{n_i}, \quad i = l, \ldots, 1 \, , \tag{2.8}$$

such that the matrices M_i are *diagonally dominant, symmetric, positive and definite*. In a finite element context, for a given level i, we can choose M_i to be the *mass matrix*. One other choice for M_i is the *identity matrix* corresponding to level i.

We have now come to a stage to combine the ideas mentioned above. Let us assume that x^0 is an initial guess to solve the equation (2.1) and x^k is the approximate solution after k iterations. Then the *residual vector* for the kth iteration is defined as:

$$r^k = b - Ax^k, \tag{2.9}$$

and we define a *multilevel residuum space* for the kth iteration as:

$$\mathcal{R}^k := span\{w_l, w_{l-1}, \ldots, w_2, w_1\}, \tag{2.10}$$

where,

$$\begin{aligned} w_l &= r^k, \\ w_i &= I_i^l \, M_i^{-1} \, I_l^i \, r^k, \quad i = (l-1), \ldots, 1 \, . \end{aligned} \tag{2.11}$$

I would like to point here that the vectors $w_i \in \mathbb{R}^n$. This is to avoid a possible confusion later on, when we deal with vectors like $v_i \in \Omega_i$, where the subscript denotes the coarse grid space in which the vector exists.

In equation (2.11), we can consider M_i (see definition (2.8)) as matrices that relax or smoothen the residual vectors at each level of the multigrid. This can be achieved, for example, using a Gauss-Seidel smoother. If M_i are chosen to be the identity matrices, then we call the space $\mathcal{R}^k$ as the *rough multilevel residuum space* $\mathcal{R}^k_{rough}$. If M_i are chosen to be finite element mass matrices, then we call the space $\mathcal{R}^k$ as the *smooth multilevel residuum space* $\mathcal{R}^k_{smooth}$. Let us see now, how we can make use of the multilevel residuum space $\mathcal{R}^k$ to construct the *multigrid gradient method*.

2.2 Multigrid Gradient Method (MGgM)

In the classical gradient method, to solve the equation (2.1) for the kth iteration with an approximation x^k, we find a correction direction d, such that the new approximation x^{k+1} minimises the functional f in equation (2.2). We do the same for MGgM, but now the correction direction d comes from the multilevel residuum space $\mathcal{R}^k$. This leads us to the following formulation of the MGgM:

▶ find $d \in \mathcal{R}^k$ such that:

$$f(x^k + d) \rightarrow \text{minimum}, \tag{2.12}$$

$$x^{k+1} = x^k + d. \tag{2.13}$$

But the above formulation does not help us much from the computational viewpoint. Let us therefore make an equivalent reformulation of MGgM as follows:

▶ find $d \in \mathcal{R}^k$ such that:

$$\begin{aligned} a(x^k + d, q) &= b^T q \quad \forall q \in \mathcal{R}^k, \\ x^{k+1} &= x^k + d. \end{aligned} \tag{2.14}$$

That is, d is the best possible correction direction that can be constructed from the space $\mathcal{R}^k$ with respect to the energy norm $\| \, . \, \|_A$. We can prove this statement and the equivalence of the formulations (2.12) and (2.14) as follows.

Let us define a function $g(\beta) = f(x^k + d + \beta q)$, where $q \in \mathcal{R}^k$ and $\beta \in \mathbf{R}$. Then we see that the function g, according to (2.12), takes a minimum for $\beta = 0$. That is $g'(\beta) = 0$ for $\beta = 0$. This implies,

$$\begin{aligned} g(\beta) &= \frac{1}{2} \, a(x^k + d + \beta q, x^k + d + \beta q) - b^T (x^k + d + \beta q) \\ \Rightarrow g(\beta) &= \frac{1}{2} \left(x^k + d + \beta q\right)^T A \left(x^k + d + \beta q\right) - b^T (x^k + d + \beta q). \end{aligned}$$

Expanding this and differentiating w.r.t β and using $g'(\beta)|_{\beta=0} = 0$, completes the proof as follows:

$$\begin{aligned} g'(\beta) &= q^T A x^k + q^T A d + \beta (d^T A q) - b^T q \\ \Rightarrow q^T A (x^k + d) &= b^T q \quad \forall q \in \mathcal{R}^k. \end{aligned}$$

How do we make use of the formulation (2.14) to compute the new correction direction d ? Since the new correction direction $d \in \mathcal{R}^k$, we can express it as a linear combination of linearly independent vectors $d_i \in \mathcal{R}^k$, that span the space $\mathcal{R}^k$, as follows:

$$\begin{aligned} d &= \sum_{i=1}^{l} \alpha_i d_i, \quad \text{where } \alpha_i \in \mathbb{R}, \\ \mathcal{R}^k &= span\{d_l, d_{l-1}, \ldots, d_2, d_1\}. \end{aligned}$$

Remark 1. From now on, we will refer to the vector d as well as any of the vector d_i as a *correction direction*.

Equation (2.13) implies:

$$x^{k+1} = x^k + \sum_{i=1}^{l} \alpha_i d_i. \tag{2.15}$$

If we assume that, the vectors d_i are *pairwise A-orthonormal*, that is:

$$d_i{}^T A d_j = \begin{cases} 0 & \text{if } i \neq j \\ 1 & \text{if } i = j, \end{cases}$$

or, are can be made so, then we can easily find the coefficients α_i using the formulation (2.14) and correction equation (2.15) as follows:

$$\begin{aligned} a(x^k + d, q) &= b^T q \quad \forall q \in \mathcal{R}^k \\ \Rightarrow \left(x^k + \sum_{i=1}^{l} \alpha_i d_i \right)^T A q &= b^T q. \end{aligned}$$

Substituting $q = d_j$ $(1 \leq j \leq l)$ in the above equation, implies:

$$\left(x^k + \sum_{i=1}^{l} \alpha_i d_i \right)^T A d_j = b^T d_j.$$

Because we assumed that the vectors d_i are pairwise A-orthonormal, this implies:

$$\begin{aligned} \left(x^k\right)^T A d_j + \alpha_j &= b^T d_j \\ \Rightarrow \alpha_j &= \left(b^T - \left(x^k\right)^T A \right) d_j \\ \Rightarrow \alpha_j &= d_j^T r^k. \end{aligned} \tag{2.16}$$

Thus the set of A-orthonormalised vectors d_i give us a clear cut approach to calculate the correction (2.15) and this means we need some procedure to construct the set of vectors d_i that are pairwise A-orthonormal. We will achieve this with the *classical Gram-Schmidt* (CGS) *method*. To construct the vectors d_i using the CGS method, we will make use of the vectors w_i $(i = 1, \ldots, l)$. This means:

▶ for $i = 1, \ldots, l$, let:

$$\hat{d}_i = w_i - \sum_{j=1}^{i-1} \left({d_j}^T A w_i \right) d_j, \tag{2.17}$$

$$d_i = \frac{\hat{d}_i}{\| \hat{d}_i \|_A}.$$

For all its advantages, the process of A-orthonormalisation is not very attractive in terms of computational complexity, that is $O(nl)$, and in terms of storage requirement, that is $O(nl)$. *The vectors d_i, just like the vectors w_i, still belong to* $\mathbb{R}^n$. We want an algorithm whose computational complexity depends only on the number of unknowns as $O(n)$ (or equivalently $O(n_l)$). One possible way out, is to somehow perform the computations on the right hand side of equation (2.17) happen on the coarser grids and then interpolate the coarse grid results to the finest grid. This will drastically reduce both the computational complexity and storage requirement, as we will see later, to $O(n_l)$.
Before we attempt to realise the above statement, we need *two sets of coarse grid residual vectors*. The first one is defined as:

$$r_i = I_l^i r^k, \quad i = l, \ldots, 1 \, .$$

That is, we just restrict the finest grid residual vector r^k to the coarser grids. We define the next set of residual vectors as:

$$\bar{r}_i = M_i^{-1} I_l^i r^k, \quad i = l, \ldots, 1 \, .$$

That is we restrict the finest grid residual vector r_k to the coarser grids and then smoothen the restricted vectors using the matrices M_i^{-1}. If we choose M_i to be the *identity matrices*, then we call the vectors $\bar{r}_i$, the *rough residual vectors*. If we choose M_i to be *finite element mass matrices*, then we call the vectors $\bar{r}_i$ the *smooth residual vectors*. To complete the picture, we observe the following relationship between the vectors w_i, r^k, r_i, and $\bar{r}_i$ (see equation (2.11)):

$$w_i \;=\; I_i^l \, M_i^{-1} \, I_l^i \, r^k \;=\; I_i^l \, M_i^{-1} \, r_i \;=\; I_i^l \bar{r}_i.$$

Before we put these sets of vectors in use, i would like to point out that the vectors

$w_i \in \mathbb{R}^n$ and the vectors $r_i,\ \bar{r}_i \in \Omega_i$, where Ω_i is the coarse grid corresponding to the level i.

Let us now look at the right hand side of the computation using the CGS method (2.17). We then observe that the vectors d_j and w_i are involved in it. If we want to perform these computations on coarser grids, then, we need to have coarse grid versions d_i^{co} and w_i^{co} $(i = l, \ldots, 1)$ for each fine grid vector d_i and w_i respectively, and the following relationship should exist between those coarse and fine grid versions of these vectors:

$$d_i = I_i^l d_i^{co}, \quad and \quad w_i = I_i^l w_i^{co}, \quad \forall i < l, \tag{2.18}$$

where $d_i, w_i \in \Omega_l$ are fine grid vectors, and $d_i^{co}, w_i^{co} \in \Omega_i$ are coarse grid vectors.

Let us draw the ideas together before we proceed further. Our original correction space was $\mathcal{R}^k$. We then wanted to build the set of A-Orthonormal vectors d_i that span the space $\mathcal{R}^k$, because this makes the correction straight forward. To achieve this, we used the CGS method (2.17). We can see in equation (2.17) that, the term w_i, by its definition (2.11), already satisfies the condition (2.18). What remains are the vectors d_i. Are there coarse grid versions of the vectors d_i? If they do exist, do they satisfy the condition (2.18)? A lemma from Pflaum [8] guarantees this, as follows:

Lemma 2.1: *There is a vector $d_i^{co} \in \mathbb{R}^{n_i}$ such that:*

$$d_i = I_i^l d_i^{co},\ i = 1, \ldots, l. \tag{2.19}$$

Furthermore, the following equations hold:

$$\begin{aligned} d_1^{co} &= \frac{\bar{r}_1}{\| \bar{r}_1 \|_A}, \\ \hat{d}_i^{co} &= \bar{r}_i - \sum_{j=1}^{i-1} I_j^i \left\{ \underbrace{\left[\left(d_j^{co}\right)^T I_i^j A_i \bar{r}_i \right]}_{a\ scalar\ value} d_j^{co} \right\}, \\ d_i^{co} &= \frac{\hat{d}_i^{co}}{\| \hat{d}_i^{co} \|_A}. \end{aligned} \tag{2.20}$$

Proof. We will prove Lemma 2.1 by mathematical induction. Let us begin with the vector

$d_1 \in \Omega_1$. From equation (2.17), we observe that,

$$\hat{d}_1 = w_1 \;\Rightarrow\; \hat{d}_1 = I_i^l \bar{r}_1.$$

Thus the coarse grid version of d_1 that also satisfies condition (2.18) is:

$$\frac{\bar{r}_1}{\| \bar{r}_1 \|_A}.$$

Let us now assume that equation (2.20) holds for all levels i such that $i < l$. Then for level $i+1$, from equation (2.17) and (2.19), we have:

$$\begin{aligned} \hat{d}_{i+1} &= w_{i+1} - \sum_{j=1}^{i} \left({d_j}^T A w_{i+1} \right) d_j \\ \Rightarrow \hat{d}_{i+1} &= I_{i+1}^l \bar{r}_{i+1} - \sum_{j=1}^{i} \left\{ \left(I_j^l d_j^{co} \right)^T A \left(I_{i+1}^l \bar{r}_{i+1} \right) \left(I_j^l d_j^{co} \right) \right\}. \end{aligned}$$

Since $\left(I_j^l\right)^T = I_l^j$, we have:

$$\hat{d}_{i+1} = I_{i+1}^l \bar{r}_{i+1} - \sum_{j=1}^{i} \left\{ \left(d_j^{co} \right)^T I_l^j A \left(I_{i+1}^l \bar{r}_{i+1} \right) \left(I_j^l d_j^{co} \right) \right\}.$$

Since $I_l^j = I_{i+1}^j I_l^{i+1}$, we have:

$$\hat{d}_{i+1} = I_{i+1}^l \bar{r}_{i+1} - \sum_{j=1}^{i} \left\{ \left(d_j^{co} \right)^T I_{i+1}^j I_l^{i+1} A \left(I_{i+1}^l \bar{r}_{i+1} \right) \left(I_j^l d_j^{co} \right) \right\},$$

and $I_l^{i+1} A I_{i+1}^l = A_{i+1}$ implies:

$$\hat{d}_{i+1} = I_{i+1}^l \bar{r}_{i+1} - \sum_{j=1}^{i} \left\{ \underbrace{\left(d_j^{co} \right)^T I_{i+1}^j \, A_{i+1} \, \bar{r}_{i+1}}_{a \;\; scalar \;\; value} \left(I_j^l d_j^{co} \right) \right\}.$$

All we need to complete the proof is to write $I_j^l = I_{i+1}^l I_j^{i+1}$. This implies:

$$\hat{d}_{i+1} = I_{i+1}^{l}\bar{r}_{i+1} - \sum_{j=1}^{i} I_{i+1}^{l}\left\{\left(d_j^{co}\right)^T I_{i+1}^{j}\, A_{i+1}\,\bar{r}_{i+1}\left(I_j^{i+1} d_j^{co}\right)\right\}$$

$$\Rightarrow \hat{d}_{i+1} = I_{i+1}^{l}\left\{\bar{r}_{i+1} - \sum_{j=1}^{i}\left(d_j^{co}\right)^T I_{i+1}^{j}\, A_{i+1}\,\bar{r}_{i+1}\left(I_j^{i+1} d_j^{co}\right)\right\}.$$

□

Let us call the vectors d_i^{co} $(i = 1 \ldots, l)$ the *level correction directions*. We call them so, because, to make the correction in equation (2.15), all we need are the vectors $d_i^{co} \in \Omega_i$ and there is one for every level i. That is we make corrections at the coarser levels, using the *level correction directions*, that we calculate using construction (2.20) and interpolate the coarse grid corrections to the finer levels recursively, until we reach the finest level. Mathematically, this is expressed using the following Lemma [8].

Lemma 2.2: *The correction of MGgM can be expresses as follows:*

$$\begin{aligned} \alpha_i &= (d_i^{co})^T r_i \text{ and} \\ x^{k+1} - x^k &= \sum_{i=1}^{l} \alpha_i d_i \\ &= \alpha_l d_l^{co} + I_{l-1}^{l}\left(\alpha_{l-1} d_{l-1}^{co} + I_{l-2}^{l-1}\left(\cdots + I_1^2(\alpha_1 d_1^{co})\right)\right). \end{aligned} \tag{2.21}$$

The coefficients α_i can be easily calculated from the equation (2.16). That is:

$$\alpha_i \;=\; (d_i)^T r^k \;=\; (I_i^l d_i^{co})^T r^k \;=\; (d_i^{co})^T I_l^i r^k \;=\; (d_i^{co})^T r_i.$$

We see from the correction equation (2.21) that:

> *the correction corresponding to level* i $= \alpha_i d_i^{co}$ + *Interpolation to level* i *of correction corresponding to level* $(i-1)$ $(i = 2, \ldots, l)$.

Results of Lemmas 2.1 and 2.2 are exactly what we wanted to formulate the MGgM efficiently; that is efficiency with respect to computational complexity and storage requirements. Before I present the algorithm for the MGgM in Algorithm 1, we will from now on assume that, in the algorithms to follow, the smoothing step with the matrices M_i is achieved using standard smoothers such as the Gauss-Seidel method.

