Smoothed Aggregation for Nonsymmetric Linear Systems

Siobhan Correnty

Master's thesis
2019:E45

Lund University

Faculty of Science
Centre for Mathematical Sciences
Numerical Analysis

Centrum Scientiarum Mathematicarum

"I ask no favor for my sex; all I ask of our brethren is that they take their feet off our necks."

Ruth Bader Ginsburg
Honorary Doctor, Lund University

Abstract

Smoothed Aggregation (SA) is a technique from algebraic multigrid methods (AMG) which has shown very promising results when solving systems of linear equations with symmetric system matrices, but suboptimal results for nonsymmetric system matrices. The purpose of this thesis is to investigate the method EMIN(r) and a closely related method EMIN proposed by Sala and Tuminaro. EMIN(r) and EMIN were built on the success of SA in symmetric systems, but perform better in nonsymmetric systems. These methods have a more costly set-up, but promising results were found, in particular when performing one time-step of linear advection at extremely high CFL numbers.

Populärvetenskaplig sammanfattning

I naturen förekommer många differentialekvationer, som man inte kan lösa exakt, men som man önskar kunna lösa approximativt. Många verktyg har utvecklas för just detta, men man måste diskretisera och lösa system av linjära ekvationer för att få igång metoden. Det här är en väldigt omfattande uppgift när systemen är stora. I det här examensarbetet har vi arbetat med en sorts iterativ method för att lösa de stora systemen av linjära ekvationer, som kommer från diskretiseringen av partiella differentiella ekvationer (PDEs).

Methoden heter multigrid och bakgrunden till den är att man delar upp problemet i två delar: 'smooth' och 'oscillatory'. Först använder man en 'smoother' (t.ex. några iterationer av Jacobi) som tar bort de 'oscillatory' felvärdena. Sedan löser man ett system, som är lika originalet men mindre, och därefter interpolerar man lösningen av det mindre problemet för att få en bättre approximation till det stora. Att lösa ett mindre problem tar bort 'smooth' felvärdena.

Om man bara använder del ett, är metoden för långsam. Om man bara använder del två, konvergerar metoden inte. Men, om man använder båda tillsammans, blir metoden kraftig och snabb.

I den här uppsatsen experimenterar vi med 'smoothed aggregation', som är en variant av den traditionella multigridmetoden. Man använder lite extra 'smoothing' när man minskar problemet och igen när man gör det stort igen. Det fungerar väldigt bra när man löser ett system av linjära ekvationer med en symmetrisk positiv definit (SPD) systemmatris. Tyvärr går det inte lika bra när systemmatrisen inte är SPD. Vi tittar också på en variant av 'smoothed aggregation' som fungerar bra när systemmatrisen inte är SPD (EMIN(r)).

Acknowledgements

To my advisor Associate Professor Philipp Birken and co-supervisor Lea Versbach for introducing me to these fascinating topics and making time for me every week. Your feedback helped me cross the finsh line and decide the right next step.

To Professor Claus Führer, for showing me the joy in mathematics and making me want to know more.

To Associate Professor Carmen Arévalo and Professor Gustaf Söderlind for their encouragement and care, and filling me with positivity.

To my high school math teachers, Ms Harrigan and Mr Schwartz, for making math class so fun and giving me a strong mathematical foundation.

To my math sisters: Chiharu Kazama, Nadia Gyllenör and Sadia Asim, for understanding every emotion I have had during these last few years and always being available for me. I knew I would meet new friends when I started these studies, but I never dreamed of meeting friends as lovely as you ladies, from backgrounds so different from my own.

To my friend Anna Hirst, for always keeping my spirits up.

To Eva, Stefan and Seymour Aronson, for giving me a Swedish family and being so encouraging of this project, from start to finish. Thank you also for helping me to learn Swedish.

To Matt, my other half, for moving to Sweden to join me on this adventure, making each day here fun. Thank you for celebrating each victory I had on this journey and laughing with me every day.

To my brother Dan, for showing me how to balance academic success with fun.

To my beautiful mother, for filling my life with books, adventures and wild craic. Thank you for insisting that I finish this degree.

And to my father, my first teacher in this world, who taught me to work hard and be kind, and who would have loved this so much.

Contents

Chapter 1

Introduction

1.1 Opening Remarks

Partial differential equations (PDEs) can be used to describe a wide variety of phenomena such as sound, heat, diffusion, electrostatics, electrodynamics, fluid dynamics, elasticity, or quantum mechanics [18]. Many PDEs cannot be solved with pen and paper, thus many methods have been developed to approximate these equations numerically to a very high degree of accuracy. In doing so, we take a continuous problem and discretize it. When solving linear PDEs numerically, we often have to solve very large systems of linear equations, and thus we are interested in ways to solve these systems efficiently. Since Gaussian elimination would be far too slow, we consider iterative methods.

One iterative method for solving these systems of equations is multigrid method. Multigrid is a very effective method for solving linear systems of equations when the system matrix is symmetric and sparse. Multigrid schemes require $\mathcal{O}(n)$ operations for sparse systems with n entries. Convergence rates as impressive as 0.1 have been observed. Multigrid also gives rise to good preconditioners for GMRES, which can be useful as a subsolver for Newton's method for the linear and nonlinear equation systems which arise from discretized PDEs [1]. A variant of multigrid, Full Approximation Scheme, can be used to solve nonlinear systems of equations, but we will not explore this.

This thesis explores some modifications to multigrid in order to solve linear systems associated with both symmetric and nonsymmetric system matrices more effectively. We will experiment with different methods for restriction and prolongation which contain extra smoothing, then test the performance of these with different pre- and post-smoothers on a variety of problems in 1D and 2D.

The aim of this thesis is to understand the results of Sala and Tuminaro in [9], and to find effective ways to solve linear equation systems with nonsymmetric system matrices. We will explain the theory in an intuitive way, and then show results of many numerical experiments which illustrate the theory.

1.2 Overview

Chapter 2 describes the equations considered throughout this thesis, and then Chapter 3 shows our discretizations of these equations. Chapter 4 provides a background on multigrid methods and Chapter 5 provides a thorough explanation of the method Smoothed Aggregation. Chapter 6 presents results of Smoothed Aggregation along with some closely related methods, and will explain the results. Chapter 7 presents a method which gener-

alizes Smoothed Aggregation for nonsymmetric system matrices, and shows how to set up this method. Chapters 8 and 9 explain how to implement these methods within a multigrid scheme, and present and explain results from testing these methods. Chapter 10 includes an experiment with one of the user-defined parameters. Chapter 11 offers some concluding remarks and ideas for future work, and The Appendix offers derivations of the different iterative methods (smoothers) used throughout this thesis. One can also find pseudo code in this section.

Chapter 2

Equations

In this chapter, we will provide background information on the three equations studied throughout this thesis. We will start with the Poisson equation, then discuss linear advection and finish with information on convection diffusion.

2.1 Poisson Equation

We begin by considering the Poisson equation and the closely related Laplace equation [5].

The Laplace equation is given by

$$\Delta u = 0$$

and the Poisson equation by

$$-\Delta u = f.$$

The Poisson equation arises, for example, when describing the potential field caused by a given charge or mass density distribution. With the potential field f known, one can then calculate the gravitational or electrostatic field u [19]. The Laplace equation can be used, for example, to model the steady-state heat equation [17].

In the two above equations, $x \in U$ and the unknown $u : \bar{U} \to \mathbb{R}, u = u(x)$, where $U \subset \mathbb{R}$ is a given, open set.

For the Laplace equation, the function $f : U \to \mathbb{R}$ is also given.

Definition:
The function

$$\Phi(x) := \begin{cases} -\frac{1}{2\pi} \log ||x|| & m = 2 \\ \frac{1}{m(m-2)\alpha(m)} \frac{1}{||x||^{m-2}} & m \geq 3, \end{cases}$$

defined for $x \in \mathbb{R}^m, x \neq 0$ is the fundamental solution to Laplace's equation. We note here that $\alpha(m)$ denotes the volume of the unit ball in $\mathbb{R}^m$.

Theorem: (Solving the Poisson equation)
Assume that $f \in C_c^2(\mathbb{R}^m)$; that is f is twice continuously differentiable with compact support.

Define u by

$$u(x) = \int_{\mathbb{R}^m} \Phi(x-y) f(y) dy = \begin{cases} -\frac{1}{2\pi} \int_{\mathbb{R}^2} \log (||x-y||) f(y) dy & m = 2 \\ \frac{1}{m(m-2)\alpha(m)} \int_{\mathbb{R}^m} \frac{f(y)}{||x-y||^{m-2}} dy & m \geq 3. \end{cases} \tag{2.1}$$

Then,

$$u \in C^2(\mathbb{R}^m)$$

and

$$-\Delta u = f \text{ in } \mathbb{R}^m.$$

So, the convolution formulas (2.1) provide us with a solution to the Poisson equation in $\mathbb{R}^m$. We take a second here to note that we do not need to address boundary conditions when using the convolution formula [16].

Following this logic, we choose $f \in C_c^2(\mathbb{R}^m)$ throughout this thesis. The problem we solve is:

$$\begin{cases} -\Delta u = 4\pi^2 \sin \pi x^2 & x \in (0,1) \\ u(x) = 0 & x = 0, x = 1, \end{cases}$$

the Poisson equation in 1D.

2.2 Linear Advection

The linear advection equation is sometimes called the transport equation because it describes taking some initial condition and moving it at a constant speed along characteristic lines in the time-space plane.

More generally [5], we have

$$u_t + a \cdot Du = 0 \text{ in } \mathbb{R}^m \times (0, \infty)$$

where a is a fixed vector in $\mathbb{R}^m$, $a = (a_1, ..., a_n)$ and $u : \mathbb{R}^m \times [0, \infty) \to \mathbb{R}$ is the unknown $u = u(x, t)$.

$Du = D_x u = (u_{x_1}, ..., u_{x_n})$ is the gradient of u with respect to the spatial variables.

In this thesis, we will consider the linear advection equation in 1D, i.e.

$$u_t + a u_x = 0$$

with initial condition

$$u(x, 0) = u_0(x) = \sin(\pi x)$$

on the domain $(0, 2]$, which has the analytical solution $u(x, t) = u_0(x - at)$. This solution reflects that we are transporting the initial values at a constant velocity a to the right if $a > 0$ and to the left if $a < 0$. In this thesis, we will consider only positive values for a.

2.3 Convection Diffusion

We consider also the equation for convection diffusion (1D),

$$\begin{cases} -\varepsilon \Delta u + b \nabla u = f & x \in (0,1) \\ u = 0 & x = 0, x = 1, \end{cases}$$

and (2D)

$$\begin{cases} -\varepsilon \Delta u + b \cdot \nabla u = f & x \in (0,1) \times (0,1) \\ u = 0 & \text{on the boundaries,} \end{cases}$$

where we take $b = 1$ in 1D and the vector fields 'bent pipe' (3.4) or 'recirc' (3.5) in 2D, which we plot in the following chapter. We will vary ε. We choose the right-hand side so that the solution u is given by $u(x) = \sin^2(\pi x)$ in 1D and $u(x, y) = \sin^2(\pi x)\sin^2(\pi y)$ in 2D. A smaller value for ε corresponds to a more convective (nonsymmetric) system, because the diffusive term is less dominant. Many flow problems (in water and air) are characterized by very small values of epsilon, so this problem is a practical one to solve. On the other hand, a larger value for ε corresponds to a more diffusive (symmetric) system. We consider $\varepsilon = 10^{-1}$ a large value for epsilon, and $\varepsilon = 10^{-5}$ a small value for epsilon.

More generally [15], the convection diffusion equation is

$$\frac{\partial c}{\partial t} = \nabla \cdot \big(D \nabla c\big) - \nabla \cdot (\mathbf{v} c) + R,$$

where $\nabla \cdot (D \nabla c)$ represents the diffusive term (divergence of the gradient), $\nabla \cdot (\mathbf{v} c)$ is the velocity field (convective term) and R describes sources or sinks.

Chapter 3

Discretizations

In this chapter, we discuss the discretizations which are used throughout this thesis. As stated in the introduction, we will take the continuous equations we considered in the previous chapter and solve these equations numerically to a very high degree of accuracy. To achieve this goal, we must begin with discretizations. For the Poisson equation and convection diffusion equations, we need only to consider space discretizations, since we have only space derivatives. When solving linear advection, we must account for both time and space derivatives, and hence the need for time and space discretizations.

We note here that, in 1D, we will always choose the number of unknowns m to be a power of 2 (i.e. $m = 2^n, n = 1, 2, ...$). In 2D, we will always choose m to be a perfect square which is also divisible by 9^l, where l represents the finest multigrid level (see Chapter 4).

3.1 Space Discretization for the Poisson Equation

The system matrix (i.e. discrete differential operator) for the discretized Poisson equation is symmetric positive definite. This is an important property which we will return to many times throughout this thesis.

Definition: Symmetric positive definite (SPD)
A symmetric matrix $A \in \mathbb{R}^{m \times m}$ is called SPD if

$$x^{\mathsf{T}} A x > 0$$

for all $x \in \mathbb{R}^m \neq 0$.

We use second order central differences (finite differences) with $m+2$ points for the Laplace operator with Dirichlet boundary conditions on an equidistant grid with domain $x \in [0, 1]$, i.e. we solve the system

$$\frac{1}{\Delta x^2} \begin{pmatrix} 2 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{pmatrix} \begin{pmatrix} u_1 \\ \vdots \\ \vdots \\ u_m \end{pmatrix} = \begin{pmatrix} 4\pi^2 \sin(\pi x_1^2) \\ \vdots \\ \vdots \\ 4\pi^2 \sin(\pi x_m^2) \end{pmatrix}, \tag{3.1}$$

where $u_j \approx u(x_j)$, and set $u_0 = u(0) = 0$ and $u_{m+1} = u(1) = 0$. A concrete example of an equidistant grid with $m = 8$ unknowns appears in Figure 3.1.

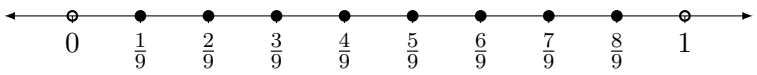


Figure 3.1: $m = 8$ unknowns on a 1D grid, $\Delta x = \frac{1}{m+1} = \frac{1}{9}$

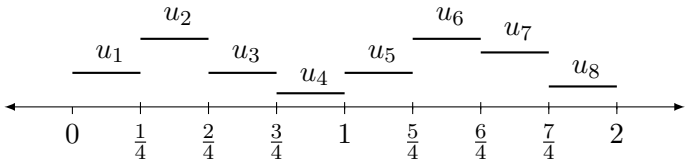


Figure 3.2: $m = 8$ unknowns on a 1D grid $x \in (0, 2]$, $\Delta x = \frac{2}{m} = \frac{1}{4}$

3.2 Time and Space Discretization for Linear Advection

Here we use a first-order finite volume discretization with upwind flux on an equidistant grid to perform one time-step of linear advection. We choose the implicit Euler method for the time-stepping because of the stability properties of this method. Since solutions are transported along characteristic lines and we consider only $a > 0$, we must use an upwind scheme, meaning we go forward in time and backward in space. Our approximations to the derivatives are then as follows:

$$\frac{\partial u}{\partial t}(x_j, t_n) = \frac{u(x_j, t_{n+1}) - u(x_j, t_n)}{\Delta t} + \mathcal{O}(\Delta t)$$

$$\frac{\partial u}{\partial x}(x_j, t_n) = \frac{u(x_j, t_n) - u(x_{j-1}, t_n)}{\Delta x} + \mathcal{O}(\Delta x).$$

Since we use the implicit Euler method, we must solve a system of linear equations on each time-step. This translates into the following system for $j = 1, ..., m$.

$$\begin{aligned}
\frac{u_j^{n+1} - u_j^n}{\Delta t} + a\frac{u_j^{n+1} - u_{j-1}^{n+1}}{\Delta x} = 0 \iff \\
u_j^{n+1} - u_j^n + a\frac{\Delta t}{\Delta x}(u_j^{n+1} - u_{j-1}^{n+1}) = 0 \iff \\
u_j^{n+1} + a\frac{\Delta t}{\Delta x}u_j^{n+1} - a\frac{\Delta t}{\Delta x}u_{j-1}^{n+1} = u_j^n \iff \\
\begin{pmatrix}
(1 + a\frac{\Delta t}{\Delta x}) & 0 & & -a\frac{\Delta t}{\Delta x} \\
-a\frac{\Delta t}{\Delta x} & \ddots & \ddots & \\
 & \ddots & \ddots & 0 \\
 & & -a\frac{\Delta t}{\Delta x} & (1 + a\frac{\Delta t}{\Delta x})
\end{pmatrix}
\begin{pmatrix} u_1^{n+1} \\ \vdots \\ \vdots \\ u_m^{n+1} \end{pmatrix} =
\begin{pmatrix} u_1^n \\ \vdots \\ \vdots \\ u_m^n \end{pmatrix},
\end{aligned} \tag{3.2}$$

where $u_j^n \approx u(x_j, t_n)$. In this thesis we take initial condition $u_j^0 = u(x_j, 0) = \sin(\pi x_j)$ on the domain $(0, 2]$.

The top right entry in the system matrix reflects the periodic boundary conditions, i.e. $u_0^k = u_m^k$ for all time-steps $k = 0, 1, 2....$ Because we are using a finite volume discretization, our solutions represent piecewise constant approximations on a cell, i.e. we have solution u_i for all $x \in [x_i, x_{i+1}]$, where $x_j \in [0, 2]$, $j = 1, ..., m+1$ and $\Delta x = \frac{2}{m}$. A concrete example of these cells with $m = 8$ unknowns is given in Figure 3.2.

3.3 Space Discretization for Convection Diffusion

3.3.1 Convection Diffusion in 1D

To form our (finite differences) discretization in the same way Sala and Tuminaro do in [9], we use second-order central differences for the diffusive term (i.e. $-\varepsilon\Delta u$) and a first-order upwind difference for the convective term (i.e. $b \cdot \nabla u$). This corresponds to

$$\frac{\varepsilon}{\Delta x^2}\begin{pmatrix} 2 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{pmatrix}\begin{pmatrix} u_1 \\ \vdots \\ \vdots \\ u_m \end{pmatrix} + \frac{b}{\Delta x}\begin{pmatrix} 1 & & & \\ -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \end{pmatrix}\begin{pmatrix} u_1 \\ \vdots \\ \vdots \\ u_m \end{pmatrix} = \begin{pmatrix} f_1 \\ \vdots \\ \vdots \\ f_m \end{pmatrix}$$

$$\Longleftrightarrow$$

$$\begin{pmatrix} (\frac{2\varepsilon}{\Delta x^2} + \frac{b}{\Delta x}) & -\frac{\varepsilon}{\Delta x^2} & & \\ (-\frac{\varepsilon}{\Delta x^2} - \frac{b}{\Delta x}) & \ddots & \ddots & \\ & \ddots & \ddots & -\frac{\varepsilon}{\Delta x^2} \\ & & (-\frac{\varepsilon}{\Delta x^2} - \frac{b}{\Delta x}) & (\frac{2\varepsilon}{\Delta x^2} + \frac{b}{\Delta x}) \end{pmatrix}\begin{pmatrix} u_1 \\ \vdots \\ \vdots \\ u_m \end{pmatrix} = \begin{pmatrix} f_1 \\ \vdots \\ \vdots \\ f_m \end{pmatrix}, \tag{3.3}$$

where $u_j \approx u(x_j)$, $f_j = f(x_j)$ and the right-hand side of the equation, f, is chosen so that the exact solution to the linear system is given by $u = \sin^2(\pi x)$, i.e.

$$f = \begin{pmatrix} -\varepsilon\Big(2\pi^2\big(\cos^2(\pi x_1) - \sin^2(\pi x_1)\big)\Big) + b\Big(2\pi\sin(\pi x_1)\cos(\pi x_1)\Big) \\ \vdots \\ \vdots \\ -\varepsilon\Big(2\pi^2\big(\cos^2(\pi x_m) - \sin^2(\pi x_m)\big)\Big) + b\Big(2\pi\sin(\pi x_m)\cos(\pi x_m)\Big) \end{pmatrix}.$$

Since we apply Dirichlet boundary conditions here, we have $u_0 = u_{m+1} = 0$. The grid in 1D is the same as the one we use for the Poisson equation, as shown in Figure 3.1.

