Positivity-preserving multigrid and multilevel methods

by

© **Dawei Wang**

A thesis submitted to the School of Graduate Studies in partial fulfillment of the requirements for the degree of Master of Science.

Department of Mathematics and Statistics

Memorial University

August 2019

St. John's, Newfoundland and Labrador, Canada

Abstract

Multigrids methods are extremely effective algorithms for solving the linear systems that arise from discretization of many differential equations in computational mathematics. A multigrid method uses a hierarchy of representations of the problem to achieve its efficiency and fast convergence. Originally inspired by a problem in adaptive mesh generation, this thesis focuses on the application of multigrid methods to a range of problems where the solution is required to preserve some additional properties during the iteration process. The major contribution of this thesis is the development of multigrid methods with the additional feature of preserving solution positivity: We have formulated both a multiplicative form multigrid method and a modified unigrid algorithm with constraints that are able to preserve positivity of the approximate solution at every iteration while maintaining convergence properties typical of normal multigrid methods. We have applied these algorithms to the 1D adaptive mesh generation problem to guarantee mesh nonsingularity, to singularly perturbed semilinear reaction-diffusion equations to compute unstable solutions, and to nonlinear diffusion equations. Numerical results show that our algorithms are effective and also possess good convergence properties.

To my parents and brother

Lay summary

Many real-world applications entail solving differential equations, which are usually so complex that we cannot get an exact formula for their solutions, but have to numerically solve for discrete approximate solutions on a computer. For example, we might be curious how a drop of ink would diffuse in water, or how long it takes for a cup of hot water to be cooled down to a certain temperature. These all require solutions to differential equations and can be simulated on a computer. Sometimes the exact solution needs to be positive so that the problem is well-posed, such as the mass of a material, or the thickness of a film, so a good approximate solution should not violate this property. In some solution procedures, however, the approximate solution can easily violate this. In addition, we want to solve for the approximate solutions quickly. One family of methods that can solve for the discrete approximate solutions extremely quickly are called multigrid and multilevel methods. These types of methods solve for the approximate solution iteratively on a hierarchy of levels. Given an initial guess of the approximate solution, they update this guess at every iteration so that the evolving approximation gets closer and closer to the true solution. This process is performed until a pre-defined stopping criterion is reached.

We wish to achieve the goal of developing a positivity-preserving method, and at the same time solve for the approximate solution quickly. We try to make the solution positive at every iteration using multigrid (or multilevel) methods, so that the approximate solution is always positive even if we stop the iterations early before an ideal stopping criteria is reached. To do this, we change the way we update the approximations on the hierarchy of levels so that the positivity condition is easier to impose, and add restrictive modifications to the approximate solution during the iterative process.

With this idea, we have developed 2 general multigrid and multilevel methods that can make sure the approximate solution is always positive. And at the same time, these methods can solve for the approximate solution to a satisfactory accuracy very quickly. We applied our methods to solve for the solutions of 3 different problems, including an equidistributing mesh problem in 1D, a nonlinear diffusion problem and a class of singularly perturbed problems. For the nonlinear diffusion problem, we also developed a two-level method that is efficient for this specific application. Numerical results of all these examples show the efficiency of our algorithms.

Acknowledgements

This thesis is an outcome of the help and support of different people. First and foremost, this work couldn't be done without the advice of my two supervisors, Dr. Scott MacLachlan and Dr. Ronald Haynes. They had the original motivation for the thesis topics and gave me guidance in the whole process of my research and thesis writing. They were always optimistic and gave constructive suggestions when I was making slow progress in the project. I would also like to thank Dr. Hans De Sterck from the University of Waterloo, who provided references on multigrid methods for Markov chains, and Dr. Niall Madden from the National University of Ireland, Galway, who suggested the semilinear reaction diffusion application.

Statement of contribution

The work presented in this thesis is the result of collaborative research among Dawei Wang, Dr. Scott MacLachlan and Dr. Ronald Haynes, with its intellectual property equally co-owned by the three. The thesis itself was written by Dawei Wang with the input and guidance of Dr. Ronald Haynes and Dr. Scott MacLachlan. Hans De Sterck provided references on multigrid methods for Markov chains. Niall Madden suggested the semilinear reaction diffusion application.

Table of contents

List of tables

List of figures

Chapter 1

Introduction

The subject of this thesis is multigrid and multilevel methods for applications where the solution is required to be positive. This was initially inspired by a problem in adaptive mesh generation in 1D, where the exact solution is monotonic at both the continuous and discrete levels, and we want the approximate discrete solution to also be monotonic so that the generated mesh is not tangled. After reformulating the equation, this becomes a positivity-preserving constraint. Due to their efficiency, we want to solve such problems with multigrid methods. However, the standard multigrid methods achieve fast convergence by progressively eliminating oscillatory error components through a hierarchical decomposition of the problem and, hence, negative components can be introduced into the approximate solution by the coarse-grid correction process.

To solve this problem, we utilize the multigrid idea but modify standard algorithms and add additional constraints to form a multiplicative-error multigrid method, so that the corrected approximate solution stays positive. We notice that the reformulated linear system has the same properties as those that occur when solving the stationary distribution equation for continuous-time Markov chains. Various direct and iterative methods exist to solve this type of problems, see [19, 14] for a detailed discussion of numerical methods for Markov chains. For the multilevel methods of interest, the iterative aggregation-disaggregation (IAD) method pioneered in [20] provides the framework for a two-level approach, and has since been studied extensively. While the IAD method is normally used as a two-level method, its multilevel version and a link to algebraic multigrid methods have also been proposed in the literature [13, 11].

The ideas of using the multiplicative-error multigrid method to solve this problem is motivated by [5, 4, 6]. With this method, we are able to achieve fast convergence that is independent of mesh size and at the same time preserve positivity. While the multiplicative-error multigrid method for Markov chains is not a new method, we extend its application to solve other types of problems with positivity-preserving requirements. To our knowledge, the ability to guarantee the monotonicity of the approximate solution to the 1D equidistributing mesh problem so that the generated mesh is not tangled is a new result.

Since the mesh qualities in 2D are not as simple as in the 1D case, it is not easy to derive a single monotonicity or positivity constraint that can describe nontangled meshes. Thus, we are not able to extend our method to multidimensional equidistributing meshes. Besides this application to 1D adaptive mesh generation, however, the ability to preserve solution positivity is important in many areas of interest in computational mathematics.

We therefore generalize the diffusion equation for equidistributing meshes, and study nonlinear diffusion equations. For these problems, while being able to preserve positivity of the approximate solutions is important, it turns out that directly applying the multiplicative-error multigrid method does not give good convergence properties for multilevel results (even though results for the two-level method are reasonable). The reason is that the pair-wise aggregation-type interpolation operators developed for Markov chain models are not a good fit for these problems. Instead of trying to construct a suitable interpolation operator, we developed an efficient two-grid method that is positivity-preserving for solving nonlinear diffusion equations. However, the generalization of this algorithm to a multigrid method does not work well.

We, therefore, turn to unigrid methods, which are another family of multilevel methods and are applicable to linear systems from the discretization of other types of differential equations. First proposed by McCormick and Ruge [17], the idea of unigrid methods is to reflect back corrections directly to the fine-grid solution instead of recursively correcting coarse-grid solutions (as in multigrid methods). This gives more control of the solution during the iteration process. Moreover, unigrid methods perform all the operations only on the fine grid, hence the name. Coarse-grid corrections are replaced by fine-grid relaxation on a hierarchy of directions. Since the unigrid cycle is equivalent to multigrid provided that the multigrid formulation satisfies variational properties, this algorithm has good convergence properties. With additional constraints developed to force the sign of the approximate solutions during the iteration process, solution positivity is able to be preserved. Combining Picard iterations and the positivity-preserving unigrid method, we get a new positivity-preserving multilevel method for solving nonlinear diffusion equations. This algorithm is also applicable to other types of nonlinear problems.

To apply this algorithm to other nonlinear problems and further motivate the importance of sign preservation in certain applications, we next apply this algorithm to solve singularly perturbed problems. By restricting the sign of the solution, we are able to solve for some solutions from initial guesses that lead to failure of Newton's method. To the author's knowledge, there is no previous work on positivity-preserving linear solvers for these problems.

We will first give a brief introduction to iterative methods in Chapter 2, where we present the (weighted) Jacobi and Gauss-Seidel relaxation methods and their smoothing properties as background and then develop the idea of the multigrid methods. Next we introduce unigrid methods, which are another family of multilevel algorithms that give the same results as multigrid methods under certain circumstances. Iterative methods for Markov chains will also be discussed, as the results for singular systems will be made use of in the applications in the later chapters. In Chapter 3, we present our positivity-preserving multilevel algorithms. We give algorithms for both singular and nonsingular systems. After introducing the ideas of the algorithms, we apply them to solve real-world applications and give numerical results to show their efficiency. In Chapter 4, equidistributing meshes in 1D are introduced and solved with a multiplicative error-correction form multigrid method to guarantee that the approximate solution gives nontangled meshes at every iteration. Numerical results are given to show the mesh-independent convergence property. For completeness, we also prove that the weighted Jacobi method with weighting parameter $\omega \leq 1/2$ and the Gauss-Seidel method are monotonicity-preserving. In Chapter 5, we use the adaptively damped Picard iterations and a restricted unigrid method to solve nonlinear diffusion problems and ensure the positivity of approximate solutions. In addition, we introduce a new two-level method that is specific to this problem. The efficiency of these algorithms are tested on model problems. In Chapter 6, we discuss singularly perturbed problems and demonstrate the importance of sign-preservation in this application. The positivity-preserving multilevel methods are also applicable in this problem. We show that a damped Picard iteration with a sign-preserving linear solver enables us to compute some solutions from initial guesses that lead to failure of Newton's method. Finally, in Chapter 7, we conclude this thesis and give an outlook of possible future work. In this thesis, unless otherwise specified, theorems that include a proof are developed by the author.

Chapter 2

Iterative methods for solving linear systems

Numerical discretization schemes such as finite-difference and finite-element methods for solving differential equations eventually lead to a large linear system to be solved,

$$A\mathbf{u} = \mathbf{b}, \tag{2.1}$$

where A is an $n \times n$ matrix, and $\mathbf{u}$ and $\mathbf{b}$ are both vectors with n components. The system matrix A is usually very sparse. Due to the matrix size and sparse structure, direct solvers quickly become ineffective, especially for problems in multiple spatial dimensions. As a result, iterative methods are usually favored in real-world applications.

In this chapter, we first talk about the properties of M-matrices that are useful in the discussion of convergence of iterative methods. We then give a brief summary of several basic iterative methods, including their smoothing properties, which form the basis of multigrid methods. Following this, we introduce the famous multigrid methods. Next we discuss unigrid algorithms, which are another class of multilevel methods we will make use of and are equivalent to multigrid methods under certain conditions. As a background for Chapter 4, we also give an introduction to iterative methods for Markov chains.

2.1 Nonnegative matrices and M-matrices

We first list some definitions and theorems that will be useful. These definitions and theorems are taken from references [21, 1], and the theorems are listed without proof.

Definition 2.1.1. *Let $A = [a_{ij}]$ be a real $m \times n$ matrix, we say $A \geq 0$ ($A > 0$) if $a_{ij} \geq 0$ ($a_{ij} > 0$) for all $1 \leq i \leq m$, $1 \leq j \leq n$, and call A a nonnegative (positive) matrix. Similarly $A = 0$ means every entry of A is 0.*

Definition 2.1.2. *For $n \geq 2$, an $n \times n$ matrix A is reducible if there exists an $n \times n$ permutation matrix P such that*

$$PAP^T = \begin{bmatrix} A_{11} & A_{12} \\ O & A_{22} \end{bmatrix},$$

where A_{11} is an $r \times r$ submatrix, and A_{22} is an $(n-r) \times (n-r)$ submatrix with $1 \leq r < n$. If no such permutation exists, then A is irreducible. If A is a 1×1 matrix, then A is irreducible if its single entry is nonzero, and reducible if it is zero.

Definition 2.1.3. *The spectral radius of an $n \times n$ matrix A is $\rho(A) = \max_i\{|\lambda_i|\}$, where the maximum is taken over all the eigenvalues, λ_i, of A.*

Definition 2.1.4. *Let $A \geq 0$ be an irreducible $n \times n$ matrix, and let k be the number of eigenvalues of A of modulus $\rho(A)$. If $k = 1$, then A is primitive. If $k > 1$, then A is cyclic of index k.*

Theorem 2.1.1. *Let A be an irreducible $n \times n$ cyclic matrix of index k, $k > 1$. Then, the k eigenvalues of modulus $\rho(A)$ are of the form*

$$\lambda_k = \rho(A) \cdot \exp\left(i\frac{2\pi j}{k}\right),\ 0 \leq j \leq k-1.$$

Moreover, all the eigenvalues of A have the property that rotations in the complex plane about the origin through angles of $2\pi/k$, but through no smaller angles, carry the set of eigenvalues into itself. Finally, there is an $n \times n$ permutation matrix P so *that*

$$PAP^T = \begin{bmatrix} O & A_{12} & O & \cdots & O \\ O & O & A_{23} & \cdots & O \\ \cdot & & & & \cdot \\ \cdot & & \ddots & & \cdot \\ \cdot & & \ddots & & \cdot \\ O & O & O & & A_{k-1,k} \\ A_{k,1} & O & O & \cdots & O \end{bmatrix}.$$

The famous Perron-Frobenious theorem tells us some important conclusions about irreducible nonnegative matrices.