Algorithm 1 Multigrid Gradient Algorithm (MGgM)

1. Calculate residuals:

$r_l := b - Ax^k$

$d_l := M_l^{-1} r_l$

for $i = l$ *to* 2 **do**

$r_{i-1} := I_i^{i-1} r_i$

$d_{i-1} := M_{i-1}^{-1} r_{i-1}$

end for

2. Calculate level correction directions:

for $i = 1$ *to* l **do**

$k_i := A_i d_i$

for $j = i$ *to* 2 **do**

$k_{j-1} := I_j^{j-1} k_j$

end for

$s_1 = 0$

for $j = 1$ *to* $(i-1)$ **do**

$s_j := s_j + d_j.(k_j^T d_j)$

$s_{j+1} := I_j^{j+1} s_j$

end for

$d_i := d_i - s_i$

3. A-Normalise level correction direction:

$k_i := A_i d_i$

$d_i := d_i.\left(k_i^T d_i\right)^{-1/2}$

end for

4. Calculate correction:

$s_1 := (d_1^T r_1).d_1$

for $i = 2$ *to* l **do**

$s_i := I_{i-1}^i s_{i-1}$

$s_i := s_i + ({d_i}^T r_i).d_i$

end for

$x^{k+1} := x^k + s_l$ ■

2.2.1 Variants of the MGgM

The major element of the MGgM is the *multilevel residuum space* $\mathcal{R}^k$ and that, is our correction space. We already noted that $\mathcal{R}^k$ can be a *rough multilevel residuum space* $\mathcal{R}^k_{rough}$ or a *smooth multilevel residuum space* $\mathcal{R}^k_{smooth}$, depending on the choice of the smoothing matrix M_i (see equation (2.11)). Based on this, using equation (2.20), we now have three possibilities to construct the set of correction directions d_i, or equivalently, the set of level correction directions d_i^{co}. They are:

1. construct the level correction directions d_i^{co} using the space $\mathcal{R}^k_{rough}$. This means, we set in equation (2.17):

$$w_i = I_i^l I_l^i \, r^k = I_i^l \bar{r}_i,$$

   and then construct d_i^{co} using equation (2.20). Let us call these d_i^{co}, the *rough level correction directions* and the corresponding fine grid vectors d_i, the *rough correction directions*.

2. construct the level correction directions d_i^{co} from the space $\mathcal{R}^k_{smooth}$.This means, we set in equation (2.17):

$$w_i = I_i^l \, M_i^{-1} \, I_l^i \, r^k = I_i^l \, M_i^{-1} \bar{r}_i,$$

   and then construct d_i^{co} using equation (2.20). Let us call these d_i^{co}, the *smooth level correction directions* and the corresponding fine grid vectors d_i, the *smooth correction directions*.

3. construct the level correction directions d_i^{co} from the space $\mathcal{R}^k_{smooth} \oplus \mathcal{R}^k_{rough}$. We will see in Section 2.2.2 how we achieve this. For the present, let us call these vectors d_i^{co}, the *rough and smooth level correction directions* and the corresponding fine grid vectors d_i, the *smooth and rough correction directions*.

Each of the three choices above respectively, lead to a variation of the MGgM. They are:

MGgM-1 Multigrid gradient method with rough correction directions.

MGgM-2 Multigrid gradient method with smooth correction directions.

MGgM-3 Multigrid gradient method with rough and smooth correction directions.

Algorithmically, the MGgM-1 and MGgM-2 are exactly the same, as described in Algorithm 1, except for the fact that, for the MGgM-1, we choose in Algorithm 1, M_i as the identity matrices and for the MGgM-2, we choose in Algorithm 1, M_i as the finite element mass matrices. The correction step (2.21) for both MGgM-1 and MGgM-2 is the same as described in Algorithm 1.

We now have three choices for the level correction directions. What choice of d_i^{co} is the best? We observe from equation (2.21) that the level correction directions $d_i^{co} (\in \Omega_i)$ determine the optimality of the correction. The more optimal the level correction directions are, the better is the correction. We will later see in the numerical results in chapter 4, what exactly is meant by optimality of the level correction directions.

The *multigrid method with rough and smooth correction directions* (MGgM-3) requires particular attention. Unlike the MGgM-1 and MGgM-2, we need to take special care in constructing the *rough and smooth level correction directions* using equation (2.20). Also the correction step (2.21) requires a small modification. We will discuss the MGgM-3 in detail now.

2.2.2 MGgM with Rough and Smooth Correction Directions (MGgM-3)

In MGgM-3, the correction space has become bigger; we now have twice more vectors to construct the correction space. That is, the correction space is $\mathcal{R}^k_{smooth} \oplus \mathcal{R}^k_{rough}$, and is defined as:

$$\breve{\mathcal{R}}^k \;:=\; \mathcal{R}^k_{smooth} \oplus \mathcal{R}^k_{rough} \;:=\; span\{w_l^r, w_l^s, w_{l-1}^r, w_{l-1}^s, \ldots, w_1^r, w_1^s\}, \tag{2.22}$$

where:

$$\begin{aligned}
w_l^r &= r^k, \\
w_l^s &= M_l^{-1} w_l^r, \\
w_i^r &= I_i^l I_l^i \, r^k, \quad i = (l-1), \ldots, 1 \qquad (2.23)\\
w_i^s &= I_i^l \, M_i^{-1} \, I_l^i \, r^k, \quad i = (l-1), \ldots, 1 \, . \qquad (2.24)
\end{aligned}$$

Similar to the formulation (2.14), we can now formulate the MGgM-3 as:

▶ find $d \in \breve{\mathcal{R}}^k$ such that:

$$\begin{aligned} a(x^k + d, q) &= b^T q \quad \forall q \in \breve{\mathcal{R}}^k, \\ x^{k+1} &= x^k + d. \end{aligned} \tag{2.25}$$

Since the new correction direction $d \in \breve{\mathcal{R}}^k$, we can express d as a linear combination of linearly independent vectors $d_i \in \breve{\mathcal{R}}^k$, that span the space $\breve{\mathcal{R}}^k$, as follows:

$$d = \sum_{i=1}^{2l} \gamma_i d_i, \quad \text{where } \gamma_i \in \mathbb{R}. \tag{2.26}$$

Let us now use the following equivalent notations for the vectors d_i:

$$\begin{aligned} d_i^r &= d_{2i-1}, \ i = 1, \ldots, l, \\ d_i^s &= d_{2i}, \ i = 1, \ldots, l, \end{aligned}$$

where the vectors d_i^r are the *rough correction directions*, and the vectors d_i^s are the *smooth correction directions*. That is we rename the vectors d_i as follows:

$$\begin{gathered} (d_1, d_2, d_3, d_4, \ldots, d_{2l-1}, d_{2l}) \\ \Downarrow \\ (d_1^r, d_1^s, d_2^r, d_2^s, \ldots, d_l^r, d_l^s\ ), \end{gathered}$$

which shows that, corresponding to every level we have two correction directions; one rough and the other smooth. Then from equations (2.25) and (2.26), we obtain the correction as:

$$x^{k+1} = x^k + \sum_{i=1}^{l} \alpha_i d_i^r + \beta_i d_i^s, \quad \text{where } \alpha_i, \beta_i \in \mathbb{R}. \tag{2.27}$$

If the vectors d_i are pairwise A-orthonormal, or, are can be made so, then similar to equation (2.16), we can estimate the coefficients α_i and β_i as:

$$\alpha_i = (d_i^r)^T r^k, \tag{2.28}$$

$$\beta_i = (d_i^s)^T r^k. \tag{2.29}$$

Similar to equation (2.17), let us construct the set of pairwise A-orthonormal vectors d_i, or equivalently the vectors d_i^r and d_i^s, using the vectors w_i^r and w_i^s. This means:

▶ for $i = 1, \ldots, l$, let:

1. Build $(d_i^r)^k$ using w_i^r, such that $(d_i^r)^k$ is made A-orthonormal to:
   (a) $(d_j^r)^k$, $j = 1, \ldots, (i-1)$.
   (b) $(d_j^s)^k$, $j = 1, \ldots, (i-1)$.
2. Build $(d_i^s)^k$ using w_i^s, such that $(d_i^s)^k$ is made A-orthonormal to:
   (a) $(d_j^r)^k$, $j = 1, \ldots, i$.
   (b) $(d_j^s)^k$, $j = 1, \ldots, (i-1)$.

We will accomplish steps $1-2$ described above, using the CGS method, as follows:

$$\hat{d}_i^r = w_i^r - \left\{ \sum_{j=1}^{i-1} \left[\left(d_j^r\right)^T A w_i^r \right] d_j^r + \sum_{j=1}^{i-1} \left[\left(d_j^s\right)^T A w_i^r \right] d_j^s \right\}, \tag{2.30}$$

$$\hat{d}_i^s = w_i^s - \left\{ \sum_{j=1}^{i} \left[\left(d_j^r\right)^T A w_i^s \right] d_j^r + \sum_{j=1}^{i-1} \left[\left(d_j^s\right)^T A w_i^s \right] d_j^s \right\}, \tag{2.31}$$

$$d_i^r = \frac{\hat{d}_i^r}{\| \hat{d}_i^r \|_A},$$

$$d_i^s = \frac{\hat{d}_i^s}{\| \hat{d}_i^s \|_A}.$$

We can see from equations (2.30) and (2.31), that in comparison to using only the space $\mathcal{R}^k_{rough}$ or $\mathcal{R}^k_{smooth}$, we have to do twice the work. We saw that we can reduce the computational complexity and storage requirement to $O(n_l)$, by finding suitable coarse grid *level correction directions*. In MGgM-3, these are the *rough and smooth level correction directions*. Before we attempt to figure them out, let us define the following sequence of coarse grid residual vectors:

$$r_i^r = I_l^i r^k, \quad i = l, \ldots, 1\,. \tag{2.32}$$

$$r_i^s = M_i^{-1} I_l^i r^k, \quad i = l, \ldots, 1\,. \tag{2.33}$$

The vectors r_i^r and r_i^s are the rough and smooth residuals respectively, that we spoke of before. From equations (2.23), (2.24) and (2.32), (2.33), we can observe the following relationship between the vectors w_i^r and r_i^r, and between the vectors w_i^s and r_i^s respectively:

$$\begin{aligned} w_i^r &= I_i^l r_i^r, \\ w_i^s &= I_i^l r_i^s. \end{aligned}$$

It is now straightforward to calculate the set of *rough and smooth level correction directions* by applying the result of Lemma 2.1 to each of equations (2.30) and (2.31). We will achieve this with a modified version of Lemma 2.1 as follows:

Lemma 2.3: *There are vectors $d_i^{rco}, d_i^{sco} \in \mathbb{R}^{n_i}$ such that:*

$$\begin{aligned} d_i^r &= I_i^l d_i^{rco}, \ i = 1, \ldots, l, \\ d_i^s &= I_i^l d_i^{sco}, \ i = 1, \ldots, l. \end{aligned}$$

Furthermore, the following equations hold:

$$\begin{aligned} d_1^{rco} &= \frac{r_1^r}{\| r_1^r \|_A}, \\ d_1^{sco} &= \frac{r_1^s}{\| r_1^s \|_A}, \\ \hat{d}_i^{rco} &= r_i^r - \sum_{j=1}^{i-1} I_j^i \left\{ \left[\left(d_j^{rco} \right)^T I_i^j A_i r_i^r \right] d_j^{rco} \right\} + \sum_{j=1}^{i-1} I_j^i \left\{ \left[\left(d_j^{sco} \right)^T I_i^j A_i r_i^r \right] d_j^{sco} \right\}, \quad (2.34) \\ \hat{d}_i^{sco} &= r_i^s - \sum_{j=1}^{i} I_j^i \left\{ \left[\left(d_j^{rco} \right)^T I_i^j A_i r_i^s \right] d_j^{rco} \right\} + \sum_{j=1}^{i-1} I_j^i \left\{ \left[\left(d_j^{sco} \right)^T I_i^j A_i r_i^s \right] d_j^{sco} \right\}, \quad (2.35) \\ d_i^{rco} &= \frac{\hat{d}_i^{rco}}{\| \hat{d}_i^{rco} \|_A}, \\ d_i^{sco} &= \frac{\hat{d}_i^{sco}}{\| \hat{d}_i^{sco} \|_A}. \end{aligned}$$

The vectors d_i^{rco} and d_i^{sco} $(i = 1, \ldots, l)$ together are the *rough and smooth level correction*

directions respectively. For the correction (2.27), I would like to state a lemma that is similar to Lemma 2.2:

Lemma 2.4: *The correction of MGgM-3 can be expresses as follows:*

$$\begin{aligned} \alpha_i &= (d_i^{rco})^T r_i^r, \\ \beta_i &= (d_i^{sco})^T r_i^r, \\ x^{k+1} - x^k &= \sum_{i=1}^{l} \{\alpha_i d_i^r + \beta_i d_i^s\} \\ &= [\alpha_l d_l^{rco} + \beta_l d_l^{sco}] + I_{l-1}^l \left(\left[\alpha_{l-1} d_{l-1}^{rco} + \beta_{l-1} d_{l-1}^{sco}\right] \right. \\ &\quad \left. + I_{l-2}^{l-1} \left(\cdots + I_1^2 \left([\alpha_1 d_1^{rco} + \beta_1 d_1^{sco}] \right) \right) \right). \end{aligned} \tag{2.36}$$

The coefficients α_i can be easily calculated from equation (2.28) as:

$$\alpha_i \;=\; (d_i^r)^T r^k \;=\; (I_i^l d_i^{rco})^T r^k \;=\; (d_i^{rco})^T I_l^i r^k \;=\; (d_i^{rco})^T r_i^r.$$

Similarly, the coefficients β_i can be easily calculated from equation (2.29) as:

$$\beta_i \;=\; (d_i^s)^T r^k \;=\; (I_i^l d_i^{sco})^T r^k \;=\; (d_i^{sco})^T I_l^i r^k \;=\; (d_i^{sco})^T r_i^r.$$

The major difference in calculating the correction in the MGgM-1 or in the MGgM-2 (2.21), and in the MGgM-3 (2.36) is that, in the MGgM-3, we make at each coarse grid level, corrections corresponding to *two level correction directions*, that is *one smooth* (d_i^{sco}) and the other *rough* (d_i^{rco}). In contrast, in the MGgM-1 or in the MGgM-2, we make at each coarse grid level, corrections corresponding to only *one level correction direction*, that is either the *rough* or the *smooth level correction direction*. Apparently, it might appear that the correction in MGgM-3 is optimal than the correction in the MGgM-1 or in the MGgM-2, since we use a relatively bigger correction space $\breve{\mathcal{R}}^k$ (2.22) in MGgM-3. This is indeed true and we will observe this clearly in the numerical results in chapter 4.

To complete our discussion on the MGgM and its variants, we will look at a theorem from Pflaum [8], that guarantees us the convergence of the MGgM and its variants for the problem (2.1), with respect to the energy norm $\| \, . \, \|_A$.

Theorem 2.1: *Let $\hat{x}$ be the exact solution of the equation $Ax = b$. Let x^k and x^{k+1} be successive iterates generated by the MGgM or its variants. There is a constant $0 < \rho < 1$ independent of x^k and number of levels l such that:*

$$\| \hat{x} - x^{k+1} \|_A \leq \rho \| \hat{x} - x^k \|_A.$$

The results of Lemmas 2.3 and 2.4 lead us to the following algorithmic representation of the MGgM-3:

Algorithm 2 Multigrid Gradient Algorithm with Rough and Smooth Correction Directions (MGgM-3)

1. Calculate rough and smooth residuals:

$r_l^r := b - Ax^k$

$r_l^s := M_l^{-1} r_l$

$d_l^r := r_l^r$

$d_l^s := r_l^s$

for $i = l$ *to* 2 **do**

$r_{i-1}^r := I_i^{i-1} r_i^r$

$r_{i-1}^s := M_{i-1}^{-1} r_{i-1}^r$

$d_{i-1}^r := r_{i-1}^r$

$d_{i-1}^s := r_{i-1}^s$

end for

2. Calculate rough and smooth level correction directions:

for $i = 1$ *to* l **do**

$k_i := A_i d_i^r$

$f_i := A_i d_i^s$

for $j = i$ *to* 2 **do**

$k_{j-1} := I_j^{j-1} k_j$

$f_{j-1} := I_j^{j-1} f_j$

end for

$s_1 = 0$

$g_1 = 0$

→ continued

for $j = 1$ *to* $(i-1)$ **do**

$s_j := s_j + d_j^r.(k_j^T d_j^r) + d_j^s.(k_j^T d_j^s)$

$g_j := g_j + d_j^r.(f_j^T d_j^r) + d_j^s.(f_j^T d_j^s)$

$s_{j+1} := I_j^{j+1} s_j$

$g_{j+1} := I_j^{j+1} g_j$

end for

$d_i^r := d_i^r - s_i$

$d_i^s := d_i^s - g_i$

$k_i := A_i d_i^s$

$s_i := d_i^s - d_i^r.(k_i^T d_i^r)$

if $\| s_i \|_\infty > \epsilon \| d_i^s \|_\infty$ **then** `(observe this)`

$d_i^s := s_i$

else

$d_i^s := 0$

end if

3. A-Normalise rough and smooth level correction directions:

$k_i := A_i d_i^r$

$d_i^r := d_i^r.\left(k_i^T d_i^r\right)^{-1/2}$

if $\| d_i^s \|_\infty \neq 0$ **then**

$k_i := A_i d_i^s$

$d_i^s := d_i^s.\left(k_i^T d_i^s\right)^{-1/2}$

end if

end for

4. Calculate correction with rough and smooth level correction directions:

$s_1 := \left((d_1^r)^T r_1^r\right).d_1^r + \left((d_1^s)^T r_1^r\right).d_1^s$

for $i = 2$ *to* l **do**

$s_i := I_{i-1}^i s_{i-1}$

$s_i := s_i + \left((d_i^r)^T r_i^r\right).d_i^r + \left((d_i^s)^T r_i^r\right).d_i^s$

end for

$x^{k+1} := x^k + s_l$ ■

Remark 2. The commented `if` condition in Algorithm 2 is essential from the computational point of view. It avoids excessive round-off error that arises from computing the A-normalisation of those *smooth level correction directions* d_i^s, that have $\| d_i^s \|_\infty \approx$ *machine precision*. This arises especially for vectors d_i^s at the coarsest level and in such cases, without the `if` condition, I observed rapid divergence in the numerical implementation.