3.3.2 Convection Diffusion in 2D

We consider the 2D convection diffusion equation,

$$\begin{cases} -\varepsilon\Delta u + b \cdot \nabla u = f & \text{in } (0,1) \times (0,1) \\ u = 0 & \text{on the boundaries} \end{cases}$$

with the two vector fields below.

$$\underline{\textbf{Bent pipe}}\textbf{:}\ b = \Big(2x(\frac{1}{2}x - 1)(1 - 2y), -4y(y-1)(1-x)\Big) \tag{3.4}$$

$$\underline{\textbf{Recirc}}\textbf{:}\ b = \Big(4x(x-1)(1-2y), -4y(y-1)(1-2x)\Big) \tag{3.5}$$

We discretize this with the following stencil:

$$\frac{1}{h^2}\begin{bmatrix} & \frac{1}{2}h(b_2 - |b_2|) - \varepsilon & \\ -\frac{1}{2}h(b_1 + |b_1|) - \varepsilon & h(|b_1| + |b_2|) + 4\varepsilon & \frac{1}{2}h(b_1 - |b_1|) - \varepsilon \\ & -\frac{1}{2}h(b_2 + |b_2|) - \varepsilon & \end{bmatrix}, \tag{3.6}$$

where b_1 refers to the first component of the vector field, and b_2 refers to the second. This represents a second-order central differences discretization for the diffusive term and a

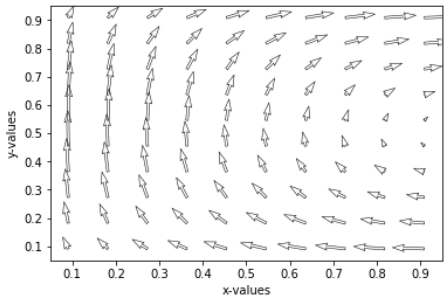


Figure 3.3: Bent Pipe

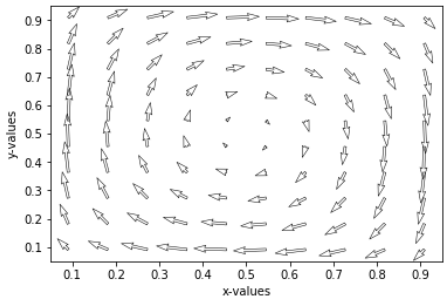


Figure 3.4: Recirc

first-order upwind discretization for the convective term, as done by Sala and Tuminaro in [9]. We use the term h here to signify that $\Delta x = \Delta y$, i.e. an equidistant grid in both directions. In this case, $h = \frac{1}{\sqrt{m}+1}$. To see what this looks like, consider Figure 3.5 with $m = 16$.

This discretization was chosen for its stability properties. Since we will be considering small values for ε, this discretization is a reasonable choice. We note, however, that this discretization is only $O(h)$ and not sufficient for many engineering applications [11].

Similar to our case in 1D, f is chosen such that the solution to our linear system is given by

$$u(x, y) = \sin^2(\pi x) \sin^2(\pi y).$$

This function is zero on the boundary.

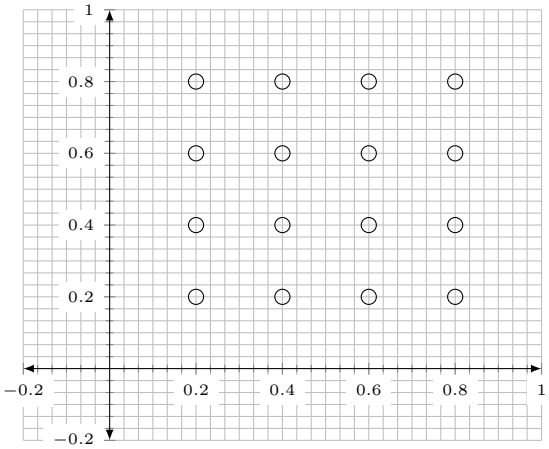


Figure 3.5: $m = 16$ unknowns on a 2D grid, $h = dx = dy = \frac{1}{\sqrt{m}+1} = \frac{1}{5}$

Chapter 4

Background on Multigrid Methods

Multigrid methods are designed for solving sparse linear systems that arise from discretized partial differential equations. These methods can solve many such systems to a high degree of accuracy in a fixed number of iterations, not growing in m, and have shown to be especially successful in solving symmetric systems [3].

To see how multigrid methods work, we take for example the Poisson equation in 1D with Dirichlet boundary conditions, i.e.

$$\begin{cases} -\Delta u = f(x), \ x \in [0,1], \ f \in \mathrm{C}([0,1], \mathbb{R}) \\ u(x) = 0, \ x = 0, x = 1. \end{cases}$$

The system matrix A which arises from the discretization (3.1) (i.e. of the discrete Laplace operator with Dirichlet boundary conditions) has m eigenvalues

$$\lambda_j^{\text{Laplace}} = \frac{4}{\Delta x^2} \sin^2\left(\frac{\pi j}{2(m+1)}\right), \ j = 1, ..., m \tag{4.1}$$

and m (scaled) eigenvectors

$$v_j^{\text{Laplace}} = \sqrt{2\Delta x} \begin{pmatrix} \sin(j\pi x_1) \\ \vdots \\ \sin(j\pi x_m) \end{pmatrix}, \ j = 1, ..., m \tag{4.2}$$

where $x_i \in (0,1)$ for $i = 1, ..., m$, $\Delta x = \frac{1}{m+1}$ [3].

4.1 Jacobi Smoothing

Multigrid methods start with pre-smoothing iterations via a linear iterative method. By smoothing, we mean a method to reduce the oscillatory error.

Definition: (Linear iterative method)
A linear iterative method for the solution of $Ax = b$, $A \in \mathbb{R}^{m \times m}$ is of the form

$$x_{k+1} = Mx_k + N^{-1}b.$$

We call M in the previous definition the iteration matrix.

Theorem:
A linear iterative method is convergent to limit $\bar{x}$ independent of x_0 iff $\rho(M) < 1$.

Thus, if $\rho(M) < 1$ we get convergence to the unique solution of

$$x = Mx + N^{-1}b.$$

This is the solution to $Ax = b$ when $x^* = A^{-1}b$. For this to happen, we must have a consistent method, i.e.

$$\begin{aligned} Mx^* + N^{-1}b = x^* &\iff \\ (M + N^{-1}A - I)x^* = 0 &\iff \\ M = I - N^{-1}A. \end{aligned}$$

Thus, any iteration with $\rho(M) < 1$ and $M = I - N^{-1}A$ for $N^{-1} \in \mathbb{R}^{m\times m}$ converges to $x^* = A^{-1}b$.

In this thesis, we use a few different linear iterative methods for smoothing (e.g. Jacobi, Gauss-Seidel, etc.). In the Appendix, we show how these methods are derived.

For now, we consider one possibility, Weighted Jacobi smoothing. Such an iteration is given by

$$x_{k+1} = x_k - wD^{-1}Ax_k + wD^{-1}b,$$

where $M = (I - wD^{-1}A) \in \mathbb{R}^{m\times m}$. The m eigenvectors of M are the same as those of A given in (4.2). The m eigenvalues of M are given by:

$$\lambda_j^M = 1 - \frac{w\Delta x^2}{2}\lambda_j^{\text{Laplace}} = 1 - 2w\sin^2\left(\frac{\pi j}{2(m+1)}\right) < 1,$$

for $w \in (0, 2)$, $j = 1, ..., m$. From the Theorem, we know Weighted Jacobi converges to the solution to $Ax = b$.

Consider now the error, i.e. $x_{k+1} - x^*$:

$$\begin{aligned} x_{k+1} - x^* &= Mx_k + wD^{-1}b - x^* + wD^{-1}Ax^* - wD^{-1}b \\ &= M(x_k - x^*) \\ &= M^{k+1}(x_0 - x^*) \\ &= \sum_{j=1}^{m} \alpha_j(\lambda_j^M)^{k+1}v_j. \end{aligned} \tag{4.3}$$

So, we can express the error as a sum of linearly independent eigenvectors and, using scaled eigenvalues as coefficients. We refer to each summand in (4.3) as an error component.

We consider now some plots (spectrum) of the m eigenvalues of M for different values of w in Figure 4.1. This represents changing the damping parameter within the Weighted Jacobi iteration. The leftmost eigenvalue in each plot represents λ_1^M and the rightmost represents λ_m^M, with λ_j^M, for $j = 2, ..., m-1$, in order, in between. The eigenvalues of low indices correspond to the slowly oscillating sine functions and the eigenvalues of high indices represent the high frequency sine waves. We refer to these terms as the low frequency and high frequency error components, respectively.

From these plots, we can see that for $w = \frac{2}{3}$, the larger half of the eigenvalues are all quite close to 0. In fact, we have

$$|\lambda_j^M| < \frac{1}{3}, \text{ for all } \frac{m+1}{2} \leq j \leq m.$$

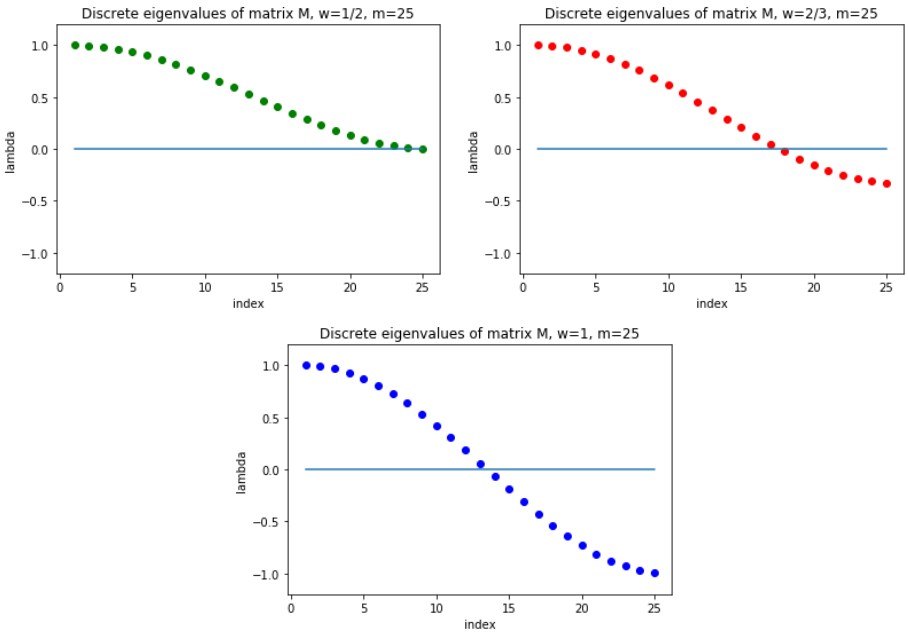


Figure 4.1: Eigenvalues for Different w

We take $w = \frac{2}{3}$ and, as a result, damp the half of the error components associated with high frequency sine functions. In fact, these oscillatory components are reduced at least by a factor of 3 with each relaxation, independent of the grid width. We call this part of our method pre-smoothing [3].

Weighted Jacobi smoothing does a great job at eliminating high frequency error, but

$$\rho(M) \to 1, \mathcal{O}(\Delta x^2).$$

So, as we refine the grid, Weighted Jacobi will become less effective, especially at eliminating low frequency error components. A similar result follows from the other linear iterative methods considered in this thesis.

4.2 Coarse Grid Correction

Now, we concern ourselves with the low frequency error components. The main idea here is to halve the dimension of our problem and solve the defect equation, i.e.

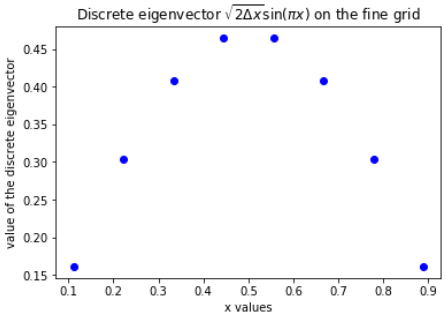
$$Ae = A(x^* - x_{k+1}) = b - Ax_{k+1} = r,$$

on a coarse grid, where this is less computational work. If we solved the defect equation on the fine grid, then we would have the solution to our original equation, but this would involve the same amount of work as solving the original problem $Ax = b$ on the fine grid, which is what we want to avoid. We instead solve the defect equation on the coarse grid, then interpolate this vector to the fine level and update our approximate solution.

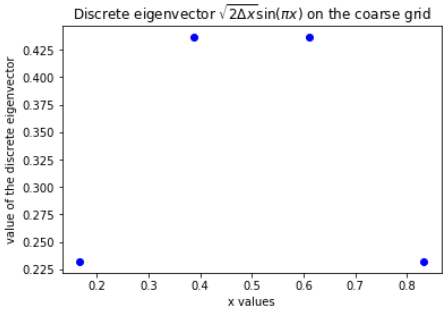
For dimension $m = 8$, consider a discretization of the lowest frequency eigenvector (error component) of the Laplace operator on $[0, 1]$.

$$v_1 = \sqrt{2\Delta x}\sin(\pi x) \approx \begin{pmatrix} \sqrt{2\Delta x}\sin(\pi x_1) \\ \vdots \\ \vdots \\ \sqrt{2\Delta x}\sin(\pi x_8) \end{pmatrix} \in \mathbb{R}^8,$$

where $x_i \in (0,1)$, $i = 1, ..., 8$, $\Delta x = \frac{1}{9}$. A plot of this follows in Figure 4.2a.



(a) Lowest frequency error component v_1, $m = 8$ Fine level



(b) $v_0 = R_1^0 v_1$ $m = 4$ Coarse level

Figure 4.2: Halving the dimension

Now, consider the restriction operator

$$R_1^0 = \frac{1}{2}\begin{pmatrix} 1 & 1 & & & & & & \\ & & 1 & 1 & & & & \\ & & & & 1 & 1 & & \\ & & & & & & 1 & 1 \end{pmatrix} \in \mathbb{R}^{4\times 8},$$

which halves the dimension. In the case of having just a fine grid and a coarse grid, one moves from the fine grid (level 1, $\mathbb{R}^8$) to the coarse grid (level 0, $\mathbb{R}^4$) by multiplying from the left with the restriction operator R_1^0. In general, the operator R_l^{l-1} takes us from level l to level $l-1$. To see what our sine function looks like on the coarse grid consider

$$R_1^0 v_1 = \frac{\sqrt{2\Delta x}}{2}\begin{pmatrix} (\sin(\pi x_1) + \sin(\pi x_2)) \\ (\sin(\pi x_3) + \sin(\pi x_4)) \\ (\sin(\pi x_5) + \sin(\pi x_6)) \\ (\sin(\pi x_7) + \sin(\pi x_8)) \end{pmatrix} \in \mathbb{R}^4.$$

The plot of this discretized sine function follows in Figure 4.2b.

On the coarse grid, our discretized sine function goes through half a period in 4 grid points, where on the fine grid this took 8 grid points. What we observe here is that the slowly oscillating error components appear half as smooth on the coarse grid. It is here on the coarse grid where we will correct them by solving $A_0 e_0 = r_0$. This is what is meant by the coarse grid correction. We note that in the experiments which follow, we omit the factor $\frac{1}{2}$ in front of the restriction operator, as done by Sala and Tuminaro in [9], unless otherwise noted.

We can write the coarse grid correction as an iterative method. This corresponds to:

$$x_{k+1} = (I - P_0^1 A^{-1}{}_0 R_1^0 A_1)x_k + P_0^1 A^{-1}{}_0 R_1^0 b_1,$$

where $P_0^1 = 2(R_1^0)^\mathsf{T}$, which represents interpolating from the coarse level to the fine level.

We can actually prove this coarse grid correction is not a convergent method. However, in combination with a smoother (such as Weighted Jacobi), we can build very powerful methods to solve linear systems. Thus we have all the tools we need in order to understand multigrid schemes.

4.3 2-Grid Method

The two grid method consists of combining the tools we have just discussed. The pseudo code follows in Algorithm 1. We note that in all of our methods, we use a Galerkin projection for the coarse grid discretization of the system matrix, i.e.

$$A_{l-1} = R_l^{l-1} A_l P_{l-1}^l \in \mathbb{R}^{\frac{m}{2} \times \frac{m}{2}},$$

where $A_l \in \mathbb{R}^{m \times m}$.

Throughout this thesis, we will sometimes use the notation $A_c = R_1^0 A_f P_0^1$, where A_f represents the system matrix on the fine level and A_c represents the system matrix on the coarse level.

Algorithm 1 2-Grid Method

Jacobi pre-smooth ν_1 times: $A_1 x_1 = b_1$
Calculate the residual: $r_1 = A_1 x_1 - b_1$
Restrict the residual: $r_0 = R_1^0 r_1$
Solve the defect equation for e_0: $A_0 e_0 = r_0$
Prolong this to correct the approximation to the solution: $x_1 = x_1 + P_0^1 e_0$
Jacobi post-smooth ν_2 times: $A_1 x_1 = b_1$

4.4 V/W Cycles

In the case of large scale systems, the coarse grid problem may be too large to solve directly. What we consider now is defining a sequence of coarser and coarser grids. We do this so we can use `solve` on a much coarser grid, which takes much less work.

To see this, consider the pseudo code in Algorithm 2 for multigrid V/W cycles (below) and Figures 4.3 and 4.4, which follow.

Algorithm 2 Multigrid Method, V/W cycles

function MG(x_l,b,l)
if $l = 0$ **then**
  Solve $A_0 x_0 = b_0$ directly
else
  Pre-smoothing: $x_l = S_l^{\nu_1}(x_l, b_l)$
  Restrict the residual: $r_{l-1} = R_l^{l-1}(A_l x_l - b_l)$
  Initial guess for e_{l-1}: $v_{l-1} = 0$
  Call Multigrid recursively: for $i = 1, ..., \gamma$ MG(v_{l-1}, r_{l-1}, $l-1$)
  Update the approximation on the fine grid: $x_l = x_l - P_{l-1}^l v_{l-1}$
  Post-smoothing: $x_l = S_l^{\nu_2}(x_l, b_l)$
end if

We note here that throughout this thesis, we will continue to iterate either the 2-grid scheme or a V/W scheme until

$$\frac{||Ax_k - b||_2}{||b||_2} < \text{tol}.$$

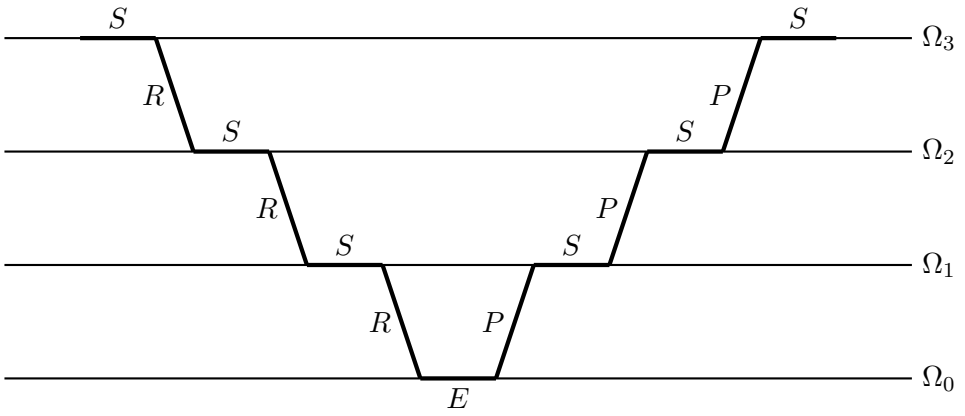


Figure 4.3: Visualization of V cycle, $\gamma = 1$
S refers to smoothing, E refers to solving (courtesy of Versbach)

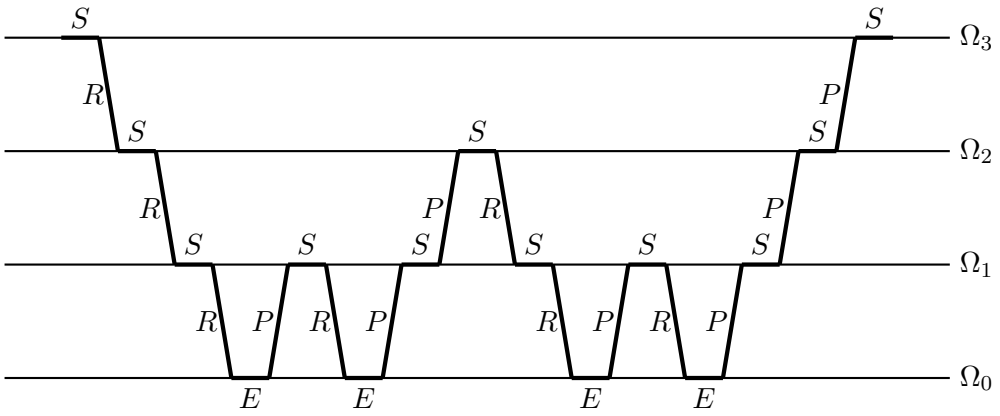


Figure 4.4: Visualization of W cycle, $\gamma = 2$
S refers to smoothing, E refers to solving (courtesy of Versbach)

Chapter 5

Smoothed Aggregation

5.1 Introduction to Smoothed Aggregation

Smoothed Aggregation (SA) is an Algebraic Multigrid (AMG) method which was designed for solving systems of linear equations with a symmetric positive definite system matrix. AMG methods automatically generate coarse levels and level transfer operators, though the coarse grids are unknown [9]. Throughout this thesis, we only consider Geometric Multigrid methods, where the grids are known. The same results we describe here hold in the case where the grid is known.

SPD systems have corresponding energy norms:

Definition: (Energy norm)
The energy norm of a vector x with respect to a symmetric positive definite (SPD) matrix A is given by

$$||x||_A = \sqrt{x^\mathsf{T} A x}.$$

In an SPD system, we can discuss high-energy and low-energy modes, which correspond to oscillatory and smooth error modes respectively [9]. Throughout this thesis, we will encounter some systems that are not SPD and thus, the energy norm is not defined. In this chapter, where we discuss only SPD systems, we will use the terms low-energy/ smooth and high-energy/ oscillatory interchangeable.

We begin by studying different methods for prolongation and restriction. The rate of convergence of an AMG method is strongly dependent on the choice of restriction and prolongation operators. A necessary condition for achieving high rate of convergence is that the prolongation operator contains in its range only vectors x such that $Ax \approx 0$, where A is the discrete differential operator. Such vectors x are called smooth [12]. By smooth, we imply that these vectors are smooth relative to the relaxation method, i.e. these are the vectors to which Jacobi smoothing does very little [7].