Theorem 2.1.2. *Let $A \geq 0$ be an irreducible $n \times n$ matrix. Then:*

1. *A has a positive real eigenvalue equal to its spectral radius.*
2. *To $\rho(A)$ there corresponds an eigenvector $\mathbf{x} > 0$.*
3. *$\rho(A)$ increases when any entry of A increases.*
4. *$\rho(A)$ is a simple eigenvalue of A.*

For convenience of discussion, we adopt the common notation and let $Z^{n\times n}$ be the set of matrices with nonpositive off-diagonal and nonnegative diagonal entries, i.e. matrices of the form

$$A = \begin{bmatrix} a_{11} & -a_{12} & -a_{13} & \cdots & -a_{1n} \\ -a_{21} & a_{22} & -a_{23} & \cdots & -a_{2n} \\ -a_{31} & -a_{32} & a_{33} & \cdots & -a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -a_{n1} & -a_{n2} & -a_{n3} & \cdots & a_{nn} \end{bmatrix},$$

where $a_{ij} \geq 0$. An important subclass of Z-matrices are the M-matrices, defined as follows.

Definition 2.1.5. *Any $n \times n$ real matrix A of form $A = sI - B$, $s > 0$, $B \geq 0$ in which $s \geq \rho(B)$ is called an M-matrix.*

In fact, M-matrices arise so often that many alternative equivalent definitions exist. See [1] for example to get a total of 50 equivalent conditions that define a nonsingular M-matrix. We list below several important theorems regarding nonsingular and singular M-matrices.

Theorem 2.1.3. *Let $n \times n$ matrix A be an M-matrix of form $A = sI - B,\ s > 0,\ B \geq 0$. Then:*

1. *A is nonsingular and $A^{-1} \geq 0$ iff $s > \rho(B)$.*
2. *A is nonsingular and $A^{-1} > 0$ iff $s > \rho(B)$ and A is irreducible.*
3. *A is singular iff $s = \rho(B)$.*
4. *A is nonsingular and $A^{-1} \geq 0$ iff $I - D^{-1}A$ is nonnegative and convergent, where D is the diagonal of A.*
5. *A is nonsingular and $A^{-1} > 0$ iff $I - D^{-1}A$ is nonnegative, irreducible and convergent, where D is the diagonal of A.*
6. *When A is nonsingular, it has all positive diagonal elements and is strictly diagonally dominant.*

Theorem 2.1.4. *Let $n \times n$ matrix A be a singular, irreducible M-matrix. Then:*

1. *A has rank $n - 1$.*
2. *There exists a vector $\mathbf{u} > 0$ such that $A\mathbf{u} = 0$.*

2.2 Basic iterative methods

Given these tools, we are now able to introduce iterative methods and discuss their convergence properties. In contrast to direct methods for solving $A\mathbf{u} = \mathbf{b}$, such as Gaussian elimination, iterative methods generally do not compute the exact solution after a finite number of steps, but try to reduce the error to a certain amount that is less than an acceptable tolerance. A good iterative method is able to reduce the error by a large amount at every iteration and requires as little work as possible.

For an iterative method to be convergent to the exact solution of the linear system in Equation (2.1), the error of the approximate solution should converge to zero. The most basic iterative methods come naturally from a splitting of A,

$$A = M - N,$$

where M is nonsingular. From this splitting, we have $A\mathbf{u} = (M - N)\mathbf{u} = M\mathbf{u} - N\mathbf{u} = \mathbf{b}$, which implies $\mathbf{u} = M^{-1}N\mathbf{u} + M^{-1}\mathbf{b}$. Therefore, we can take

$$\mathbf{u}^{(k+1)} = M^{-1}N\mathbf{u}^{(k)} + M^{-1}\mathbf{b} \tag{2.2}$$

as an iterative scheme, where $\mathbf{u}^{(k)}$ for $k \geq 1$ is the approximate solution at the k-th iteration step, and $\mathbf{u}^{(0)}$ is the initial approximation.

Generally, an iterative scheme

$$\mathbf{u}^{(k+1)} = H\mathbf{u}^{(k)} + \mathbf{c}, \tag{2.3}$$

where H is an $n \times n$ iteration matrix, and $\mathbf{c} = (I - H)A^{-1}\mathbf{b}$, is a convergent scheme if the sequence, $\mathbf{u}^{(0)}, \mathbf{u}^{(1)}, ..., \mathbf{u}^{(k)}, ...$, converge to the exact solution of the linear system $A\mathbf{u} = \mathbf{b}$, i.e.,

$$\lim_{k\to\infty} \mathbf{u}^{(k)} = \mathbf{u}.$$

This is true if and only if the iteration matrix satisfies $\lim_{k\to\infty} H^k = 0$, because

$$\mathbf{e}^{(k)} = \mathbf{u}^{(k)} - \mathbf{u} = H(\mathbf{u}^{(k-1)} - \mathbf{u}) = H^k(\mathbf{u}^{(0)} - \mathbf{u}) = H^k\mathbf{e}^{(0)}.$$

When $H = M^{-1}N$, we have $\mathbf{c} = M^{-1}\mathbf{b}$. It is worth noting that analyzing the exact error directly requires the exact solution $\mathbf{u}$, which, in general, we do not know. Instead, an indirect approach is required to stop the iteration in a practical implementation, such as the norm of the residual. We give the following definition and related theorem from [1] regarding convergent matrices.

Definition 2.2.1. *We say an $n \times n$ matrix H is a convergent matrix if*

$$\lim_{k\to\infty} H^k = 0.$$

It is possible to show that the condition $\lim_{k\to\infty} H^k = 0$ is equivalent to $\rho(H) < 1$.

Therefore, the iteration described in Equation (2.3) is a convergent iterative scheme if its iteration matrix H is a convergent matrix. More importantly, we have the following theorem.

Theorem 2.2.1. *Let A be an $n \times n$ nonsingular linear system. The iteration $\mathbf{u}^{(k+1)} = H\mathbf{u}^{(k)} + \mathbf{c}$ as defined in Equation (2.3) to solve $A\mathbf{u} = \mathbf{b}$ is convergent for all starting vectors $\mathbf{u}^{(0)}$ iff $\rho(H) < 1$.*

An important class of splittings that can give convergent iteration matrices are the regular splittings [18].

Definition 2.2.2. *Let $A = M - N$. The pair of matrices M, N is called a regular splitting of A if M is nonsingular and M^{-1} and N are nonnegative.*

Theorem 2.2.2. *Let M, N be a regular splitting of matrix A. Then $\rho(M^{-1}N) = \rho(I - M^{-1}A) < 1$ if and only if A is nonsingular and A^{-1} is nonnegative.*

2.2.1 Weighted Jacobi and Gauss-Seidel methods

Now consider a particular splitting of A as

$$A = D - (L + U),$$

in which D is the diagonal of A, $-L$ is the strictly lower triangular part of A, and $-U$ is the strictly upper triangular part of A. Following the previous notation, this splitting gives the Jacobi method by taking

$$M_J = D. \tag{2.4}$$

The Jacobi iteration matrix is

$$H_J = D^{-1}(L + U) = I - D^{-1}A.$$

Therefore, the Jacobi iteration is

$$\mathbf{u}^{(k+1)} = D^{-1}(L + U)\mathbf{u}^{(k)} + D^{-1}\mathbf{b} = (I - D^{-1}A)\mathbf{u}^{(k)} + D^{-1}\mathbf{b} = \mathbf{u}^{(k)} + D^{-1}\mathbf{r}^{(k)},$$

where $\mathbf{r}^{(k)} = \mathbf{b} - A\mathbf{u}^{(k)}$ is the residual at the k-th step. From Theorem 2.2.2, we can see that if A is a nonsingular M-matrix, the Jacobi method is guaranteed to be convergent.

The weighted Jacobi iteration takes

$$M_\omega = \frac{1}{\omega}D, \tag{2.5}$$

The weighted Jacobi iteration matrix is

$$H_\omega = \omega D^{-1}\Big((\frac{1}{\omega} - 1)D + L + U\Big) = (I - \omega D^{-1}A).$$

Therefore, the weighted Jacobi iteration is

$$\mathbf{u}^{(k+1)} = (I - \omega D^{-1}A)\mathbf{u}^{(k)} + \omega D^{-1}\mathbf{b} = \mathbf{u}^{(k)} + \omega D^{-1}\mathbf{r}^{(k)}.$$

The weighted Jacobi method can also be written in the form of a weighted average of the previous approximation and the unweighted Jacobi approximation by noticing that

$$\begin{aligned}
\mathbf{u}^{(k+1)} &= (I - \omega D^{-1}A)\mathbf{u}^{(k)} + \omega D^{-1}\mathbf{b} \\
&= \mathbf{u}^{(k)} - \omega D^{-1}A\mathbf{u}^{(k)} + \omega D^{(-1)}\mathbf{b} \\
&= \mathbf{u}^{(k)} - \omega D^{-1}\big(D - (L+U)\big)\mathbf{u}^{(k)} + \omega D^{-1}\mathbf{b} \\
&= (1-\omega)\mathbf{u}^{(k)} + \omega\big[D^{-1}(L+U)\mathbf{u}^{(k)} + D^{-1}\mathbf{b}\big].
\end{aligned}$$

The Gauss-Seidel method takes

$$M_{GS} = D - L, \tag{2.6}$$

so that the iteration matrix is

$$H_{GS} = (D-L)^{-1}U = I - (D-L)^{-1}A.$$

Therefore, the Gauss-Seidel iteration is

$$\begin{aligned}\mathbf{u}^{(k+1)} &= \big[I-(D-L)^{-1}A\big]\mathbf{u}^{(k)}+(D-L)^{-1}\mathbf{b}\\ &= \mathbf{u}^{(k)}+(D-L)^{-1}\mathbf{r}^{(k)}.\end{aligned}$$

Other choices of the nonsingular matrix M give other commonly used iterative schemes, which we do not discuss in this thesis.

2.2.2 A component-wise interpretation of Jacobi and Gauss-Seidel method

We take a closer look at the Jacobi and Gauss-Seidel methods in this section. The Jacobi iterative scheme can be written component-wise as

$$u_i^{(k+1)} = \frac{1}{a_{ii}}\bigg(\sum_{j\neq i} a_{ij}u_j^{(k)} + b_i\bigg).$$

Moving all the elements to one side of the equation, we get

$$a_{ii}u_i^{(k+1)} - \bigg(\sum_{j\neq i} a_{ij}u_j^{(k)} + b_i\bigg) = 0,$$

which is the negative of the i-th component of the residual. This iterative scheme determines the i-th component of the next approximate solution vector by setting the i-th component of the residual to zero, while keeping all other components of the approximation at their old values.

Similarly, the Gauss-Seidel iterative scheme can written component-wise as

$$-\sum_{j<i} a_{ij}u_j^{(k+1)} + a_{ii}u_i^{(k+1)} - \sum_{j>i} a_{ij}u_j^{(k)} - b_i = 0,$$

which is also the negative of the i-th component of the residual. The difference from the Jacobi iteration is that the approximate solution is updated immediately after each new component is obtained, and this new information is incorporated into the residual for next iteration immediately.

For the Gauss-Seidel iteration, the k-th step within an iteration makes the k-th component of the residual zero, which can also be written as an orthogonality condition,

$$< \mathbf{r}^{(k)}, \mathbf{1}_k >= 0, \tag{2.7}$$

where $\mathbf{r}^{(k)}$ is the residual at the k-th step within the current iteration, $\mathbf{1}_k$ is the k-th column of the identity matrix, and $< \cdot, \cdot >$ is the l_2 inner product. Therefore, the update rule in this interpretation is to make the residual at the k-th step be orthogonal to $\mathbf{1}_k$. In addition, writing the update at the k-th step of the iteration as

$$\mathbf{u}^{(k)} = \mathbf{u}^{(k-1)} + \omega \mathbf{1}_k,$$

and substituting this into Equation (2.7), we get

$$< \mathbf{r}^{(k-1)} - \omega A \mathbf{1}_k, \mathbf{1}_k >= 0,$$

where we can deduce that

$$\omega = \frac{< \mathbf{r}^{(k-1)}, \mathbf{1}_k >}{< A\mathbf{1}_k, \mathbf{1}_k >}.$$

Thus, the k-th step of one sweep of the Gauss-Seidel iteration can be written compactly as

$$\mathbf{u} \leftarrow \mathbf{u} + \frac{< \mathbf{r}, \mathbf{1}_k >}{< A\mathbf{1}_k, \mathbf{1}_k >} \mathbf{1}_k. \tag{2.8}$$

We notice that the Gauss-Seidel iteration updates one component at a time, and new information at one grid point takes many sweeps to have an effect on points far away. This restriction is due to the "narrow" shape of the basis vector $\mathbf{1}_k$. With this in mind, the unigrid method discussed in Section 2.6 tries to avoid this disadvantage by introducing broader directions on coarse levels so that new local information is accounted for faster.

2.3 Iterative methods for Markov chains

While these basic iterative methods discussed above can be applied to solve both singular and nonsingular systems, the convergence properties for a singular system are different from those for a nonsingular system. For example, when solving a singular system, the iteration matrix does not have to be a convergent matrix. Instead, a semiconvergent iteration matrix, which we will define in the following, can give us a convergent iterative scheme.