2.3 Multigrid Conjugate Gradient Method (MGgM)

It has been reported by Pflaum [8] and so shall we observe in the results of the numerical experiments in chapter 4, that, though the MGgM and its variants have an optimal asymptotic convergence rate with respect to the number of unknowns n, they display poor convergence behavior when it comes to problems with extreme parameters such as P.D.Es with large jumping coefficients. This can be attributed to the *optimality of the level correction directions* that I mentioned before.
The correction directions, constructed from the correction space of the MGgM (and its variants), are not optimal because, we could end up making corrections corresponding to successive iterations in the same correction direction(s). Just as we do it in the *classical conjugate gradient method*, we will make a correction in a particular correction direction only once and be done with it. If $\mathcal{D}^k$ is a correction space for iteration k, then this means, we need to construct the correction directions corresponding to the space $\mathcal{D}^k$, A-orthogonal (or A-orthonormal) to the correction directions corresponding to the space $\mathcal{D}^{k-1}$. That is:

$$\mathcal{D}^k \perp_A \mathcal{D}^{k-1} \Rightarrow faster\ convergence.$$

The following theorem by Pflaum [8] guarantees us that, successively A-orthogonal correction spaces result in a faster convergence rate compared to successive correction spaces that are not (or may not be) A-orthogonal, such as the *multilevel residuum spaces* $\mathcal{R}^k$ (2.10) in the MGgM and its variants.

Theorem 2.2: *Let $\hat{x} \in \mathbb{R}^n$ be the exact solution of the equation $Ax = b$. Let $W, V \subset \mathbb{R}^n$ be subspaces. Furthermore, let $\tilde{x}$ be an approximation of $\hat{x}$ such that:*

$$a(\tilde{x}, v) = b^T v \quad \forall v \in V. \tag{2.37}$$

Construct the subspace $W^{\perp}$ such that:

$$span(W, V) = span(W^{\perp}, V) \quad and \quad W^{\perp} \perp_A V. \tag{2.38}$$

Now, let $w_r \in W$ and $w_d \in W^{\perp}$ such that:

$$\begin{aligned} a(\tilde{x} + w_r, w) &= b^T w \quad \forall w \in W \\ a(\tilde{x} + w_d, w) &= b^T w \quad \forall w \in W^{\perp}. \end{aligned}$$

Then $(\tilde{x} + w_r)$ and $(\tilde{x} + w_d)$ are approximations of the exact solution $\hat{x}$, such that:

$$\| \hat{x} - (\tilde{x} + w_d) \|_A \leq \| \hat{x} - (\tilde{x} + w_r) \|_A. \tag{2.39}$$

As a consequence of Theorem 2.2, in the MGcgM, the correction space for iteration k will no more be the *multilevel residuum space* $\mathcal{R}^k$, as in the MGgM and its variants, but an l-dimensional *multilevel correction space* $\mathcal{D}^k$, defined as:

$$\mathcal{D}^k := span\{d_l, d_{l-1}, \ldots, d_2, d_1\}, \tag{2.40}$$

where $d_i \in \mathbb{R}^n,\ i = l, \ldots, 1$ and $\mathcal{D}^k \perp_A \mathcal{D}^{k-1}$.

Before we start constructing the *multilevel correction space* $\mathcal{D}^k$, let us formulate the MGcgM as follows:

▶ find $d \in \mathcal{D}^k$ such that:

$$a(x^k + d, q) = b^T q \quad \forall q \in \mathcal{D}^k, \tag{2.41}$$

$$x^{k+1} = x^k + d. \tag{2.42}$$

Let us now see how we can make use of Theorem 2.2 to construct the *multilevel correction space* $\mathcal{D}^k$ and hence the MGcgM. Assume that we are in iteration k of the MGcgM. Let us consider that the space V in Theorem 2.2 corresponds to the *multilevel correction space* $\mathcal{D}^{k-1}$. Then, according to equation (2.37), we have an approximate solution $\tilde{x} \in \mathcal{D}^{k-1}$ that satisfies the problem (2.41) for iteration $(k-1)$. We now have to construct a new approximation $(\tilde{x} + d)$, where d is the correction direction that is to be constructed from the correction space corresponding to iteration k. Given this, we have two possible

correction spaces for iteration k. One is the *multilevel residuum space* $\mathcal{R}^k$, that we discussed in Section 2.2 and this corresponds to the space W in Theorem 2.2. This choice leads us to the MGgM or one of its variants. Theorem 2.2 gives us another choice for the correction space, that is optimal and that is the space $W^{\perp}$ in Theorem 2.2. Let us consider that the space $W^{\perp}$ corresponds to the new *multilevel correction space* $\mathcal{D}^k$. We observe from construction (2.38) that, Theorem 2.2 asks us to construct the *multilevel correction space* $\mathcal{D}^k$ for iteration k in such a way that:

$$span(\mathcal{R}^k, \mathcal{D}^{k-1}) = span(\mathcal{D}^k, \mathcal{D}^{k-1}) \quad and \quad \mathcal{D}^k \perp_A \mathcal{D}^{k-1}. \tag{2.43}$$

Then, according to the inequality (2.39), the correction direction constructed from the space $\mathcal{D}^k$ (which is w_d in inequality (2.39)) results in a faster convergence rate compared to one constructed from the space $\mathcal{R}^k$ (which is w_r in inequality (2.39)).

Therefore, according to Theorem 2.2, a natural approach to construct the space $\mathcal{D}^k$, A-orthogonal to the space $\mathcal{D}^{k-1}$, is to make use of the space $\mathcal{R}^k$ as follows:

▶ for $i = 1, \ldots, l$:

$$\begin{aligned} \hat{d}_i^k &= \underbrace{w_i - \sum_{j=1}^{i-1} \left\{ \left(d_j^k\right)^T A w_i \right\} d_j^k}_{term\ 1} - \underbrace{\sum_{j=1}^{l} \left\{ \left(d_j^{k-1}\right)^T A w_i \right\} d_j^{k-1}}_{term\ 2}, \\ d_i^k &= \frac{\hat{d}_i^k}{\| \hat{d}_i^k \|_A}, \end{aligned} \tag{2.44}$$

where $d_i^k \in \mathcal{D}^k$, $w_i \in \mathcal{R}^k$, and $d_i^{k-1} \in \mathcal{D}^{k-1}$.

We see that the construction (2.44) leads to the space $\mathcal{D}^k$ that satisfies condition (2.43). But the construction (2.44) results in a computational complexity of $O(nl)$. To reduce the computational complexity to $O(n_l)$, we will again perform the computations on the right hand side of construction (2.44) happen at coarser grids and interpolate the coarse grid results to the finest grid. Before we do that, if we observe the right hand side of construction (2.44), we see that the *term 1* can be computed on the coarser grids using a formula similar to (2.20) that we saw in Lemma 2.1. The problem now is with the *term 2*, because we cannot apply Lemma 2.1 to it. We overcome this by replacing

condition (2.43) by the following modified condition from Pflaum [8]:

$$\begin{aligned}
\mathcal{D}^1 &= \mathcal{R}^1, \\
\mathcal{D}^k &:= span\{d_l^k, d_{l-1}^k, \ldots, d_1^k\}, \\
\| d_i^k \|_A &= 1, & i &= 1, \ldots, l, \\
d_i^k &\in I_i^l(\mathbb{R}^{n_i}), & i &= 1, \ldots, l, \\
d_i^k &\perp_A d_j^k, & i &\neq j, \\
d_i^{k-1} &\perp_A d_j^k, & i &\leq j,\ k > 1, \\
span(\mathcal{R}^k, \mathcal{D}^{k-1}) &= span(\mathcal{D}^k, \mathcal{D}^{k-1}).
\end{aligned} \tag{2.45}$$

That is, *we do not make the spaces $\mathcal{D}^k$ and $\mathcal{D}^{k-1}$ exactly A-orthogonal, rather make them partially A-orthogonal.* Then, the new correction space $\mathcal{D}^k$ can now be constructed as follows [8]:

▶ for $i = 1, \ldots, l$:

$$\tilde{d}_i^{k-1} = d_i^{k-1} - \sum_{j=1}^{i-1} \left\{ \left(d_i^k\right)^T A d_i^{k-1} \right\} d_i^k \tag{2.46}$$

$$\hat{d}_i^{k-1} = \frac{\tilde{d}_i^{k-1}}{\| \tilde{d}_i^{k-1} \|_A},$$

$$\hat{d}_i^k = w_i - \sum_{j=1}^{i-1} \left\{ \left(d_j^k\right)^T A w_i \right\} d_j^k - \sum_{j=1}^{i} \left\{ \left(\hat{d}_j^{k-1}\right)^T A w_i \right\} \hat{d}_j^{k-1}, \tag{2.47}$$

$$d_i^k = \frac{\hat{d}_i^k}{\| \hat{d}_i^k \|_A}.$$

Constructions (2.46) and (2.47) give us the space $\mathcal{D}^k$ that satisfies the modified condition (2.45).

Remark 3. With the modified condition (2.45), it might be tempting to construct thee new correction space $\mathcal{D}^k$ using a construction similar to (2.44), instead of using the constructions (2.46) and (2.47) mentioned above, as follows:

▶ for $i = 1, \ldots, l$:

$$\begin{aligned} \hat{d}_i^k &= w_i - \sum_{j=1}^{i-1} \left\{ \left(d_j^k\right)^T A w_i \right\} d_j^k - \sum_{j=1}^{i} \left\{ \left(d_j^{k-1}\right)^T A w_i \right\} d_j^{k-1}, \\ d_i^k &= \frac{\hat{d}_i^k}{\| \hat{d}_i^k \|_A}. \end{aligned}$$

But this means that:

$$span(\mathcal{R}^k, \mathcal{D}^{k-1}) \neq span(\mathcal{D}^k, \mathcal{D}^{k-1}),$$

and the modified condition (2.45) is not satisfied.

After we construct the correction space, we have to make the correction (2.42). Though the correction space changes (or becomes bigger) in the MGcgM, its correction step remains the same as in the MGgM (2.21).

We also observe that we can perform the constructions (2.46) and (2.47) on coarser grids, using formulas similar to (2.20), that we saw in Lemma 2.1. This leads us to the following algorithmic representation of the MGcgM:

Algorithm 3 Multigrid Conjugate Gradient Algorithm (MGcgM)

---

if $k = 0$ **then**

Perform one step of the Multigrid Gradient Algorithm (Algorithm 1)..

else

Proceed as follows.

end if

1. Calculate residuals:

$r_l := b - Ax^k$

$d_l^{new} := M_l^{-1} r_l$

for $i = l$ *to* 2 **do**

$r_{i-1} := I_i^{i-1} r_i$

$d_{i-1}^{new} := M_{i-1}^{-1} r_{i-1}$

end for → continued

---

2. Calculate new level correction directions:

for $i = 1$ *to* l **do**

3. Old direction A-orthonormal to new coarser directions:

$k_i := A_i d_i$

for $j = i$ *to* 2 **do**

$k_{j-1} := I_j^{j-1} k_j$

end for

$s_1 = 0$

for $j = 1$ *to* $(i-1)$ **do**

$s_j := s_j + d_j^{new}.(k_j^T d_j^{new})$

$s_{j+1} := I_j^{j+1} s_j$

end for

$d_i := d_i - s_i$

4. A-Normalise old modified level correction direction:

$k_i := A_i d_i$

$d_i := d_i.\left(k_i^T d_i\right)^{-1/2}$

5. New direction A-orthonormal to new and old coarser directions:

$k_i := A_i d_i^{new}$

for $j = i$ *to* 2 **do**

$k_{j-1} := I_j^{j-1} k_j$

end for

$s_1 = 0$

for $j = 1$ *to* $(i-1)$ **do**

$s_j := s_j + d_j^{new}.(k_j^T d_j^{new})$

$s_j := s_j + d_j.(k_j^T d_j)$

$s_{j+1} := I_j^{j+1} s_j$

end for

$d_i^{new} := d_i^{new} - s_i$ → `continued`

6. New direction A-orthonormal to old modified direction:

$k_i := A_i d_i^{new}$

$s_i := d_i^{new} - d_i.(k_i^T d_i)$

if $\| s_i \|_\infty > \epsilon \| d_i^{new} \|_\infty$ **then** `(observe this)`

$d_i^{new} := s_i$

else

$d_i^{new} := d_i$

$d_i := 0$

end if

7. A-Normalise new level correction direction:

$k_i := A_i d_i^{new}$

$d_i := d_i^{new}.\left(k_i^T d_i^{new}\right)^{-1/2}$

end for

8. Calculate correction:

$s_1 := (d_1^T r_1).d_1$

for $i = 2$ *to* l **do**

$s_i := I_{i-1}^i s_{i-1}$

$s_i := s_i + ({d_i}^T r_i).d_i$

end for

$x^{k+1} := x^k + s_l$ ■

Remark 4. The commented `if` condition in Algorithm 3 is essential from the computational point of view. If the coarse grid consists of a very small number of grid points, then it is possible that the vector d_i^{new} becomes the *null vector*. The `if` condition avoids this possibility.

2.3.1 Variants of the MGcgM

We discussed in Section 2.2.1, three variants of the MGgM depending on the choice of the correction space. The correction space there was the *multilevel residuum space* $\mathcal{R}^k$. Similarly, we can derive three variants of the MGcgM depending on the choice of the correction space. But now our correction space is not $\mathcal{R}^k$, but the *multilevel*

correction space $\mathcal{D}^k$ (2.40). Since we constructed the space $\mathcal{D}^k$, using the space $\mathcal{R}^k$, as in equation (2.47), we have three different possibilities for constructing the space $\mathcal{D}^k$ (or equivalently the level correction directions), depending on how we construct the space $\mathcal{R}^k$. They are:

1. $\mathcal{D}^k_{rough} \rightarrow$ constructed using $\mathcal{R}^k_{rough}$.
2. $\mathcal{D}^k_{smooth} \rightarrow$ constructed using $\mathcal{R}^k_{smooth}$.
3. $\mathcal{D}^k_{rough} \oplus \mathcal{D}^k_{smooth} \rightarrow$ constructed using $\mathcal{R}^k_{rough} \oplus \mathcal{R}^k_{smooth}$.

Each of the three choices above for the *multilevel correction space* $\mathcal{D}^k$ respectively, lead us to a variation of the MGcgM. They are:

MGcgM-1 Multigrid conjugate gradient method with rough correction directions.

MGcgM-2 Multigrid conjugate gradient method with smooth correction directions.

MGcgM-3 Multigrid conjugate gradient method with rough and smooth correction directions.

Similar to what we discussed before about the variants of the MGgM, algorithmically, the MGcgM-1 and MGcgM-2 are exactly the same, as described in Algorithm 3, except for the fact that, for the MGcgM-1, we choose in Algorithm 3, M_i as the identity matrices and for the MGcgM-2, we choose in Algorithm 3, M_i as the finite element mass matrices. The correction step (2.21) for both MGcgM-1 and MGcgM-2 is the same as described in Algorithm 3.

Just like the MGgM-3, the MGcgM-3 also requires a different approach to construct the *multilevel correction space* $\mathcal{D}^k$ and the correction step itself. We will discuss the MGcgM-3 in detail now.

2.3.2 MGcgM with Smooth and Rough Correction Directions (MGcgM-3)

We again have a bigger correction space in the MGcgM-3. The correction space now is the $2l$-dimensional *multilevel correction space* $\mathcal{D}^k_{rough} \oplus \mathcal{D}^k_{smooth}$, and is defined as:

$$\breve{\mathcal{D}}^k \;:=\; \mathcal{D}^k_{smooth} \oplus \mathcal{D}^k_{rough} \;:=\; span\{d^r_l, d^s_l, d^r_{l-1}, d^s_{l-1}, \ldots, d^r_1, d^s_1\},$$

where:

1. the vectors d_i^r $(i = 1, \ldots, l)$, are constructed using the space $\mathcal{R}_{rough}^k$ (that is using the vectors w_i^r described in (2.23)), and using equations (2.46) and (2.47). The vectors d_i^r are the *rough correction directions*.

2. the vectors d_i^s $(i = 1, \ldots, l)$, are constructed using the space $\mathcal{R}_{smooth}^k$ (that is using the vectors w_i^s described in (2.24)), and using equations (2.46) and (2.47). The vectors d_i^s are the *smooth correction directions*.