Smooth error components can be efficiently corrected on coarser levels. If we include them in the range space of the prolongator, they are represented throughout the mesh hierarchy [8] and rapid convergence is achieved [4]. SA can be thought of as a regular multigrid method with extra smoothing in prolongation and restriction.

In this chapter, we will first show how to form the prolongation and restriction operators for the method SA. We will then look at a concrete example to appreciate the effectiveness of the damping parameter defined by the method SA. After that, we will experiment with multiple iterations of Weighted Jacobi on the prolongation basis functions (i.e. columns of

the prolongator). We conclude this section with some error estimates from [12].

5.2 Prolongation Operator, SA

To create such a prolongation operator, we start by constructing a tentative prolongator matrix $P^{(\text{tent})}$ based on a user-defined parameter: the near kernel. For this we will follow the explanation from [9].

Definition: (Near kernel)
All vectors x such that $Ax \approx 0$, where A is the discrete differential operator.

The near kernel is an assumption about the problem rather than the use of geometrical information [14]. By this we mean the near kernel is based on the differential operator and not on the grid we are using. In practice, the near kernel will be a user-defined vector (or set of vectors). Throughout this thesis, we take the constant vector as our user-defined near kernel, unless otherwise specified. This is the standard choice of near kernel for the Laplace operator [9], but others are possible, namely the constant vector plus a linear term, which we will show in Chapter 10.

Consider a 2-grid method. In the method SA, we first construct the tentative prolongator from the constant vector in a way such that the near kernel is in the range of $P^{(\text{tent})}$. The constant vector is thus accurately interpolated from the coarse level to the fine level. We do this such that the columns of $P^{(\text{tent})}$ are linearly independent, i.e. we define $P^{(\text{tent})}$ as

$$P^{(\text{tent})} = \begin{pmatrix} 1 & & & & \\ 1 & & & & \\ & 1 & & & \\ & 1 & & & \\ & & & \ddots & \\ & & & & 1 \\ & & & & 1 \end{pmatrix} \in \mathbb{R}^{m \times \frac{m}{2}}. \tag{5.1}$$

Consider solving the discrete Poisson equation in 1D (3.1), where A is SPD. We can see these columns in (5.1) as sine waves which go through one period over one grid point. Unfortunately, the tentative prolongator columns have high energy (oscillatory sine waves). This implies that some smooth error components (low frequency sine waves) are not accurately represented throughout the mesh hierarchy even though the user-defined near kernel always is. One step of Weighted Jacobi (iteration matrix) applied to all columns of the tentative prolongator reduces the energy in each of the columns while preserving the interpolation of user-defined near kernel (far from boundaries). This is because the constant vector is still in the range of the prolongation operator, even after applying Weighted Jacobi. The result of applying Weighted Jacobi is the columns of P more closely span the smooth error subspace [8], and thus smooth error components (solved for on the coarse grid) can be better interpolated to the fine grid, where we can update our approximation to the solution. Thus, for the method SA, we take

$$P = (I - w^{(p)} D^{-1} A) P^{(\text{tent})},$$

where $w^{(p)}$ is the damping parameter and D is the diagonal of A.

To better understand this, we consider a more concrete example: a 2-grid method from

[13], where we seek the solution to $Ax = b$, and A is SPD.

We consider again our smoothing iterative method (e.g. Weighted Jacobi), which can be written as

$$J(x) = Mx + N^{-1}b,$$

where

$$M = (I - w^{(p)}D^{-1}A). \tag{5.2}$$

The error on the fine grid is defined as

$$e(x) = x - x^* \in \mathbb{R}^m.$$

Error components which are not efficiently resolved by smoothing, i.e. those such that

$$Me \approx e$$

are the smooth error components. These are the components we must resolve on the coarse grid.

What we seek now is a prolongation operator such that

$$e_1 = Pe_0,$$

so we can properly prolong the smooth error from the coarse grid (correction) and thus update the approximate solution on the fine grid, i.e.

$$x_1 = x_1 + Pe_0.$$

The subscript here refers to which level we are on, with subscript 1 referring to the fine grid and subscript 0 referring to the coarse grid.

In the ideal case, the range of P always contains all smooth components of an arbitrary error $e_1 \in \mathbb{R}^m$, but in practice this requirement is unnecessarily strong. Typically, a weaker condition is satisfactory, i.e. find a P such that

$$MPv \approx Pv. \tag{5.3}$$

Our matrix $P^{(\text{tent})}$ from (5.1) is very bad from the point of view of (5.3). We take instead our prolongation operator P to be

$$P = MP^{(\text{tent})},$$

which is very good from the point of view of (5.3), since one iteration of Weighted Jacobi is very effective in decreasing the energy of the columns of $P^{(\text{tent})}$, thus smoothing them. In a later section, we will experiment with applying multiple iterations of Weighted Jacobi to the columns of $P^{(\text{tent})}$.

5.3 Restriction Operator, SA

Since SA was developed for SPD systems, it is natural that we take our restriction matrix to be the transpose of our prolongation matrix. In other words,

$$R^{(\text{tent})} = (P^{(\text{tent})})^{\mathsf{T}} = \begin{pmatrix} 1 & 1 & & & & & \\ & & 1 & 1 & & & \\ & & & & \ddots & \ddots & \\ & & & & & & 1 & 1 \end{pmatrix} \in \mathbb{R}^{\frac{m}{2} \times m},$$

and

$$R = P^{\mathsf{T}} = (MP^{(\text{tent})})^{\mathsf{T}} = R^{(\text{tent})}M^{\mathsf{T}},$$

where M is defined as in (5.2).

5.4 Poisson Equation

Since SA was designed for SPD systems [9], we consider first our discretization of the Poisson equation, since the system matrix A defined in (3.1) is SPD. P and R as defined by the method SA are given by:

$$P = MP^{(\text{tent})} = \begin{pmatrix} 1-.5w^{(p)} & & & & \\ 1-.5w^{(p)} & .5w^{(p)} & & & \\ .5w^{(p)} & 1-.5w^{(p)} & & & \\ & 1-.5w^{(p)} & & & \\ & .5w^{(p)} & & & \\ & & \ddots & & \\ & & & .5w^{(p)} & \\ & & & 1-.5w^{(p)} & \\ & & & 1-.5w^{(p)} & \end{pmatrix} \in \mathbb{R}^{m\times\frac{m}{2}},$$

with

$$w^{(p)} = \frac{4/3}{\rho(D^{-1}A)}, \tag{5.4}$$

and $R = P^{\mathsf{T}}$, where M is defined as in (5.2).

The damping parameter $w^{(p)}$ for the method SA is based on an eigenvalue calculation. We will now compare the energy of the first column of the tentative prolongator with respect to the system matrix A to the energy over the first column of the prolongator matrix from the method SA with respect to the same SPD matrix. We do this to show how effective Jacobi energy-minimization is in this problem, with the given damping parameter. Using first the tenative prolongator, we get:

$$\sqrt{P_1^{(\text{tent})\mathsf{T}} A P_1^{(\text{tent})}} = \sqrt{\frac{2}{\Delta x^2}}.$$

Using the prolongator matrix from the method SA we get:

$$\sqrt{P_1^{\mathsf{T}} A P_1} = \sqrt{\frac{2-3w^{(p)}+1.5w^{(p)}}{\Delta x^2}}. \tag{5.5}$$

To compare the two values for energy, we consider the one variable function

$$f(w^{(p)}) = 2 - 3w^{(p)} + 1.5w^{(p)},$$

i.e. the quantity under the square root symbol in the numerator of (5.5). We also look at our possible values for $w^{(p)}$. As stated above in (5.4), $w^{(p)}$ in the method SA is based on an eigenvalue calculation. We are interested in the eigenvalues of $D^{-1}A$, which are just scaled version of the eigenvalues for the discrete Laplace operator with Dirichlet boundary conditions (recall from (4.1)). Thus, the eigenvalues of $D^{-1}A$ are

$$\lambda_j^{D^{-1}A} = \frac{\Delta x^2}{2}\lambda_j^{\text{Laplace}} = 2\sin^2\left(\frac{\pi j}{2(m+1)}\right),\ j = 1, ..., m.$$

Since $w^{(p)}$ relies on the spectral radius of $D^{-1}A$, we only care about the largest of these eigenvalues. From the formula above, we see that $\rho(D^{-1}A) \to 2$ as our dimension increases,

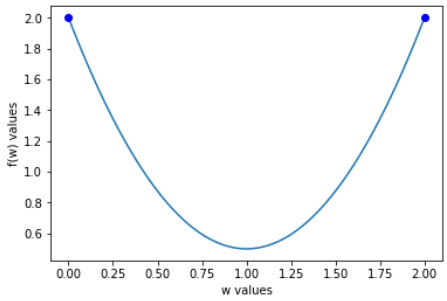


Figure 5.1: Plot of $f(w^{(p)}) = 2 - 3w^{(p)} + 1.5w^{(p)}$

and thus $w^{(p)} \to \frac{2}{3}$.

Consider now the plot in Figure 5.1. With $w^{(p)} = \frac{2}{3}$ we can see the energy of the first column of P will be significantly less than the energy over the first column of $P^{(\text{tent})}$, with respect to the system matrix A. In other words, one iteration of Weighted Jacobi on the prolongation basis functions with the damping parameter defined for the method SA significantly decreases the energy of the columns of P with respect to the matrix associated with the discretized Laplace operator.

As shown previously, Weighted Jacobi is very effective at eliminating high frequency error components. We can now assume that the low-energy subspace is in the range of P, and the smooth error components (including the user-defined near kernel) are represented throughout the mesh hierarchy. These are the key concepts of the successful method SA for SPD systems [9]. Later in this thesis, we will develop similar methods for nonsymmetric systems which are built on the success of SA.

We now consider what happens if we were to repeat the process, smoothing the columns even further.

5.5 Multiple Iterations of Weighted Jacobi

The method SA applies just 1 iteration of Weighted Jacobi to the prolongation basis functions, effectively damping the high frequency error. The prolongation matrix this is associated with can be seen in the previous section. In this section, we consider what happens if we were to apply more than one iteration of Weighted Jacobi to the basis functions, reducing the energy even more. Our prolongation matrix is thus given by the formula

$$\begin{aligned} P_i^{k+1} &= (I - w^{(p)} D^{-1} A) P_i^k \\ &= P_i^k - w^{(p)} D^{-1} A P_i^k, \end{aligned}$$

where P_i^0 is the i-th column of $P^{(\text{tent})}$. We consider the prolongation matrix which corresponds to 2 iterations of Weighted Jacobi on each prolongation basis function. This matrix

we will call P^2 and is given by

$$P^2 = \begin{pmatrix} a & d & & \\ b & c & & \\ c & b & d & \\ d & b & c & \ddots \\ & c & b & \ddots \\ & d & b & \ddots \\ & & c & \ddots \\ & & d & \ddots \\ & & & \ddots \end{pmatrix} \in \mathbb{R}^{m \times \frac{m}{2}}$$

where $a = \frac{w^2-4w+4}{4}$, $b = \frac{w^2-2w+2}{2}$, $c = \frac{-3w^2+4w}{4}$ and $d = \frac{w^2}{4}$.

To appreciate what is going on here, consider the second column of P with 1 iteration of Weighted Jacobi (i.e. P from the method SA), when A is the discrete Laplace operator,

$$P_2^1 = \begin{pmatrix} 0 \\ 0.33333412 \\ 0.66666588 \\ 0.66666588 \\ 0.33333412 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \in \mathbb{R}^m$$

and then the second column of P^2 (2 iterations of Weighted Jacobi) when A is the same as above:

$$P_2^2 = \begin{pmatrix} 0.11111163 \\ 0.33333333 \\ 0.55555503 \\ 0.55555503 \\ 0.33333333 \\ 0.11111163 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \in \mathbb{R}^m.$$

From this we can see the columns of the prolongation basis functions actually begin to resemble sine waves of lower frequencies as we iterate Weighted Jacobi. From an energy minimization perspective this makes sense, since higher frequency sine waves have higher energy than low frequency, slowly oscillating sine waves. This pattern continues as we increase our number of iterations. This is beneficial for convergence because these slowly oscillating sine functions are the same error components that are slow-to-converge on the fine grid. Having these sine functions as the columns of the prolongation matrix helps ensure the low energy eigenmodes (i.e. smooth error components) are accurately represented throughout the mesh hierarchy [8], and thus can be corrected. In Chapter 10 we will experiment in taking the lowest frequency eigenvector and constructing our P based solely on this vector.

Next, consider the coarse matrix corresponding to solving the Poisson equation using the

Table 5.1: Jacobi Iterations on the Prolongation Basis Functions, Solving the Poisson Equation (3.1)

Number of Jacobi iterations	Number of 2-grid iterations	CPU time
0	43	25.207526000000144
1	17	18.04043499999989
2	13	16.02034300000014
3	12	15.640740999999707
4	11	15.008073999999851
5	11	15.396749

1 pre- and post-smoothing step (Weighted Jacobi), $m = 5120$, $\Delta x = \frac{1}{1+m}$, `solve` used on the coarse level, CPU time includes set-up

2-grid method, first with $A_c^{\text{SA}} = RA_fP$ where $R = P^{\mathsf{T}}$ and P is from the method SA and then $A_c^{\text{SA2}} = R^2A_fP^2$, where $R^2 = (P^2)^{\mathsf{T}}$. We get:

$$A_c^{\text{SA}} = \begin{pmatrix} a & b & c & & & & \\ b & d & b & c & & & \\ c & b & d & b & c & & \\ & c & b & d & b & c & \\ & & \ddots & \ddots & \ddots & \ddots & \ddots \end{pmatrix} \in \mathbb{R}^{\frac{m}{2}\times\frac{m}{2}}$$

where $a = \frac{3w^2-6w+4}{2\Delta x^2}$, $b = \frac{-w^2+2w-1}{\Delta x^2}$, $c = \frac{-w^2}{4\Delta x^2}$ and $d = \frac{5w^2-8w+4}{2\Delta x^2}$ and

$$A_c^{\text{SA2}} = \begin{pmatrix} a & b & c & d & & & & \\ b & e & f & c & d & & & \\ c & f & e & f & c & d & & \\ d & c & f & e & f & c & d & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \end{pmatrix} \in \mathbb{R}^{\frac{m}{2}\times\frac{m}{2}}$$

where $a = \frac{11w^4-30w^3+36w^2-24w+8}{4\Delta x^2}$, $b = \frac{-23w^4+72w^3-96w^2+64w-16}{16\Delta x^2}$, $c = \frac{-13w^4+24w^3-12w^2}{8\Delta x^2}$, $d = \frac{-w^4}{16\Delta x^2}$, $e = \frac{21w^4-56w^3+60w^2-32w+8}{4\Delta x^2}$ and $f = \frac{-15w^4+64w^3-96w^2+64w-16}{16\Delta x^2}$.

From this, we see that doing another iteration of Weighted Jacobi on the prolongation basis functions results in more non-zero elements on the coarse matrix, complicating the `solve` function. We are not surprised by this result, since as we damp the columns of P, we increase the number of non-zero elements in P (and R). Then, the product $A_c = RA_fP$ has more non-zero elements as a result.

From the results of Table 5.1 and the plot of iterations of Weighted Jacobi vs energy in the first column of P (Figure 5.2), we see a large improvement in CPU time to convergence after applying just 1 iteration of Weighted Jacobi to the prolongation basis functions when solving the discrete Poisson equation (3.1).

Iterating the procedure continues to decrease the energy, but we see only marginal gains in CPU time when we do this, due to the longer set-up. With this in mind, we apply just 1 iteration for the remainder of this thesis.

5.6 Error Estimates

In [12] Vanek compares how the error is reduced with each iteration of the 2-grid method when solving $Ax = b$ for an SPD system, using first tentative prolongation and restriction,

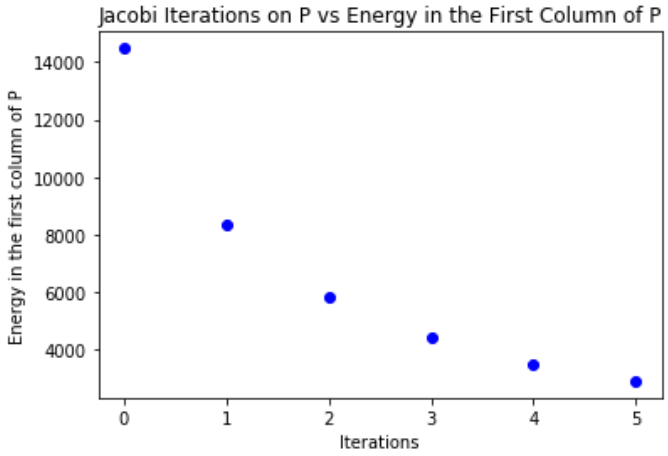


Figure 5.2: Energy Plot, $m = 5120$

and then using the prolongation and restriction operators as defined by the method SA. The results follow without proof. We use the notation ν_1, ν_2 to indicate the number of pre-smoothing and post-smoothing iterations, respectively. M corresponds to the iteration matrix of Weighted Jacobi.

Lemma: (Error estimate, Tentative prolongation and restriction)
Let restriction and prolongation be achieved using a tentative operators, $\nu_1 = 0$, $T = \ker(RA)$. Then the following error estimate is valid:

$$\frac{||e(x_{k+1})||_2}{||e(x_k)||_2} \leq \sup_{x \in T \setminus \{0\}} \frac{||M^{\nu_2} x||_2}{||x||_2}.$$

Theorem: (Error estimate, Smoothed Aggregation)
Let $\nu_1 = \nu_2 = 2$, $T = \ker(RA)$. Then the following error estimate holds for the 2-level method with operators $R = R^{(\text{tent})}M$, $P = MP^{(\text{tent})}$:

$$\frac{||e(x_{k+1})||_2}{||e(x_k)||_2} \leq \Big(\sup_{x \in T \setminus \{0\}} \frac{||Mx||_2}{||x||_2} \Big)^4.$$

In reality, we hope to do much better than these error estimates.

Chapter 6

Smoothed Aggregation and Closely Related Methods

6.1 SA, NSA and NSR

We now discuss two variants of SA, Nonsmoothed Restriction Aggregration (NSR) and Nonsmoothed Aggregation (NSA). These methods require only very simple adaptations from SA.

In NSR, we keep P as defined in SA, i.e.

$$P = MP^{(\text{tent})}$$

$$M = (I - w^{(p)} D^{-1} A)$$

$$w^{(p)} = \frac{4/3}{\rho(D^{-1}A)},$$

where $P^{(\text{tent})}$ is as defined in (5.1). But, instead of defining R as

$$R = P^{\mathsf{T}} = R^{(\text{tent})} M^{\mathsf{T}},$$

as in SA, we use

$$R = R^{(\text{tent})},$$

where $R^{(\text{tent})} = P^{(\text{tent})\mathsf{T}}$. Thus, we have removed the extra smoothing step in the restriction, but keep it in the prolongation.

In NSA, we remove the extra smoothing altogether and take

$$R = R^{(\text{tent})}$$

and

$$P = P^{(\text{tent})}.$$

This corresponds to the classic agglomeration multigrid scheme.

6.2 SA, NSA and NSR for Symmetric System Matrices

To see the effects of removing the extra smoothing, we consider the discrete Poisson equation in one dimension with Dirichlet boundary conditions (3.1). A plot of this figure follows in Figure 6.1.

The discrete Laplace operator is an SPD matrix, so we expect SA to perform well. Because NSR removes some extra damping, we can expect it to perform suboptimally, but better than Nonsmoothed Aggregation (NSA). To test this, a simple 2-grid method was implemented with 1 pre- and post-smoothing iteration of Weighted Jacobi. Sparse solve (`spsolve`) from Scipy was used on the coarsest level. To see the results of this experiment with $m = 1024$ unknowns, see Table 6.1. The results are as predicted: SA performed best, followed by NSR. NSA required the most iterations to converge in this experiment.

Table 6.1: Poisson Equation (3.1) (Symmetric)

Method	Number of 2-grid iterations
SA	16
NSA	41
NSR	23

1 pre- and post-smoothing step (Weighted Jacobi), $m = 1024$, $\Delta x = \frac{1}{1+m}$, `solve` used on the coarse level, tol $= 10^{-8}$

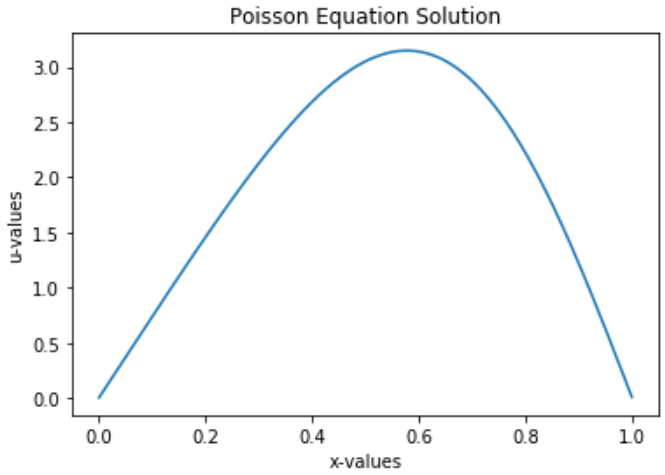


Figure 6.1: Solution to (3.1)

6.3 SA, NSA and NSR for Nonsymmetric System Matrices

So far, we have only considered symmetric systems, such as the one formed by the discrete Laplace operator. Consider now linear advection

$$u_t + au_x = 0$$

with periodic boundary conditions and initial condition

$$u_0 = u(x, 0) = \sin(\pi x)$$

on the domain $x \in (0, 2]$.