In this section, we will discuss a special application, the Markov chain problem, and focus on the computation of stationary probability distributions, where the linear system to be solved is singular. Both the discrete-time and continuous-time Markov chain models will be introduced. As we will see, the continous-time model is just an extension of the discrete-time model from discrete-time space to continuous-time space. The discussion of iterative methods for Markov chains is important not only in the sense that it is an important application of iterative methods for solving singular linear systems, but our first positivity-preserving multigrid method introduced in Chapter 3 is also inspired by this.

2.3.1 Discrete- and Continuous-Time Markov chain models

A Markov chain is a stochastic model that describes a sequence of possible states, with the assumption that only knowledge of the current state is relevant to a prediction of the future of the system, and past information does not matter. Discrete-time, discrete-state space Markov processes are generally called discrete-time Markov chains (DTMC), which move through a countable number of states. Formally, a finite DTMC can be specified by a tuple $(S, B, \boldsymbol{\pi})$, where S is the space consisting of n possible states, B is the transition probability matrix, which is a stochastic matrix satisfying $0 \leq B_{ij} \leq 1$ and

$$\mathbf{1}^T B = \mathbf{1},$$

and $\boldsymbol{\pi}$ is an initial probability distribution satisfying $\pi_i \geq 0$ and summing up to 1. The problem of interest is to find the stationary distribution $\mathbf{x}$ of the model, such that

$$B\mathbf{x} = \mathbf{x}, \quad x_i \geq 0, \; \forall i, \quad ||\mathbf{x}||_1 = 1. \tag{2.9}$$

The existence of a solution will be discussed later. We first give several important definitions adapted from [8] here concerning the states of DTMC that will be useful later for deciding whether the steady-state distribution exists.

Definition 2.3.1. *A DTMC $(S, B, \boldsymbol{\pi})$ is called irreducible if, from each state, one can reach any other state in a finite number of steps, i.e. there exists an integer $n \geq 1$ such that $(B^n)_{ij} \neq 0$ for any $i \in S \backslash \{j\}$.*

Definition 2.3.2. *For a DTMC, a state is said to be recurrent if the Markov chain is guaranteed to return to that state infinitely often. A state is called transient if there is a nonzero probability that the Markov chain will not return to that state again. A recurrent state j is called a positive-recurrent state if the average number of steps taken to return to state j for the first time after leaving it is finite; Otherwise, it is called null-recurrent.*

Definition 2.3.3. *A state j is said to be periodic with period p, or cyclic of index p if, on leaving state j, a return is possible only in a number of transitions that is a multiple of integer $p > 1$. A state whose period is $p = 1$ is said to be aperiodic.*

Definition 2.3.4. *A state that is positive-recurrent and aperiodic is said to be ergodic. If all the states of a Markov chain are ergodic, then the Markov chain itself is said to be ergodic.*

Now we present a theorem regarding the existence and uniqueness of a stationary distribution of a DTMC. For a proof of this theorem, see [8, 19].

Theorem 2.3.1. *In an irreducible and aperiodic DTMC $(S, B, \boldsymbol{\pi})$ with positive-recurrent states, the stationary probability distribution does exist and is unique independent of the initial probability distribution $\boldsymbol{\pi}$.*

When the time space is continuous, we have continuous-time, discrete-state space Markov processes, which are generally called continuous-time Markov chains (CTMC). This model is different from DTMC in the sense that every state is now associated with a residence time, which complies with an exponential distribution. CTMC can be described by a tuple $(S, Q, \boldsymbol{\pi})$, with the new component Q here being the infinitesimal generator matrix. The definition of the generator matrix has the form

$$[Q]_{ij} = \begin{cases} q_{ij} \geq 0, & i \neq j, \\ -\sum_{k \neq i} q_{kj}, & i = j, \end{cases} \tag{2.10}$$

where q_{ij} is the rate of exponential distribution of residence time in state i when going from state i to state j, and $\sum_{k \neq i} q_{kj}$ is thus the exit rate of state i. We notice that generator matrix Q satisfies

$$\mathbf{1}^T Q = 0,$$

i.e. every column sums up to 0.

The transient-state probabilities in a CTMC can be described by (the details can be found in [8])

$$\mathbf{p}(t) = e^{Qt}\boldsymbol{\pi}. \tag{2.11}$$

From the fact that

$$\mathbf{p}'(t) = Q\mathbf{p}(t),$$

we can compute the stationary probability $\mathbf{x}$ of a CTMC by solving

$$Q\mathbf{x} = \mathbf{0}, \quad x_i \geq 0,\ \forall i, \quad ||\mathbf{x}||_1 = 1. \tag{2.12}$$

If we rewrite Equation (2.9) as

$$(B - I)\mathbf{x} = 0,$$

we can see that $B - I$ satisfies all the properties of Q and thus can also be viewed as a generator matrix. On the other hand, if we construct a state-transition probability matrix B from Q by setting $[B]_{ij} = q_{ij}/|q_{jj|}$ when $i \neq j$ and $[B]_{ii} = 0$, we get the embedded DTMC corresponding to this CTMC, which provides an alternative way to solve for its stationary distribution. For a CTMC with a finite state space, the steady state probabilities always exist, and if the CTMC is, in addition, irreducible, then the steady-state probability is independent of the initial state $\boldsymbol{\pi}$.

Equation (2.11) is not appropriate to use for computing transient state probabilities. Instead, a more efficient technique is generally used by performing a *uniformization* step to get a uniformized state transition probability matrix, B, corresponding to a DTMC. To do the uniformization step, we define the matrix

$$B = I + \frac{Q}{\lambda}, \tag{2.13}$$

where λ is chosen such that

$$\lambda \geq \max_i\{|q_{ii}|\}. \tag{2.14}$$

With this choice of λ, the entries in matrix B are always in the interval $[0, 1]$, and every column of B sums up to 1. Hence it is a stochastic matrix corresponding to a DTMC and describes the evolution of the CTMC in time-steps of mean length $1/\lambda$. The detailed implementation of using this method to compute transient state probabilities is not of interest in this thesis. However, it will be a useful method for us to construct a convergent relaxation algorithm, as will be seen in Chapter 4.

2.3.2 Basic iterative methods for Markov chains

For large Markov chain models, iterative methods are usually applied to solve for the stationary probability vector. Commonly used methods include the power method, Jacobi, Gauss-Seidel and SOR methods, and many other common methods for solving linear systems. Because we will use multigrid methods to solve the problem, we only discuss (weighted) Jacobi and Gauss-Seidel methods as relaxation schemes.

Since Equation (2.9) for solving DTMCs is a special case of Equation (2.12) for CTMCs, as already mentioned, we only discuss iterative methods for solving $Q\mathbf{x} = 0$ where Q satisfies the properties of being a generator matrix, i.e., it satisfies the conditions in Equation (2.10). We also write $A = -Q$, and try to solve $A\mathbf{x} = 0$ in the following. We give a proof in Theorem 2.3.2 that matrix A is a singular M-matrix.

Theorem 2.3.2. *Given a $Z^{n\times n}$ matrix A with $\max\limits_{1\le i\le n}\{a_{ii}\} > 0$ if, in addition, either every row or column of A sums up to zero, then A is a singular M-matrix.*

Proof. Inspired by the uniformization technique in CTMC, let $\alpha = \max\limits_{1\le i\le n}\{a_{ii}\}$, and define

$$B = I - \frac{A}{\alpha} = \begin{bmatrix} 1-\beta_{11} & \beta_{12} & \beta_{13} & \cdots & \beta_{1n} \\ \beta_{21} & 1-\beta_{22} & \beta_{23} & \cdots & \beta_{2n} \\ \beta_{31} & \beta_{32} & 1-\beta_{33} & \cdots & \beta_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \beta_{n1} & \beta_{n2} & \beta_{n3} & \cdots & 1-\beta_{nn} \end{bmatrix},$$

where $\beta_{ij} = \frac{a_{ij}}{\alpha}$. Then, B is a nonnegative matrix with every row summing up to 1, which means its spectral radius $\rho(B) = 1$. Since $\frac{A}{\alpha} = I - B$, by definition, $\frac{A}{\alpha}$ is a singular M-matrix, so A is a singular M-matrix. $\square$

As shown in Section 2.2, the Jacobi iteration matrix is given by

$$H_J = D^{-1}(L + U),$$

and Gauss-Seidel iteration matrix is given by

$$H_{GS} = (D - L)^{-1}U.$$

We saw in Section 2.2 that, for nonsingular systems, when the spectral radius of the iteration matrix is less than 1 (e.g. $\rho(H_J) < 1$ or $\rho(H_{GS} < 1)$), the iterative scheme (e.g. Jacobi or Gauss-Seidel method) is convergent. However, this is not the case when solving singular systems. Instead, we should now consider the property of semiconvergence [1].

Definition 2.3.5. *A matrix $H \in \mathbb{R}^{n\times n}$ is said to be semiconvergent if*

$$\lim_{n\to 0} H^n$$

exists.

Theorem 2.3.3 gives the properties that a matrix needs to satisfy in order to be semiconvergent.

Theorem 2.3.3. *A matrix H is semiconvergent iff all of the following conditions hold*

1. $\rho(H) \leq 1$;
2. *If $\rho(H) = 1$, then all the elementary divisors associated with the unit eigenvalue of H are linear, that is, $rank(I - H)^2 = rank(I - H)$;*
3. *If $\rho(H) = 1$, then $\lambda \in \sigma(H)$ where $|\lambda| = 1$ implies $\lambda = 1$.*

For the iterative scheme in Equation (2.2) to be convergent to some solution, the iteration matrix $H = M^{-1}N$ has to be semiconvergent.

Theorem 2.3.4. *Let $A = M - N \in \mathbb{R}^{n\times n}$ with M nonsingular. Then, the iterative method in Equation (2.2) converges to some solution $\mathbf{u}$ of $A\mathbf{u} = \mathbf{b}$ for each $\mathbf{u}^{(0)}$ if and only if $H = M^{-1}N$ is semiconvergent.*

2.4 Smoothing properties of basic iterative methods

As standalone iterative schemes, the aforementioned Jacobi and Gauss-Seidel methods are too slow to use in many real world applications. The motivation for multigrid methods lies in the fact that these basic iterations are very effective smoothers, i.e., they can damp out high frequency error components very quickly. As a result, they can be used as relaxation schemes for more effective multigrid methods.

Suppose the iteration matrix H is diagonizable and has eigenvectors $\mathbf{v}_i$ with the corresponding eigenvalues λ_i for all i. That is, $H\mathbf{v}_i = \lambda_i \mathbf{v}_i$. And suppose the eigenvector expansion of the initial error is

$$\mathbf{e}^{(0)} = \sum_i c_i \mathbf{v}_i.$$

Then, the error at the k-th iteration can be expressed by

$$\mathbf{e}^{(k)} = H^k \mathbf{e}^{(0)} = \sum_i c_i \lambda_i^k \mathbf{v}_i. \tag{2.15}$$

A good relaxation scheme should damp out those components $\mathbf{v}_i$ for many i.

To be more concrete, take the example $(n-1) \times (n-1)$ matrix

$$A = \frac{1}{h^2} \begin{bmatrix} 2 & -1 & & \\ -1 & 2 & -1 & \\ & & \ddots & -1 \\ & & -1 & 2 \end{bmatrix}$$

from the discretization of $-u''(x) = b(x)$ with boundary conditions $u(0) = u(1) = 0$, where $h = 1/n$ is the mesh size. The eigenvectors of matrix A are

$$(\mathbf{v}_i)_j = \sin\left(\frac{ij\pi}{n}\right), \quad 1 \le j \le n-1,$$

for $1 \le i \le n-1$, where $(\mathbf{v}_i)_j$ denotes the j-th component of the eigenvector $\mathbf{v}_i$, and the corresponding eigenvalues are

$$\lambda_i(A) = \frac{4}{h^2} \sin^2\left(\frac{i\pi}{2n}\right),$$

for $1 \le i \le n-1$.

Therefore, for the weighted Jacobi method, the eigenvalues of the iteration matrix are

$$\lambda_i(I - \omega D^{-1} A) = 1 - 2\omega \sin^2\left(\frac{i\pi}{2n}\right).$$

By choosing $\omega = 2/3$, we get

$$|\lambda_i(I - \omega D^{-1}A)| \leq \frac{1}{3},$$

for $n/2 \leq i \leq n-1$. This implies that after every iteration of the weighted Jacobi method with relaxation parameter $\omega = 2/3$, those modes in the upper half of the spectrum of A (in error expansion Equation (2.15)) are damped out to a third of their magnitude from the previous iteration. Hence, we can see that the weighted Jacobi method is very effective for eliminating high-frequency error components within only a few iterations.

On the other hand, on the lower half of the spectrum of A, the corresponding eigenvalues are close to 1, hence λ_i^k in Equation (2.15) is close to 1. As a result, after damping out those oscillatory modes, the error reduction becomes very slow because the remaining smooth modes in the error are hard to decrease. This justifies the phenomenon that these basic iterative methods are usually slow to converge.

Similarly, for the Gauss-Seidel method, when convergence is described in terms of the modes of A, the oscillatory modes will decay rapidly as in the weighted Jacobi method, while smooth modes will be damped slowly [3].