Similar to the formulation (2.14), we can now formulate the MGcgM-3 as:

▶ find $d \in \breve{\mathcal{D}}^k$ such that:

$$\begin{aligned} a(x^k + d, q) &= b^T q \quad \forall q \in \breve{\mathcal{D}}^k, \\ x^{k+1} &= x^k + d. \end{aligned}$$

Similar to the correction for the MGgM-3 (2.27), we obtain the correction for the MGcgM-3 as:

$$x^{k+1} - x^k = \sum_{i=1}^{l} \alpha_i d_i^r + \beta_i d_i^s, \quad \text{where } \alpha_i, \beta_i \in \mathbb{R},$$

where:

$$\begin{aligned} \alpha_i &= (d_i^r)^T r^k, \\ \beta_i &= (d_i^s)^T r^k. \end{aligned}$$

Let us now construct the correction space $\breve{\mathcal{D}}^k$, using the space $\mathcal{R}_{rough}^k \oplus \mathcal{R}_{smooth}^k$, and the modified condition (2.45), as follows:

▶ for $i = 1, \ldots, l$:

1. Construct $(\hat{d}_i^r)^{k-1}$ using $(d_i^r)^{k-1}$, such that $(\hat{d}_i^r)^{k-1}$ is made A-orthonormal to:

   (a) $(d_j^r)^k$, $j = 1, \ldots, (i-1)$.

   (b) $(d_j^s)^k$, $j = 1, \ldots, (i-1)$.

2. Construct $(\hat{d}_i^s)^{k-1}$ using $(d_i^s)^{k-1}$, such that $(\hat{d}_i^s)^{k-1}$ is made A-orthonormal to:

   (a) $(d_j^r)^k$, $j = 1, \ldots, i$.

   (b) $(d_j^s)^k$, $j = 1, \ldots, (i-1)$.

3. Construct $(d_i^r)^k$ using w_i^r, such that $(d_i^r)^k$ is made A-orthonormal to:

   (a) $(d_j^r)^k$, $j = 1, \ldots, (i-1)$.

   (b) $(d_j^s)^k$, $j = 1, \ldots, (i-1)$.

   (c) $(\hat{d}_j^r)^{k-1}$, $j = 1, \ldots, i$.

   (d) $(\hat{d}_j^s)^{k-1}$, $j = 1, \ldots, (i-1)$.

4. Construct $(d_i^s)^k$ using w_i^s, such that $(d_i^s)^k$ is made A-orthonormal to:

   (a) $(d_j^r)^k$, $j = 1, \ldots, i$.

   (b) $(d_j^s)^k$, $j = 1, \ldots, (i-1)$.

   (c) $(\hat{d}_j^r)^{k-1}$, $j = 1, \ldots, i$.

   (d) $(\hat{d}_j^s)^{k-1}$, $j = 1, \ldots, i$.

We will accomplish steps $1 - 4$ described above, using the CGS method and equations similar to (2.46) and (2.47), as follows:

$$\begin{aligned}
(\tilde{d}_i^r)^{k-1} &= (d_i^r)^{k-1} - \sum_{j=1}^{i-1} \left\{ \left[(d_j^r)^k \right]^T A (d_i^r)^{k-1} \right\} (d_j^r)^k \\
&\quad - \sum_{j=1}^{i-1} \left\{ \left[(d_j^s)^k \right]^T A (d_i^r)^{k-1} \right\} (d_j^s)^k, \qquad (2.48) \\
(\hat{d}_i^r)^{k-1} &= \frac{(\tilde{d}_i^r)^{k-1}}{\| (\tilde{d}_i^r)^{k-1} \|_A},
\end{aligned}$$

$$
\begin{aligned}
(\tilde{d}_i^s)^{k-1} &= (d_i^s)^{k-1} - \sum_{j=1}^{i} \left\{ \left[(d_j^r)^k \right]^T A (d_i^s)^{k-1} \right\} (d_j^r)^k \\
&\quad - \sum_{j=1}^{i-1} \left\{ \left[(d_j^s)^k \right]^T A (d_i^s)^{k-1} \right\} (d_j^s)^k, \qquad (2.49) \\
(\hat{d}_i^s)^{k-1} &= \frac{(\tilde{d}_i^s)^{k-1}}{\| (\tilde{d}_i^s)^{k-1} \|_A}, \\
\\
(\hat{d}_i^r)^k &= w_i^r - \sum_{j=1}^{i-1} \left\{ \left[(d_j^r)^k \right]^T A w_i^r \right\} (d_j^r)^k \\
&\quad - \sum_{j=1}^{i-1} \left\{ \left[(d_j^s)^k \right]^T A w_i^r \right\} (d_j^s)^k \\
&\quad - \sum_{j=1}^{i} \left\{ \left[(\hat{d}_j^r)^{k-1} \right]^T A w_i^r \right\} (\hat{d}_j^r)^{k-1} \\
&\quad - \sum_{j=1}^{i-1} \left\{ \left[(\hat{d}_j^s)^{k-1} \right]^T A w_i^r \right\} (\hat{d}_j^s)^{k-1}, \qquad (2.50) \\
(d_i^r)^k &= \frac{(\hat{d}_i^r)^k}{\| (\hat{d}_i^r)^k \|_A}, \\
\\
(\hat{d}_i^s)^k &= w_i^s - \sum_{j=1}^{i} \left\{ \left[(d_j^r)^k \right]^T A w_i^s \right\} (d_j^r)^k \\
&\quad - \sum_{j=1}^{i-1} \left\{ \left[(d_j^s)^k \right]^T A w_i^s \right\} (d_j^s)^k \\
&\quad - \sum_{j=1}^{i} \left\{ \left[(\hat{d}_j^r)^{k-1} \right]^T A w_i^s \right\} (\hat{d}_j^r)^{k-1} \\
&\quad - \sum_{j=1}^{i} \left\{ \left[(\hat{d}_j^s)^{k-1} \right]^T A w_i^s \right\} (\hat{d}_j^s)^{k-1}, \qquad (2.51) \\
(d_i^s)^k &= \frac{(\hat{d}_i^s)^k}{\| (\hat{d}_i^s)^k \|_A}.
\end{aligned}
$$

All the computations in step 1-4 involve vectors at the finest level. This results in a computational complexity of $O(nl)$. As before, we will reduce the computational complexity to $O(n_l)$, by performing the computations (2.48)–(2.51) on coarser grids, using formulas similar to (2.34) and (2.35), that we saw in Lemma 2.3. This means we will calculate the *rough and smooth level correction directions*, d_i^{rco} and d_i^{sco} respectively, that we introduced in formulating the MGgM-3 in Section 2.2.2. After finding the *rough and smooth level correction directions*, we we can compute the correction using the same correction formula that we derived for the MGgM-3 (2.36). We will conclude our discussion on the MGcgM-3, with an algorithmic representation of it in Algorithm 4.

Algorithm 4 Multigrid Conjugate Gradient Algorithm with Rough and Smooth Correction Directions (MGcgM-3)

if $k = 0$ **then**

perform one step of the MGgM-3 Algorithm (Algorithm 2).

else

proceed as follows.

end if

1. Calculate rough and smooth residuals:

$r_l^r := b - Ax^k$

$r_l^s := M_l^{-1} r_l$

$(d_l^r)^{new} := r_l^r$

$(d_l^s)^{new} := r_l^s$

for $i = l$ *to* 2 **do**

$r_{i-1}^r := I_i^{i-1} r_i^r$

$r_{i-1}^s := M_{i-1}^{-1} r_{i-1}^r$

$(d_{i-1}^r)^{new} := r_{i-1}^r$

$(d_{i-1}^s)^{new} := r_{i-1}^s$

end for → continued

2. Calculate new rough and smooth level correction directions:

for $i = 1$ *to* l **do**

3. old rough and smooth directions A-orthonormal to new rough and smooth coarser directions:

$k_i := A_i d_i^r$

$f_i := A_i d_i^s$

for $j = i$ *to* 2 **do**

$k_{j-1} := I_j^{j-1} k_j$

$f_{j-1} := I_j^{j-1} f_j$

end for

$s_1 := 0$

$g_1 := 0$

for $j = 1$ *to* $(i-1)$ **do**

$s_j := s_j + (d_j^r)^{new}.\left(k_j^T (d_j^r)^{new}\right) + (d_j^s)^{new}.\left(k_j^T (d_j^s)^{new}\right)$

$g_j := g_j + (d_j^r)^{new}.\left(f_j^T (d_j^r)^{new}\right) + (d_j^s)^{new}.\left(f_j^T (d_j^s)^{new}\right)$

$s_{j+1} := I_j^{j+1} s_j$

$g_{j+1} := I_j^{j+1} g_j$

end for

$d_i^r := d_i^r - s_i$

$d_i^s := d_i^s - g_i$

4. old smooth direction A-orthonormal to new rough direction:

$k_i := A_i (d_i^r)^{new}$

$norm := \left(k_i^T (d_i^r)^{new}\right)^{1/2}$

$k_i := (A_i d_i^s)/norm$

$d_i^s := d_i^s - (d_i^r)^{new}.\left(k_i^T (d_i^r)^{new}\right)$

5. A-Normalise old modified rough and smooth level correction directions:

$k_i := A_i d_i^r$

$f_i := A_i d_i^s$

$d_i^r := d_i^r.\left(k_i^T d_i^r\right)^{-1/2}$

$d_i^s := d_i^s.\left(k_i^T d_i^s\right)^{-1/2}$ → `continued`

6. New rough and smooth directions A-orthonormal to all new and old coarser directions:

$k_i := A_i(d_i^r)^{new}$

$f_i := A_i(d_i^s)^{new}$

for $j = i$ *to* 2 **do**

$k_{j-1} := I_j^{j-1} k_j$

$f_{j-1} := I_j^{j-1} f_j$

end for

$s_1 = 0$

$g_1 = 0$

for $j = 1$ *to* $(i-1)$ **do**

$s_j := s_j + (d_j^r)^{new}.\left(k_j^T (d_j^r)^{new}\right) + (d_j^s)^{new}.\left(k_j^T (d_j^s)^{new}\right)$

$s_j := s_j + d_j^r.(k_j^T d_j^r) + d_j^s.(k_j^T d_j^s)$

$g_j := g_j + (d_j^r)^{new}.\left(f_j^T (d_j^r)^{new}\right) + (d_j^s)^{new}.\left(f_j^T (d_j^s)^{new}\right)$

$g_j := g_j + d_j^r.(f_j^T d_j^r) + d_j^s.(f_j^T d_j^s)$

$s_{j+1} := I_j^{j+1} s_j$

$g_{j+1} := I_j^{j+1} g_j$

end for

$(d_i^r)^{new} := (d_i^r)^{new} - s_i$

$(d_i^s)^{new} := (d_i^s)^{new} - g_i$

7. New rough direction A-orthonormal to old modified rough direction:

$k_i := A_i(d_i^r)^{new}$

$s_i := (d_i^r)^{new} - d_i^r.(k_i^T d_i^r)$

if $\| s_i \|_\infty > \epsilon \| (d_i^r)^{new} \|_\infty$ **then**

$(d_i^r)^{new} := s_i$

else

$(d_i^r)^{new} := d_i^r$

$d_i^r := 0$

end if

→ continued

8. New smooth direction A-orthonormal to new rough direction:

$f_i := A_i(d_i^s)^{new}$

$g_i := (d_i^s)^{new} - (d_i^r)^{new}.\left(f_i^T(d_i^r)^{new}\right)$

if $\| g_i \|_\infty > \epsilon \| (d_i^s)^{new} \|_\infty$ **then**

$(d_i^s)^{new} := g_i$

else

$(d_i^s)^{new} := 0$

end if

9. New smooth direction A-orthonormal to old modified rough direction:

$f_i := A_i(d_i^s)^{new}$

$g_i := (d_i^s)^{new} - d_i^r.(f_i^T d_i^r)$

if $\| g_i \|_\infty > \epsilon \| (d_i^s)^{new} \|_\infty$ **then**

$(d_i^s)^{new} := g_i$

else

$(d_i^s)^{new} := d_i^r$

$d_i^r := 0$

end if

10. New smooth direction A-orthonormal to old modified smooth direction:

$f_i := A_i(d_i^s)^{new}$

$g_i := (d_i^s)^{new} - d_i^s.(f_i^T d_i^s)$

if $\| g_i \|_\infty > \epsilon \| (d_i^s)^{new} \|_\infty$ **then**

$(d_i^s)^{new} := g_i$

else

$(d_i^s)^{new} := d_i^s$

$d_i^s := 0$

end if

11. A-Normalise new rough and smooth level correction directions:

$k_i := A_i(d_i^r)^{new}$

$f_i := A_i(d_i^s)^{new}$

$d_i^r := (d_i^r)^{new}.\left(k_i^T(d_i^r)^{new}\right)^{-1/2}$

$d_i^s := (d_i^s)^{new}.\left(f_i^T(d_i^s)^{new}\right)^{-1/2}$

end for

→ continued

12. Calculate correction with rough and smooth level correction directions:

$s_1 := \left((d_1^r)^T r_1^r\right).d_1^r + \left((d_1^s)^T r_1^r\right).d_1^s$

for $i = 2$ *to* l **do**

$s_i := I_{i-1}^i s_{i-1}$

$s_i := s_i + \left((d_i^r)^T r_i^r\right).d_i^r + \left((d_i^s)^T r_i^r\right).d_i^s$

end for

$x^{k+1} := x^k + s_l$ ■

2.4 Computational Complexity of the MGgM and MGcgM

Matrix-vector multiplications are the computationally intensive components of the *multi-grid gradient based methods*. Let us consider the MGgM algorithm (Algorithm 1). We observe that there are:

1. 3 matrix-vector multiplications, $A_l x$ at the finest level.
2. 2 matrix-vector multiplications, $A_i x$ for levels $i = (l-1), \ldots, 1$.

In addition, the operators I_i^{i-1} and I_i^{i+1} are applied $O\left(n_i * (l-i)\right)$ times. If we assume that the multiplications:

$$\begin{aligned} &A_i x \ ,i = 1, \ldots, l \\ &I_i^{i-1} \ ,(i = 2, \ldots, l) \\ &I_i^{i+1} \ ,(i = 1, \ldots, (l-1)) \end{aligned}$$

can be performed in $O(n_i)$ operations, then we see that the computational complexity of the MGgM algorithm is $O(n_l)$. Since the MGcgM algorithm (Algorithm 3) is based on formulas similar to that of the MGgM (see Equation (2.20), (2.46), (2.47)), we have a computational complexity of $O(n_l)$ for the MGcgM also.

Chapter 3

Multilevel V-cycle based Space Correction Methods

3.1 Multilevel V-Cycle based Space Correction Method (MLV-SCOM)

From the correction formulas (2.21) and (2.36) in the previous chapter, we observe the following principle common to the iterative methods that we have discussed so far:

> *Corrections are made at all the levels of the multigrid using the smooth and\or rough level correction directions, and are then appropriately interpolated to the finest grid.*

This can viewed as solving the original problem $Ax = b$, by solving a finite number of *auxiliary problems* on subspaces $\Omega_l, \ldots, \Omega_1$, and then linking the individual solutions together. This is the basic idea behind the *method of subspace corrections* [14]. Let us quickly recall a correction term involving only the *smooth level correction directions* (2.36).

$$\sum_{i=1}^{l} \beta_i d_i^{sco} = \beta_l d_l^{sco} + I_{l-1}^{l} \left(\beta_{l-1} d_{l-1}^{sco} + I_{l-2}^{l-1} \left(\cdots + I_1^2 (\beta_1 d_1^{sco}) \right) \right). \tag{3.1}$$

I would like to point here that the above correction term occurs in the MGgM-2, MGgM-3, MGcgM-2, and MGcgM-3 (2.36).

The correction above can be regarded as a subspace correction equation. That is, in each subspace Ω_i, we solve an *independent* auxiliary problem; we do not know yet how these individual problems are defined. But what we know from the above correction equation is that, *the solutions to those auxiliary problems* are known; that is $\beta_i d_i^{sco}$ for the subspace problem on Ω_i. Individual subspace solutions are then linked together by means of prolongation operators. But, why are the individual subspace problems with the smooth level correction directions, in equation (3.1) independent?

Referring to the correction equation above, solving a subspace problem in Ω_i means we make a correction in the direction of the *smooth level correction direction* d_i^{sco}. We saw that the *smooth level correction direction* d_i^{sco} are constructed using the *smooth residual vectors* r_j^s (for example, using the formula (2.35) in Lemma 2.3), that are defined as follows:

$$r_i^s = M_i^{-1} I_l^i r^k, \quad i = l, \ldots, 1 \, .$$

We observe from the above definition that the *smooth residual vectors* r_i^s are defined independent of each other; that is they are just the smoothened versions of the restrictions of the finest grid residual vector r^k. This implies that the subspace problems, and hence their solutions corresponding to the *smooth level correction directions*, are independent of each other; that is the resulting correction using the *smooth level correction directions* (3.1) is an *additive preconditioner type*.