Our system matrix A corresponding to a first-order finite volume discretization with upwind flux and one time-step (implicit Euler method) is no longer symmetric (3.2).

6.3.1 SA Prolongation for Linear Advection (Nonsymmetric)

The prolongation operator for the method SA is again given by

$$P = (I - w^{(p)} D^{-1} A) P^{(\text{tent})}.$$

In the symmetric case, this prolongator reduces energy of the prolongation basis functions over the SPD system matrix A. As we no longer have an SPD system, we can no longer speak of an energy norm with respect to A or high energy error components, since the energy norm is no longer defined. However, the method still attempts to damp oscillatory eigenmodes (analogous to high energy components in symmetric systems) of A via Weighted Jacobi, which can be seen directly from the equation for P [9]. The goal of this is still to represent better the slowly-oscillating smooth error components throughout the mesh hierarchy [8].

In the case of linear advection, where A is defined as in (3.2), P for the method SA becomes

$$P = \begin{pmatrix} 1-w & & & \frac{wa\Delta t}{a\Delta t+\Delta x} \\ 1-\frac{w\Delta x}{a\Delta t+\Delta x} & & & \\ \frac{wa\Delta t}{a\Delta t+\Delta x} & 1-w & & \\ & 1-\frac{w\Delta x}{a\Delta t+\Delta x} & & \\ & \frac{wa\Delta t}{a\Delta t+\Delta x} & \ddots & \\ & & & 1-w \\ & & & 1-\frac{w\Delta x}{a\Delta t+\Delta x} \end{pmatrix} \in \mathbb{R}^{m \times \frac{m}{2}}.$$

6.3.2 SA Restriction for Linear Advection (Nonsymmetric)

As stated previously, the system matrix for linear advection associated with one time-step using our discretization is no longer symmetric. We can still apply the method SA to our system with the restriction matrix given by

$$R = P^{\mathsf{T}} = P^{(\text{tent})\mathsf{T}} (I - w^{(p)} D^{-1} A)^{\mathsf{T}}.$$

Because D^{-1} is just the identity matrix times a constant, we have

$$(D^{-1}A)^{\mathsf{T}} = A^{\mathsf{T}} (D^{-1})^{\mathsf{T}} = A^{\mathsf{T}} (D^{-1}) = (D^{-1}) A^{\mathsf{T}}.$$

Thus, we have the relation

$$\begin{aligned} R &= P^{(\text{tent})\mathsf{T}} (I^{\mathsf{T}} - w^{(p)} A^{\mathsf{T}} D^{-1^{\mathsf{T}}}) \\ &= R^{(\text{tent})} (I - w^{(p)} A^{\mathsf{T}} D^{-1}) \\ &= R^{(\text{tent})} (I - w^{(p)} D^{-1} A^{\mathsf{T}}). \end{aligned}$$

Looking carefully at the equation above for R, we notice that this corresponds to damping the oscillatory eigenmodes of A^{T}, a matrix which is not equal to A. Thus, we are effectively damping the wrong error components when we apply the restriction operator from SA in the nonsymmetric case. Because of this property of the restriction operator, we can expect suboptimal results of SA in nonsymmetric systems [9].

6.4 Results: Nonsymmetric Systems

6.4.1 Data Tables

Table 6.2: Linear Advection (3.2) (Nonsymmetric)

Method	Number of 2-grid iterations
SA	81
NSA	85
NSR	81

Iterations required to perform one time-step with $\Delta t = .01$, $m = 1024$, $a = 2$, 1 pre- and post-smoothing step (Non-Weighted Jacobi), $\Delta x = \frac{2}{m}$, `solve` used on the coarse level, initial condition $u_0 = \sin(\pi x)$, for $x \in (0, 2]$, periodic boundary conditions, tol $= 10^{-8}$

Table 6.3: Number of 2-grid Iterations, Linear Advection (3.2) (Nonsymmetric)

Method	$m = 512$	$m = 1024$	$m = 2048$	$m = 4096$	$m = 8192$
SA	6	7	7	8	8
NSA	6	7	8	8	8
NSR	6	7	7	8	8

Iterations required to perform one time-step with $\Delta t = .01$, $a = 2$, 1 pre- and post-smoothing step (Weighted Jacobi), $\Delta x = \frac{2}{m}$, `solve` used on the coarse level, initial condition $u_0 = \sin(\pi x)$, for $x \in (0, 2]$, periodic boundary conditions, tol $= 10^{-8}$

Table 6.4: 1D Convection Diffusion (3.3)

Method	Number of 2-grid iterations, $\varepsilon = 10^{-5}$, highly convective	Number of 2-grid iterations, $\varepsilon = 10^{-1}$, nearly symmetric
SA	24	13
NSA	27	30
NSR	9	16

1 pre- and post-smoothing step of Weighted Jacobi, $m = 1024$, $\Delta x = \frac{1}{1+m}$, $b = 1$, tol $= 10^{-8}$

6.4.2 Discussion of the Results

To visualize how the methods perform in a simple case, we consider a 2-grid method for performing one time-step of linear advection with Jacobi smoothing. A plot of the calculated solution with $\Delta t = .1$ can be seen in Figure 6.2. Table 6.2 shows us how many iterations we need to perform one time-step ($\Delta t = .01$) of linear advection using the previously described methods. As we can see, SA actually takes almost as many iterations as NSA, and NSR takes as many iterations as SA. We can see that extra smoothing within restriction and/ or prolongation does help, but we hope to do better.

Table 6.3 shows a nice property of multigrid methods. In this simple 2-grid scheme, we see mesh width independence. As we vary how many unknowns we have on the same space domain, our iterations stay roughly the same when solving an otherwise identical problem. We use this property to our advantage when solving more complicated (and larger) problems.

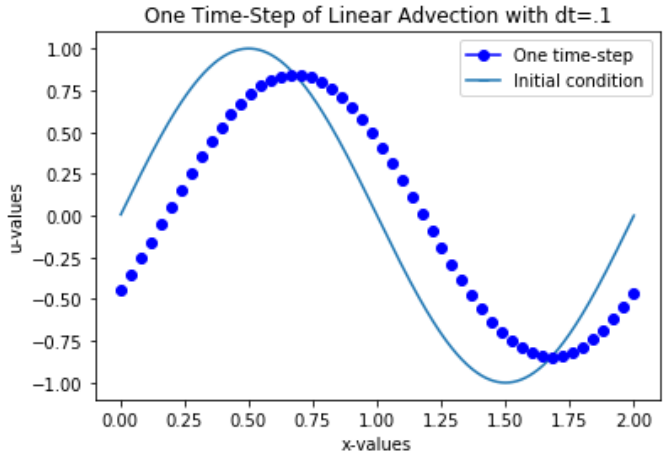


Figure 6.2: Solution to (3.2)

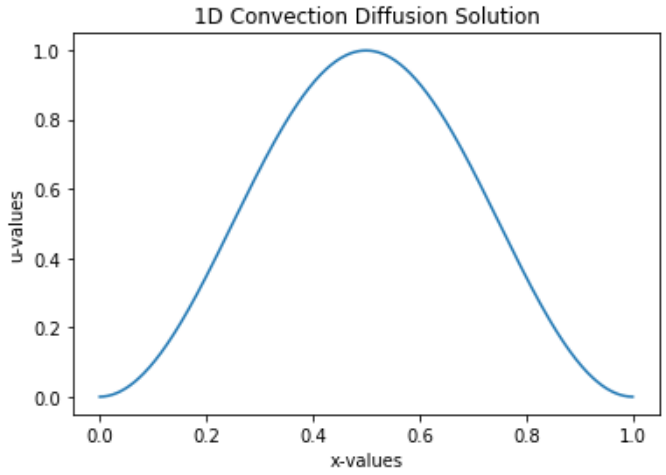


Figure 6.3: Solution to (3.3)

Following the same ideas as before, we test this equation in the 2-grid method for different values of ε in the convection diffusion equation. A plot of the calculated solution can be found in Figure 6.3. We recall that the discretization matrix for convection diffusion from (3.3) is nonsymmetric. Table 6.4 shows how many iterations we need for convergence in solving both cases of the convection diffusion equation, using our previously described methods. In highly convective systems (i.e. $\varepsilon = 10^{-5}$), SA performs poorly and NSR, skipping the suboptimal smoothing related to A^{T}, performs better than both SA and NSA. In more symmetric systems (i.e. $\varepsilon = 10^{-1}$), SA performs better than NSA and NSR.

6.5 Motivation for a New Method for Nonsymmetric Systems

What we would like now is a method which builds on the success of SA, but can achieve good results in nonsymmetric systems. Such a method should effectively smooth the tentative restrictor and prolongator basis functions in nonsymmetric systems, so that the smooth error components can be represented throughout the mesh hierarchy [8]. From our previous work, we understand that such a method should not involve A^{T} in the restriction operator. We understand that such a method should also accurately interpolate the user-defined near kernel.

Chapter 7

Generalization of SA to Nonsymmetric Systems

In [9], Sala and Tuminaro suggest 2 new methods for nonsymmetric systems which are based on the success of SA in symmetric systems. These methods are built on the same two key components as SA: energy minimization (smoothing) and the accurate interpolation of the user-defined near kernel. In their experiments, the methods show results which are comparable to SA in diffusive systems and much better than SA in convective systems (1D).

We first explain the method called EMIN(r) then the closely related method EMIN. We will consider how these methods handle energy minimization (smoothing) in systems that are not SPD, and the interpolation of the user-defined near kernel. Just like in the method SA, we seek to minimize the energy of our columns of P so that we can better represent smooth error components throughout the mesh hierarchy, with the goal of correcting them on the coarsest grid, then interpolating them back to the fine grid to update our approximation. However, unlike in SA, where we chose a single damping parameter based on an eigenvalue calculation, EMIN(r) will incorporate a new idea of local damping parameters. The idea here is that different regions (highly convective or highly diffusive) may benefit from different damping parameters [9]. We will close this section by discussing how these methods perform for different systems.

7.1 Damping Parameters for EMIN(r)

Before we show how to set-up the prolongation and restriction operators for our new methods, we consider how to calculate the local damping parameters. To mimic the method SA with nonsymmetric systems, Sala and Tuminaro selected damping parameters in such a way to minimize AP in the Frobenius norm [9].

The following relationship holds:

$$||AP||_F^2 = \text{trace}([AP]^\mathsf{T}[AP]) = \sum_{j=1}^{n_c} P_j^\mathsf{T} A^\mathsf{T} A P_j = \sum_{j=1}^{n_c} ||P_j||_{A^\mathsf{T}A}^2, \tag{7.1}$$

where $P_j = (I - w_j^{(p)} D^{-1} A) P_j^{(tent)}$ and n_c refers to the number of columns of $P^{(\text{tent})}$. Defining P_j as such reminds us of the method SA, which applied one iteration of Weighted Jacobi to each column of the tentative prolongator, damping the high energy (symmetric system matrices) and smoothing the columns of the prolongator. The $w_j^{(p)}$ represent the local damping parameters. The method SA did not use local damping parameters, but

instead a single damping parameter based on an eigenvalue computation.

The last item in (7.1) reflects a generalization of SA's energy minimization of the prolongation basis functions to nonsymmetric systems, since $A^\mathsf{T}A$ is always an SPD matrix.

The local damping parameters are chosen such that:

$$w_j^{(p)} = \arg\min_{w_j^{(p)}} ||AP||_F^2, \ j = 1, ..., n_c$$

which corresponds to:

$$\begin{aligned} w_j^{(p)} &= \sum \arg\min_{w_j^{(p)}} ||P_j||_{A^\mathsf{T}A}^2, \ j = 1, ..., n_c \\ &= \sum \arg\min_{w_j^{(p)}} ||(I - w_j^{(p)} D^{-1} A) P_j^{(tent)}||_{A^\mathsf{T}A}^2, \ j = 1, ..., n_c \\ &= \sum \arg\min_{w_j^{(p)}} ||P_j^{(tent)} - w_j^{(p)} D^{-1} A P_j^{(tent)}||_{A^\mathsf{T}A}^2, \ j = 1, ..., n_c. \end{aligned} \tag{7.2}$$

We can solve this minimization by an orthogonal projection, i.e. projecting the vector $P_j^{(tent)}$ onto $D^{-1}AP_j^{(tent)}$ with respect to $A^\mathsf{T}A$. The length of this projection corresponds to the $w_j^{(p)}$ which minimizes the j-th summand in (7.2).

Projecting $P_j^{(tent)}$ onto $D^{-1}AP_j^{(tent)}$ with respect to $A^\mathsf{T}A$ results in a vector which we can calculate as follows:

$$v = \frac{< P_j^{(\text{tent})}, D^{-1}AP_j^{(\text{tent})} >_{A^\mathsf{T}A}}{||D^{-1}AP_j^{(\text{tent})}||_{A^\mathsf{T}A}^2} D^{-1}AP_j^{(\text{tent})}.$$

Since we are interested in the length of this vector, i.e. the scalar quantity $w_j^{(p)}$ for each j, this gives us a nice closed formula for our damping parameters. We get:

$$w_j^{(p)} = \frac{< P_j^{(\text{tent})}, D^{-1}AP_j^{(\text{tent})} >_{A^\mathsf{T}A}}{||D^{-1}AP_j^{(\text{tent})}||_{A^\mathsf{T}A}^2}$$

for the prolongation and

$$w_i^{(r)} = \frac{< R_i^{(\text{tent})\mathsf{T}}, D^{-1}A^\mathsf{T}R_i^{(\text{tent})\mathsf{T}} >_{AA^\mathsf{T}}}{||D^{-1}A^\mathsf{T}R_i^{(\text{tent})\mathsf{T}}||_{AA^\mathsf{T}}^2}.$$

in the case of restriction (analogous case).

This results in a large set-up cost for the method EMIN(r), compared to our previously defined methods. However, as shown in the results in the next chapter, the set-up cost is justified for some problems (in particular, linear advection with large CFL numbers), due to the impressive results of the method.

7.2 Setting up EMIN(r)

Now that we have our damping parameters, we must set up our restriction and prolongation in a way that guarantees the interpolation of the user-defined near kernel vector(s),

i.e. that this vector(s) is accurately represented throughout the mesh hierarchy. In this section we focus on how to set up prolongation and in the next section, we explain why this way accurately interpolates the user-defined near kernel.

If we follow an analogous idea to SA, it seems reasonable to set up our prolongation as

$$P = P^{(\text{tent})} - D^{-1}AP^{(\text{tent})}\Omega^{(\text{p})},$$

where $\Omega^{(\text{p})}$ is a diagonal $(\frac{m}{2} \times \frac{m}{2})$ matrix with $w_j^{(p)}$, $j = 1, ..., n_c$ as entries. Since our matrix $\Omega^{(\text{p})}$ has dimensions $(\frac{m}{2} \times \frac{m}{2})$ we need to put the this matrix on the right of the matrix $P^{(\text{tent})}$. This formula, unfortunately, proves to be problematic (see following section). We decide instead to set up our prolongation as

$$P = P^{(\text{tent})} - D^{-1}\Omega^{(\text{fine})}AP^{(\text{tent})} = (I - D^{-1}\Omega^{(\text{fine})}A)P^{(\text{tent})}, \tag{7.3}$$

where $\Omega^{(\text{fine})}$ is an $(m \times m)$ matrix with diagonal entries $w_j^{(\text{fine})}$ defined by

$$\Omega^{(\text{int})}P^{(\text{tent})} = P^{(\text{tent})}\Omega^{(p)} \tag{7.4}$$

and

$$w_j^{(\text{fine})} = \max\{0, \min_{j,\, a_{ij}\neq 0} w_j^{(\text{int})}\}, \tag{7.5}$$

where $\Omega^{(\text{int})}$ is an $(m \times m)$ diagonal matrix with entries $w_j^{(\text{int})}$. Because of this, we need m damping parameters, though we only have $\frac{m}{2}$. Thus, we must do an interpolation.

To see what this interpolation looks like, consider using solving (7.4) for the entries of $\Omega^{(\text{int})}$ (i.e. $w_j^{(\text{int})}$) in a concrete example ($m = 8$). We have:

$$\Omega^{(\text{int})}\begin{pmatrix} 1 & & & & & & & \\ 1 & & & & & & & \\ & 1 & & & & & & \\ & 1 & & & & & & \\ & & & 1 & & & & \\ & & & 1 & & & & \\ & & & & & & 1 & \\ & & & & & & 1 & \end{pmatrix} = \begin{pmatrix} 1 & & & & & & & \\ 1 & & & & & & & \\ & 1 & & & & & & \\ & 1 & & & & & & \\ & & & 1 & & & & \\ & & & 1 & & & & \\ & & & & & & 1 & \\ & & & & & & 1 & \end{pmatrix}\begin{pmatrix} w_1^{(p)} & & & & & & & \\ & w_2^{(p)} & & & & & & \\ & & & w_3^{(p)} & & & & \\ & & & & & & w_4^{(p)} & \end{pmatrix} \tag{7.6}$$

$$\Longleftrightarrow \begin{pmatrix} w_1^{(\text{int})} & & & & & & & \\ w_2^{(\text{int})} & & & & & & & \\ & & w_3^{(\text{int})} & & & & & \\ & & w_4^{(\text{int})} & & & & & \\ & & & & w_5^{(\text{int})} & & & \\ & & & & w_6^{(\text{int})} & & & \\ & & & & & & w_7^{(\text{int})} & \\ & & & & & & w_8^{(\text{int})} & \end{pmatrix} = \begin{pmatrix} w_1^{(p)} & & & & & & & \\ w_1^{(p)} & & & & & & & \\ & & w_2^{(p)} & & & & & \\ & & w_2^{(p)} & & & & & \\ & & & & w_3^{(p)} & & & \\ & & & & w_3^{(p)} & & & \\ & & & & & & w_4^{(p)} & \\ & & & & & & w_4^{(p)} & \end{pmatrix}, \tag{7.7}$$

and therefore $w_{2j-1}^{(\text{int})} = w_j^{(p)}$ and $w_{2j}^{(\text{int})} = w_j^{(p)}$ for $j = 1, ..., n_c$. Now we have our $w_j^{(\text{int})}$, and we use (7.5) to calculate $w_j^{(\text{fine})}$.

Unfortunately, we have no closed formula to calculate the optimal $w_j^{(\text{int})}$ (interpolation).

So, setting up P in the way described above does lose some of the optimality of the energy minimization.

We define R in an analogous way:

$$R = R^{(\text{tent})} - R^{(\text{tent})} A D^{-1} \Omega^{(\text{fine})} = R^{(\text{tent})} (I - A D^{-1} \Omega^{(\text{fine})}), \tag{7.8}$$

noting that A^{T} is not included in the above equation.

7.3 EMIN

Our second proposed method, EMIN is almost identical to EMIN(r). EMIN avoids some of the setup cost of EMIN(r) by taking the local damping parameters calculated for the prolongation and using them directly in the restriction. We will see a savings in setup cost associated with EMIN (one half), without sacrificing on performance.

7.4 Near Kernel Interpolation, EMIN(r)/ EMIN

In a previous section we discussed choosing the damping parameters to minimize $||AP||_F^2$ and thus smooth the prolongation basis functions. Using SA as our guide, we must also accurately interpolate the user-defined near kernel. By this we mean

$$\min_P ||AP||_F^2 \text{ subject to } \mathcal{N} \subset \mathcal{R}(P)$$

where $\mathcal{N}$ is the user-defined near kernel (the constant vector, unless otherwise specified) and $\mathcal{R}(P)$ is the range of our prolongation operator.

For the methods EMIN(r) and EMIN, the prolongation operator is

$$P = (I - D^{-1} \Omega^{(\text{fine})} A) P^{(\text{tent})}$$

instead of the original idea which gave us optimal energy minimization,

$$P = P^{(\text{tent})} - D^{-1} A P^{(\text{tent})} \Omega^{(\text{p})} = (I - D^{-1} A) P^{(\text{tent})} \Omega^{(\text{p})}. \tag{7.9}$$

To see why the original idea for P does not accurately interpolate the user-defined near kernel, consider calculating P from (7.9) when A is the system matrix used to solve the discrete Poisson equation. This matrix P corresponds to:

$$P = \begin{pmatrix} .5w_1^{(\text{p})} & & & & & \\ .5w_1^{(\text{p})} & .5w_2^{(\text{p})} & & & & \\ .5w_1^{(\text{p})} & .5w_2^{(\text{p})} & & & & \\ & .5w_2^{(\text{p})} & .5w_3^{(\text{p})} & & & \\ & .5w_2^{(\text{p})} & .5w_3^{(\text{p})} & & & \\ & & .5w_3^{(\text{p})} & & & \\ & & .5w_3^{(\text{p})} & & & \\ & & & \ddots & .5w_{n_c}^{(\text{p})} & \\ & & & & .5w_{n_c}^{(\text{p})} & \\ & & & & .5w_{n_c}^{(\text{p})} & \end{pmatrix}.$$

We see above that multiplying with $\Omega^{(p)}$ on the right multiplies in the w_i's column-wise. Because our damping parameters are not always equal for all i, this does not guarantee

that the constant vector will be in the range of our prolongation operator. This gives us the idea to change the placement of the diagonal matrix constructed from the damping parameters.