2.5 Multigrid methods

In order to be able to remove smooth modes of the error quickly, a natural idea is to apply a relaxation scheme recursively on different levels of resolution so that smooth modes on a finer level becomes oscillatory modes on coarser levels, which forms the basic idea of the extremely efficient family of multigrid methods: relax on a fine grid, then restrict the problem to a coarser grid and, finally, correct the fine-grid solution. To implement this idea, two more pieces are needed. We first need to express the fine-grid system on a coarser level and, then, be able to reflect coarse-grid corrections on fine-grid solutions, i.e., to construct restriction and interpolation operators to transfer information back-and-forth between grids.

2.5.1 Restriction and interpolation operators

In this section, we only discuss geometric multigrid methods, where there exist physical grids for the problem we are solving. First, consider the interpolation operator, and assume that the coarse grid has twice the grid spacing of the next finer grid. Denote the coarse-grid space as Ω^{2h}, and the next finer grid space as Ω^h, where the superscripts h and $2h$ denote grid spacings. Then as shown in Figure 2.1, solutions on fine-grid points are interpolated from neighboring coarse-grid points by linear interpolation

$$\begin{aligned} u_{2j}^h &= u_j^{2h}, \\ u_{2j+1}^h &= \frac{1}{2}(u_j^{2h} + u_{j+1}^{2h}), \end{aligned}$$

for $1 \le j \le n/2 - 1$, where u_{2j}^h is the $2j$-th component of the fine-grid approximation $\mathbf{u}^h$, and u_j^{2h} is the j-th component of the next coarse-grid approximation $\mathbf{u}^{2h}$. In matrix form, this defines

$$P\mathbf{u}^{2h} = \frac{1}{2}\begin{bmatrix} 1 & & \\ 2 & & \\ 1 & 1 & \\ & 2 & \\ & 1 & \\ & & \ddots \end{bmatrix} \cdot \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{\frac{n}{2}-1} \end{bmatrix}^{2h} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ \vdots \\ u_{n-1} \end{bmatrix}^{h} = \mathbf{u}^h.$$

To restrict from the fine grid to the coarse grid, the value at a coarse-grid point can simply be taken as the value at its corresponding fine-grid point, i.e.

$$u_j^{2h} = u_{2j}^h,$$

This is called injection. Another common restriction operator is full weighting, which also takes into consideration the neighboring points on the fine grid, and computes the value at a coarse-grid point as

$$u_j^{2h} = \frac{1}{4}(u_{2j-1}^h + 2u_{2j}^h + u_{2j+1}^h).$$

In matrix form, this defines

$$\mathbf{u}^{2h} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{\frac{n}{2}-1} \end{bmatrix}^{2h} = R\mathbf{u}^h = \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 & & & \\ & & 1 & 2 & 1 & \\ & & & & & \ddots \end{bmatrix} \cdot \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ \vdots \\ u_{n-1} \end{bmatrix}^{h}.$$

We notice that the restriction and interpolation operators are transposes of one other up to a constant

$$R = \frac{1}{2}P^T. \tag{2.16}$$

As we will see, dropping the constant factor generally does not influence the solution. The relationship $R = cP^T$ is called the variational property due to the minimization principle introduced in the next section.

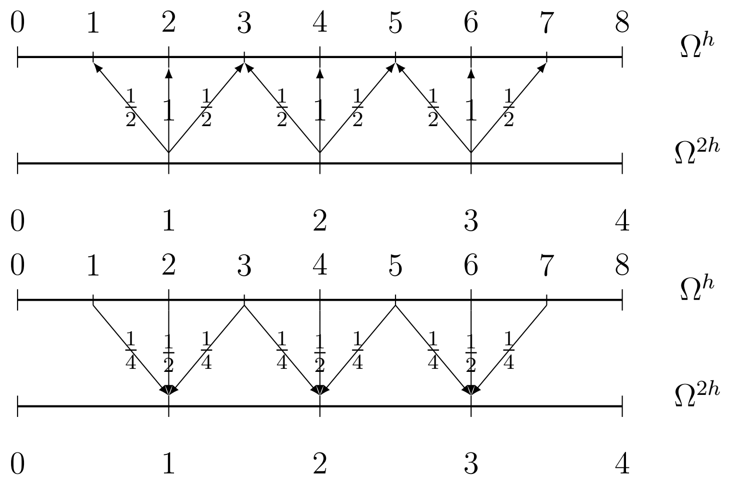

0 1 2 3 4 5 6 7 8
Ω^h
$\frac{1}{2}$ 1 $\frac{1}{2}$
Ω^{2h}
0 1 2 3 4
$\frac{1}{4}$ $\frac{1}{2}$ $\frac{1}{4}$

Figure 2.1: Linear interpolation by $P\mathbf{u}^{2h}$ and full weighting restriction by $R\mathbf{u}^h$.

2.5.2 Galerkin property

Consider the two-grid case: given an approximate solution $\mathbf{u}^h$ on the fine grid, and the interpolation operator as introduced in the last section. The coarse-grid corrected approximation to the solution is given by

$$\hat{\mathbf{u}}^h = \mathbf{u}^h + P\mathbf{u}^{2h},$$

for an unknown correction $\mathbf{u}^{2h}$. We want $\hat{\mathbf{u}}^h$ to be close to the true solution $\mathbf{u}$, i.e. the error norm $||\mathbf{u} - \hat{\mathbf{u}}^h||_F$ to be as small as possible, where F is some appropriate positive definite matrix, and the matrix-norm is defined by $||\mathbf{v}||_F^2 = \mathbf{v}^T F\mathbf{v}$.

Theorem 2.5.1. *For the linear system $A\mathbf{u} = \mathbf{b}$, let A be symmetric and positive definite. Given an approximate solution, $\mathbf{u}^h$, on fine grid Ω^h, and coarse grid to fine grid interpolation operator, P. Of all the vectors, $\mathbf{u}^{2h}$, on coarse grid Ω^{2h}, the vector $\mathbf{u}^{2h}$ computed by*

$$P^T AP\mathbf{u}^{2h} = P^T(\mathbf{b} - A\mathbf{u}^h) \tag{2.17}$$

minimizes the error norm

$$||\mathbf{u} - (\mathbf{u}^h + P\mathbf{u}^{2h})||_A.$$

That is,

$$(P^T AP)^{-1}P^T(\mathbf{b} - A\mathbf{u}^h) = \arg\min_{\mathbf{u}^{2h}} ||\mathbf{u} - (\mathbf{u}^h + P\mathbf{u}^{2h})||_A.$$

Proof. Compute the error norm

$$\begin{aligned} ||\mathbf{u} - (\mathbf{u}^h + P\mathbf{u}^{2h})||_A^2 &= ||\mathbf{e}^h - P\mathbf{u}^{2h}||_A^2 = (\mathbf{e}^h - P\mathbf{u}^{2h})^T A(\mathbf{e}^h - P\mathbf{u}^{2h}) \\ &= (\mathbf{e}^h)^T A\mathbf{e}^h - 2\mathbf{u}^{2h} P^T A\mathbf{e}^h + (\mathbf{u}^{2h})^T P^T AP\mathbf{u}^{2h}. \end{aligned}$$

Taking the derivative with respect to $\mathbf{u}^{2h}$ and setting it to zero gives

$$P^T AP\mathbf{u}^{2h} - P^T A\mathbf{e}^h = P^T AP\mathbf{u}^{2h} - P^T(\mathbf{b} - A\mathbf{u}^h) = 0.$$

Therefore, the solution of

$$P^T AP\mathbf{u}^{2h} = P^T(\mathbf{b} - A\mathbf{u}^h)$$

minimizes the error norm of the coarse-grid corrected solution $\mathbf{u}^h + P\mathbf{u}^{2h}$. $\square$

When we define the coarse-grid operator as

$$A^{2h} = P^T AP, \tag{2.18}$$

this is called the Galerkin condition, and the coarse-grid system to be solved is

$$A^{2h}\mathbf{u}^{2h} = P^T(\mathbf{b} - A\mathbf{u}^h) = \mathbf{r}^{2h}.$$

As mentioned before, we can see that dropping the constant scaling in front of the restriction operator in Equation (2.16) does not influence the solution.

2.5.3 Multigrid algorithms

Given the above formulation, it is now easy to state the algorithm pseudocode for a general class of multigrid methods. First, consider the two-grid case given by the following algorithm:

Algorithm 1: Two-grid Galerkin Multigrid Algorithm

```
    $\mathbf{MGT}(A, \mathbf{b}, \mathbf{u}^h, \nu_1, \nu_2)$ :
  $\mathbf{u}^h = \text{Relax}(A, \mathbf{b}, \mathbf{u}^h, \nu_1)$;   %Pre-relax $\nu_1$ times
  Get $A^{2h} = P^TAP$, $\mathbf{r}^{2h} = P^T(\mathbf{b} - A\mathbf{u}^h)$;
  Solve: $A^{2h}\mathbf{u}^{2h} = \mathbf{r}^{2h}$;
  $\mathbf{u}^h = \mathbf{u}^h + P\mathbf{u}^{2h}$;   %Coarse-grid correction
  $\mathbf{u}^h = \text{Relax}(A, \mathbf{b}, \mathbf{u}^h, \nu_2)$;   %Post-relax $\nu_2$ times
```

The function "Relax$(A, \mathbf{b}, \mathbf{u}^h, \nu)$" in Algorithm 1 can be any relaxation method, such as the weighted Jacobi and Gauss-Seidel methods. Recursively implementing this idea gives us multigrid algorithms:

Algorithm 2: Multigrid Algorithm

```
$\mathbf{MG}(A, \mathbf{b}, \mathbf{u}^h, \nu_1, \nu_2, \mu, l)$ :
if $l == 1$ then
    Solve: $A\mathbf{u}^h = \mathbf{b}$;
else
    $\mathbf{u}^h = \text{Relax}(A, \mathbf{b}, \mathbf{u}^h, \nu_1)$;    %Pre-relax $\nu_1$ times
    Set $\mathbf{r}^{2h} = P^T(\mathbf{b} - A\mathbf{u}^h), \mathbf{u}^{2h} = 0$;
    Solve $\mathbf{u}^{2h} = \mathbf{MG}(A^{2h}, \mathbf{r}^{2h}, \mathbf{u}^{2h}, \nu_1, \nu_2, \mu, l-1)$ $\mu$ times;
    $\mathbf{u}^h = \mathbf{u}^h + P\mathbf{u}^{2h}$;    %Coarse-grid correction
    $\mathbf{u}^h = \text{Relax}(A, \mathbf{b}, \mathbf{u}^h, \nu_2)$;    %Post-relax $\nu_2$ times
end
```

The parameter μ in Algorithm 2 is usually chosen to be 1 or 2. When $\mu = 1$, it gives the V-cycle multigrid method; when $\mu = 2$, it gives the W-cycle multigrid method. For Galerkin multigrid methods, the coarse-grid operator is $A^{2h} = P^T AP$. Also note that operators A and P on different levels can be precomputed before the iteration starts and stored in a dictionary data structure, so that we can directly access them during iterations.

Another form of multigrid method that is also important is the full multigrid algorithm, which starts iterations from an approximation on the coarsest level instead of the finest level, and progresses to finer levels with V-cycles or W-cycles. In this way, iterations on fine grids are started with improved initial guesses, which has a hope of reducing iteration numbers on costly fine grids. Algorithm 3 gives the pseudocode of the FMG algorithm, where the superscripts $2^k h$ denote the grid spacings on grid level k, and $A^h = A$ is defined to be the finest-grid operator.

Algorithm 3: FMG Multigrid Algorithm

```
FMG:
Solve: $A^{2^l h}\mathbf{u}^{2^l h} = \mathbf{b}^{2^l h}$;    %Solve on coarsest level
for $k = l-1, ..., 1, 0$ do
    Solve $\mathbf{u}^{2^k h} = \mathbf{MG}(A^{2^k h}, P\mathbf{u}^{2^{(k+1)} h}, \nu_1, \nu_2, \mu, l-k+1)$;
end
```

Figure 2.2 shows the grid schedules of various forms of multigrid methods in comparison with one another on 4 levels of grids.

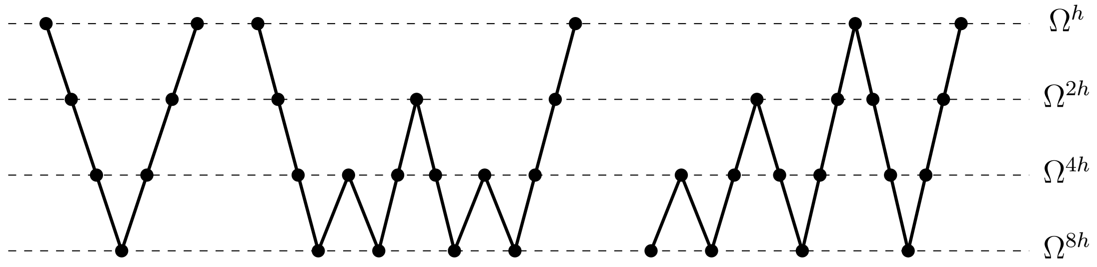


Figure 2.2: Grid schedules for V-cycle, W-cycle and FMG

2.6 Unigrid methods

Another multilevel method that we will make use of in this thesis is the so-called unigrid method introduced by McCormick and Ruge [17]. The basic idea is to do iterations as seen in Equation (2.8) for the Gauss-Seidel method, but to introduce broader directions for corrections from coarse levels, and write

$$\mathbf{u}^h = \mathbf{u}^h + \boldsymbol{\delta}_j^h,$$

where $\boldsymbol{\delta}_j^h$ is the directional correction

$$\boldsymbol{\delta}_j^h = \frac{< \mathbf{r}^h, \mathbf{d} >}{< A\mathbf{d}, \mathbf{d} >}\mathbf{d},$$

with direction vector $\mathbf{d}$, so that the resulting fine-grid residual is orthogonal to the direction $\mathbf{d}$ after correction. We can also add a weighting to the correction term, i.e., write

$$\mathbf{u}^h = \mathbf{u}^h + \omega\boldsymbol{\delta}_j^h, \tag{2.19}$$

to damp the iteration. The correction directions used in a unigrid method are generally formed from fine-grid interpolation of unit vectors on the coarse levels, which we denote by the columns of

$$\Phi^k = [\mathbf{d}_1^k, \mathbf{d}_2^k, \cdots, \mathbf{d}_{n_k}^k], \ 0 \leq k \leq l, \tag{2.20}$$

where $\mathbf{d}_i^k$ is the i-th iteration direction on level k, and n_k is the total number of iteration directions on level k, and l is the coarsest level number.