Can the corrections due to the *smooth level correction directions* (3.1) be improved any further? The answer is yes and we will draw our motivation from the *V-cycle multigrid method*. Instead of constructing the *smooth level correction directions* d_i^{sco} from the independently defined *smooth residual vectors* r_i^s, we will construct them from a set of dependent vectors r_i^v. To be precise, the vectors r_i^v will depend on each other just like the residual vectors at different levels of a standard V-cycle multigrid method. Let us call the vectors r_i^v, the *V-cycle based residual vectors*. For example, this means, equation (2.35) in Lemma 2.3, that we used to calculate the *smooth level correction directions* for the MGgM-3, changes as follows:

$$\hat{d}_i^{sco} = r_i^v - \sum_{j=1}^{i} I_j^i \left\{ \left[\left(d_j^{rco} \right)^T I_i^j A_i r_i^v \right] d_j^{rco} \right\} + \sum_{j=1}^{i-1} I_j^i \left\{ \left[\left(d_j^{sco} \right)^T I_i^j A_i r_i^v \right] d_j^{sco} \right\} .$$

That is, a correction space corresponding to the *smooth correction directions* can now be defined as:

$$\begin{aligned} \mathcal{R}^k_{vsmooth} &:= span\{w^v_l, \ldots, w^v_1\}, \text{where,} \\ w^v_i &= I^l_i r^v_i,\ i = l, \ldots, 1. \end{aligned} \tag{3.2}$$

We will see in a moment in Algorithm 5, how the *V-cycle based residual vectors* r^v_i are computed. Let us quickly recall that $\mathcal{R}^k_{rough}$ is the *rough residual space* and the definition of the vectors w^r_i from (2.23). The correction space $\mathcal{D}^k$ for the MLV-SCOM is now constructed using the spaces $\mathcal{R}^k_{rough}$ and $\mathcal{R}^k_{vsmooth}$ (3.2), as follows:

▶ for $i = 1, \ldots, l$, let:

1. Build $(d^r_i)^k$ using w^r_i, such that $(d^r_i)^k$ is made A-orthonormal to:
   (a) $(d^r_j)^k$, $j = 1, \ldots, (i-1)$.
   (b) $(d^s_j)^k$, $j = 1, \ldots, (i-1)$.
2. Build $(d^s_i)^k$ using w^v_i, such that $(d^s_i)^k$ is made A-orthonormal to:
   (a) $(d^r_j)^k$, $j = 1, \ldots, i$.
   (b) $(d^s_j)^k$, $j = 1, \ldots, (i-1)$.

We will accomplish steps $1-3$ described above, using the CGS method, as follows:

$$\begin{aligned} \hat{d}^r_i &= w^r_i - \left\{ \sum_{j=1}^{i-1} \left[\left(d^r_j\right)^T A w^r_i \right] d^r_j + \sum_{j=1}^{i-1} \left[\left(d^s_j\right)^T A w^r_i \right] d^s_j \right\}, \\ \hat{d}^s_i &= w^v_i - \left\{ \sum_{j=1}^{i} \left[\left(d^r_j\right)^T A w^v_i \right] d^r_j + \sum_{j=1}^{i-1} \left[\left(d^s_j\right)^T A w^v_i \right] d^s_j \right\}, \\ d^r_i &= \frac{\hat{d}^r_i}{\| \hat{d}^r_i \|_A}, \\ d^s_i &= \frac{\hat{d}^s_i}{\| \hat{d}^s_i \|_A}. \end{aligned}$$

Again, we can implement the above equations efficiently by $O(n_l)$ operations by calculating the corresponding *rough and smooth level correction directions* using equations

similar to (2.34) and (2.35), that we saw in the last chapter. The correction step for the MLV-SCOM remains the same as that of the MGgM-3 (2.36). This leads us to the following algorithmic representation of the MLV-SCOM:

Algorithm 5 Multilevel V-cycle based Space Correction Method (MLV-SCOM)

1. Calculate rough and V-cycle based residual vectors:

$r_l^r := b - Ax^k$

$d_l^r := r_l^r$

$q_l := r_l^r$

Perform the relaxation $r_l^v := M_l^{-1} q_l$, with an initial guess $r_l^v := 0$.

$d_l^s := r_l^v$

for $i = l$ *to* 2 **do**

$r_{i-1}^r := I_i^{i-1} r_i^r$

$d_{i-1}^r := r_{i-1}^r$

$q_{i-1} := I_i^{i-1}(q_i - M_i r_i^v)$

Perform the relaxation $r_{i-1}^v := M_{i-1}^{-1} q_{i-1}$, with an initial guess $r_{i-1}^v := 0$.

$d_{i-1}^s := r_{i-1}^v$

end for

2. Proceed from step 2 of the MGgM-3 algorithm (Algorithm 2). ■

3.2 Multilevel V-Cycle based Conjugate Space Correction Methods (MLV-CSCOM)

A natural extension of the MLV-SCOM is to make the correction spaces $\mathcal{D}^k$ for successive iterations (partially) A-orthonormal as described in Section 2.3 of the previous chapter. This leads us to the *Multilevel V-cycle based conjugate space correction method* (MLV-CSCOM). The MLV-CSCOM has one of the following correction spaces:

$$\begin{aligned} \breve{\mathcal{D}}^k &:= \mathcal{D}^k_{vsmooth} := span\{d_l, d_{l-1}, \ldots, d_1\} := span\{d_l^s, d_{l-1}^s, \ldots, d_1^s\}, \\ \breve{\mathcal{D}}^k &:= \mathcal{D}^k_{vsmooth} \oplus \mathcal{D}^k_{rough} := span\{d_{2l}, d_{2l-1}, \ldots, d_1\} := span\{d_l^r, d_l^s, \ldots, d_1^r, d_1^s\}, \end{aligned} \tag{3.3}$$

where the space $\mathcal{D}^k_{vsmooth}$ is constructed using the space $\mathcal{R}^k_{vsmooth}$ (3.2). Should a recap of the naming convention of the correction directions d_i be required, then I would like to refer the reader to our discussion in Section 2.2.2.

Let us also recall that d^r_i and d^s_i are the *rough and smooth correction directions* respectively. We will be particularly interested in those variants of the MLV-CSCOM where we make a correction using both the *rough and smooth level correction directions*. But the variants will differ in the way the correction spaces for successive iterations ($\breve{\mathcal{D}}^k$ and $\breve{\mathcal{D}}^{k-1}$) are A-orthonormalised, as described below:

MLV-CSCOM-3A $\Rightarrow$ both the *rough and smooth correction directions* from $\breve{\mathcal{D}}^k$ and $\breve{\mathcal{D}}^{k-1}$ are partially A-orthonormalised. This corresponds to the modified condition (2.45) of the MGcgM.

MLV-CSCOM-3B $\Rightarrow$ only the *rough correction directions* from $\breve{\mathcal{D}}^k$ and $\breve{\mathcal{D}}^{k-1}$ are partially A-orthonormalised.

The correction step for both the variants of the MLV-CSCOM remains the same as that of the MGgM-3 or MGcgM-3 (2.36). I will describe these variants in detail in the following sections.

3.2.1 Variants of the MLV-CSCOM

3.2.2 MLV-CSCOM-3A

The MLV-CSCOM-3A has exactly the same formulation as the *multigrid conjugate gradient method with rough and smooth correction directions* (MGcgM-3), except for its correction space, that is $\breve{\mathcal{D}}^k$, as described above (3.3). We will construct the successive correction spaces such that they satisfy the modified condition (2.45) in the last chapter. This leads us to Algorithm 6 of the MLV-CSCOM-3A.

3.2.3 MLV-CSCOM-3B

Let us recall again that $\mathcal{R}^k_{rough}$ is the *rough residual space*. The only change now is that, we construct the correction spaces $\breve{\mathcal{D}}^k$ such that, they satisfy the following condition

Algorithm 6 Multilevel V-cycle based Space Correction Method with Rough and Smooth Correction Directions Partially A-orthonormalised (MLV-CSCOM-3A)

if $k = 0$ **then**

perform one step of the MGgM-3 algorithm (Algorithm 2)

else

proceed as follows.

end if

1. Calculate rough and V-cycle based residuals:

Perform step 1 of the MLV-SCOM algorithm (Algorithm 5).

2. Calculate new rough and smooth level correction directions:

Proceed from step 2 of the MGcgM-3 algorithm (Algorithm 4). ■

instead of the modified condition (2.45):

$$\begin{aligned}
\breve{\mathcal{R}}^1 &:= \mathcal{R}^1_{rough} \oplus \mathcal{R}^1_{vsmooth} \\
\breve{\mathcal{D}}^1 &= \breve{\mathcal{R}}^1, \\
\breve{\mathcal{D}}^k &:= span\{d^k_{2l}, d^k_{2l-1}, \ldots, d^k_1\} \\
&:= \ span\{d^s_l, d^r_l, \ldots, d^s_1, d^r_1\}, \\
\| d^k_i \|_A &= 1, && i = 1, \ldots, 2l, \\
d^k_p,\ d^k_{p+1} &\in I^l_p(\mathbb{R}^{n_p}), && p = 1, \ldots, l, \\
d^k_i &\perp_A d^k_j, && i \neq j.
\end{aligned}$$

and the new condition:

$$d^{k-1}_i \perp_A d^k_j, \qquad i \leq j,\ k > 1,$$

is valid only when: $(i+1) \bmod 2 \ = \ (j+1) \bmod 2 \ = 0$. We also have:

$$span(\breve{\mathcal{R}}^k, \breve{\mathcal{D}}^{k-1}) = span(\breve{\mathcal{D}}^k, \breve{\mathcal{D}}^{k-1}).$$

The condition above guarantees us that, only the *rough correction directions* of the successive correction spaces are made partially A-orthonormal to each other. Let us now

construct the correction space $\breve{\mathcal{D}}^k$ or equivalently the *rough and smooth correction directions* d_i^r and d_i^s for the MLV-CSCOM-3B, using the space $\mathcal{R}^k_{rough} \oplus \mathcal{R}^k_{vsmooth}$, and the condition above, as follows:

▶ for $i = 1, \ldots, l$:

1. Construct $(\hat{d}_i^r)^{k-1}$ using $(d_i^r)^{k-1}$, such that $(\hat{d}_i^r)^{k-1}$ is made A-orthonormal to:
   (a) $(d_j^r)^k$, $j = 1, \ldots, (i-1)$.
2. Construct $(d_i^r)^k$ using w_i^r (see 2.23), such that $(d_i^r)^k$ is made A-orthonormal to:
   (a) $(d_j^r)^k$, $j = 1, \ldots, (i-1)$.
   (b) $(d_j^s)^k$, $j = 1, \ldots, (i-1)$.
   (c) $(\hat{d}_j^r)^{k-1}$, $j = 1, \ldots, i$.
3. Construct $(d_i^s)^k$ using w_i^v, such that $(d_i^s)^k$ is made A-orthonormal to:
   (a) $(d_j^r)^k$, $j = 1, \ldots, i$.
   (b) $(d_j^s)^k$, $j = 1, \ldots, (i-1)$.

We will accomplish steps $1-3$ described above, using the CGS method, as follows:

$$
\begin{aligned}
(\tilde{d}_i^r)^{k-1} &= (d_i^r)^{k-1} - \sum_{j=1}^{i-1} \left\{ \left[(d_j^r)^k \right]^T A (d_i^r)^{k-1} \right\} (d_j^r)^k, \qquad (3.4)\\
(\hat{d}_i^r)^{k-1} &= \frac{(\tilde{d}_i^r)^{k-1}}{\| (\tilde{d}_i^r)^{k-1} \|_A},
\end{aligned}
$$

$$
\begin{aligned}
(\hat{d}_i^r)^k &= w_i^r - \sum_{j=1}^{i-1} \left\{ \left[(d_j^r)^k \right]^T A w_i^r \right\} (d_j^r)^k - \sum_{j=1}^{i-1} \left\{ \left[(d_j^s)^k \right]^T A w_i^r \right\} (d_j^s)^k \\
&\quad - \sum_{j=1}^{i} \left\{ \left[(\hat{d}_j^r)^{k-1} \right]^T A w_i^r \right\} (\hat{d}_j^r)^{k-1}, \\
(d_i^r)^k &= \frac{(\hat{d}_i^r)^k}{\| (\hat{d}_i^r)^k \|_A},
\end{aligned}
$$

$$
\begin{aligned}
\left(\hat{d}_i^s\right)^k &= w_i^v - \sum_{j=1}^{i} \left\{ \left[(d_j^r)^k \right]^T A w_i^v \right\} (d_j^r)^k - \sum_{j=1}^{i-1} \left\{ \left[(d_j^s)^k \right]^T A w_i^v \right\} (d_j^s)^k, \qquad (3.5) \\
(d_i^s)^k &= \frac{\left(\hat{d}_i^s\right)^k}{\| \left(\hat{d}_i^s\right)^k \|_A}.
\end{aligned}
$$

The computations (3.4)–(3.5) can be performed efficiently by $O(n_l)$ operations, by using equations similar to (2.34) and (2.35), that we saw in Lemma 2.3 of the last chapter. This leads us to the following algorithmic representation of the MLV-CSCOM-3B:

Algorithm 7 Multilevel V-cycle based Space Correction Method with Rough Correction Directions Partially A-orthonormalised (MLV-CSCOM-3B)

---

if $k = 0$ **then**

perform one step of the MLV-SCOM Algorithm (Algorithm 5)

else

proceed as follows.

end if

1. Calculate rough and V-cycle based residuals:

$r_l^r := b - Ax^k$

$(d_l^r)^{new} := r_l^r$

$q_l := r_l^r$

Perform the relaxation $r_l^v := M_l^{-1} q_l$, with an initial guess $r_l^v := 0$.

$(d_l^s)^{new} := r_l^v$

for $i = l$ *to* 2 **do**

$r_{i-1}^r := I_i^{i-1} r_i^r$

$(d_{i-1}^r)^{new} := r_{i-1}^r$

$q_{i-1} := I_i^{i-1}(q_i - M_i r_i^v)$

Perform the relaxation $r_{i-1}^v := M_{i-1}^{-1} q_{i-1}$, with an initial guess $r_{i-1}^v := 0$.

$(d_{i-1}^s)^{new} := r_{i-1}^v$

end for → `continued`

---

2. Calculate new rough and smooth level correction directions:

for $i = 1$ *to* l **do**

3. old rough directions A-orthonormal to new rough coarser directions:

$k_i := A_i d_i^r$

for $j = i$ *to* 2 **do**

$k_{j-1} := I_j^{j-1} k_j$

end for

$s_1 := 0$

for $j = 1$ *to* $(i-1)$ **do**

$s_j := s_j + (d_j^r)^{new} . \left(k_j^T (d_j^r)^{new}\right)$

$s_{j+1} := I_j^{j+1} s_j$

end for

$d_i^r := d_i^r - s_i$

4. A-Normalise old modified rough level correction directions:

$k_i := A_i d_i^r$

$d_i^r := d_i^r . \left(k_i^T d_i^r\right)^{-1/2}$

5. New rough directions A-orthonormal to all new directions and old rough coarser directions. New smooth directions A-orthonormal to all new directions:

$k_i := A_i (d_i^r)^{new}$

$f_i := A_i (d_i^s)^{new}$

for $j = i$ *to* 2 **do**

$k_{j-1} := I_j^{j-1} k_j$

$f_{j-1} := I_j^{j-1} f_j$

end for

$s_1 = 0$

$g_1 = 0$

→ continued

for $j = 1$ *to* $(i-1)$ **do**

$s_j := s_j + (d_j^r)^{new}.\left(k_j^T(d_j^r)^{new}\right) + (d_j^s)^{new}.\left(k_j^T(d_j^s)^{new}\right)$

$s_j := s_j + d_j^r.(k_j^T d_j^r)$

$g_j := g_j + (d_j^r)^{new}.\left(f_j^T(d_j^r)^{new}\right) + (d_j^s)^{new}.\left(f_j^T(d_j^s)^{new}\right)$

$s_{j+1} := I_j^{j+1} s_j$

$g_{j+1} := I_j^{j+1} g_j$

end for

$(d_i^r)^{new} := (d_i^r)^{new} - s_i$

$(d_i^s)^{new} := (d_i^s)^{new} - g_i$

6. New rough direction A-orthonormal to old modified rough direction:

$k_i := A_i(d_i^r)^{new}$

$s_i := (d_i^r)^{new} - d_i^r.(k_i^T d_i^r)$

if $\| s_i \|_\infty > \epsilon \| (d_i^r)^{new} \|_\infty$ **then**

$(d_i^r)^{new} := s_i$

else

$(d_i^r)^{new} := d_i^r$

$d_i^r := 0$

end if

7. New smooth direction A-orthonormal to new rough direction:

$f_i := A_i(d_i^s)^{new}$

$g_i := (d_i^s)^{new} - (d_i^r)^{new}.\left(f_i^T(d_i^r)^{new}\right)$

if $\| g_i \|_\infty > \epsilon \| (d_i^s)^{new} \|_\infty$ **then**

$(d_i^s)^{new} := g_i$

else

$(d_i^s)^{new} := 0$

end if

→ continued

8. A-Normalise new rough and smooth level correction directions:

$k_i := A_i(d_i^r)^{new}$

$f_i := A_i(d_i^s)^{new}$

$d_i^r := (d_i^r)^{new}.\left(k_i^T(d_i^r)^{new}\right)^{-1/2}$

$d_i^s := (d_i^s)^{new}.\left(f_i^T(d_i^s)^{new}\right)^{-1/2}$

end for

9. Calculate correction with rough and smooth level correction directions:

$s_1 := \left((d_1^r)^T r_1^r\right).d_1^r + \left((d_1^s)^T r_1^r\right).d_1^s$

for $i = 2$ *to* l **do**

$s_i := I_{i-1}^i s_{i-1}$

$s_i := s_i + \left((d_i^r)^T r_i^r\right).d_i^r + \left((d_i^s)^T r_i^r\right).d_i^s$

end for

$x^{k+1} := x^k + s_l$ ■

Chapter 4

Numerical Results and Observations

Applying standard multigrid (MG) to problems with large jumping coefficients leads to a poor convergence rate [13]. A common approach to achieve high convergence rates for such problems is to use a multigrid preconditioned conjugate gradient method (MGPcg). Other strategies for such problems include, using jump-interface preserving coarsening techniques [13], or special smoothers such as block/line relaxation, or problem dependent restriction and prolongation operators [2]. But all of them involve complicated implementation techniques.