We return now to our proposed formula for P in EMIN(r)/EMIN. For simplicity's sake, let's refer to this matrix as

$$S = I - D^{-1}\Omega^{(\text{fine})}A.$$

In the case of the Poisson equation, we get

$$SP^{(\text{tent})} = \begin{pmatrix} 1-.5w_1^{(\text{fine})} & & & & \\ 1-.5w_2^{(\text{fine})} & .5w_2^{(\text{fine})} & & & \\ .5w_3^{(\text{fine})} & 1-.5w_3^{(\text{fine})} & & & \\ & 1-.5w_4^{(\text{fine})} & & & \\ & .5w_5^{(\text{fine})} & & & \\ & & \ddots & .5w_{m-2}^{(\text{fine})} & \\ & & & 1-.5w_{m-1}^{(\text{fine})} & \\ & & & 1-.5w_m^{(\text{fine})} & \end{pmatrix}.$$

For linear advection (system matrix corresponds to 1 time-step using the implicit Euler method with periodic boundary conditions), we get

$$SP^{(\text{tent})} = \begin{pmatrix} \frac{a\Delta t+\Delta x-aw\Delta t-w\Delta x}{a\Delta t+\Delta x} & & & \frac{aw\Delta t}{a\Delta t+\Delta x} \\ \frac{a\Delta t-w\Delta x+\Delta x}{a\Delta t+\Delta x} & \frac{a\Delta t+\Delta x-aw\Delta t-w\Delta x}{a\Delta t+\Delta x} & & \\ \frac{aw\Delta t}{a\Delta t+\Delta x} & \frac{a\Delta t-w\Delta x+\Delta x}{a\Delta t+\Delta x} & & \\ & \frac{aw\Delta t}{a\Delta t+\Delta x} & & \\ & & \ddots & \\ & & & \frac{a\Delta t+\Delta x-aw\Delta t-w\Delta x}{a\Delta t+\Delta x} \\ & & & \frac{a\Delta t-w\Delta x+\Delta x}{a\Delta t+\Delta x} \end{pmatrix},$$

where w is used in the previous matrix because $w_i^{(\text{fine})}$ are equal for all i in this case.

For convection diffusion, we get

$$SP^{(\text{tent})} = \begin{pmatrix} \frac{2\varepsilon-\varepsilon w_1^{(\text{fine})}+b\Delta x-bw_1^{(\text{fine})}\Delta x}{2\varepsilon+b\Delta x} & & & \\ \frac{2\varepsilon-\varepsilon w_2^{(\text{fine})}+b\Delta x}{2\varepsilon+b\Delta x} & \frac{\varepsilon w_2^{(\text{fine})}}{2\varepsilon+b\Delta x} & & \\ \frac{\varepsilon w_3^{(\text{fine})}+bw_3^{(\text{fine})}\Delta x}{2\varepsilon+b\Delta x} & \frac{2\varepsilon-\varepsilon w_3^{(\text{fine})}+b\Delta x-bw_3^{(\text{fine})}\Delta x}{2\varepsilon+b\Delta x} & & \\ & \frac{2\varepsilon-\varepsilon w_4^{(\text{fine})}+b\Delta x}{2\varepsilon+b\Delta x} & & \\ & \frac{\varepsilon w_5^{(\text{fine})}+bw_5^{(\text{fine})}\Delta x}{2\varepsilon+b\Delta x} & & \\ & & \ddots & \\ & & & \frac{\varepsilon w_{m-2}^{(\text{fine})}}{2\varepsilon+b\Delta x} \\ & & & \frac{2\varepsilon-\varepsilon w_{m-1}^{(\text{fine})}+b\Delta x-bw_{m-1}^{(\text{fine})}\Delta x}{2\varepsilon+b\Delta x} \\ & & & \frac{2\varepsilon-\varepsilon w_m^{(\text{fine})}+b\Delta x}{2\varepsilon+b\Delta x} \end{pmatrix}.$$

Because of the structure of the linearly independent columns of $P^{(\text{tent})}$ (i.e. repeating patterns of 0's and 1's) and the different matrices for S shown above, where the damping parameters appear row-wise (instead of column-wise), we can see that the constant vector

is in the range of our new prolongation operators (excluding the first and last entry, which correspond to boundaries). Thus, we can count on our user-defined near kernel vector being accurately represented throughout the mesh hierarchy, just as it was in the method SA. Unfortunately, this result is not true in general for $S \in \mathbb{R}^{m \times m}$, which is why we chose to show the previous three examples.

7.5 EMIN and EMIN(r) for Symmetric Systems

Table 7.1: Poisson Equation (3.1) (Symmetric)

Method	Number of 2-grid iterations
SA	16
NSA	41
NSR	23
EMIN(r)	18
EMIN	18

1 pre- and post-smoothing step (Weighted Jacobi), $m = 1024$, $\Delta x = \frac{1}{1+m}$, `solve` used on the coarse level, tol $= 10^{-8}$

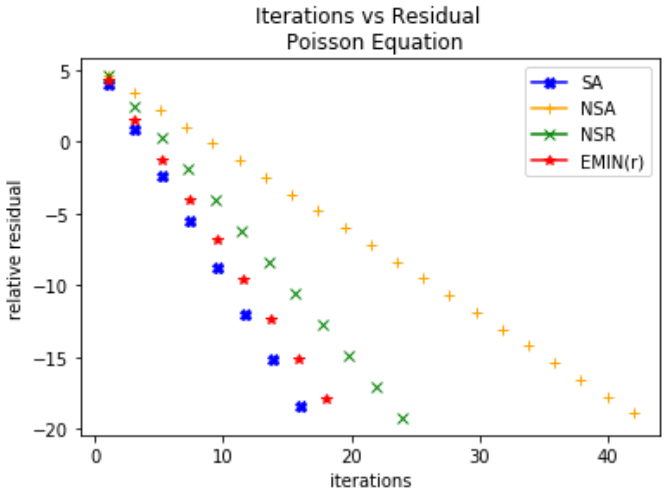


Figure 7.1: Iterations vs Residual for solving (3.1) with Jacobi smoothing, $\nu_1 = \nu_2 = 1$ $l = 1$, $\gamma = 1$, $m = 1024$, $\Delta x = \frac{1}{m+1}$

Now that we have shown how to set up our new methods, we can visualize their effectiveness first (for simplicity's sake) in the 2-grid method. The results of this follow in Table 7.1 and Figure 7.1, where we considered the Poisson equation.

Our new methods perform almost as well as SA in the symmetric case. This is very promising because SA is a powerful method for symmetric systems and it is our hope that our newly proposed methods will perform well in both symmetric and nonsymmetric systems. We note, however that EMIN(r) and EMIN require more set-up than SA.

7.6 EMIN and EMIN(r) for Nonsymmetric Systems

Table 7.2: Linear Advection (3.2) (Nonsymmetric)

Method	Number of 2-grid iterations
SA	81
NSA	85
NSR	81
EMIN(r)	68
EMIN	68

Iterations required to perform one time-step with $\Delta t = .01$, $m = 1024$, $a = 2$, 1 pre- and post-smoothing step (Non-Weighted Jacobi), $\Delta x = \frac{2}{m}$, `solve` used on the coarse level, initial condition $u_0 = \sin(\pi x)$, for $x \in (0, 2]$, periodic boundary conditions, tol $= 10^{-8}$

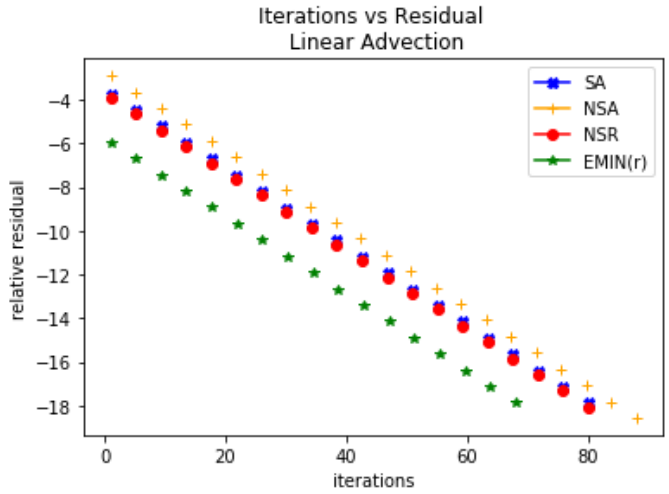


Figure 7.2: Iterations vs Residual for solving (3.2), Non-Weighted Jacobi smoothing, $\nu_1 = \nu_2 = 1$, $l = 1$, $\gamma = 1$, $m = 1024$, $\Delta x = \frac{2}{m}$, $\Delta t = .01$, $a = 2$

Table 7.3: Number of 2-grid Iterations, Linear Advection (3.2) (Nonsymmetric)

Method	$m = 512$	$m = 1024$	$m = 2048$	$m = 4096$	$m = 8192$
SA	6	7	7	8	8
NSA	6	7	8	8	8
NSR	6	7	7	8	8
EMIN(r)	5	6	6	6	6
EMIN	5	6	6	6	6

Iterations required to perform one time-step with $\Delta t = .01$, $a = 2$, 1 pre- and post-smoothing step (Weighted Jacobi), $\Delta x = \frac{2}{m}$, `solve` used on the coarse level, initial condition $u_0 = \sin(\pi x)$, for $x \in (0, 2]$, periodic boundary conditions, tol $= 10^{-8}$

Table 7.4: 1D Convection Diffusion (3.3)

Method	Number of 2-grid iterations, $\varepsilon = 10^{-5}$, highly convective	Number of 2-grid iterations, $\varepsilon = 10^{-1}$, nearly symmetric
SA	24	13
NSA	27	30
NSR	9	16
EMIN(r)	7	14
EMIN	7	14

1 pre- and post-smoothing step of Weighted Jacobi, $m = 1024$, $\Delta x = \frac{1}{1+m}$, $b = 1$, tol $= 10^{-8}$

If we now refer to Table 7.2 and Figure 7.2, we can see that our new methods perform well when performing one time-step of linear advection. Our new methods are better than our previous best methods here. Table 7.3 shows that our new methods also preserve mesh width independence in the case of one time-step of linear advection. Looking at Table 7.4 we can see that in the highly convective case of convection diffusion, our new methods also perform better than the previously described methods. In the nearly symmetric case of convection diffusion, our newly proposed methods do almost as well as SA, the method designed for such systems. Thus, we have promising new methods to try out on more complicated problems.

Chapter 8

Multigrid Method Results (Part 1)

In this chapter, we begin by transitioning from the 2-grid method to a multigrid scheme. We will explain what this entails, and then prove the invertibility of the coarse system matrices in this scheme. After this, we will present some basic results using our previously presented methods.

8.1 Implementation of the Multigrid Method

Now that we have seen many simple examples using just the 2-grid method, we change our focus to the multigrid method. It is here that we incorporate V and W cycles for more efficient solving.

It is also important to point out how we are defining our coarse matrices. Following Sala and Tuminaro in [9], we set up the coarse matrices recursively. By this, in the 4-level model (i.e. $l = 3$ on the finest level), we mean A_3 is constructed according to the specified discretization and dimensions on the fine grid, and then

$$A_2 = R_3^2 A_3 P_2^3,$$

$$A_1 = R_2^1 R_3^2 A_3 P_2^3 P_1^2 = R_2^1 A_2 P_1^2$$

and

$$A_0 = R_1^0 R_2^1 R_3^2 A_3 P_2^3 P_1^2 P_0^1 = R_1^0 A_1 P_0^1,$$

where R_l^{l-1} and P_{l-1}^l are the method-specific restriction and prolongation operators. Because of this, sometimes the matrices on the coarsest level no longer have the same structure as the fine level system matrix they originated from.

It is here that we remind the reader to always use sparse matrices when working with a multigrid method. If one fails to do this, the code will be far, far too slow for practical purposes, especially if one wants to solve for thousands of unknowns (or more). Additionally, since the numerical experiments in this thesis were implemented in Python, it should be pointed out that for loops should be avoided whenever possible, as Python is a scripting language.

8.2 Multigrid Method Compared to 2-grid Method

We also note here the differences between the multigrid method and the 2-grid method. Multigrid methods are extremely useful when we have large dimensions, since we can reduce the size of our `solve` problem immensely by going down multiple levels. However, with

the change of dimension comes the fact that the `solve` on the coarsest level will not be as effective as in the 2-grid method, since we have strayed so far from our original problem. What multigrid methods offer us is a comparatively shorter computation time.

8.3 Proof of the Invertibility of the Coarse Matrices

8.3.1 Introduction to Null Space Preservation

As stated in the previous section, going down multiple levels using R and P as defined by the methods in this thesis can result in a coarsest grid matrix that is quite different from the system matrix on the finest level. We will prove in this section that, as long as the fine grid matrix is full rank, we need not worry that our matrices will become singular on the coarse levels.

Consider first P from the methods EMIN(r) and EMIN. We take this prolongator first because it is the most complicated. Setting up P as

$$P = P^{(\text{tent})} - D^{-1}\Omega^{(\text{fine})}AP^{(\text{tent})} \tag{8.1}$$

allows us to show the desired property, whereas the original idea (presented in the previous chapter)

$$P = P^{(\text{tent})} - D^{-1}AP^{(\text{tent})}\Omega^{(\text{p})} \tag{8.2}$$

does not.

We shall now show that for (8.1), the null space of A is in the range of P when it is in the range of $P^{(\text{tent})}$. We also show that this is not necessarily so for (8.2). In the following subsection, we show the importance of this property.

Proof:
Assume that the (not necessarily trivial) null space of A is in the range of $P^{(\text{tent})}$, that is assume $AP^{(\text{tent})}c = 0$ where $A \in \mathbb{R}^{m\times m}$ and $c \in \mathbb{R}^m$. Then, using P from (8.1) we have:

$$\begin{aligned} Pc &= P^{(tent)}c - D^{-1}\Omega^{(\text{fine})}AP^{(\text{tent})}c \\ &= P^{(tent)}c - 0 \\ &= P^{(tent)}c, \end{aligned}$$

or that the null space of A is in the range of P when it is in the range of $P^{(\text{tent})}$.

Now consider again our original idea for P from (8.2). Here we have:

$$\begin{aligned} Pc &= P^{(\text{tent})}c - D^{-1}AP^{(\text{tent})}\Omega^{(\text{p})}c \\ &= P^{(\text{tent})}c - D^{-1}AP^{(\text{tent})} \begin{pmatrix} w_1^{(p)} & & & \\ & \ddots & & \\ & & \ddots & \\ & & & w_{n_c}^{(p)} \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_{n_c} \end{pmatrix} \\ &= P^{(\text{tent})}c - D^{-1}AP^{(\text{tent})} \begin{pmatrix} w_1^{(p)}c_1 \\ w_2^{(p)}c_2 \\ \vdots \\ w_{n_c}^{(p)}c_{n_c} \end{pmatrix}. \end{aligned}$$

From our assumption, we have that $AP^{(\text{tent})}c = 0$. From the above, we can see that if there is variation in the $w_j^{(p)}$'s, then $P^{(\text{tent})}\Omega^{(\text{p})}c$ might not be in the null space of A, and thus the null space of A is not necessarily in the range of P when it is in the range of $P^{(\text{tent})}$, which is the property we wanted to show. It is for this reason we use

$$P = P^{(\text{tent})} - D^{-1}\Omega^{(\text{fine})}AP^{(\text{tent})}.$$

We note here that it is trivial to see that our previous methods (SA, NSA and NSR) all share this property.

8.3.2 Null Space Preservation

Define P from SA, NSR, NSA, EMIN(r) or EMIN.

In order to show the relationship between the null space vectors throughout the mesh hierarchy, assume

$$A_f P^{(\text{tent})}c = 0$$

where $P^{(\text{tent})}c$ is not necessarily the zero vector (i.e. A_f is not necessarily full rank) and $A_f \in \mathbb{R}^{m\times m}$.

From the previous subsection and the above, we have:

$$Pc = P^{(tent)}c.$$

We also have:

$$RA_f Pc = 0.$$

But we notice that

$$A_c = RA_f P \in \mathbb{R}^{\frac{m}{2}\times\frac{m}{2}}$$

and

$$A_c c = RA_f Pc = R0 = 0,$$

showing that c is the (not necessarily trivial) null space of A_c.

From this, we see that we have a relationship between the null space of A_f and the null space of A_c. That is, if

$$AP^{(\text{tent})}c = A\begin{pmatrix} 1 & & & \\ 1 & & & \\ & 1 & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \\ & & & 1 \end{pmatrix}\begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_{\frac{m}{2}} \end{pmatrix} = A\begin{pmatrix} c_1 \\ c_1 \\ c_2 \\ c_2 \\ \vdots \\ c_{\frac{m}{2}} \\ c_{\frac{m}{2}} \end{pmatrix} = 0$$

then

$$A_c c = A_c\begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_{\frac{m}{2}} \end{pmatrix} = 0.$$

All of the examples presented in this thesis have an invertible matrix on the fine level. The above shows that, if we start with a full rank matrix on the fine level, then we need not worry that the coarse level matrices will become singular, since the null space of the coarse system matrices will also be only the zero vector (of proper dimension).

This holds throughout the mesh hierarchy, since we define coarse level matrices recursively using P and R from the specified method (i.e. $A_0 = R_1^0 R_2^1 A_2 P_1^2 P_0^1$). This is advantageous, as we use `solve` on the coarsest level. (For the interested reader, an alternative proof of this concept follows in the next subsection.)

8.4 Alternative Proof of the Invertibility of the Coarse Matrices

We provide here a sketch of another proof of the invertibility of the coarse matrices. Though the null space preserving property of these methods (previous subsection) shows the same thing, this proof provides an alternative way of looking at things.

A sketch was chosen because our matrices R, A_f and P are different depending on the problem. First, we chose an A_f based on the problem we wish to solve, but for each A_f of a given dimension, we have 4 different R's and 3 different P's to choose from (if we consider only the methods presented in this thesis). The following result holds in all cases.

Proof:
Assume A_f is a matrix with dimensions $(m \times m)$ and that A_f has full rank.

Due to the constraint on the near kernel and the structure of P, P has always $\frac{m}{2}$ linearly independent columns and only the trivial null space (of dimension $(\frac{m}{2} \times 1)$).

$A_f P$ has only the trivial null space since we assumed A_f is full rank. This means $A_f P$ maps only the zero vector of dimension $(\frac{m}{2} \times 1)$ to the zero vector of dimension $(m \times 1)$ since $A_f P$ has dimension $(m \times \frac{m}{2})$.

R has a nontrivial null space of dimension $\frac{m}{2}$, i.e. we have $\frac{m}{2}$ linearly independent non-zero vectors $(v_1, ..., v_{\frac{m}{2}})$ of dimension $(m \times 1)$ such that $Rv_i = 0$, $i = 1, ..., \frac{m}{2}$.

We can show that the non-trivial null space of R is not within the range of $A_f P$, i.e. $A_f P$ cannot map any vectors of dimension $(\frac{m}{2} \times 1)$ to any of the vectors $(v_1, ..., v_{\frac{m}{2}})$ of dimension $(m \times 1)$ which are the elements of the null space of R.

Thus, $A_c = R A_f P \in \mathbb{R}^{(\frac{m}{2} \times \frac{m}{2})}$ has only the trivial null space of dimension $(\frac{m}{2} \times 1)$.

$\implies$ A_c is invertible. $\square$

8.5 Results: Multigrid (Basic Examples)

We include the data tables which show the effect of the different methods on multigrid iterations when solving the Poisson equation, linear advection and 1D convection diffusion. All of these data tables represent a 4-level model ($l = 3$ on the finest level) with $\gamma = 2$, i.e. a 'W-cycle.' The text n/c indicates a failure to convergence after 300 multigrid iterations.

Table 8.1: Number of Multigrid Iterations, Poisson Equation (3.1)

Method	$m = 512$	$m = 1024$	$m = 2048$	$m = 4096$	$m = 8192$
SA	15	16	16	16	17
NSA	70	72	74	76	79
NSR	22	22	23	24	24
EMIN(r)	17	18	18	19	19
EMIN	17	18	18	19	19

1 pre- and post-smoothing step (Weighted Jacobi), $l = 3$, $\gamma = 2$, $\Delta x = \frac{1}{1+m}$, `solve` on the coarsest level, tol $= 10^{-8}$

Table 8.2: Number of Multigrid Iterations, Linear Advection (3.2)

Method	$m = 512$	$m = 1024$	$m = 2048$	$m = 4096$	$m = 8192$
SA	9	21	n/c	n/c	n/c
NSA	16	25	33	40	44
NSR	6	7	7	8	8
EMIN(r)	14	8	6	6	6
EMIN	14	8	6	6	6

1 pre- and post-smoothing step (Weighted Jacobi), iterations required to perform one time-step with $a = 2$, $\Delta t = .01$, $l = 3$, $\gamma = 2$, $\Delta x = \frac{2}{m}$, initial condition $u_0 = \sin(\pi x)$, for $x \in (0, 2]$, periodic boundary conditions, `solve` on the coarsest level, tol $= 10^{-8}$

Table 8.3: Number of Multigrid Iterations, 1D Convection Diffusion (3.3), $\varepsilon = 10^{-5}$ (Convective Case)

Method	$m = 512$	$m = 1024$	$m = 2048$	$m = 4096$	$m = 8192$
SA	n/c	n/c	n/c	n/c	n/c
NSA	54	50	44	35	26
NSR	9	9	9	9	11
EMIN(r)	7	8	8	9	11
EMIN	7	8	8	9	11

1 pre- and post-smoothing step (Weighted Jacobi), $l = 3$, $\gamma = 2$, $\Delta x = \frac{1}{1+m}$, $b = 1$, `solve` on the coarsest level, tol $= 10^{-8}$

Table 8.4: Number of Multigrid Iterations, 1D Convection Diffusion (3.3), $\varepsilon = 10^{-1}$ (Diffusive Case)

Method	$m = 512$	$m = 1024$	$m = 2048$	$m = 4096$	$m = 8192$
SA	12	13	13	13	14
NSA	55	57	59	60	62
NSR	13	13	14	15	16
EMIN(r)	14	14	14	15	15
EMIN	14	14	14	15	15

1 pre- and post-smoothing step (Weighted Jacobi), $l = 3$, $\gamma = 2$, $\Delta x = \frac{1}{1+m}$, $b = 1$, `solve` on the coarsest level, tol $= 10^{-8}$

8.5.1 Discussion of the Results (Basic Examples)

If we consider Table 8.1, we see that SA performs best when solving the Poisson equation, as expected. NSR, removing the restriction smoothing does not perform as well as SA. EMIN(r) and EMIN perform well, though not as well as SA and have a significantly larger set-up cost.