For convenience, we introduce some new notations here that will be useful in the following discussion. Denote $A^{(k)} = A^{2^k h}$ as the system matrix on level k, with $A^{(0)} = A^h = A$ as the finest-grid operator. Let I_k^l for $k, l \geq 0$ denote the interpolation operator from level k to level l if $k > l$, and restriction operator from level k to level l if $k < l$, and $I_k^l = I$ be the identity operator if $k = l$. Also denote $\mathbf{u}^k = \mathbf{u}^{2^k h}$ for $k \geq 0$ to simplify the notation when necessary.

Because the interpolation operators are the same for all the direction vectors on the same level, if the corresponding interpolation of these vectors in Equation (2.20) to the fine grid is done by operators I_k^0 for $k = 0, 1, \cdots, l$, we can write

$$\boldsymbol{\delta}_j^h = \frac{\langle \mathbf{b} - A\mathbf{u}, I_k^0 \mathbf{d}_j^k \rangle}{\langle A I_k^0 \mathbf{d}_j^k, I_k^0 \mathbf{d}_j^k \rangle} I_k^0 \mathbf{d}_j^k, \tag{2.21}$$

for $1 \leq j \leq n_k$, where $I_k^0 = I_1^0 I_2^1 \cdots I_k^{k-1}$, when $k \neq 0$, is the interpolation operator to the fine grid for all vectors on level k. Equation (2.19) gives one step of the relaxation on the fine grid. We can see that $I_k^0 \Phi^k = [I_k^0 \mathbf{d}_1^k, I_k^0 \mathbf{d}_2^k, \cdots, I_k^0 \mathbf{d}_{n_k}^k]$ forms the subspace of corrections to the fine-grid approximation at level k for $k = 0, 1, ..., l$. As a summary, Algorithm 4 gives the pseudocode of the unigrid method.

Algorithm 4: Unigrid method

```
UG_SOLVE($A, \mathbf{b}, \mathbf{u}, \nu, \omega$):
for $k = 0, 1, ..., l$ do
    for $i = 1, ..., \nu$ do
        for $j = 1, ..., n_k$ do
            Compute correction: $\boldsymbol{\delta}_j^h = \frac{\langle \mathbf{b} - A\mathbf{u}, I_k^0 \mathbf{d}_j^k \rangle}{\langle A I_k^0 \mathbf{d}_j^k, I_k^0 \mathbf{d}_j^k \rangle} I_k^0 \mathbf{d}_j^k$;
            Update approximation: $\mathbf{u} = \mathbf{u} + \omega \boldsymbol{\delta}_j^h$;
        end
    end
end
Return $\mathbf{u}$;
```

Note that we are starting the iteration from the finest level to the coarsest level in the algorithm here. In fact, this is not necessary. We can also start from the coarsest level and progress to the finest level or even randomly pick the next iteration direction. An alternative interpretation is to treat unigrid method as a subspace correction scheme, where the subspace is $P\Phi = [\Phi^0, I_1^0\Phi^1, ..., I_k^0\Phi^k, ..., I_l^0\Phi^l]$, and choose the next correction direction according to some reasonable pre-defined rules. For example, we can greedily choose the direction with the largest (or at least relatively large) residual norm for the next update step, which is also known as the Gauss-Southwell method, or we can even choose the next iteration direction randomly, which can be shown to have similar estimates for the error reduction, as discussed in [16, 7].

We now show that the unigrid method is theoretically equivalent to the multigrid method constructed with the variational conditions. It is worth re-writing the variational conditions which were defined recursively before with the new notation,

$$A^{(k+1)} = I_k^{k+1} A^{(k)} I_{k+1}^k, \; (I_k^{k+1})^T = I_{k+1}^k. \tag{2.22}$$

As already shown in [17], one relaxation step of the so-called immediate replacement multigrid (MGIR) process, which is the same as MG except that each change in the coarse-grid correction is immediately reflected in the fine-grid approximation and is then used to update the fine-grid residual and the coarse-grid equation, can be described by

$$\mathbf{u}_{MGIR}^0 = \mathbf{u}^h + \omega \frac{\langle I_0^k(\mathbf{b} - A\mathbf{u}^h), \mathbf{d}_i^k \rangle}{\langle A^{(k)}\mathbf{d}_i^k, \mathbf{d}_i^k \rangle} I_k^0 \mathbf{d}_i^k, \tag{2.23}$$

where $\mathbf{u}_{MGIR}^0$ is the corrected fine-grid approximation after one MGIR step. This update can be understood in two steps: First, perform the coarse-grid relaxation on grid level k such that the residual on level k is orthogonal to $\mathbf{d}_i^k$; and, then, interpolate this coarse-grid correction to the fine grid by the interpolation operator I_k^0.

The corresponding change for a standard multigrid method is

$$\mathbf{u}_{MG}^0 = \mathbf{u}^h + \omega \frac{\langle \mathbf{r}^k, \mathbf{d}_i^k \rangle}{\langle A^{(k)}\mathbf{d}_i^k, \mathbf{d}_i^k \rangle} I_k^0 \mathbf{d}_i^k.$$

The difference here is that the multigrid residual $\mathbf{r}^k$ on level k is different from that in MGIR process, because it is now computed recursively instead of directly restricted from the fine grid. Using the recursive definitions that $\mathbf{r}^q = \mathbf{b}^q - A^{(q)}\tilde{\mathbf{u}}^q$, $\mathbf{b}^{q+1} = I_q^{q+1}\mathbf{r}^q$

and $\bar{\mathbf{u}}^q = \tilde{\mathbf{u}}^q + I_{q+1}^q \bar{\mathbf{u}}^{q+1}$, where $\tilde{\mathbf{u}}^q$ is the approximate solution of level q without coarse-grid correction (i.e. the solution immediately after relaxation on level q), and $\bar{\mathbf{u}}^q$ is the approximate solution on level q after coarse grid correction, we can deduce that

$$\begin{aligned}\mathbf{r}^k &= I_0^k(\mathbf{b}^0 - A^{(0)}\bar{\mathbf{u}}^0) + \sum_{q=1}^{k}(I_{q-1}^k A^{(q-1)} I_q^{q-1} - I_q^k A^{(q)})\bar{\mathbf{u}}^q \\ &= I_0^k(\mathbf{b} - A\mathbf{u}^h) + \sum_{q=1}^{k} I_q^k (I_{q-1}^q A^{(q-1)} I_q^{q-1} - A^{(q)})\bar{\mathbf{u}}^q.\end{aligned}$$

Therefore

$$\mathbf{u}_{MG}^0 = \mathbf{u}^h + \omega \frac{\langle I_0^k(\mathbf{b} - A\mathbf{u}^h), \mathbf{d}_i^k\rangle}{\langle A^{(k)}\mathbf{d}_i^k, \mathbf{d}_i^k\rangle} I_k^0 \mathbf{d}_i^k,$$

which is the same as Equation (2.23) for MGIR. This proves that MGIR is equivalent to MG.

For the unigrid algorithm, from Equation (2.21), one relaxation step can be expressed by

$$\mathbf{u}^h = \mathbf{u}^h + \omega \frac{\langle \mathbf{b} - A\mathbf{u}_h, I_k^0\mathbf{d}_i^k\rangle}{\langle A I_k^0 \mathbf{d}_i^k, I_k^0\mathbf{d}_i^k\rangle} I_k^0 \mathbf{d}_i^k.$$

Then, the variational conditions in Equation (2.22) give

$$\mathbf{u}_{Uni}^0 = \mathbf{u}^h + \omega \frac{\langle I_0^k(\mathbf{b} - A\mathbf{u}_h), \mathbf{d}_i^k\rangle}{\langle I_0^k A I_k^0 \mathbf{d}_i^k, \mathbf{d}_i^k\rangle} I_k^0 \mathbf{d}_i^k = \mathbf{u}^h + \omega \frac{\langle I_0^k(\mathbf{b} - A\mathbf{u}_h), \mathbf{d}_i^k\rangle}{\langle A^{(k)}\mathbf{d}_i^k, \mathbf{d}_i^k\rangle} I_k^0 \mathbf{d}_i^k,$$

which is the same as Equation (2.23) for the MGIR update. This proves that unigrid and MGIR are also equivalent. Therefore, the unigrid method is equivalent to the multigrid method, given the variational conditions in Equation (2.22).

Chapter 3

Positivity-Preserving Multilevel Methods

In this chapter, we develop multilevel algorithms for solving $A\mathbf{u} = \mathbf{b}$ that can preserve solution positivity. We first deal with the situation when A is a singular, irreducible M-matrix. With the application of a modified multiplicative-form algebraic multigrid method (AMG) to the system, we prove that positivity can be preserved at each iteration and every grid level. We then apply a unigrid method to solve general nonsingular systems, and add additional constraints at every iteration such that the approximate solution stays positive.

3.1 Singular systems

In this section, we solve for a nontrivial solution of

$$A\mathbf{u} = \mathbf{0},$$

where A is an irreducible $Z^{n\times n}$ matrix with positive diagonals, and every column sums to zero. That is

$$\mathbf{1}^T A = \mathbf{0}.$$

From Theorem 2.3.2, we know that A is a singular M-matrix. This system corresponds to the CTMC steady-state in Equation (2.12). Note that the results in this section also apply to the DTMC steady-state in Equation (2.9) as we can always write $B\mathbf{x} = \mathbf{x}$ as $(I - B)\mathbf{x} = \mathbf{0}$, with $I - B$ satisfying all of the conditions here.

By the property of singular M-matrices from Theorem 2.1.4, we know that this system has a unique solution, $\mathbf{u}$, up to a constant scaling. In addition, the true solution satisfies $\mathbf{u} > 0$. We now introduce a multiplicative-error multigrid method that is able to preserve this property given a positive initial guess.

3.1.1 Positivity preserving relaxation

To be able to do this, we first need an appropriate relaxation scheme that is positivity preserving. It is already known that if A is a nonsingular M-matrix, then the Jacobi iteration is guaranteed to be convergent from Theorem 2.1.3. For the current case, A is a singular M-matrix. Although the Jacobi method is not guaranteed to be convergent anymore, the following theorem [19, 1] tells us that weighted Jacobi iteration is semiconvergent.

Theorem 3.1.1. *If H is the iteration matrix arising from a regular splitting of an irreducible singular M-matrix A, then the transformed matrix*

$$H_\alpha = (1 - \alpha)I + \alpha H \tag{3.1}$$

is semiconvergent for all $\alpha \in (0, 1)$.

The Jacobi iteration matrix of A is $H = D^{-1}(L + U)$ arising from the $A = D - (L + U)$, which is a regular splitting by Definition 2.2.2. Therefore, the weighted Jacobi method with weight parameter $\omega \in (0, 1)$ is semiconvergent.

We now look to make sure that the weighted Jacobi method is positivity preserving, which we prove in the following result.

Theorem 3.1.2. *For the linear system $A\mathbf{u} = \mathbf{b}$, where A is an $n \times n$ M-matrix with $a_{ii} > 0$, and $\mathbf{b} \geq 0$, the approximate solution given by the weighted Jacobi method with weight parameter $\omega \in (0, 1)$ is always positive if the initial guess, $\mathbf{u}^{(0)}$, is positive.*

Proof. Because A is an M-matrix, it is also a $Z^{n\times n}$ matrix. Therefore, it has all positive diagonals, $a_{ii} > 0$, by assumption, and nonpositive off-diagonals $-a_{ij} \leq 0$ for $i \neq j$.