We have so far discussed several combinations of the multigrid and gradient based methods. The major advantage of these methods, in comparison to the above mentioned strategies, is the *ease of implementation*, in particular for problems with large jumping coefficients. For such types of problems, it has already been shown by Pflaum [8] that, the MGcgM-1, with one standard V(1,1) multigrid cycle inserted after every sth iteration, leads to a faster convergence compared to the MGPcg, MGgM, and its variants. Let us call this algorithm MGcgM-1+MG(s). For the numerical experiments described here, I used a standard V(2,2) multigrid cycle after every sth iteration.

I performed all the numerical experiments using the C++ expression templates based library 2D-EXPDE [4]. I will present the numerical results for the following Poisson's equation with a large jumping coefficients:

$$\begin{aligned} \nabla.\left(\mu(x,y)\nabla u(x,y)\right) &= f(x,y) \text{ on } \Omega, \\ u(x,y) &= 0 \text{ on } \partial\Omega, \end{aligned} \tag{4.1}$$

where $\Omega =]0,1[^2$ and $\mu(x,y)$ is the (variable) coefficient. Let us consider the following cases for the coefficient $\mu(x,y)$ and the right hand side $f(x,y)$:

Example 1:

$$f(x,y) = 1,$$
$$\mu(x,y) = \begin{cases} \alpha \text{ as shown in Figure 4.1,} \\ 1.0 \text{ else.} \end{cases}$$

Example 2:

$$\gamma(x,y) = \frac{200\alpha}{e^{100(x-0.5)^2} e^{100(y-0.5)^2}},$$

$$f(x,y) = \gamma(x,y)\left\{cos(\pi x)sinh(\pi y)(x-0.5) + sin(\pi x)cosh(\pi y)(y-0.5)\right\},$$

$$\mu(x,y) = 1 + \frac{\alpha}{e^{100(x-0.5)^2} e^{100(y-0.5)^2}} \frac{sinh(\pi)}{\pi},$$

and the exact solution is:

$$u(x,y) = \frac{sinh(\pi y)sin(\pi x)}{sinh(\pi)}. \tag{4.2}$$

The coefficient $\mu(x,y)$ in Example 2 is also controlled by the parameter α as shown in Figure 4.2. That is $\mu(x,y)$ is of the same order as the parameter α for both the examples. A large value of the parameter α implies a large value of the jumping coefficient μ in the regions of jump shown in Figure 4.1 and 4.2.

I used bilinear finite elements for discretising Equation 4.1 and a Gauss-Seidel smoother with two smoothing iterations to calculate the *smooth and V-cycle based residuals*. For both the examples, we will start with an initial guess $x^0 = \overrightarrow{0}$. To study the convergence behaviour, for Example 1, we will consider the L2 norm of the residual $\| b - Ax^k \|_2$ and

for Example 2, the L2 norm of the algebraic error $\| x^k - \hat{x} \|_2$, where $\hat{x}$ is the exact solution obtained from equation (4.2). I would like to point here that the computational amounts of all the algorithms considered in our numerical experiments are roughly the same.

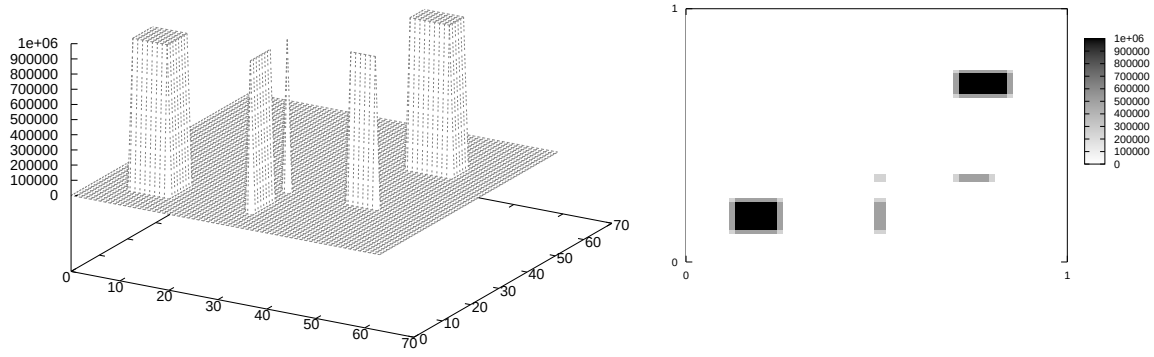

Figure 4.1: Jumping coefficient and its contour, for $\alpha = 10^6$ in Example 1.

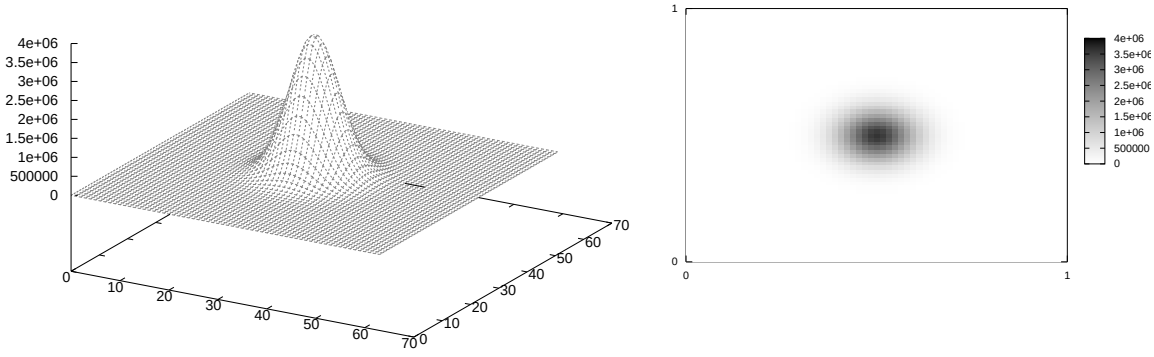

Figure 4.2: Jumping coefficient and its contour, for $\alpha = 10^6$ in Example 2.

Table 1 and 2 clearly show us the poor convergence behaviour of the multigrid algorithm (here a V(2,2) multigrid cycle) for large values of α. We can see that the multigrid algorithm, for values of $\alpha \geq 10^3$, is practically not a good choice. Theorems 2.1 and 2.2

guarantee us that, all the algorithms that we discussed in the last chapters, have a convergence rate that is independent of the number of unknowns N. This can be confirmed from Table 3. For the calculation of the convergence rate after k iterations, I used the following approach:

$$\text{convergence rate } = \left(\frac{\| b - Ax^k \|_2}{\| b - Ax^0 \|_2}\right)^{1/(k-k_0)}, \; k_0 \approx 1 - 5.$$

Table 1, 2, and 3 also give us a strong indication of the superior performance of the MGcgM-3 and MLV-CSCOM-3A compared to the the MGcgM-1 and MGcgM-2. This is a first hint of achieving improved convergence rates with a bigger correction space, as in the MGcgM-3 and MLV-CSCOM-3A (3.3).

Algorithm	$\alpha = 10^2$	$\alpha = 10^3$	$\alpha = 10^5$
MG	28	258	> 2000
MGcgM-1	164	493	> 2000
MGcgM-3	29	79	296
MLV-CSCOM-3A	17	49	238

Table 1: **Example 1**, number of iterations to achieve $\| b - Ax^k \|_2 < 10^{-8}$, with $N \approx 2^{18}$.

Algorithm	$\alpha = 10^2$	$\alpha = 10^3$	$\alpha = 10^5$
MG	36	276	> 3000
MGcgM-1	537	1942	> 2000
MGcgM-3	10	16	47
MLV-CSCOM-3A	7	12	63

Table 2: **Example 2**, number of iterations to achieve $\| x^k - \hat{x} \|_2 \approx 10^{-5}$, with $N \approx 2^{18}$.

We observe in Figure 4.3 and 4.4, an improved convergence behaviour using *smooth correction directions* that are constructed using the *smooth multilevel residual space* $\mathcal{R}^k_{smooth}$ (2.10). That is, the MGgM-2 and MGgM-3 converge faster than the MGgM-1. This is due to the optimality of the *smooth correction directions* that I mentioned in

Algorithm	$N \approx 2^{12}$	$N \approx 2^{14}$	$N \approx 2^{16}$	$N \approx 2^{18}$
MGgM-2	0.97	0.98	0.98	0.98
MGcgM-2	0.93	0.94	0.95	0.90
MGgM-3	0.97	0.98	0.98	0.98
MGcgM-3	0.88	0.90	0.91	0.87
MLV-CSCOM-3A	0.77	0.78	0.78	0.79

Table 3: **Example 1**, optimal convergence rate ($\| b - Ax^k \|_2 < 10^{-12}$) independent of number of unknowns N, with $\alpha = 10^3$.

Section 2.2.1 of Chapter 2. Apparently, it might appear that using only the *smooth correction directions* as in the MGgM-2, leads to an equally good performance compared to using both the *rough and smooth correction directions* as in the MGgM-3. But by comparing the convergence rates of the MGcgM-2 and MGcgM-3 in Table 3, we see that, that for $\alpha = 10^3$, we obtain a relatively poor convergence rate using only *smooth correction directions*.

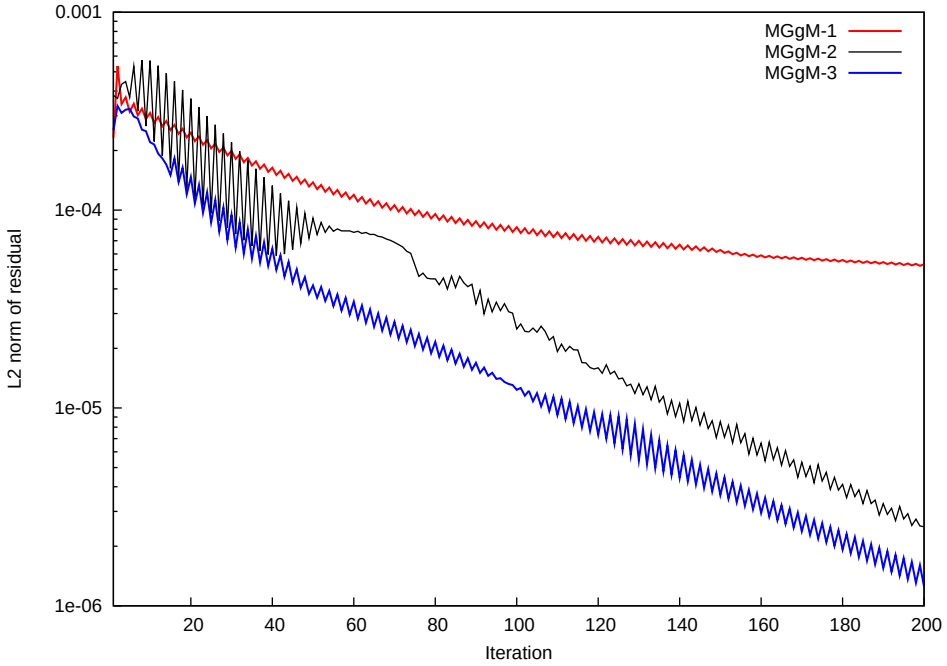

Figure 4.3: **Example 1**, improved MGgM convergence with smooth correction directions constructed using $\mathcal{R}^k_{smooth}$, with $\alpha = 10^3$, $N \approx 2^{18}$.

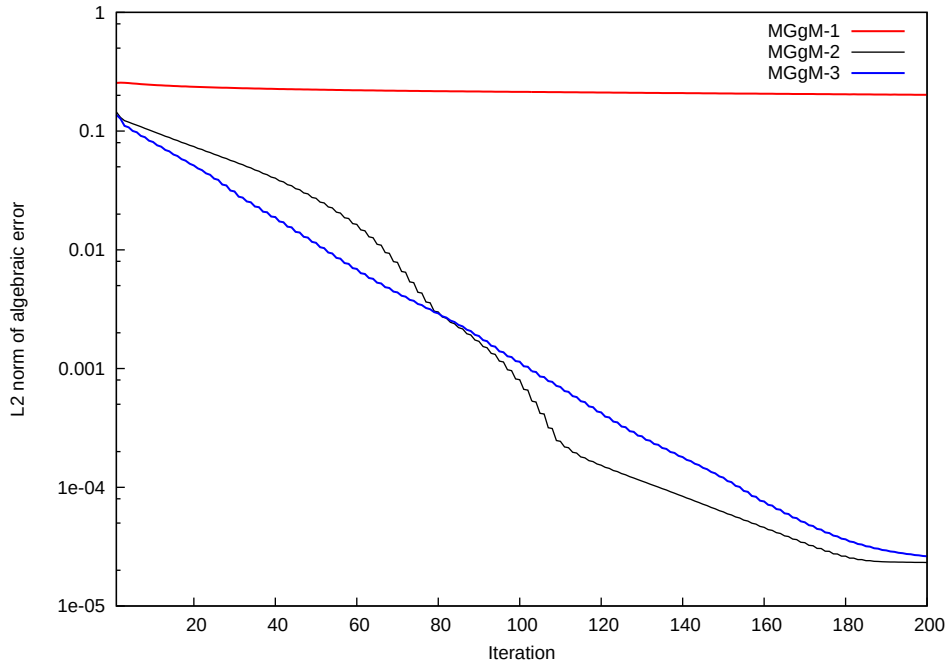

Figure 4.4: **Example 2**, improved MGgM convergence with smooth correction directions constructed using $\mathcal{R}^k_{smooth}$, with $\alpha = 10^3$, $N \approx 2^{18}$.

Let us proceed now by analysing the *multigrid gradient based algorithms* that make use of both *rough and smooth correction directions*. As a consequence of Theorem 2.2, we observe in Figure 4.5 and 4.6, an improved convergence behaviour by A-orthonormalising correction spaces for successive iterations. We also observe in Figure 4.6 and 4.6 that, for large values of α, the MGgM-3 is not a suitable algorithm. We also observe here that, using an additional V(2,2) multigrid cycle after every 10th iteration, as in the MGcgM-3+MG(10), improves the convergence rate. We also observe for Example 1 that, the MGcgM-3+MG(10) results in a drastic but only a momentary reduction in the residual norm, as indicated by the spikes in Figure 4.5.

This is the motivation behind the algorithms that we discussed in Chapter 3, that make use of the *V-cycle based residual vectors* to construct a correction space. By constructing the correction spaces for these algorithms (3.2 and 3.3), we observe that, by doing so, we implicitly include the correction space of a standard V-cycle multigrid *without postsmoothing*. It is also known that, under certain conditions on the smoothing operator and using the Galerkin variational form as the coarse grid operator, a standard V-cycle multigrid method without postsmoothing always converges [3]. This means, we

may still obtain a better convergence rate for the algorithms described in Chapter 3, without inserting an additional V(2,2) or V(1,1) multigrid cycle after every sth iteration.