If we look to Table 8.2, we see that our newly proposed methods EMIN(r) and EMIN perform well when performing one time-step of linear advection, particularly as the dimension grows. This corresponds to solving linear advection at a larger CFL, where

$$\text{CFL} := \frac{a\Delta t}{\Delta x}.$$

We will investigate this idea further in the following chapter. Our previously discussed methods do not perform as well, with the exception of NSR, which skips the suboptimal restriction smoothing related to A^{T}.

If we look to Table 8.3, we see that EMIN and EMIN(r) perform extremely well in the highly convective case, though not with mesh width independence. Our method SA does not converge here. The method NSR, skipping the suboptimal smoothing on restriction, also performs well and is less costly to set up, compared to EMIN(r) and EMIN.

If we look to Table 8.4, we see that EMIN and EMIN(r) perform almost as well as SA in the diffusive case. This is promising, since SA was designed for symmetric systems. NSR performs as well or better than EMIN(r) and EMIN, and has a simpler set-up.

8.5.2 A Note on EMIN vs EMIN(r)

In all of the experiments presented in this thesis, the two newly proposed methods perform identically, though EMIN saves considerably on setup cost (one half). It is therefore advised that EMIN be used over EMIN(r), even if perhaps we lose some optimality in energy minimization.

In the case of linear advection with periodic boundary conditions, the damping parameters on restriction and prolongation are all identical anyway. In the case of 1D convection diffusion, the largest difference between damping parameters was found in the first decimal place. This is interesting to note, considering these local damping parameters were designed for the convection diffusion equation. This result is true both when $\varepsilon = 10^{-1}$ and $\varepsilon = 10^{-5}$, i.e. in the convective and diffusive case.

Chapter 9

Multigrid Method Results (Part 2)

In this chapter, we go a little bit further in our investigation of the different methods for prolongation and restriction. We first apply some more advanced smoothers in pre-smoothing when performing one time-step of linear advection, then we experiment with residual smoothing when performing one time-step of the same equation. We finish this chapter by explaining how to solve a 2D convection diffusion equation in a multigrid setting, and providing results on how our different methods performed. We note that, just as we had previously, the text n/c in the data tables indicates a failure to convergence after 300 multigrid iterations.

9.1 Linear Advection with Optimized Runge-Kutta Smoothers

9.1.1 Background on Runge-Kutta Smoothers

In [2], Birken used a multigrid method with s-stage explicit Runge-Kutta methods for presmoothing and a smoothing step on the coarsest level (instead of a direct `solve`), in order to solve the linear advection (3.2). These smoothers require more set-up than Weighted Jacobi, but are also more powerful. The smoothers approximate the solution to the initial value problem

$$u_t = f(u),\ u_n = u(t_n),$$

and are of the form

$$u_0 = u_n$$
$$u_j = u_n + \alpha_j \Delta t^* f(u_{j-1}),\ j = 1, ..., s-1$$
$$u_{n+1} = u_n + \Delta t^* f(u_{s-1}),$$

where α_j and Δt^* are free parameters and we make the consistency requirement that $\alpha_j \in [0, 1]$. As Birken explains in [2], one step of the Runge-Kutta smoother consists of performing one step of the Runge-Kutta scheme for solving

$$u_{t^*} = u^n - Au(t^*) = f(u),\ u(t_0^*) = u^n.$$

We note here that Δt^* refers to a pseudo time-step and actually has no physical meaning. We define Δt^* on each level as

$$\Delta t_l^* = c\Delta t^*/\nu,$$

where $\nu = a\Delta t$. Thus, it is the c and α_j which Birken has chosen in an optimal way.

Two different strategies are presented for optimizing these free parameters. First, the free parameters α_j and c were optimized such that the smoothers removed nonsmooth

error components fast, for a variety of CFL numbers. Then, the free parameters were optimized in order to minimize the spectral radius of M, where

$$u^{(k+1)} = Mu^{(k)} + N^{-1}b,$$

corresponds to one iteration of our multigrid scheme. This optimization was more costly.

We note that in [2], Birken took $R^{(\text{tent})} = \frac{1}{2}P^{(\text{tent})}$, so we do this also in the experiments involving the Runge-Kutta smoothers which follow.

9.1.2 Experiment with Runge-Kutta Smoothers

We would like to see if using Birken's optimized Runge-Kutta smoothers can speed up convergence when using our methods to solve (3.2). We know that the methods SA, NSR, EMIN(r) and EMIN provide extra smoothing within prolongation, restiction or both. Since the method NSA uses only a tentative restrictor and tentative prolongator, the method NSA corresponds to the same problem presented in [2].

The quantity

$$\text{CFL} := \frac{a\Delta t}{\Delta x}$$

refers to a ratio between time-step and cells on a domain when solving PDEs with explicit methods. Since we use a discretization with the implicit Euler method, we do not need to consider a certain maximum CFL for stability. Instead, it is interesting to look at what happens for large CFL values, since the method can handle large time-steps. The data tables for medium, large and extremely large CFL numbers follow in the next section.

9.1.3 Data Tables, Linear Advection with R-K Smoothers

We provide here the data tables which compare the results of performing one time-step of linear advection using different methods for prolongation and restriction and different smoothers for pre-smoothing and smoothing on the coarsest grid. The code for the Runge-Kutta smoothers was written by Versbach. We have optimized parameters α_j and c for CFL $= 3$ and CFL $= 24$. CFL $= 240$ represents an extremely large CFL, for which we do not have optimized parameters. We instead use the optimized parameters we have from CFL $= 24$, the largest CFL Birken considered in [2].

We note that the abbreviation 'RK2 (sm.)' refers to using a 2-stage explicit Runge-Kutta method with optimized parameters such that the smoother removes the nonsmooth error components fast and 'RK2 (rho)' refers to a 2-stage explicit Runge-Kutta method with optimized parameters to minimize the spectal radius of M.

We include here a column which corresponds to using Weighted Jacobi as a smoother (abbreviation 'W.J.'). This was done so that the reader would have a basis from which to interpret the new results.

Table 9.1: Number of Multigrid Iterations, Linear Advection (3.2), CFL=3 (medium)

Method	RK2 (sm.)	RK3 (sm.)	RK4 (sm.)	RK2 (rho)	RK3 (rho)	RK4 (rho)	W.J.
SA	7	4	4	12	4	4	18
NSA	8	5	5	13	5	5	19
NSR	7	5	4	8	4	4	8
EMIN(r)	7	5	4	10	5	5	13
EMIN	7	5	4	10	5	5	13

Iterations required to perform one time-step with $m = 48$, $a = 2$, $\nu_1 = 2$ pre- and $\nu_2 = 0$ post-smoothing step, $\Delta x = \frac{2}{m}$, $\Delta t = \frac{1}{16}$, ν_1 smoothing steps used on the coarse level, initial condition $u_0 = \sin(\pi x)$, for $x \in (0, 2]$, periodic boundary conditions, tol $= 10^{-8}$, $l = 2$, $\gamma = 1$

Table 9.2: Number of Multigrid Iterations, Linear Advection (3.2), CFL=24 (large)

Method	RK2 (sm.)	RK3 (sm.)	RK4 (sm.)	RK2 (rho)	RK3 (rho)	RK4 (rho)	W.J.
SA	n/c	n/c	n/c	n/c	n/c	n/c	n/c
NSA	31	20	15	22	20	14	63
NSR	21	14	12	20	12	13	36
EMIN(r)	19	13	10	19	9	13	22
EMIN	19	13	10	19	9	13	22

Iterations required to perform one time-step with $m = 48$, $a = 2$, $\nu_1 = 2$ pre- and $\nu_2 = 0$ post-smoothing step, $\Delta x = \frac{2}{m}$, $\Delta t = \frac{1}{2}$, ν_1 smoothing steps used on the coarse level, initial condition $u_0 = \sin(\pi x)$, for $x \in (0, 2]$, periodic boundary conditions, tol $= 10^{-8}$, $l = 2$, $\gamma = 1$

Table 9.3: Number of Multigrid Iterations, Linear Advection (3.2), CFL=240 (extremely large)

Method	RK2 (sm.)	RK3 (sm.)	RK4 (sm.)	RK2 (rho)	RK3 (rho)	RK4 (rho)	W.J.
SA	n/c	n/c	n/c	n/c	n/c	n/c	n/c
NSA	165	79	48	128	53	76	n/c
NSR	50	33	27	51	33	44	107
EMIN(r)	35	23	18	28	18	21	60
EMIN	35	23	18	28	18	21	60

Iterations required to perform one time-step with $m = 48$, $a = 20$, $\nu_1 = 2$ pre- and $\nu_2 = 0$ post-smoothing step, $\Delta x = \frac{2}{m}$, $\Delta t = \frac{1}{2}$, ν_1 smoothing steps used on the coarse level, initial condition $u_0 = \sin(\pi x)$, for $x \in (0, 2]$, periodic boundary conditions, tol $= 10^{-8}$, $l = 2$, $\gamma = 1$

9.1.4 Discussion of the Data Tables, Linear Advection with R-K Smoothers

When we consider a medium CFL condition, we can see above that the choice of method for restriction and prolongation does not make a large difference in the number of multigrid iterations required for convergence. The exception here is 'RK2 (rho),' which performs best with the method NSR. The optimized smoothers perform much better than Weighted Jacobi in this example, which is not surprising considering these methods were optimized for solving this particular problem.

At a large CFL, the choice of method begins to matter a bit more. Our newly proposed methods, EMIN(r) and EMIN are the best performing methods at large CFL (though they

tie with NSR for 'RK4 (rho)'). All the Runge-Kutta smoothers perform significantly better than Weighted Jacobi at CFL = 24.

At extremely large CFL (i.e. CFL = 240), we see that the newly proposed methods EMIN(r) and EMIN really pay off. We need to consider that these methods have a much larger set-up cost compared to the others, but the convergence at extremely large CFL numbers makes them worthwhile.

9.2 Residual Smoothing, Linear Advection

The method SA adds additional smoothing on restriction and prolongation. In this section we will explore what happens when we do an extra smoothing of the residual within the multigrid method.

9.2.1 Optional Second Order Lowpass Filter

In [10], Söderlind suggests adding a second-order lowpass filter to smooth the residual before restriction and thus speed up convergence. By this, we mean update r_l with

$$r_l \leftarrow \frac{1}{4}\begin{pmatrix} 2 & 1 & & \\ 1 & \ddots & \ddots & \\ & \ddots & \ddots & 1 \\ & & 1 & 2 \end{pmatrix} r_l.$$

The above equation shows that this optional step adds additional matrix-vector products, and is, thus, more costly. However, this simple modification can be a cheap way to speed up NSA when solving the linear advection equation, as we will see in the results in Table 9.4.

9.2.2 Residual Averaging

Similarly, in [6], Jameson suggests implicit averaging of the residuals with their neighbors, i.e.

$$-\varepsilon\bar{r}_{j-1} + (1+2\varepsilon)\bar{r}_j - \varepsilon\bar{r}_{j+1} = \bar{r}_j$$

with $\varepsilon = .5$. This increases the implicitness of the method, which allows us to take a larger time-step.

In practice, we apply two iterations of Weighed Jacobi to the following system:

$$\begin{pmatrix} 1+2\varepsilon & -\varepsilon & & \\ -\varepsilon & \ddots & \ddots & \\ & \ddots & \ddots & -\varepsilon \\ & & -\varepsilon & 1+2\varepsilon \end{pmatrix} \begin{pmatrix} \hat{r}_{h_1} \\ \vdots \\ \vdots \\ \hat{r}_{h_m} \end{pmatrix} = \begin{pmatrix} r_{h_1} \\ \vdots \\ \vdots \\ r_{h_m} \end{pmatrix}$$

in order to solve for $\hat{r}_h$. We then take this vector $\hat{r}_h$ to be our residual and proceed with the algorithm as usual. Just like with Söderlind's lowpass filter for smoothing the residual, we apply this step before we restrict the residual.

We test these ideas when performing one time-step of the linear advection equation with periodic boundary conditions, i.e. solving (3.2) with our previously described methods for restriction and prolongation. We use a simple Weighted Jacobi smoother for pre-smoothing

and then this smoother again on the coarsest level, instead of solving. First, we look at what happens with time-step $\Delta t = \frac{1}{16}$ using no filter, Jameson's residual averaging and Söderlind's lowpass filter, then again at $\Delta t = \frac{1}{2}$. These results are in Tables 9.4 and 9.5. Table 9.6 compares how many iterations our methods need to converge to solve 1 iteration of the same equation without any filter and $\Delta t = \frac{1}{16}$, then how many iterations using Jameson's implicit averaging but with a time-step which is twice as large ($\Delta t = \frac{1}{8}$).

9.2.3 Results, Linear Advection with Residual Smoothing

Table 9.4: Number of Multigrid Iterations, Linear Advection (3.2), CFL=3

Method	No filter	Implicit residual averaging	Lowpass filter
SA	18	13	13
NSA	19	14	14
NSR	8	13	13
EMIN(r)	13	13	13
EMIN	13	13	13

Iterations required when performing one time-step with $m = 48$, $a = 2$, $\nu_1 = 2$ pre- and $\nu_2 = 0$ post-smoothing step (Weighted Jacobi relaxation), $\Delta x = \frac{2}{m}$, $\Delta t = \frac{1}{16}$, ν_1 iterations of the smoother used on the coarse level, initial condition $u_0 = \sin(\pi x)$, for $x \in (0, 2]$, periodic boundary conditions, tol $= 10^{-8}$, $l = 2$, $\gamma = 1$, $\varepsilon = .5$

Table 9.5: Number of Multigrid Iterations, Linear Advection (3.2), CFL=24

Method	No filter	Implicit residual averaging	Lowpass filter
SA	n/c	n/c	n/c
NSA	63	61	62
NSR	35	39	38
EMIN(r)	22	27	26
EMIN	22	27	26

Iterations required when performing one time-step with $m = 48$, $a = 2$, $\nu_1 = 2$ pre- and $\nu_2 = 0$ post-smoothing step (Weighted Jacobi relaxation), $\Delta x = \frac{2}{m}$, $\Delta t = \frac{1}{2}$, ν_1 iterations of the smoother used on the coarse level, initial condition $u_0 = \sin(\pi x)$, for $x \in (0, 2]$, periodic boundary conditions, tol $= 10^{-8}$, $l = 2$, $\gamma = 1$, $\varepsilon = .5$

Table 9.6: Number of Multigrid Iterations, Linear Advection (3.2)

Method	No filter $\Delta t = \frac{1}{16}$, CFL=3	Implicit residual averaging $\Delta t = \frac{1}{8}$, CFL=6
SA	18	17
NSA	19	18
NSR	8	17
EMIN(r)	13	15
EMIN	13	15

Iterations required when performing one time-step with $m = 48$, $a = 2$, $\nu_1 = 2$ pre- and $\nu_2 = 0$ post-smoothing step (Weighted Jacobi relaxation), $\Delta x = \frac{2}{m}$, ν_1 iterations of the smoother on the coarsest level, initial condition $u_0 = \sin(\pi x)$, for $x \in (0, 2]$, periodic boundary conditions, tol $= 10^{-8}$, $l = 2$, $\gamma = 1$, $\varepsilon = .5$

9.2.4 Discussion of the Results, Residual Smoothing

From Table 9.4, we can see that for CFL= 3, applying either Söderlind's second-order lowpass filter or Jameson's implicit residual averaging can be a cheap way to speed up convergence when using the methods NSA or SA. Unfortunately, this filter does not help us in conjunction with NSR, EMIN(r) or EMIN. Since the method SA requires extra set-up cost, the method NSA with the lowpass filter is our best option here.

In Table 9.5, we see that at CFL= 24, we do not benefit from applying a lowpass filter or implicit residual averaging in the same way we did at CFL= 3. Thus, residual smoothing is not recommended here.

From Table 9.6, we can see that applying Jameson's idea of residual averaging with the method NSA allows us to take a time-step which is twice as large (double the CFL) in 1 fewer multigrid iteration. Since residual averaging corresponds to just 2 iterations of Weighted Jacobi on the fine grid, this can be seen as a cheap way to speed up convergence of the method NSA, which is our cheapest option for restriction and prolongation. SA also performs slightly better with residual averaging, though SA is not an ideal method for the (nonsymmetric) linear advection equation. Unfortunately, our preferred methods for linear advection (NSR, EMIN(r) and EMIN) do not show any improvement with residual averaging. Instead, residual averaging in these cases corresponds to more work for a slower convergence and is, thus, not recommended.

9.3 Convection Diffusion in 2D

9.3.1 Tentative Restriction and Prolongation in 2D

We begin here by noting that the tentative prolongator and tentative restrictor are different in 2D, since we need to prolong and restrict in both the x and y directions. To see this, we provide a concrete example with 36 unknowns.

Consider Figure 9.1, where there are 36 grid points on the fine grid (black and white circles), which get reduced to 4 grid points on the coarse grid (black cirles). In this way, we divide the number of unknowns by 9 each time we move to the next coarsest grid, and each contributing point on the 'finer' grid has equal weight on the next coarsest grid. This is the same coarsening Sala and Tuminaro used in [9]. The restriction operator in this case is thus:

$$R_{2D}^{(\text{tent})} = \frac{1}{9}\begin{bmatrix} \hat{R} & & \\ & \hat{R} & \end{bmatrix} \in \mathbb{R}^{4\times 36},$$

where

$$\hat{R} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}.$$

We note here that $P_{2D}^{(\text{tent})} = 9R_{2D}^{(\text{tent})\mathsf{T}}$ (different from 1D where $P^{(\text{tent})} = R^{(\text{tent})\mathsf{T}}$), and we continue to use a Galerkin projection, i.e. $A_{l-1} = R_{2D}A_lP_{2D}$. Additionally, we note that forming the tentative restrictor and prolongator as described above keeps the constant vector (user-defined near kernel) in the range of both of these operators, thus preserving the properties we have previously discussed (namely, invertible coarse grid matrices, under the assumption of a full rank fine grid system matrix). We note that we use the discretization given in (3.6) and to solve this problem.

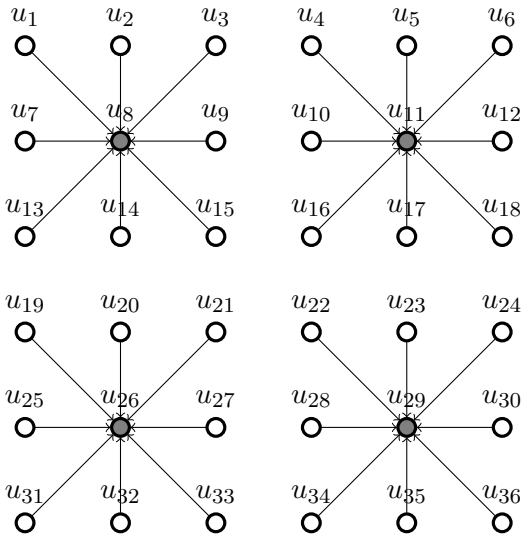


Figure 9.1: Coarsening of the Grid in 2D

9.3.2 Methods for Restriction and Prolongation in 2D

The methods NSA, SA and NSR in 2D are formed in a way analogous to 1D, i.e. the prolongators and restrictors are either taken as the tentative operators, or one iteration of Jacobi damping is applied to the columns and/ or rows of the tentative operators. Our newly proposed methods EMIN(r) and EMIN, require a bit more consideration in 2D, but we note here that the formulas for restriction and prolongation (i.e. (7.3) and (7.8)), still hold in 2D for all of our methods.

The difficulty with EMIN(r) and EMIN in 2D comes from setting up the matrix $\Omega^{(\text{fine})}$. Consider again (7.4) through (7.7), where we first found the entries $w_j^{(\text{int})}$ of the $(m \times m)$ diagonal matrix of damping parameters $\Omega^{(\text{int})}$ from the entries of $\Omega^{(p)}$. The entries of $\Omega^{(p)}$ represented optimal damping parameters in terms of energy minimization and the entries of $\Omega^{(\text{int})}$ represented interpolated values, which we need in order to meet the null space preserving property (see Chapter 7). Once we had the entries of $\Omega^{(\text{int})}$, finding the entries of $\Omega^{(\text{fine})}$ was simple (7.5).