Therefore, given a positive approximation $\mathbf{u}^{(k)} > 0$, each intermediate Jacobi iteration preserves positivity because every component update is of the form

$$u_i^{(k^*)} = \frac{1}{a_{ii}} \left(\sum_{j \neq i} a_{ij} u_j^{(k)} + b_i \right) \geq 0.$$

Hence, the approximation from the next iteration,

$$u_i^{(k+1)} = (1-\omega) u_i^{(k)} + \omega u_i^{(k^*)} > 0,$$

when $\omega \in (0,1)$. As a result,the weighted Jacobi approximation $\mathbf{u}^{(k+1)}$ is positive. $\square$

3.1.2 Multiplicative coarse-grid correction

Having shown that, with weighted Jacobi relaxation on the system $A\mathbf{u} = \mathbf{0}$, positivity of the solution can be preserved in each iteration, we come to the question of preserving positivity in the coarse-grid correction step of a multigrid method

If we use the standard multigrid method, the coarse-grid correction is of the form $\mathbf{u}^h \leftarrow \mathbf{u}^h + P\mathbf{u}^{2h}$. Given a positive fine-grid approximation, it is not convenient to control the coarse-grid correction $\mathbf{u}^{2h}$ so that the improved fine-grid approximation is guaranteed to be positive after correction. This motivates us to use a multiplicative error-correction form [5, 4], by writing the system equation on the fine grid as

$$A[\text{diag}(\mathbf{u}^h)]\mathbf{e}^h = \mathbf{0}, \tag{3.2}$$

where $\mathbf{u}^h$ is an approximation of the true solution on the fine grid, diag($\mathbf{u}$) is a diagonal matrix with its diagonal entries being the components of vector $\mathbf{u}$, i.e., $[\text{diag}(\mathbf{u})]_{ii} = u_i$, and $\mathbf{e}^h$ now denotes the multiplicative-form error vector of the approximate solution, $\mathbf{u}^h$, namely, $u_i = u_i^h e_i^h$. We note that $A[\text{diag}(\mathbf{u}^h)]$ still satisfies the properties in Theorem 2.3.2, so it is still a singular M-matrix. Therefore, the true solution to Equation (3.2) is positive, $\mathbf{e}^h > 0$, from Theorem 2.1.4. Then by writing

$$\mathbf{e}^h = P\mathbf{e}^{2h},$$

where P is the interpolation operator, we have

$$P^T A[\mathrm{diag}(\mathbf{u}^h)]P\mathbf{e}^{2h} = \mathbf{0}.$$

We leave the detailed construction of an interpolation operator P to be discussed in specific applications later, but only give the properties it should satisfy:

$$P \geq 0, \text{ and } P\mathbf{1} = \mathbf{1},$$

i.e. P is nonnegative and has row sums equal to 1.

Denote

$$\tilde{A} = A[\mathrm{diag}(\mathbf{u})],$$

and

$$\tilde{A}^{2h} = P^T A[\mathrm{diag}(\mathbf{u})]P = P^T \tilde{A} P.$$

We then have coarse-grid system

$$\tilde{A}^{2h}\mathbf{e}^{2h} = \mathbf{0},$$

with corresponding coarse-grid correction

$$\mathbf{e}^h = P\mathbf{e}^{2h}.$$

The approximate solution generated by coarse-grid correction using this two-level method is

$$\mathbf{u}^h \leftarrow \mathrm{diag}(\mathbf{u}^h)\mathbf{e}^h.$$

This form of multiplicative error-correction gives us more control over the final solution in terms of positivity. Now in order to guarantee that the approximate solution is positive, we only need to make sure that the interpolated error vector, $\mathbf{e}^h$, is positive. In addition, because the interpolation operation obviously preserves the sign of the approximate solution, we only need to make sure that the coarse-grid solution $\mathbf{e}^{2h}$ is positive. We therefore want coarse-grid matrix $\tilde{A}^{2h} = P^T A[\mathrm{diag}(\mathbf{u})]P$ to be an M-matrix, so we know that $\tilde{A}^{2h}\mathbf{e}^{2h}$ will have a positive solution.

Note that

$$\mathbf{1}^T P^T A[\mathrm{diag}(\mathbf{u})]P = (P\mathbf{1})^T A[\mathrm{diag}(\mathbf{u})]P = \mathbf{1}^T A[\mathrm{diag}(\mathbf{u})]P = 0,$$

because $\mathbf{1}^T A = \mathbf{0}$. Therefore,

$$\mathbf{1}^T \tilde{A}_c = \mathbf{0}. \tag{3.3}$$

In addition, we have that

$$P^T \tilde{A} P = P^T (\tilde{D} - (\tilde{L} + \tilde{U}))P = P^T \tilde{D} P - (P^T \tilde{L} P + P^T \tilde{U} P),$$

where $\tilde{D}$, $-\tilde{L}$ and $-\tilde{U}$ are the diagonal, lower-triangular and upper-triangular matrix of $\tilde{A}$ respectively. We can see that $P^T\tilde{D}P$, $P^T\tilde{L}P$ and $P^T\tilde{U}P$ are all nonnegative matrices. If $P^T\tilde{D}P$ is diagonal and does not contribute to off-diagonals, then the off-diagonals can only come from $-(P^T\tilde{L}P + P^T\tilde{U}P)$, so the off-diagonals of $P^T\tilde{A}P$ are nonpositive. Hence, by Equation (3.3) we are sure $\tilde{A}_c$ has nonnegative diagonals because its columns sum to zero. Then, by Theorem 2.3.2, the coarse-grid operator $\tilde{A}_c$ is also a singular M-matrix. However, this deduction only holds in special situations where $P^T\tilde{D}P$ is diagonal, such as when P is the pairwise aggregation operator as we will see in Chapter 4.

In general, $P^T\tilde{D}P$ is not diagonal, so it will contribute positive entries to the off-diagonals of $\tilde{A}^{2h}$, meaning $\tilde{A}^{2h}$ may have positive off-diagonal entries. Therefore, $\tilde{A}^{2h}$ is not guaranteed to be a singular M-matrix. Moreover, $\tilde{A}^{2h}$ may lose irreducibility due to new zero entries being introduced. This problem can be solved by introducing a *lumping method* [4] to the coarse-grid operator $\tilde{A}^{2h}$ so that the obtained lumped coarse-grid operator $\breve{A}^{2h}$ is an irreducible singular M-matrix. In addition, the exact solution is a fixed point of the V-cycle with the lumped coarse-grid error equation, $\breve{A}^{2h}\mathbf{e}^{2h} = \mathbf{0}$.

3.1.3 Positivity-preserving multigrid algorithm

With the tools formulated above, our relaxation scheme is positivity preserving, and our two-grid coarse-grid correction scheme is also positivity preserving. Some details for the implementation of the multigrid algorithm still need to be dealt with. For the direct solve on the coarsest grid, the system is singular and of rank $n-1$. Since we know the true solution with a direct solve is unique up to a scaling, we can scale the solution $\mathbf{e}^{(i)}$ such that its last entry is 1 by solving

$$\bar{A}\mathbf{d} = \begin{bmatrix} \tilde{A}(1:n-1,:) \\ \mathbf{z}^T \end{bmatrix} \mathbf{d} = \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix} = \mathbf{z},$$

which has a unique positive solution, where $\tilde{A}(1:n-1,:)$ denotes the first $n-1$ rows of $\tilde{A}$. However, in case the entries of $\tilde{A}$ are much smaller than 1, it is better to use

$$\bar{A}\mathbf{d} = \begin{bmatrix} \tilde{A}(1:n-1,:) \\ \mathbf{z}^T \end{bmatrix} \mathbf{d} = \begin{bmatrix} \mathbf{0} \\ A_{11} \end{bmatrix} = \mathbf{z}, \tag{3.4}$$

so that floating point arithmetic does not lead to an effectively singular matrix, where A_{11} is the entry of A in the first row and first column.

The last detail is the initial approximation that we use on the coarse grid. Because $A\mathbf{u} = A\text{diag}(\mathbf{u})\mathbf{1} = A\text{diag}(\mathbf{u})P\mathbf{1}$, relaxation on $A\mathbf{u} = \mathbf{0}$ with initial guess $\mathbf{u}$ is equivalent to relaxation on $P^T A\text{diag}(\mathbf{u})P\mathbf{e} = 0$ with initial guess $\mathbf{1}$. Therefore, an appropriate choice of initial guess on coarse grids should be $\mathbf{1}$. This is similar to standard multigrid with the choice of a zero vector $\mathbf{0}$.

Now we are able to give a multigrid algorithm that is able to preserve solution positivity. Algorithms 5 and 6 show the pseudocode of the two-grid and multigrid methods, respectively. The function "Relax$(A, \mathbf{u}, \nu)$" means applying weighted Jacobi relaxation. Also note that, for relaxation, normalization of the solution is needed at either each iteration or on the final step to make the solution sum up to one.

Algorithm 5: Two-grid Positivity-preserving Method for Solving $A\mathbf{u} = \mathbf{0}$

```
**MGT_SIN**$(A, \mathbf{u}, \nu_1, \nu_2)$ :
$\mathbf{u} = \text{Relax}(A, \mathbf{u}, \nu_1)$;   %Pre-relax $\nu_1$ times
$\tilde{A} = A[\text{diag}(\mathbf{u})]$;
$\tilde{A}^{2h} = P^T \tilde{A} P$;
Replace the last row of $\tilde{A}^{2h}$ to get $\bar{A}^{2h}$;
Solve: $\bar{A}^{2h}\mathbf{e}^{2h} = \mathbf{z}$;
$\mathbf{u} = [\text{diag}(\mathbf{u})]P\mathbf{e}^{2h}$;   %Coarse-grid correction
$\mathbf{u} = \text{Relax}(A, \mathbf{u}, \nu_2)$;   %Post-smoothing $\nu_2$ times
```

Algorithm 6: Positivity-preserving Multigrid Method for Solving $A\mathbf{u} = \mathbf{0}$

```
MGN_SIN$(A, \mathbf{u}, \nu_1, \nu_2, l)$ :
if $l == 1$ then
    Replace the last row of $A$ to get $\bar{A}$;
    Solve: $\bar{A}\mathbf{u} = \mathbf{z}$;
else
    $\mathbf{u} = \text{Relax}(A, \mathbf{u}, \nu_1)$;    %Pre-relax $\nu_1$ times
    $\tilde{A} = A[\text{diag}(\mathbf{u})]$;
    $\tilde{A}_c = P^T \tilde{A} P$;
    $\mathbf{e}_c = \textbf{MGN\_SIN}(\tilde{A}_c, \mathbf{1}, \nu_1, \nu_2, l-1)$;
    $\mathbf{u} = [\text{diag}(\mathbf{u})] P \mathbf{e}_c$;    %Coarse-grid correction
    $\mathbf{u} = \text{Relax}(A, \mathbf{u}, \nu_2)$;    %Post-smoothing $\nu_2$ times
end
```

3.2 Nonsingular systems

When system matrix A is a nonsingular matrix, the problem $A\mathbf{u} = \mathbf{0}$ has only trivial solution, and matrix A will not satisfy the properties required in Section 3.1 to apply the multigrid method developed there. In this section, we make use of the unigrid method to derive a positivity preserving multilevel method. Theorem 3.2.1 describes the class of problems that we are interested in for the discussion in this section. We will in this section always assume that we are in one of the two cases described.

Theorem 3.2.1. *Given a nonsingular $n \times n$ M-matrix, A, and a vector $\mathbf{b} > 0$, then the solution to the system $A\mathbf{u} = \mathbf{b}$ is positive. If, in addition, A is irreducible, then for any nonzero right-hand side $\mathbf{b} \geq 0$, the solution to the system $A\mathbf{u} = \mathbf{b}$ is positive.*

Proof. Because matrix A is a nonsingular M-matrix, from Theorem 2.1.3, we know $A^{-1} \geq 0$. When $\mathbf{b} > 0$, we have $\mathbf{u} = A^{-1}\mathbf{b} > 0$.

If, in addition, A is irreducible, by Theorem 2.1.3, we have $A^{-1} > 0$. Therefore, we only need $\mathbf{b} \geq 0$ with at least one nonzero entry to have $\mathbf{u} = A^{-1}\mathbf{b} > 0$. □

3.2.1 Unigrid method to preserve positivity with uniform thresholding

Unlike the standard multigrid method that solves for corrections at each level, the unigrid method introduced in Section 2.6 gives us more control over the fine-grid solution itself.

In order to use a unigrid method to preserve solution positivity, we try to threshold the relaxation correction so that negative entries in the correction do not contaminate the positivity of the approximate solution. That is, we choose the weighting parameter ω such that

$$\mathbf{u}^h \leftarrow \mathbf{u}^h + \omega\boldsymbol{\delta}_j^h > \mathbf{0}, \tag{3.5}$$

for each j, where $\boldsymbol{\delta}_j^h$ is given in Equation (2.21). To achieve this, suppose, at some stage of the iteration, if the i-th component of the correction $\boldsymbol{\delta}_j^h$ satisfies $\delta_{j,i} \geq 0$, because $\mathbf{u}^h$ is positive by assumption, we have $u_i + \omega\delta_{j,i} > 0$, where u_i is the i-th component of $\mathbf{u}^h$; if $\delta_{j,i} < 0$, then $u_i + \omega\delta_{j,i} > 0$ gives $\omega < -u_i/\delta_{j,i}$. Letting $M_j = \{i \mid \delta_{j,i} < 0\}$, we define m so that

$$-\frac{u_m}{\delta_{j,m}} = \min_{i\in M_j}\left\{-\frac{u_i}{\delta_{j,i}}\right\},$$

and set the damping parameter as

$$\omega = -\varepsilon\frac{u_m^h}{\delta_{j,m}^h}, \tag{3.6}$$

where $\varepsilon < 1$ is some constant close to 1, e.g. $\varepsilon = 0.9999$. Algorithm 7 gives the pseudocode of this positivity preserving unigrid method.