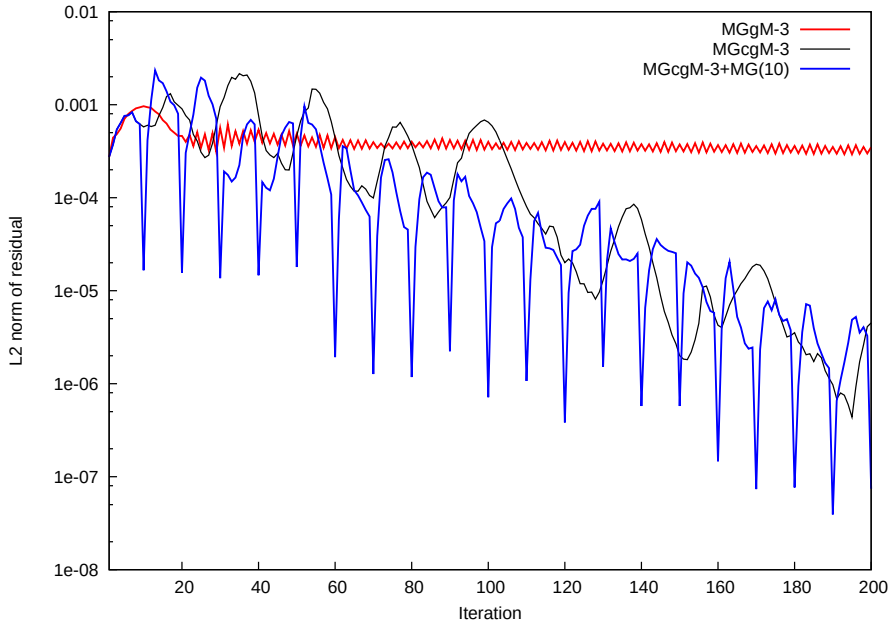

Figure 4.5: **Example 1**, improved convergence with partial A-orthonormalisation of successive correction spaces, with $\alpha = 10^5$.

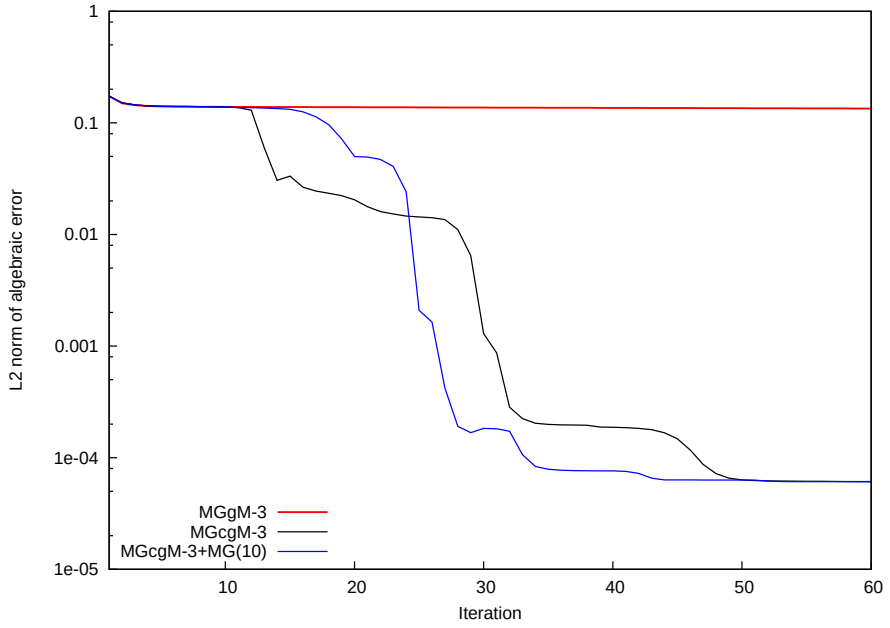

Figure 4.6: **Example 2**, improved convergence with partial A-orthonormalisation of successive correction spaces, with $\alpha = 10^5$.

Figures 4.7, 4.8, 4.9, and 4.10 show us a comparison of the performances of the MG(2,2), MG(2,0), and a variant of the MLV-SCOM, where we make corrections using only the smooth correction directions constructed from the *V-cycle based residual vectors*. Let us call this variant, the MLV-SCOM-2. Observe that the correction step for the MLV-SCOM-2 is the same as that of the MGgM-1 (2.21), except for the type of *smooth correction direction* that is used. This correction, for instance, is also included in the total correction for the MLV-CSCOM and its variants. We observe in Figure 4.8 and 4.10 that, for small and large values of α for Example 2, the MLV-SCOM-2 outperforms both MG(2,0) and MG(2,2). We observe the same for Example 1 in Figure 4.7, with $\alpha = 10^3$. But for Example 1 with $\alpha = 10^5$, as in Figure 4.9, we observe that the MLV-SCOM-2 performs poorly in comparison to both MG(2,0) and MG(2,2). In fact the residual oscillates rapidly and periodically, as seen in Figure 4.9. This could be because of a loss of A-orthonormalisation in constructing the correction directions using the *classical Gram-Schmidt method*, which is not numerically stable. An alternative is to construct the A-orthonormal correction directions using the *classical Gram-Schmidt method with reorthogonalisation*, which is numerically stable against round-off errors [5].

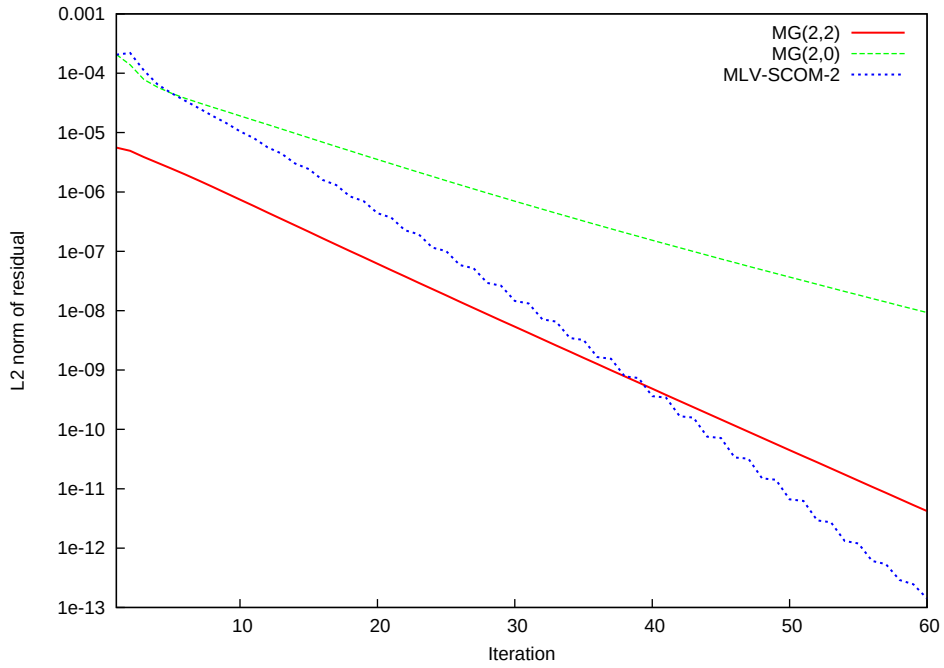

Figure 4.7: **Example 1**, Comparison of V-cycle multigrid method with and without postsmoothing, with $\alpha = 10^3$.

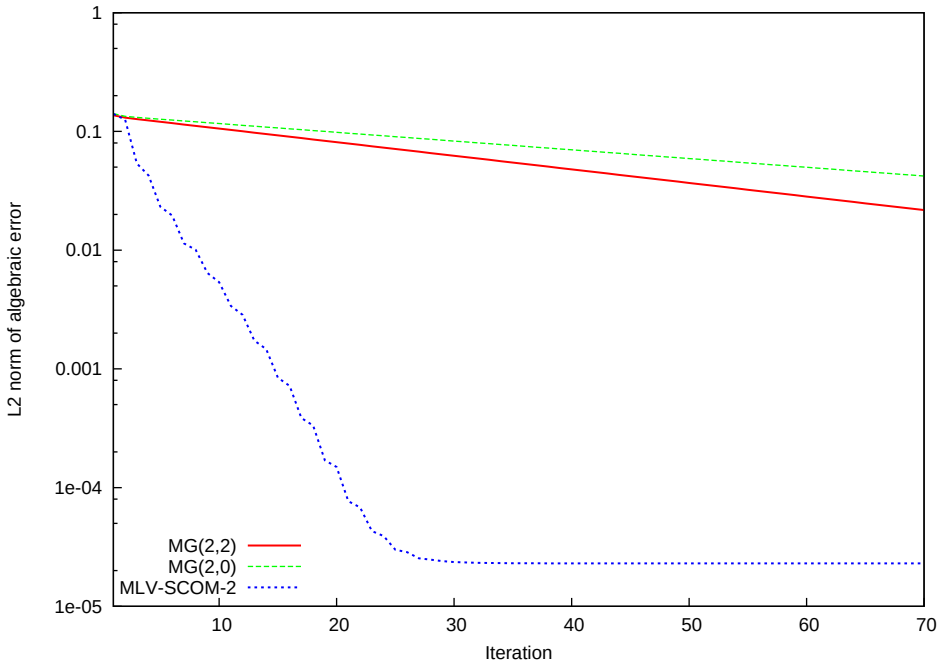

Figure 4.8: **Example 2**, Comparison of V-cycle multigrid method with and without postsmoothing, with $\alpha = 10^3$.

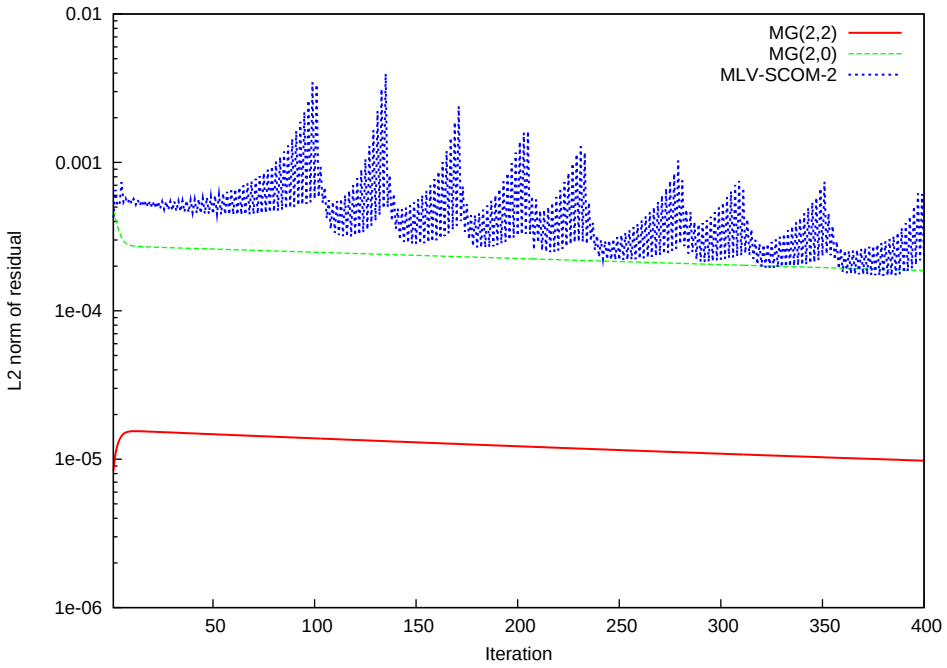

Figure 4.9: **Example 1**, Comparison of V-cycle multigrid method with and without postsmoothing, with $\alpha = 10^5$.

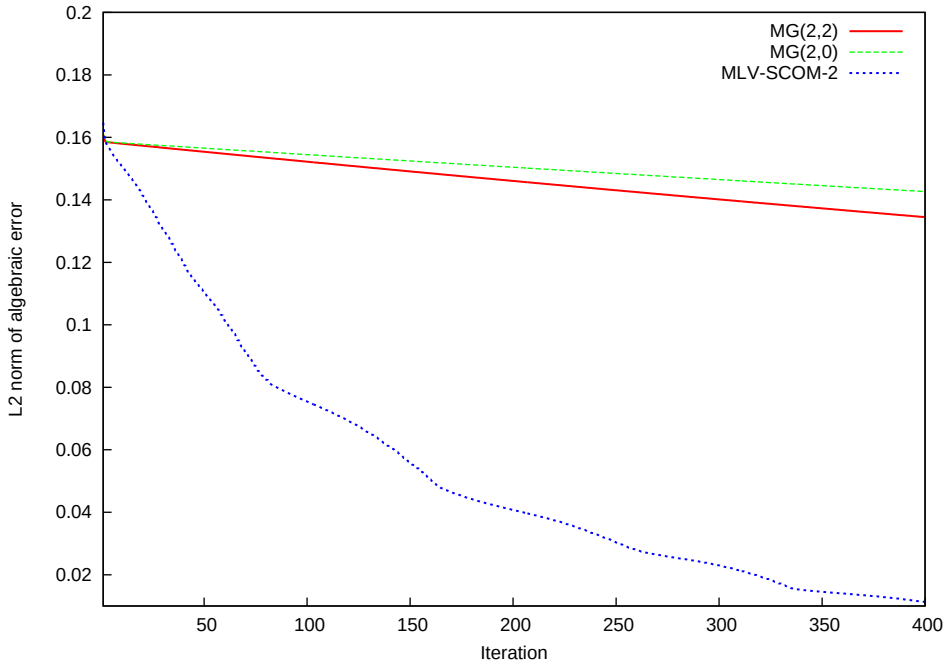

Figure 4.10: **Example 2**, Comparison of V-cycle multigrid method with and without postsmoothing, with $\alpha = 10^5$.

Let us now consider the performances of the MLV-CSCOM-3A, MLV-CSCOM-3B, MGcgM-3 and their variants that include a standard V(2,2) multigrid cycle after every 10th iteration. For Example 1 with $\alpha = 4, 5,$ and 6, we see in Figure 4.11, 4.12, and 4.13, that the MGcgM-1+MG(10) is completely outperformed by the MGcgM-3+MG(10), MLV-CSCOM-3A, and MLV-CSCOM-3A+MG(10). The poor convergence behaviour of MLV-CSCOM-3B and MLV-CSCOM-3B+MG(10) seen in Figure 4.11, 4.12, and 4.13 indicate us that, A-orthonormalising only the *rough correction directions* of the successive correction spaces, as we discussed in Section 3.2.3, is not a good choice.

Looking at the convergence behaviour of the MGcgM-3+MG(10), MLV-CSCOM-3A, and MLV-CSCOM-3A+MG(10) with respect to the L2 norm of the residual in Figure 4.11, 4.12, and 4.13, we observe that, a correction that is made using both the *rough and smooth correction directions* results in a relatively good convergence behaviour. In particular, when the *smooth correction directions* in such bigger correction spaces are constructed using the *V-cycle based residual vectors* described in Chapter 3, we achieve the best convergence rates, as it can be seen through the convergence behaviour of the MLV-CSCOM-3A and MLV-CSCOM-3A+MG(10). Also observe that the MLV-CSCOM-3A,

without the additional MG(2,2) V-cycle after every 10th iteration, performs as good as the MLV-CSCOM-3A+MG(10). As I pointed out before, the convergence behaviour of the MLV-CSCOM-3A+MG(10) or the MLV-CSCOM-3B+MG(10), is expected not to have sharp spikes for every 10th iteration, where an MG(2,2) V-cycle is inserted. Because, by their definitions (3.2, 3.3), the correction spaces of the MLV-CSCOM-3A and MLV-CSCOM-3B, already include a standard V(2,0) multigrid method's correction space. In spite of this, we observe rapidly oscillating spikes for every 10th iteration. This behaviour has to be investigated further.

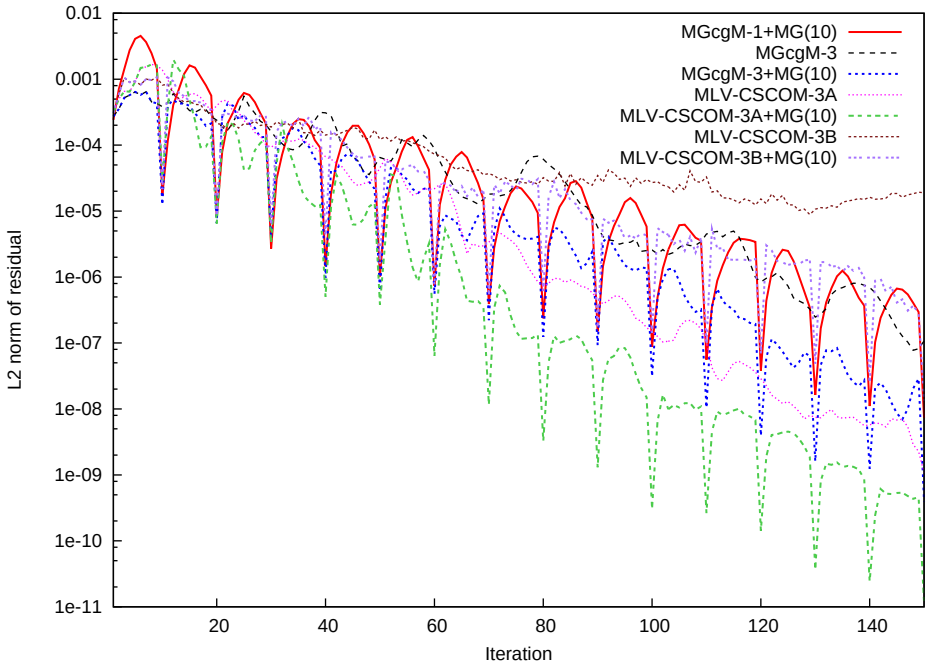

Figure 4.11: **Example 1**, improved convergence with V-cycle based smooth correction directions, with $\alpha = 10^4$.