In 2D, finding the proper $w_j^{(\text{int})}$ is more challenging. We include here the closed formula:

$$w^{(\text{int})}_{3i\sqrt{\frac{m}{9}}+3k+j+lm\sqrt{\frac{m}{9}}} = w^{(p)}_{l\sqrt{\frac{m}{9}}+k},\ l=(1,...,\sqrt{\frac{m}{9}}), k=(1,...,\sqrt{\frac{m}{9}}), i=(1,...,3), j=(1,...,3)$$

for all possible unknowns, m.

9.3.3 Results of 2D Convection Diffusion

We provide in this section data tables showing how our methods performed, error plots and plots of the solutions found for the 2D convection diffusion equation, i.e.

$$\begin{cases} -\varepsilon\Delta u + b\cdot\nabla u = f & \text{in } (0,1)\times(0,1) \\ u = 0 & \text{on the boundaries,} \end{cases}$$

over the vector fields 'bent pipe' (3.4) and 'recirc' (3.5) with discretization given in (3.6).

We provide also a table showing the relative error after multigrid terminates for both

vector fields. We note that we provide data tables on iterations for convergence of this experiment performed with two different smoothers, i.e. Weighted Gauss-Seidel (SOR) and Symmetric Gauss-Seidel (SSOR). Discussion of these results follows after. We note that we limit the error plots and solution plots in the remainder of this section to those corresponding to the method EMIN with smoother SSOR. This is to avoid being repetitive, as the plots look identical when using the other methods. We note, however, that this has been checked. Additionally, Figure 9.4 shows the plots of iterations vs residual for the 2D convection diffusion equation.

Additionally, we note that the experiments in this section consider $m = 50625$ unknowns, a comparatively much larger quantity of unknowns than what was considered in the other experiments in this thesis. This choice of quantity of unknowns was made due to the large relative error upon termination (when compared to the discrete representation of the exact solution, see Table 9.11) of some of the experiments, and the order one character of the discretization for convection diffusion in 2D (3.6).

Table 9.7: Number of Multigrid Iterations, 2D Convection Diffusion (3.6) (Bent pipe) with SOR smoother

Method	$\varepsilon = 10^{-1}$	$\varepsilon = 10^{-2}$	$\varepsilon = 10^{-3}$	$\varepsilon = 10^{-4}$	$\varepsilon = 10^{-5}$
SA	50	47	n/c	n/c	n/c
NSA	191	135	93	161	n/c
NSR	81	67	70	150	220
EMIN(r)	37	37	66	145	217
EMIN	37	37	66	145	217

Iterations required to solve 2D convection diffusion (bent pipe), $m = 225^2 = 50625$, $\nu_1 = 1$ pre- and $\nu_2 = 1$ post-smoothing step, $\Delta x = \frac{1}{\sqrt{m}+1} = \frac{1}{226}$, `solve` used on the coarse level, tol $= 10^{-8}$, $l = 2$, $\gamma = 1$, n/c indicates no convergence after 300 iterations

Table 9.8: Number of Multigrid Iterations, 2D Convection Diffusion (3.6) (Bent pipe) with SSOR smoother

Method	$\varepsilon = 10^{-1}$	$\varepsilon = 10^{-2}$	$\varepsilon = 10^{-3}$	$\varepsilon = 10^{-4}$	$\varepsilon = 10^{-5}$
SA	36	31	27	43	59
NSA	157	105	49	55	66
NSR	66	48	32	51	65
EMIN(r)	24	23	27	49	64
EMIN	24	23	27	49	64

Iterations required to solve 2D convection diffusion (bent pipe), $m = 225^2 = 50625$, $\nu_1 = 1$ pre- and $\nu_2 = 1$ post-smoothing step, $\Delta x = \frac{1}{\sqrt{m}+1} = \frac{1}{226}$, `solve` used on the coarse level, tol $= 10^{-8}$, $l = 2$, $\gamma = 1$

Discussion of the Results, 'Bent pipe'

When solving the 2D convection diffusion equation with the SOR or SSOR smoother over the vector field 'bent pipe' (Table 9.7 and Table 9.8), we notice that for the methods SA, NSA and NSR, as we decrease epsilon such that $\varepsilon = 10^{-1}$, $\varepsilon = 10^{-2}$, $\varepsilon = 10^{-3}$, the number of iterations required for convergence decreases. As we then further decrease epsilon such that $\varepsilon = 10^{-4}$, $\varepsilon = 10^{-5}$, the number of iterations required for convergence increases. This is due to the character of the vector field. We do not see such a pattern with the methods EMIN(r) or EMIN combined with either smoother. Instead, with our newly proposed

Table 9.9: Number of Multigrid Iterations, 2D Convection Diffusion (3.6) (Recirc) with SOR smoother

Method	$\varepsilon = 10^{-1}$	$\varepsilon = 10^{-2}$	$\varepsilon = 10^{-3}$	$\varepsilon = 10^{-4}$	$\varepsilon = 10^{-5}$
SA	47	51	n/c	n/c	n/c
NSA	202	202	213	n/c	n/c
NSR	85	87	133	289	n/c
EMIN(r)	36	40	101	284	n/c
EMIN	36	40	101	284	n/c

Iterations required to solve 2D convection diffusion (recirc), $m = 225^2 = 50625$, $\nu_1 = 1$ pre- and $\nu_2 = 1$ post-smoothing step, $\Delta x = \frac{1}{\sqrt{m}+1} = \frac{1}{226}$, `solve` used on the coarse level, tol $= 10^{-8}$, $l = 2$, $\gamma = 1$

Table 9.10: Number of Multigrid Iterations, 2D Convection Diffusion (3.6) (Recirc) with SSOR smoother

Method	$\varepsilon = 10^{-1}$	$\varepsilon = 10^{-2}$	$\varepsilon = 10^{-3}$	$\varepsilon = 10^{-4}$	$\varepsilon = 10^{-5}$
SA	37	38	55	88	123
NSA	167	165	160	157	177
NSR	70	71	85	126	164
EMIN(r)	22	23	55	115	158
EMIN	22	23	55	115	158

Iterations required to solve 2D convection diffusion (recirc), $m = 225^2 = 50625$, $\nu_1 = 1$ pre- and $\nu_2 = 1$ post-smoothing step, $\Delta x = \frac{1}{\sqrt{m}+1} = \frac{1}{226}$, `solve` used on the coarse level, tol $= 10^{-8}$, $l = 2$, $\gamma = 1$

methods, as epsilon decreases, the number of iterations required for convergence increases.

For larger values of epsilon (i.e. the diffusive case) and the smoother SOR or SSOR, the newly proposed methods actually perform better than SA. For smaller values of epsilon (i.e. the convective case) and the smoother SSOR, SA actually performs best, though SA does not converge after 300 iterations with the smoother SOR.

We note here that at larger values of epsilon, we recommend the smoother SOR. This is because SOR is half of the work of SSOR, yet we can converge in fewer than double iterations using SOR compared to SSOR. However, when solving problems with smaller values of epsilon, SSOR is recommend. In some cases, we can converge in nearly a quarter as many iterations using SSOR, compared to SOR (e.g. EMIN(r)/EMIN with $\varepsilon = 10^{-5}$).

Discussion of the Results, 'Recirc'

When solving the 2D convection diffusion equation with the SOR or SSOR smoother over the vector field 'recirc' (Table 9.9 and Table 9.10), we see that as epsilon becomes smaller, we need more iterations to converge. When we consider larger values of epsilon (e.g. $\varepsilon = 10^{-1}$, $\varepsilon = 10^{-2}$), EMIN(r) and EMIN perform the best. We must note that SA with smaller values of epsilon (e.g. $\varepsilon = 10^{-3}$, $\varepsilon = 10^{-4}$, $\varepsilon = 10^{-5}$) performs well when combined with SSOR smoothing, but does not converge within 300 iterations when combined with SOR smoothing. In general, for either smoother, as epsilon gets smaller, NSR performs almost as well as EMIN(r) and EMIN, and is less costly to implement, though the relative error upon termination of this experiment is very important to note (see following section).

Just as when we considered the 'bent pipe' problem above, we have savings when us-

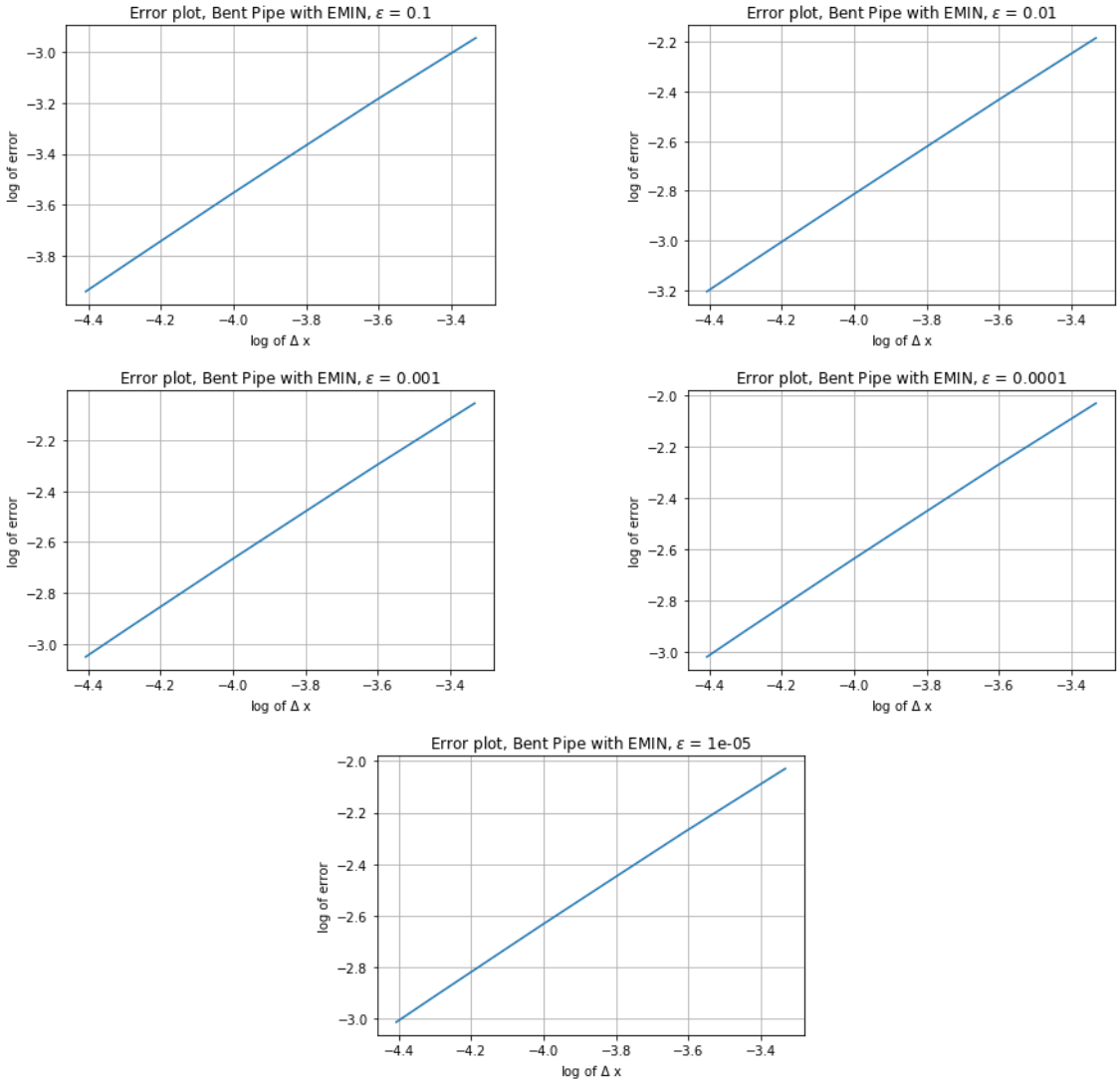


Figure 9.2: Error plots for the solutions found for 2D convection diffusion (3.6) ('bent pipe') using the method EMIN, SSOR smoothing

ing the SOR smoother with larger values of epsilon. However, as epsilon gets smaller, SSOR is worth the cost. We can see this in particular for $\varepsilon = 10^{-4}$ solved with the methods EMIN(r) and EMIN. When using SSOR, we needed 115 iterations to converge, while SOR required more than double this (284 iterations).

General Reflections, 2D Convection Diffusion

The previously described results are not exactly what we expected would happen, especially when we consider both vector fields with the more powerful smoother SSOR, which Sala and Tuminaro considered in [9]. We thought SA would perform very well in the diffusive case (symmetric) and horribly in the convective case (nonsymmetric), while EMIN/EMIN(r) would perform quite well in the diffusive case and very well in the convective case. What our prediction failed to take into account was that we would need to adapt the discretization in order to preserve stability in 2D convection diffusion with small values for epsilon (i.e. the coefficient in front of the Laplace operator). One can see directly from looking at the stencil in (3.6) that the discretization matrix will be rather symmetric (especially compared to the discretization matrix for convection diffusion in 1D, (3.3)) and thus, SA

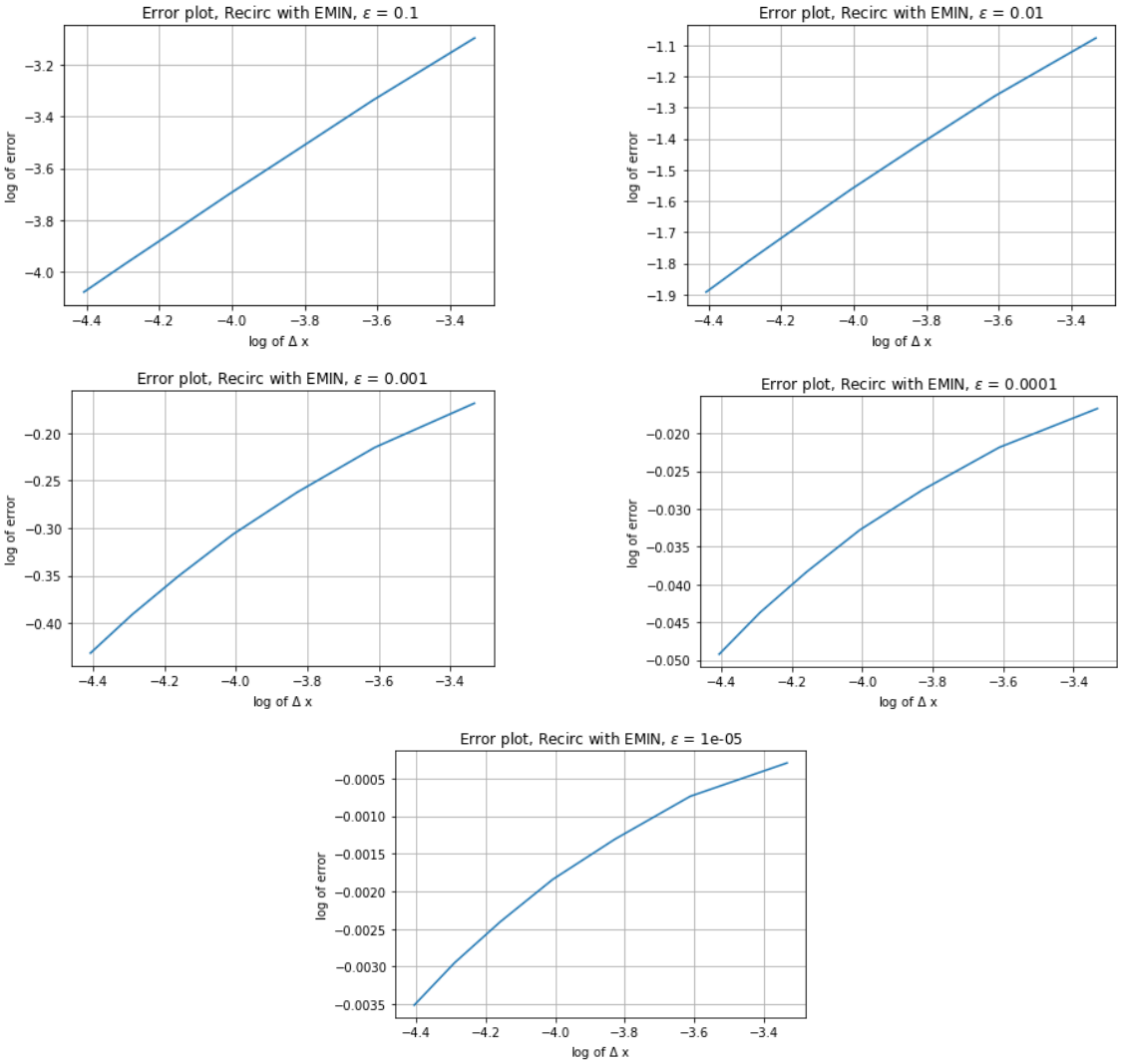
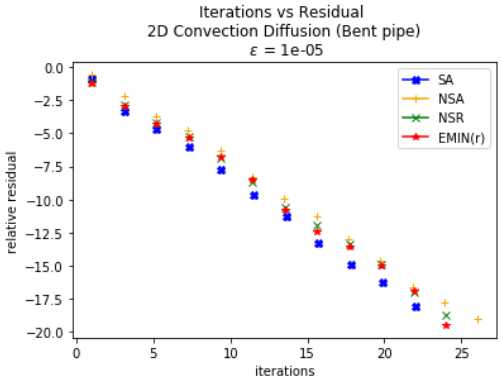


Figure 9.3: Error plots for the solutions found for 2D convection diffusion (3.6) ('recirc') using the method EMIN, SSOR smoothing

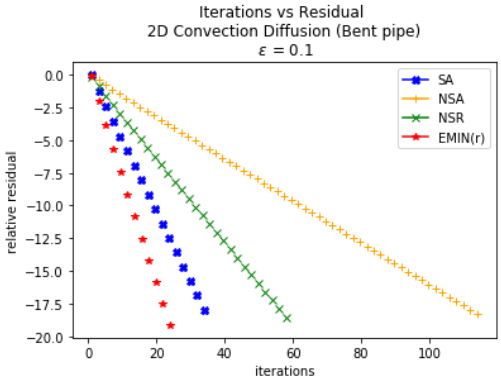
will be a reasonable choice, even in the convective case. This reflection matches the results we see with the smoother SSOR.

We note that, though EMIN/EMIN(r) always performed better than NSA (i.e. no additional smoothing on restriction and prolongation), sometimes the gain was only marginal. SA requires less in terms of set-up cost when compared to EMIN(r) and EMIN, and is a good method for 2D convection diffusion, especially for small values of epsilon in conjunction with the more costly smoother SSOR.

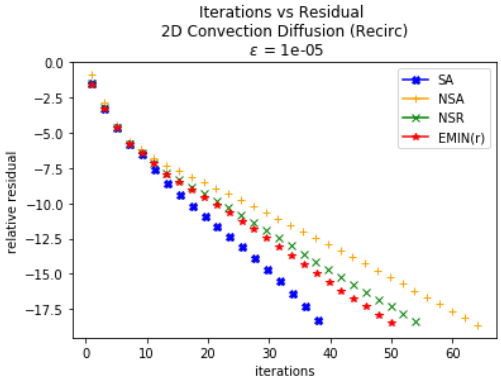
We take a second to note that the error plots for solving the 2D convection diffusion equation over the vector field 'bent pipe' (Figure 9.2) all have slope one, indicating a first order discretization, as we anticipated. The error plots for the same equation over the vector field 'recirc' (Figure 9.3) show slope one for large values of epsilon and slopes smaller than one for smaller values of epsilon. This corresponds to the results we get from the plots of the solution, i.e. for very small epsilon, we get a solution which converges in multigrid (small relative residual), but which still has a rather large relative error (measured by comparing our solution to the discrete representation of the solution we know we should



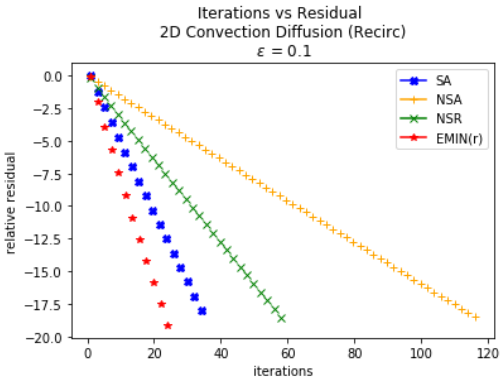
(a) Symmetric Gauss-Seidel smoothing, $\nu_1 = \nu_2 = 1$, $l = 2$, $\gamma = 1$, $m = 2025$, $\Delta x = \frac{1}{\sqrt{m}+1}$



(b) Symmetric Gauss-Seidel smoothing, $\nu_1 = \nu_2 = 1$, $l = 2$, $\gamma = 1$, $m = 2025$, $\Delta x = \frac{1}{\sqrt{m}+1}$



(c) Symmetric Gauss-Seidel smoothing, $\nu_1 = \nu_2 = 1$, $l = 2$, $\gamma = 1$, $m = 2025$, $\Delta x = \frac{1}{\sqrt{m}+1}$



(d) Symmetric Gauss-Seidel smoothing, $\nu_1 = \nu_2 = 1$, $l = 2$, $\gamma = 1$, $m = 2025$, $\Delta x = \frac{1}{\sqrt{m}+1}$

Figure 9.4: Iterations vs Residual, 2D Convection Diffusion (3.6)

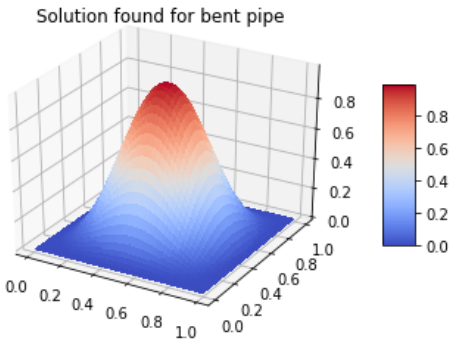
expect, namely $u(x, y) = \sin^2(\pi x)\sin^2(\pi y)$).