3.2.2 Unigrid method to preserve positivity with local correction

Since the uniform thresholding technique relies heavily on a single parameter, ω, and tries to threshold all of the corrections at once (possibly forced by only a few points), it is possible that this might hamper the convergence because of a "bad" choice of ω. An alternative approach to avoid this is to correct those negative components locally by interpolation. Because we are expecting that the solution varies smoothly, it makes sense to just locally correct those negative entries by interpolating from their positive neighbors.

Algorithm 7: Unigrid Method with Uniform Threasholding

```
UG_SOLVE($A, \mathbf{b}, \mathbf{u}, \nu, \omega$):
for $k = 0, 1, ..., l$ do
    for $i = 1, ..., \nu$ do
        for $j = 1, ..., n^k$ do
            Compute correction: $\boldsymbol{\delta}_j^h = \frac{\langle \mathbf{b} - A\mathbf{u}, I_k^0 \mathbf{d}_j^k \rangle}{\langle A I_k^0 \mathbf{d}_j^k, I_k^0 \mathbf{d}_j^k \rangle} I_k^0 \mathbf{d}_j^k$;
            if $\min\{\mathbf{u} + \boldsymbol{\delta}_j^h\} \leq 0$ then
                Set of negative-correction positions: $M_j = \{i \mid \delta_{j,i}^h < 0)\}$;
                Weighting parameter: $\omega = \varepsilon \min_{i \in M_j} \{ -\frac{u_i^h}{\delta_{j,i}^h} \}$;
                Update approximation: $\mathbf{u} = \mathbf{u} + \omega \boldsymbol{\delta}_j^h$;
            else
                Update approximation: $\mathbf{u} = \mathbf{u} + \boldsymbol{\delta}_j^h$;
            end
        end
    end
end
Return $\mathbf{u}$;
```

We use Figure 3.1 to explain this idea. This figure shows an example approximate solution on a mesh, where, after a step of unigrid relaxation, there are two groups of nodes that have negative approximate solutions. We first iterate through the approximate solution to find the two groups of nodes {4,5,6} and {11,12,13} that have a nonpositive approximation. Then, for group {4,5,6}, we use their positive neighboring nodes 3 and 7 to linearly interpolate and give their new values; for group {11,12,13}, we use their positive neighboring nodes (indexed 10 and 14) to linearly interpolate their new values. This gives a local correction scheme for a unigrid method that is able to preserve the positivity of the approximate solution.

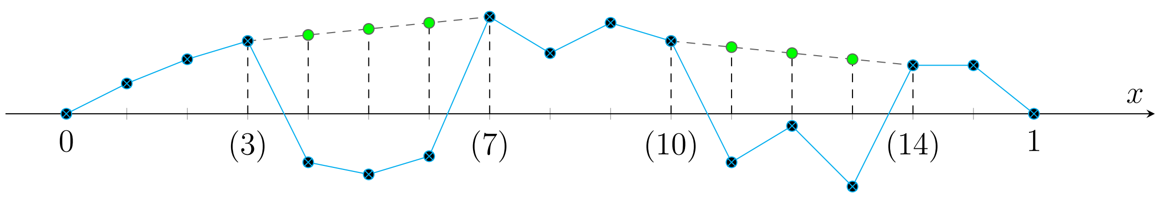


Figure 3.1: Local corrections on an example approximate solution

Algorithm 8: Unigrid Method with Uniform Threasholding

```
UG_SOLVE($A, \mathbf{b}, \mathbf{u}, \nu, \omega$):
for $k = 0, 1, ..., l$ do
    for $i = 1, ..., \nu$ do
        for $j = 1, ..., n^k$ do
            Compute correction: $\boldsymbol{\delta}_j^h = \frac{\langle \mathbf{b} - A\mathbf{u}, I_k^0 \mathbf{d}_j^k \rangle}{\langle A I_k^0 \mathbf{d}_j^k, I_k^0 \mathbf{d}_j^k \rangle} I_k^0 \mathbf{d}_j^k$;
            Update approximation: $\mathbf{u} = \mathbf{u} + \boldsymbol{\delta}_j^h$;
            if $\min\{\mathbf{u}\} \leq 0$ then
                Find nonpositive groups in the approximate solution $\mathbf{u}$;
                Perform local interpolation correction for these groups;
            end
        end
    end
end
Return $\mathbf{u}$;
```

Chapter 4

Application to 1D Equidistributing Meshes

This chapter applies Algorithms 5 and 6 to the problem of 1D adaptive mesh generation. This idea of solution-adapted grid generation is important when the solution of the given equation varies rapidly in some parts of the physical region. In such cases, it is reasonable to choose finer grids in that part of the region to reduce the error in the numerical solution. Mesh nontangling in 1D simply requires the mesh to be monotonic, which is true at the continuous level for standard models. We develop an efficient multigrid method that guarantees monotonicity also at the discrete level for the approximate solution. For completeness, we will first prove that the weighted Jacobi method with a suitable relaxation parameter and the Gauss-Seidel method can both preserve monotonicity of the solution at every iteration. Then, we reformulate the original problem into an incremental form, transform the requirement of monotonicity to one of positivity, and make use of the positivity preserving multigrid Algorithms 5 and 6. Hence, we get a multigrid algorithm that can preserve solution monotonicity. Numerical results are given to show the efficiency of this method.

4.1 An introduction to adaptive equidistributing meshes in 1D

The concept of equidistribution plays an important role in adaptive mesh generation. Let $\mathbb{R}^d$ denote the d-dimensional real space, and "$\subset$" denote the subset relation. One interpretation of equidistribution [10] is, given a metric space in the computational domain $\Omega \subset \mathbb{R}^d (d \geq 1)$ with a matrix-valued function $M = M(\mathbf{x})$ defined on it, we wish to find a mesh, $\mathcal{J}_h$, such that all elements have a constant volume in the defined metric:

$$\int_K \rho(\mathbf{x}) d\mathbf{x} = \frac{\sigma}{N}, \quad \forall K \in \mathcal{J}_h,$$

where σ is some positive constant, N is the number of elements, $M(\mathbf{x})$ is called the monitor function (or metric tensor) and $\rho(\mathbf{x}) = \sqrt{\det(M(\mathbf{x}))}$ is the corresponding mesh density function, and "$\in$" is the set membership relation. A mesh satisfying this condition is called an *M-uniform mesh*.

Taking the 1D case as an example, the equidistributing condition translates to: Given an integer $N > 1$, and a continuous function $\rho = \rho(x) > 0$ on interval $[\theta_0, \theta_1]$, the equidistributing mesh $\mathcal{J}_h : \theta_0 = x_0 < x_1 < \cdots < x_N = \theta_1$ evenly distributes the mesh density function ρ among the subintervals determined by the mesh points such that

$$\int_{x_0}^{x_1} \rho(x) dx = \cdots = \int_{x_{N-1}}^{x_N} \rho(x) dx = \frac{\sigma}{N}, \tag{4.1}$$

where

$$\sigma = \int_{\theta_0}^{\theta_1} \rho(x) dx.$$

The geometric meaning of these identities is that the area under function $\rho(x)$ is the same for every subinterval. Here the monitor function is chosen as $M(x) = \rho(x)^2$. It can be proven that, given a fixed integer N that is large enough, under certain conditions, there is a unique mesh that satisfies the equidistribution condition in Equation (4.1) [10].

It is easy to see from Equation (4.1) that

$$\int_{\theta_0}^{x_j} \rho(x) dx = \sigma \frac{j}{N}.$$

Looking at this equation, if we define a continuous mapping $x = x(u) : [0,1] \to [\theta_0, \theta_1]$ that satisfies

$$\int_{\theta_0}^{x(u)} \rho(x)dx = \sigma u,$$

this mapping is called an equidistributing coordinate transformation for $\rho(x)$. Differentiating this with respect to u on both sides of this integral equation gives

$$\rho(x)\frac{dx}{du} = \sigma. \tag{4.2}$$

Note that this is nonlinear. It is sometimes more useful to formulate the equidistributing condition in terms of inverse coordinate transformation $u = u(x) : [\theta_0, \theta_1] \to [0,1]$. Therefore, we rewrite Equation (4.2) as

$$\frac{1}{\rho(x)}\frac{du}{dx} = \frac{1}{\sigma}, \tag{4.3}$$

which now becomes linear. Differentiating this with respect to x on both sides of Equation (4.3) and writing

$$a(x) = \frac{1}{\rho(x)},$$

we get the elliptic equidistributing mesh generator in 1D with boundary conditions

$$\frac{d}{dx}\left(a(x)\frac{du}{dx}\right) = 0,\ u(\theta_0) = 0,\ u(\theta_1) = 1. \tag{4.4}$$

It should be noted that this equation uses inverse coordinate transformation $u = u(x)$ from physical domain $[\theta_0, \theta_1]$ to computational domain $[0,1]$, so it does not directly give node locations on the physical domain. Note in passing that in the multiple dimensional case, the elliptic mesh generator generalizes to

$$-\nabla \cdot (a(\mathbf{x})\nabla \mathbf{u}) = b(\mathbf{x}), \quad \mathbf{x} \in \Omega \subset \mathbb{R}^d.$$

This is called the Winslow grid generator [22]. They can be used to construct solution-adaptive meshes. For the determination of the weight function, the simplest is the idea of feature-adaptive weights in which the weight function depends upon the features of the solution such as the function itself, its gradient or second derivative.

4.2 Discretization

First, consider Equation (4.4) in 1D with Dirichlet boundary conditions at the boundaries $\theta_0 = -1$ and $\theta_1 = 1$

$$u(-1) = u_0 = 0, \quad u(1) = u_N = 1.$$

Using the second-order finite-difference discretization on a uniform mesh with step size h, we get

$$\begin{aligned}
-\frac{d}{dx}\left(a(x)\frac{du}{dx}\right)\bigg|_{x_j} &\approx -\frac{a(x_{j+1/2})u'(x_{j+1/2}) - a(x_{j-1/2})u'(x_{j-1/2})}{h} \\
&\approx -\frac{1}{h}\left[a(x_{j+1/2})\frac{u(x_{j+1}) - u(x_j)}{h} - a(x_{j-1/2})\frac{u(x_j) - u(x_{j-1})}{h}\right] \\
&= \frac{1}{h^2}\big[-a_{j-1/2}u(x_{j-1}) + \big(a_{j-1/2} + a_{j+1/2}\big)u(x_j) - a_{j+1/2}u(x_{j+1})\big].
\end{aligned}$$

where $x_{j\pm\frac{1}{2}} = (x_j + x_{j\pm1})/2$, and $a_{j\pm\frac{1}{2}} = a(x_{j\pm\frac{1}{2}})$. Therefore, the finite-difference approximation of the differential equation in (4.4) is

$$\frac{1}{h^2}\big[-a_{j-1/2}u_{j-1} + \big(a_{j-1/2} + a_{j+1/2}\big)u_j - a_{j+1/2}u_{j+1}\big] = 0,$$

where u_j is the approximation of $u(x_j)$. Therefore, the linear system to be solved is $A\mathbf{u} = \mathbf{b}$ with

$$A = \frac{1}{h^2}\begin{bmatrix}
a_{\frac{1}{2}} + a_{\frac{3}{2}} & -a_{\frac{3}{2}} & 0 & \cdots & \cdots & 0 \\
-a_{\frac{3}{2}} & a_{\frac{3}{2}} + a_{\frac{5}{2}} & -a_{\frac{5}{2}} & & & 0 \\
\vdots & \ddots & \ddots & \ddots & & \vdots \\
\vdots & & \ddots & \ddots & \ddots & \vdots \\
0 & & & -a_{N-\frac{5}{2}} & a_{N-\frac{5}{2}} + a_{N-\frac{3}{2}} & -a_{N-\frac{3}{2}} \\
0 & \cdots & \cdots & 0 & -a_{N-\frac{3}{2}} & a_{N-\frac{3}{2}} + a_{N-\frac{1}{2}}
\end{bmatrix}, \tag{4.5}$$

and

$$\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ \vdots \\ u_{N-2} \\ u_{N-1} \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} \frac{a_{1/2}u_0}{h^2} \\ 0 \\ \vdots \\ \vdots \\ 0 \\ \frac{a_{N-1/2}u_N}{h^2} \end{bmatrix}. \tag{4.6}$$

4.3 Monotonicity of exact solutions

Regarding the exact solutions of both the continuous boundary value problem (BVP) problem in Equation (4.4) and its discretized analogue, the following results are known.