Let us now consider for Example 2, the performances of the MLV-CSCOM-3A, MLV-CSCOM-3B, MGcgM-3, and their variants that include a standard V(2,2) multigrid cycle after every 10th iteration. Looking at the convergence behaviour with respect to the L2 norm of the algebraic error in Figure 4.14, we observe for $\alpha = 4$, a convergence behaviour similar to that of Example 1. But for $\alpha = 5, 6$, we observe in Figure 4.15 and 4.16 that, the MGcgM-3 and MGcgM-3+MG(10) perform as good as the MLV-CSCOM-3A+MG(10). Here again, the MGcgM-1+MG(10) is completely outperformed by the other algorithms.

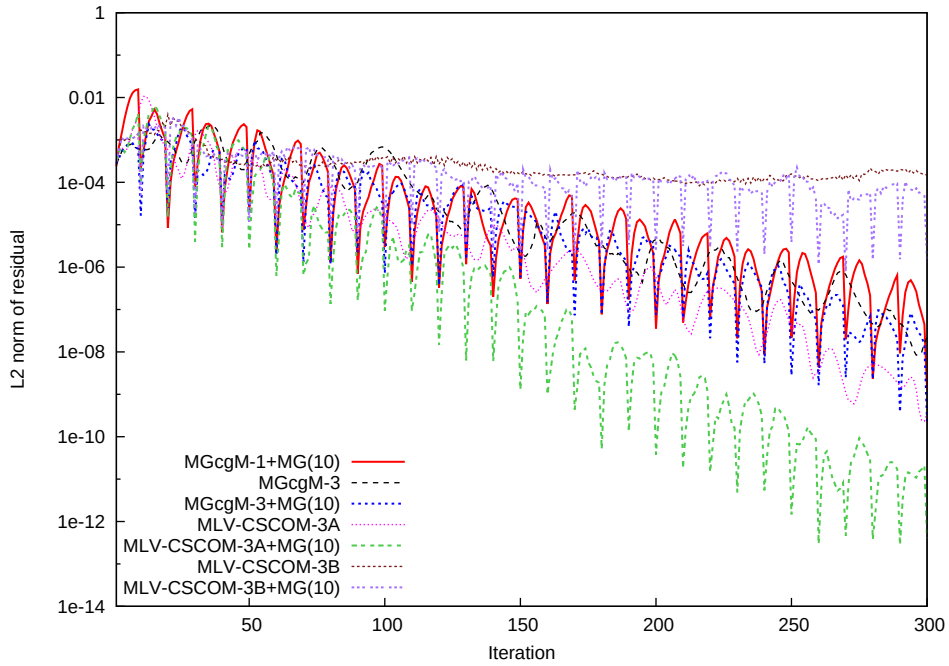

Figure 4.12: **Example 1**, improved convergence with V-cycle based smooth correction directions, with $\alpha = 10^5$.

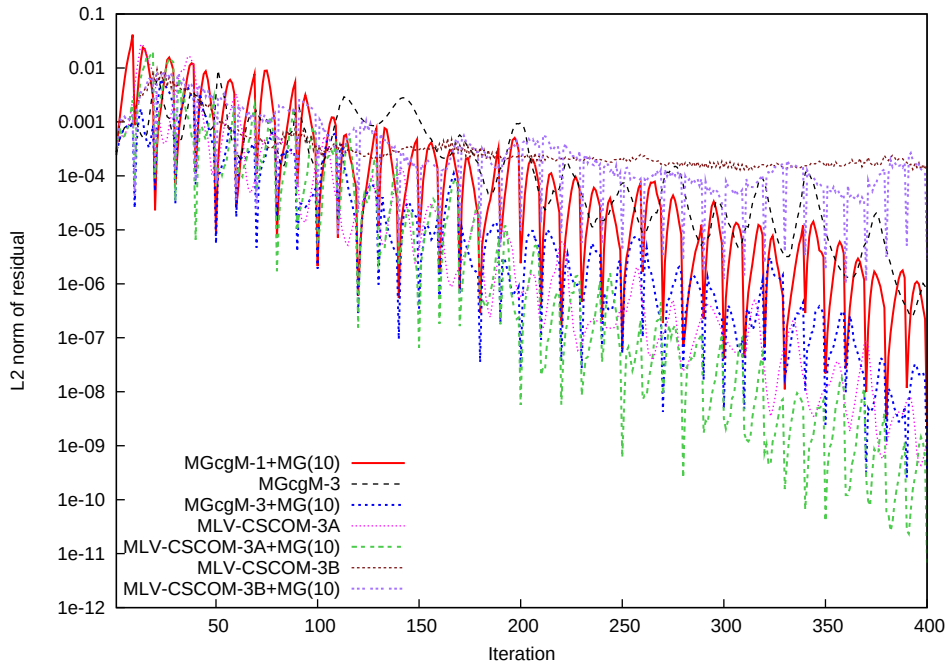

Figure 4.13: **Example 1**, improved convergence with V-cycle based smooth correction directions, with $\alpha = 10^6$.

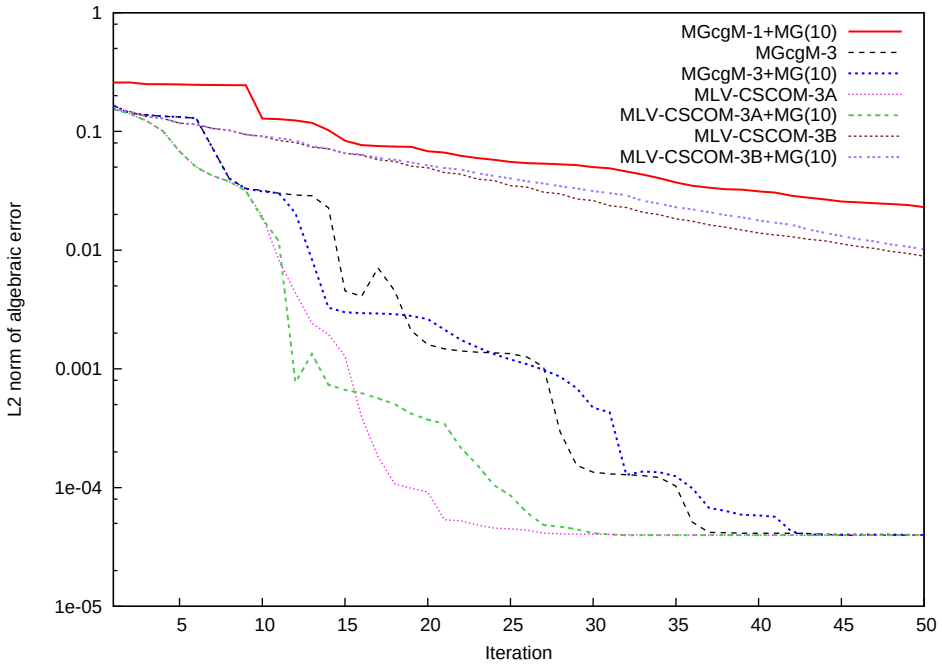

Figure 4.14: **Example 2**, improved convergence with V-cycle based smooth correction directions, with $\alpha = 10^4$.

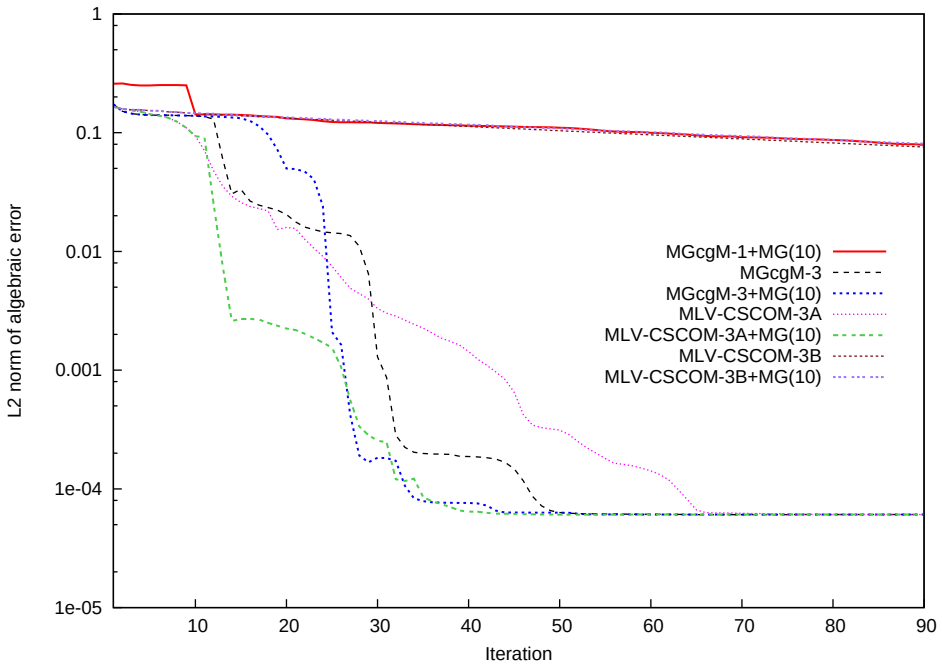

Figure 4.15: **Example 2**, improved convergence with V-cycle based smooth correction directions, with $\alpha = 10^5$.

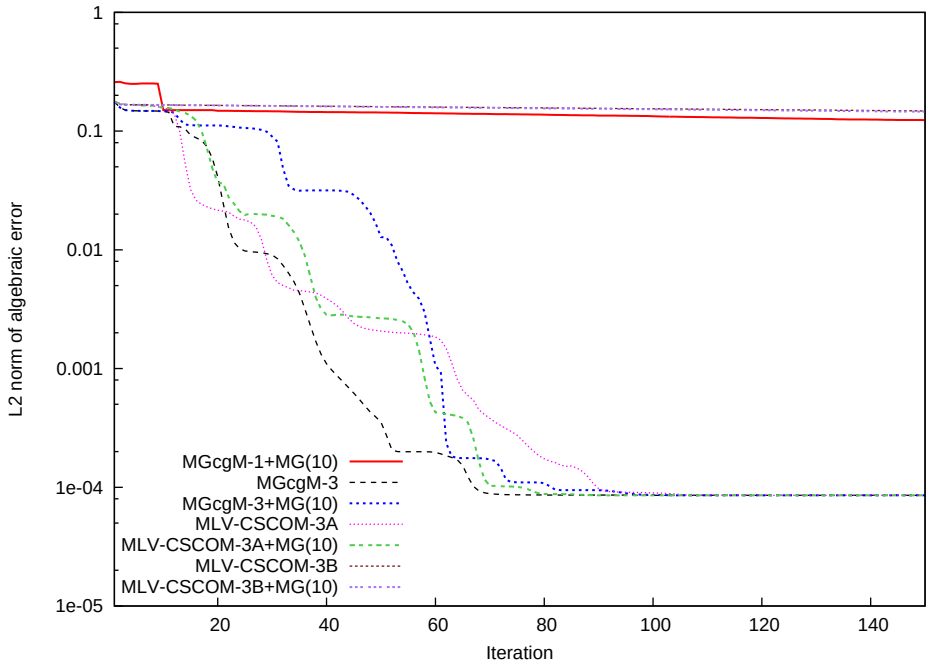

Figure 4.16: **Example 2**, improved convergence with V-cycle based smooth correction directions, with $\alpha = 10^6$.

Until now, we have seen numerical results only for extremely large values of the parameter α. As a preliminary test case for extremely small values of the parameter α, we will consider the MGcgM-1+MG(10) for Example 1. Table 5 shows us the outcome.

α	10^{-2}	10^{-3}	10^{-4}	10^{-5}
Iterations	44	84	114	121

Table 4: **Example 1**, MGcgM-1+Mg(10) - number of iterations required to achieve $\| b - Ax^k \|_2 < 10^{-9}$ for different values of the parameter α, with $s = 10$.

From the numerical results that we have seen so far, we can conclude that the MGcgM-3+MG(s), MLV-CSCOM-3A, and MLV-CSCOM-3A+MG(s) are the algorithms that perform better for large values of α ($\geq 10^3$). Therefore, we will now look at Table 5, where these three algorithms are compared for Example 1. For the MGcgM-3+MG(s) and MLV-CSCOM-3A+MG(s), we also compare the performance by varying the parameters α and s. We immediately observe from Table 5 that the MLV-CSCOM-3A and MLV-CSCOM-3A+MG(s) perform better than the MGcgM-3+MG(s), for values of α ranging

from 10^{-5} to 10^{6}. We again observe that the MLV-CSCOM-3A performs nearly as good as the MLV-CSCOM-3A+MG(10). It is also instructive to observe that, for values of α ranging from 10^{-2} to 10^{6}, the MGcgM-3+MG(s) shows a relatively better convergence behaviour for smaller values of the parameter s. Whereas the MLV-CSCOM-3A+MG(s), shows an erratic behaviour with respect to the parameter s.

α	s	MGcgM-3+MG(s)	MLV-CSCOM-3A+MG(s)	MLV-CSCOM-3A
10^{-5}	10	19	11	12
10^{-4}	10	18	11	12
10^{-3}	10	18	11	11
10^{-2}	5	14	11	10
10^{-2}	10	16	11	-
10^{2}	5	28	28	20
10^{2}	10	31	19	-
10^{3}	10	81	54	62
10^{3}	20	102	59	-
10^{4}	10	176	131	152
10^{4}	20	207	124	-
10^{5}	10	331	201	271
10^{5}	15	399	256	-
10^{5}	20	391	261	-
10^{6}	10	498	322	377
10^{6}	20	591	303	-
10^{6}	30	541	257	-

Table 5: **Example 1**, number of iterations required to achieve $\| b - Ax^k \|_2 < 10^{-9}$ for different values of the parameters α and s.

To complete our discussion, let us look at the results for Example 1 in Table 4 and 5. We observe that the MGcgM-1+MG(s) has an inferior performance compared to the MGcgM-3+MG(s), MLV-CSCOM-3A, and MLV-CSCOM-3A+MG(s), also for extremely small values of α (10^{-2} to 10^{-5}).

Bibliography

[1] S.F. Ashby, R. D. Falgout, S. G. Smith, and A. F. B. Tompson. The Parallel Performance of a Groundwater Flow Code on the Cray T3D. Technical report, Lawrence Livermore National Laboratory.

[2] Tony F. Chan and W.L. Wan. Robust Multigrid Methods for Nonsmooth Coefficient Elliptic Linear Systems. *Journal of Computational and Applied Mathematics*, 123:323–352, 2000.

[3] Björn Engquist and Erding Luo. Convergence of the multigrid method with a wavelet coarse grid operator.

[4] 2D-EXPDE, A C++ Expression Templates Based Scientific Library. WWW page. http://www10.informatik.uni-erlangen.de/ pflaum/expde/public_html/index.html/.

[5] Luc Giraud, Julien Langou, and Miroslav Rozložńki. On the Loss of Orthogonality in the Gram-Schmidt Orthogonalization Process. Technical report, CERFACS.

[6] Thomas Y. Hou and Xiao-Hui Wu. A Multiscale Finite Element Method for Elliptic Problems in Composite Materials and Porous media. *Journal of Computational Physics*, 134(CP975682):169–189, 1997.

[7] Claes Johnson. *Numerical Solution of Partial Differential Equations by the Finite Element Method*. Cambridge University Press, Cambridge, 1995.

[8] C. Pflaum. A Multigrid Conjugate Gradient Method. To appear in Elsevier Science, 2006.

[9] C. Pflaum. Multigrid Methods, 2006. Lecture Notes, Universität Erlangen-Nürnberg.

[10] C. Pflaum. Simulation und Wissenschaftliches Rechnen (SiwiR), 2006. Lecture Notes, Universität Erlangen-Nürnberg.

[11] Gilbert Strang. *Linear Algebra And Its Applications*. Harcourt Brace Jovanovich Publishers, San Diego, thrid edition edition, 1988.

[12] Ulrich Trottenberg, Cornelis Oosterlee, and Anton Schüller. *Multigrid*. Academic Press, San Diego, 2001.

[13] Wing Lok Wan. Interface preserving coarsening multigrid for elliptic problems with highly discontinuous coefficients. *Numerical Linear Algebra with Applications*, 7:727–741, 2000.

[14] Jinchao Xu. The Method of Subspace Corrections. *Journal of Computational and Applied Mathematics*, 128:335–362, 2001.