Following this idea, we can see the size of the relative error when solving the 2D convection diffusion equation for varying values of epsilon (SSOR smoother) in Table 9.11. We can see that the relative error when solving the 2D convection diffusion equation over the vector field 'recirc' with small values of epsilon is rather large. This explains the plots of the solutions we found for this equation and the corresponding error plots.

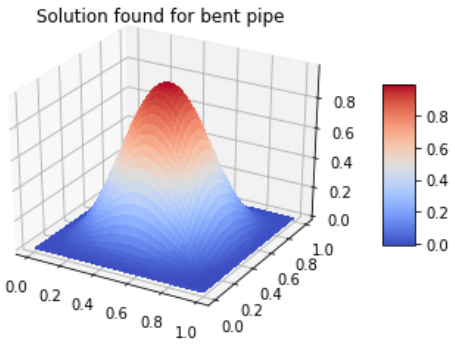
Figure 9.5 and Figure 9.6 show the plots for various values of epsilon when solving the 2D convection diffusion problem with the method EMIN. In the case of 'bent pipe,' all of the plots look the same, though one can see when $\varepsilon = 10^{-4}$ or $\varepsilon = 10^{-5}$, there is some variation on the boundary. In the case of 'recirc,' one can see that the relative error is rather large as epsilon becomes small, though we still converge in a multigrid scheme.

A Final Word on 2D Convection Diffusion

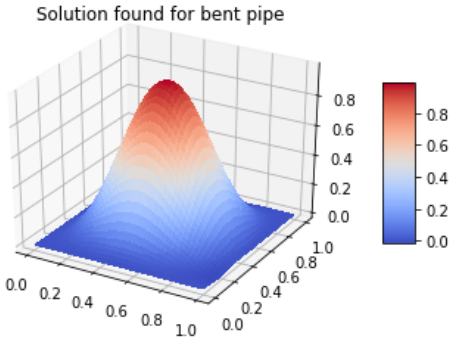
We note that in [9], Sala and Tuminaro reported very exciting results when using EMIN(r)/EMIN to solve the 2D convection diffusion equation over both vector fields. We solved the same problem they did, but in a simpler way, i.e. we solved the linear equation systems that arised from this discretized PDE via a multigrid scheme with different methods for prolongation and restriction. They on the other hand used the multigrid schemes with different methods for prolongation and restriction as preconditioners for GMRES, a different algorithm to solve the linear equation systems that arise from this discretized PDE. Their



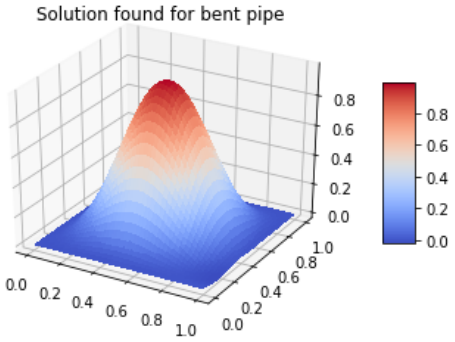
(a) $\varepsilon = 10^{-1}$



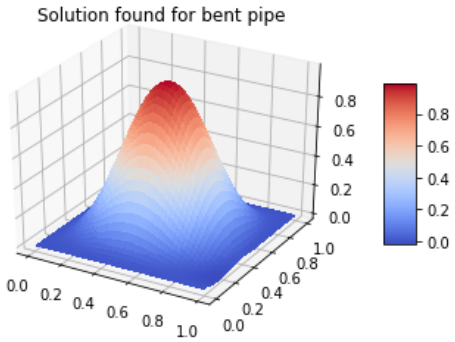
(b) $\varepsilon = 10^{-2}$



(c) $\varepsilon = 10^{-3}$



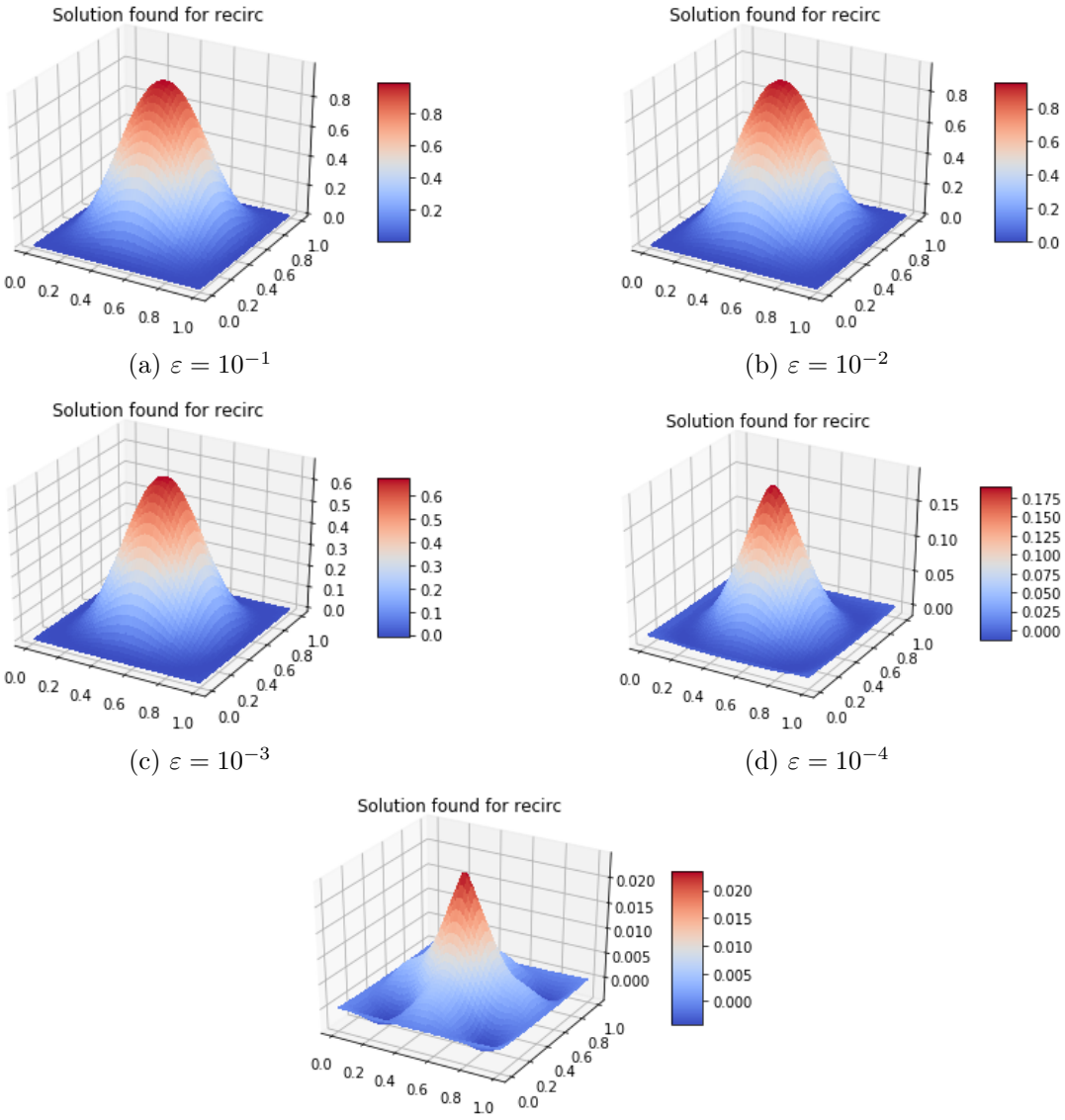
(d) $\varepsilon = 10^{-4}$



(e) $\varepsilon = 10^{-5}$

Figure 9.5: Plots of the solutions found for 2D convection diffusion (3.6) ('bent pipe') using the method EMIN, $m = 50625$, SSOR smoothing

results are more exciting, but this work offers a different perspective.



(a) $\varepsilon = 10^{-1}$

(b) $\varepsilon = 10^{-2}$

(c) $\varepsilon = 10^{-3}$

(d) $\varepsilon = 10^{-4}$

(e) $\varepsilon = 10^{-5}$

Figure 9.6: Plots of the solutions found for 2D convection diffusion (3.6) ('recirc') using the method EMIN, $m = 50625$, SSOR smoothing

Table 9.11: Relative Error upon Termination, 2D Convection Diffusion (3.6) with SSOR Smoother

Vector field	$\varepsilon = 10^{-1}$	$\varepsilon = 10^{-2}$	$\varepsilon = 10^{-3}$	$\varepsilon = 10^{-4}$	$\varepsilon = 10^{-5}$
Bent pipe	.00726	.01499	.01771	.01836	.01845
Recirc	.00633	.06036	.39695	.87472	.98767

Relative error rounded to 5 decimal places, $m = 225^2 = 50625$, $\nu_1 = 1$ pre- and $\nu_2 = 1$ post-smoothing step, $\Delta x = \frac{1}{\sqrt{m}+1} = \frac{1}{226}$, `solve` used on the coarse level, tol $= 10^{-8}$, $l = 2$, $\gamma = 1$

Chapter 10

Experiment with the Near Kernel, Poisson Equation

As mentioned previously, the constant vector is not the only possible user-defined near kernel for the discrete Laplace operator [9]. We can choose the near kernel as any vector(s) x such that $Ax \approx 0$, with the implication that this vector(s) will be represented exactly within the mesh hierarchy. We experiment below with taking a discrete representation of a linear function plus a constant as the near kernel vector for the Laplace operator. For example, take:

$$P^{(\text{tent})} = \begin{pmatrix} 1 & & & & \\ 1+\frac{1}{m} & & & & \\ & 1+\frac{2}{m} & & & \\ & 1+\frac{3}{m} & & & \\ & & & \ddots & \\ & & & & 1+\frac{m-2}{m} \\ & & & & 1+\frac{m-1}{m} \end{pmatrix} \in \mathbb{R}^{m \times \frac{m}{2}}$$

with $R^{(\text{tent})} = (P^{(\text{tent})})^{\mathsf{T}}$.

We take a second to remember that the goal of our restriction is to resolve low energy eigenmodes (i.e. error components) on the coarse grid, where they appear to have higher energy. We must ensure that they are represented on the coarse grid. Instead of considering the near kernel as defined above, we experiment below with replacing the definition of the near kernel with a discrete representation of the lowest energy eigenmode. This vector, x is not such that $Ax \approx 0$, but it is a smooth error component, and this tentative prolongator will help ensure it is represented on the coarse grid, where we can resolve it. In the case of the Laplace operator, our tentative prolongator will look like:

$$P^{(\text{tent})} = \begin{pmatrix} \sin(\pi x_1) & & & & \\ \sin(\pi x_2) & & & & \\ & \sin(\pi x_3) & & & \\ & \sin(\pi x_4) & & & \\ & & & \ddots & \\ & & & & \sin(\pi x_{m-1}) \\ & & & & \sin(\pi x_m) \end{pmatrix} \in \mathbb{R}^{m \times \frac{m}{2}},$$

with $x_1, ..., x_m$ taken to be m equally-spaced points in the interval $(0, 1)$, and $R^{(\text{tent})} = (P^{(\text{tent})})^{\mathsf{T}}$.

Table 10.1: Number of 2-Grid Iterations, Poisson Equation (3.1) with Different Tenative Prolongators

Method	Near kernel: constant vector	Near kernel: x + constant vector	Near kernel: Lowest frequency eigenmode
SA	16	16	14
NSA	41	41	33
NSR	23	23	17
EMIN(r)	18	18	15
EMIN	18	18	15

1 pre- and post-smoothing step (Weighted Jacobi), $m = 1024$, $\Delta x = \frac{1}{1+m}$, `solve` on the coarse level, tol $= 10^{-8}$

For the results of these experiments in the 2-grid method, refer to Table 10.1. Using the lowest energy eigenmode does correspond to a modest improvement in convergence. It must also be noted that using this instead of the traditional definition of the near kernel will result in considerably more setup cost for only a slight improvement. Of course, it would have been fun to experiment with this idea on a hard-to-converge example, such as the 2D convection diffusion equation over the vector field 'recirc' with $m = 50625$ unknowns. However, the eigenvectors of the system matrix are not nearly as simple to calculate, so such an example was not possible. Thus, all other experiments in this thesis are performed with the constant vector as the user-defined near kernel.

Chapter 11

Concluding Remarks

11.1 Summary of Findings

Our new methods, EMIN(r)/EMIN offer good results in the 1D convection diffusion problem. However, in the convective case NSR performs almost identically and is cheaper, while SA performs better in the diffusive case and is also cheaper than the new methods. In the 2D convection diffusion experiment, EMIN(r) and EMIN performed competitively in terms of iterations, but cheaper alternatives such as SA, again, performed nearly as well or slightly better. It is therefore not recommended that one use the new methods in solving 2D convection diffusion with a multigrid scheme as carried out in this thesis.

One problem which showed exciting results was performing one time-step of linear advection with Birken's optmized smoothers from [2]. Here we saw great performance gains from EMIN(r)/EMIN over the other methods at extremely large CFL numbers.

11.2 Future Work

In this thesis, we used a multgrid scheme with different operators for prolongation and restriction to solve the linear equation systems associated with the discretized PDEs. It must be noted, however, that Sala and Tuminaro in [9] considered these schemes as preconditioners for GMRES, the tool they used to solve the same linear equation systems.

Our results shed light on the new methods' benefits and drawbacks, but except for in the case of linear advection at extremely large CFL's, we do not see any very exciting gains from using the newly proposed methods EMIN(r) and EMIN. It is thus highly recommended that this work be continued, using these multigrid schemes as preconditioners for GMRES. In [9], great results were documented in 2D convection diffusion and in solving the Euler equations when these methods were used in this way.

Chapter 12

Appendix

12.1 Derivation of the Smoothers

12.1.1 Jacobi

$$A = D + R,$$

$$D = \begin{pmatrix} a_{1,1} & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & a_{m,m} \end{pmatrix}, \; R = \begin{pmatrix} 0 & a_{1,2} & \cdots & a_{1,m} \\ a_{2,1} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & a_{m-1,m} \\ a_{m,1} & \cdots & a_{m,m-1} & 0 \end{pmatrix}$$

$$Ax = b$$

$$Dx = Dx + (b - Ax)$$

$$Dx = Dx + (b - Dx - Rx)$$

$$Dx = b - Rx$$

$$x = D^{-1}(b - Rx)$$

$$x^{(k+1)} = D^{-1}(b - Rx^{(k)})$$

12.1.2 Gauss-Seidel

$$A = D + L + U,$$

$$L = \begin{pmatrix} 0 & 0 & \cdots & 0 \\ a_{2,1} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ a_{m,1} & \cdots & a_{m,m-1} & 0 \end{pmatrix}, \; U = \begin{pmatrix} 0 & a_{1,2} & \cdots & a_{1,m} \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & a_{m-1,m} \\ 0 & \cdots & 0 & 0 \end{pmatrix}$$

$$Ax = b$$

$$(D + L)x = (D + L)x + (b - Ax)$$

$$(D + L)x = (D + L)x + (b - (D + L + U)x)$$

$$(D + L)x = b - Ux$$

$$(D + L)x^{(k+1)} = b - Ux^{(k)}$$

$$x^{(k+1)} = -(D + L)^{-1}Ux^{(k)} + (D + L)^{-1}b$$

12.1.3 Weighted Jacobi

This method is almost the same as Jacobi Smoothing, except we add a parameter w (and an additional term) to speed up convergence. This corresponds to:

$$\begin{aligned} x^{(k+1)} &= wD^{-1}(b - Rx^{(k)}) + (1-w)x^{(k)} \\ &= wD^{-1}(b - (D-A)x^{(k)}) + (1-w)x^{(k)} \\ &= wD^{-1}b + wx^{(k)} - wD^{-1}Ax^{(k)} + x^{(k)} - wx^{(k)} \\ &= (I - wD^{-1}A)x^{(k)} + wD^{-1}b. \end{aligned}$$

12.1.4 Weighted Gauss-Seidel

In an analogous way to Weighted Jacobi, an iteration of Weighted Gauss-Seidel corresponds to:

$$x^{(k+1)} = w\big((D-L)^{-1}Ux^{(k)} + (D-L)^{-1}b\big) + (1-w)x^{(k)}.$$

12.1.5 Symmetric Gauss-Seidel

For each iteration of Symmetric Gauss-Seidel, we need to solve two systems of linear equations in order to get $x^{(k+1)}$. This corresponds to doing one iteration of backward weighted Gauss-Seidel, followed by an iteration of forward Gauss-Seidel. These systems are as follows.

$$(D - wL)x^{(k+\frac{1}{2})} = [wU + (1-w)D]x^{(k)} + wb$$

$$(D - wU)x^{(k+1)} = [wL + (1-w)D]x^{(k+\frac{1}{2})} + wb$$

12.2 Symmetric Gauss-Seidel Implementation

Start with an initial guess $x^{(0)}$. Then,

for $k = 1, 2, ..., \nu_1$
  for $i = 1, 2, ..., n$
    $\sigma = 0$
    for $j = 1, 2, ..., i-1$
      $\sigma = \sigma - (D - wL)_{i,j}\ x_j^{(k-\frac{1}{2})}$
    end
    for $j = i, i+1, ..., n$
      $\sigma = \sigma + (wU + (1-w)D)_{i,j}\ x_j^{(k-1)}$
    end
    $x_i^{(k-\frac{1}{2})} = (wb_i + \sigma)/A_{i,i}$
  end
  for $i = n, n-1, ..., 1$
    $\sigma = 0$
    for $j = 1, 2, ..., i$
      $\sigma = \sigma + (wL + (1-w)D)_{i,j}\ x_j^{(k-\frac{1}{2})}$
    end
    for $j = i+1, i+2, ..., n$
      $\sigma = \sigma - (D - wU)_{i,j}\ x_j^{(k)}$
    end
    $x_i^{(k)} = (wb_i + \sigma)/A_{i,i}$

```
    end
end
```

Bibliography

[1] Philipp Birken. *Iterative Solution of Large Scale Systems in Scientific Computing*. Lund University, 2018.

[2] Philipp Birken. *Optimizing Runge-Kutta Smoothers for Unsteady Flow Problems*. Kent State University, 2012.

[3] William Briggs, Van Emden Henson and Steve F. McCormick. *A Multigrid Tutorial Second Edition*. Society for Industrial and Applied Mathematics, 2000.

[4] Dutra, I., Camacho, R., Barbosa, J., Marques, O. *High Performance Computing for Computational Science – VECPAR 2016*. Springer 2017.

[5] Lawrence C. Evans *Partial Differential Equations*. American Mathematical Society 1997.

[6] Antony Jameson and Timothy J. Baker. Solution of the Euler equations for complex configurations. *Proceedings of AlAA 6th Computational Fluid Dynamics Conference, Danvers*, p. 296, 1983.

[7] Luke N. Olson, Jacob B. Schroder, and Raymond S. Tuminaro. *A general interpolation strategy for algebraic multigrid using energy-minimization*

[8] David Padua. *Encyclopedia of Parallel Computing*. Springer Science+Business Media, LLC.

[9] Marzio Sala and Raymond S. Tuminaro. A New Petrov Galerkin Smoothed Aggregation Preconditioner for Nonsymmetric Systems. *SIAM J. Scientific Computing*, 31(1):143-166, 2008.

[10] Gustaf Söderlind. *Numerical Methods for Differential Equations* Lund University, 2011-2012.

[11] Ulrich Trottenberg, Cornelius W. Oosterlee and Anton Schuller. *Multigrid* Academic Press; 1st edition, December 2000.

[12] Petr Vanek. Acceleration of convergence of a two-level algorithm by smoothing transfer operators. *Appl. Math.*, pp. 265-274, 1992.

[13] Petr Vanek. Fast multigrid solver. *Appl. Math.*, 40:1-10, 1995.

[14] Petr Vanek, Marian Brezina, Jan Mandel. Convergence of algebraic multigrid based on smoothed aggregation. *Numer. Math.*, 88: 559–579, 2001.

[15] *Convection-diffusion equation*. Wikipedia 2019.
`https://en.wikipedia.org/w/index.php?title=Convection%E2%80%93diffusion_equationoldid=892138892`

[16] *Fundamental solution*. Wikipedia 2019.
https://en.wikipedia.org/w/index.php?title=Fundamental_solutionoldid=857106258

[17] *Laplace's Equation*. Wikipedia 2019.
https://en.wikipedia.org/w/index.php?title=Laplace%27s_equationoldid=893937784

[18] *Partial Differential Equations*. Wikipedia 2019.
https://en.wikipedia.org/w/index.php?title=Partial_differential_equationoldid=893813173

[19] *Poisson's equation*. Wikipedia 2019.
https://en.wikipedia.org/w/index.php?title=Poisson%27s_equationoldid=895287301

Master's Theses in Mathematical Sciences 2019:E45
ISSN 1404-6342

LUNFNA-3029-2019

Numerical Analysis
Centre for Mathematical Sciences
Lund University
Box 118, SE-221 00 Lund, Sweden

http://www.maths.lth.se/