Theorem 4.3.1. *For the model BVP in Equation (4.4), when $a(x) > 0$ on $[-1, 1]$, the solution, $u(x)$, is monotonically increasing, $\frac{du}{dx} > 0$.*

Proof. Integrating twice on both sides of this equation gives

$$\frac{1}{\rho(x)}\frac{du}{dx} = \frac{1}{\sigma}, \quad u(x) = \frac{1}{\sigma}\int_{-1}^{x} \rho(y)dy + c.$$

Substituting the boundary conditions, $u(-1) = 0$ and $u(1) = 1$, gives $c = 0, \sigma = \int_{-1}^{1} \rho(x)dx$. Therefore, since $\rho(x) > 0$, we have

$$\frac{du}{dx} = \frac{\rho(x)}{\sigma} > 0.$$

This proves the statement. □

Theorem 4.3.2. *For the finite difference discretization $A\mathbf{u} = \mathbf{b}$ of the model problem, the exact solution of this linear system is monotonically increasing, i.e. $u_{j+1} > u_j$ for $1 \leq j \leq N - 1$.*

Proof. In the discretized linear system, we have the equations

$$-a(x_{j-1/2})u_{j-1} + \big(a(x_{j-1/2}) + a(x_{j+1/2})\big)u_j - a(x_{j+1/2})u_{j+1} = 0.$$

for $1 \leq j \leq N-1$. Rearranging this equation, we have

$$\frac{u_{j+1}-u_j}{u_j-u_{j-1}} = \frac{a(x_{j-1/2})}{a(x_{j+1/2})} > 0. \tag{4.7}$$

This implies $u_j \neq u_{j+1}$ for all j. Given the boundary conditions

$$1 = u(1) = u_N > u_0 = u(-1) = 0,$$

and using proof by contradiction, if $u_0 > u_1$, then, from Equation (4.7), $u_1 > u_2$, $u_2 > u_3$, ..., $u_{N-1} > u_N$. This implies $u_0 > u_N$, which is in contradiction with the boundary conditions. Thus, we can conclude that $u_1 > u_0$ and, $u_{j+1} > u_j$, for all j. Therefore, u_j is increasing. □

4.4 Monotonicity preserving property of the Jacobi and Gauss-Seidel methods

These results give us the conclusion that the exact solution of Equation (4.4) is monotonic at both the continuous and discrete level. However, in practice, we usually solve for its numerical solution with iterative methods, which means we cannot get the exact solution, but need to stop the iteration at some point where the approximate solution is within a satisfactory accuracy. Although we can sometimes theoretically analyze an upper bound of the error of the approximate solution, it is not clear how accurately we need to solve the problem in order to get a monotonic approximate solution.

Therefore, to ensure that the approximate solution of the discretized system is monotonic even if we stop early, we try to make the approximate solution monotonic at every iteration, given a monotonic initial guess. For completeness, we prove in this section that the weighted Jacobi and Gauss-Seidel methods are able to preserve solution monotonicity.

Theorem 4.4.1. *Consider the discretized linear system $A\mathbf{u} = \mathbf{f}$ given in Equations (4.5) and (4.6), and a monotonically increasing initial guess. When using the weighted Jacobi method to solve this system, the approximate solution will preserve the monotonicity property in each iteration if the parameter, ω, satisfies $\omega \leq \frac{1}{2}$. If, in addition,*

$a(x)$ is also monotonic, then monotonicity is preserved for $\omega \leq \frac{2}{3}$.

Proof. The weighted Jacobi iterative algorithm can be expressed as

$$\mathbf{u}^{(k+1)} = (I - \omega D^{-1}A)\mathbf{u}^{(k)} + \omega D^{-1}\mathbf{b} = D^{-1}\big[(D-\omega A)\mathbf{u}^{(k)} + \omega \mathbf{b}\big].$$

and the matrix $D - \omega A$ is

$$D - \omega A = \begin{bmatrix} (1-\omega)s_1 & \omega a_{\frac{3}{2}} & 0 & \cdots & \cdots & 0 \\ \omega a_{\frac{3}{2}} & (1-\omega)s_2 & \omega a_{\frac{5}{2}} & & & 0 \\ \vdots & \ddots & \ddots & \ddots & & \vdots \\ 0 & & \omega a_{j-\frac{1}{2}} & (1-\omega)s_j & \omega a_{j+\frac{1}{2}} & 0 \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ 0 & & & \omega a_{N-\frac{5}{2}} & (1-\omega)s_{N-2} & \omega a_{N-\frac{3}{2}} \\ 0 & \cdots & \cdots & 0 & \omega a_{N-\frac{3}{2}} & (1-\omega)s_{N-1} \end{bmatrix},$$

where $s_j = a_{j-\frac{1}{2}} + a_{j+\frac{1}{2}}$. If we let $d_j = u_j - u_{j-1}$ and using the facts that $u_0 = 0,\ u_N = 1$, by shift subtracting the entries of $\mathbf{u}$, we can easily get

$$\begin{cases} d_1^{(k+1)} &= \left(1 - \frac{\omega a_{1/2}}{s_1}\right) d_1^{(k)} + \frac{\omega a_{3/2}}{s_1} d_2^{(k)}, \\ d_j^{(k+1)} &= \frac{\omega a_{j-\frac{1}{2}}}{s_j} d_{j-1}^{(k)} + \left(1 - \omega a_{j+\frac{1}{2}}\big(\frac{1}{s_j} + \frac{1}{s_{j+1}}\big)\right) d_j^{(k)} + \frac{\omega a_{j+\frac{3}{2}}}{s_{j+1}} d_{j+1}^{(k)}, \qquad 2 \leq j \leq N-1, \\ d_N^{(k+1)} &= \frac{\omega a_{N-\frac{3}{2}}}{s_{N-1}} d_{N-1}^{(k)} + \left(1 - \frac{\omega a_{N-\frac{1}{2}}}{s_{N-1}}\right) d_N^{(k)}. \end{cases}$$

Since

$$a_{j+\frac{1}{2}}\left(\frac{1}{s_j} + \frac{1}{s_{j+1}}\right) = \frac{a_{j+\frac{1}{2}}}{a_{j-\frac{1}{2}} + a_{j+\frac{1}{2}}} + \frac{a_{j+\frac{1}{2}}}{a_{j+\frac{1}{2}} + a_{j+\frac{3}{2}}} < 2,$$

when $\omega \leq \frac{1}{2}$, we have $1 - \omega a_{j+\frac{1}{2}}\big(\frac{1}{s_j} + \frac{1}{s_{j+1}}\big) > 0$. Given $\omega \leq \frac{1}{2}$, it is obvious that $1 - \frac{\omega a_{1/2}}{s_1} > 0$ and $1 - \frac{\omega a_{N-\frac{1}{2}}}{s_{N-1}} > 0$. Therefore, given a monotonically increasing initial guess, $\mathbf{u}^{(0)}$, the approximate solution $\mathbf{u}^{(k)}$ is also monotonically increasing for all k.

If, in addition, $a(x)$ is not only positive but monotonically increasing, then we can see that

$$\frac{a_{j+\frac{1}{2}}}{a_{j-\frac{1}{2}} + a_{j+\frac{1}{2}}} + \frac{a_{j+\frac{1}{2}}}{a_{j+\frac{1}{2}} + a_{j+\frac{3}{2}}} < \frac{a_{j+\frac{1}{2}}}{a_{j+\frac{1}{2}}} + \frac{a_{j+\frac{1}{2}}}{a_{j+\frac{1}{2}} + a_{j+\frac{1}{2}}} = 1 + \frac{1}{2} = \frac{3}{2},$$

with a similar bound for $a(x)$ monotonically decreasing. Therefore, if $a(x)$ is monotonic, when $\omega \leq \frac{2}{3}$, the monotonicity of $\mathbf{u}$ is preserved in each iteration. $\square$

Theorem 4.4.2. *Consider the discretized linear system $A\mathbf{u} = \mathbf{b}$ given in Equations (4.5) and (4.6), and a monotonically increasing initial guess. When using the Gauss-Seidel method to solve this system, the approximate solution is always monotonically increasing in each iteration.*

Proof. The iterative scheme for the Gauss-Seidel method is given by

$$-a_{j-\frac{1}{2}} u_{j-1}^{(k+1)} + (a_{j-\frac{1}{2}} + a_{j+\frac{1}{2}}) u_j^{(k+1)} = a_{j+\frac{1}{2}} u_{j+1}^{(k)},$$

for $1 \leq j \leq N-1$. Let $d_j = u_j - u_{j-1}$ and using the fact that $u_0 = 0, u_N = 1$, we can easily get

$$\left(1 + \frac{a_{j+\frac{1}{2}}}{a_{j+\frac{3}{2}}}\right) d_{j+1}^{(k+1)} = d_{j+2}^{(k)} + \frac{a_{j-\frac{1}{2}}}{a_{j+\frac{1}{2}}} d_j^{(k+1)},$$

for $1 \leq j \leq N-2$. Hence, if we can show that $d_1^{(k+1)} > 0$ and $d_N^{(k+1)} > 0$, the result follows. To prove $d_1^{(k+1)} > 0$, notice that $(a_{\frac{1}{2}} + a_{\frac{3}{2}}) u_1^{(k+1)} = a_{\frac{3}{2}} u_2^{(k)}$. Because by assumption $u_j^{(k)} > u_0^{(k)} = 0$ for all j, it follows that $d_1^{(k+1)} = u_1^{(k+1)} > 0$, which again gives $d_j^{(k+1)} > 0$ for $1 \leq j \leq N-1$. To prove $d_N^{(k+1)} > 0$, rearrange the last equation

$$-a_{N-\frac{3}{2}} u_{N-2}^{(k+1)} + (a_{N-\frac{3}{2}} + a_{N-\frac{1}{2}}) u_{N-1}^{(k+1)} = a_{N-\frac{1}{2}} u_N,$$

and subtract $(a_{N-\frac{3}{2}} + a_{N-\frac{1}{2}}) u_N$ from both sides to get

$$d_N^{(k+1)} = \frac{a_{N-\frac{3}{2}}}{a_{N-\frac{1}{2}}} d_{N-1}^{(k+1)},$$

therefore, $d_N^{(k+1)} > 0$. This completes the proof. $\square$

4.5 Monotonicity preserving multigrid methods

4.5.1 Algorithm description

Because the Jacobi and Gauss-Seidel methods are very slow to converge in practice, we design an efficient multigrid algorithm which has the nice property of preserving monotonicity at each iteration and every grid level. We first write the original linear system in mesh-increment form as $\hat{A}\mathbf{d} = \mathbf{0}$, with $\hat{A}$ given by

$$\hat{A} = \begin{bmatrix} -a_{1/2} & a_{3/2} & & & \\ & -a_{3/2} & a_{5/2} & & \\ & & \ddots & & \\ & & & -a_{N-3/2} & a_{N-1/2} \\ a_{1/2} & \cdots & & \cdots & -a_{N-1/2} \end{bmatrix}, \tag{4.8}$$

and $d_j = u_j - u_{j-1}$. This singular matrix $\hat{A}$ can be seen as the generator matrix of a CTMC. Moreover, from Theorem 2.3.2, it can be shown that $-\hat{A}$ is an irreducible, singular M-matrix. By Theorem 2.1.4, $\hat{A}\mathbf{d} = \mathbf{0}$ has a unique positive solution up to a constant scaling.

We use this concrete example to analyze carefully the necessity of Theorem 3.1.1 for a semiconvergent relaxation scheme. The Jacobi iteration matrix for $\hat{A}$ is

$$H_J = I - D^{-1}\hat{A} = \begin{bmatrix} 0 & t_1 & & & \\ & 0 & t_2 & & \\ & & \ddots & & \\ & & & 0 & t_{N-1} \\ t_N & \cdots & \cdots & \cdots & 0 \end{bmatrix},$$

where $t_j = \frac{a_{j+1/2}}{a_{j-1/2}}$ for $1 \leq j \leq N-1$ and $t_N = \frac{a_{1/2}}{a_{N-1/2}}$. We see that H_J is a cyclic matrix of index N. According to the properties of a cyclic matrix in Theorem 2.1.1, all of the eigenvalues of the Jacobi iteration matrix, H_J, with magnitude equal to its spectral radius, $\rho(H_J)$, are

$$\lambda_k = \rho(H_J)e^{\frac{i2\pi k}{N}}.$$

Therefore, without even knowing the spectral radius of H_J, we can conclude that the Jacobi iteration matrix, H_J, is not semiconvergent, because it obviously could not satisfy the third condition of semiconvergence in Theorem 2.3.3, which requires that it should have a simple eigenvalue with modulus equal to its spectral radius. In fact, from numerical experiments, we find that $\rho(H_J) = 1$. This can again be verified by the computation that $H_J^N = sI$ with $s = t_1 t_2 \cdots t_N = \frac{a_{3/2}}{a_{1/2}}\frac{a_{5/2}}{a_{3/2}} \cdots \frac{a_{5/2}}{a_{3/2}}\frac{a_{1/2}}{a_{N-1/2}} = 1$. Therefore, we have $H_J^N\mathbf{v} = \mathbf{v}$, which implies 1 is an eigenvalue of H_J^N, so the eigenvalues of H_J satisfy $\lambda_k^N = 1$, hence $\lambda_k = e^{\frac{i2\pi k}{N}}$, which are the n-th roots of unity.

However, Theorem 3.1.1 tells us that the weighted Jacobi iteration matrix $H_\omega = (1-\omega)I + \omega H_J = (1-\omega)I + \omega(I - D^{-1}\hat{A})$ is semiconvergent for $\omega \in (0,1)$, and the iterative scheme is taken as

$$\mathbf{d}^{(k+1)} = (1-\omega)\mathbf{d}^{(k)} + \omega(I - D^{-1}\hat{A})\mathbf{d}^{(k)} = \mathbf{d}^{(k)} - \omega D^{-1}\hat{A}\mathbf{d}^{(k)},$$

which is obviously positivity preserving when $0 < \omega < 1$ (also proved in Theorem 3.1.2).

Before making use of Algorithms 5 and 6, we still need to decide on the interpolation operator to be used. For this application, we consider the pairwise aggregation operator,

$$P = \begin{bmatrix} 1 & & & \\ 1 & & & \\ & 1 & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \\ & & & 1 \end{bmatrix},$$

and the restriction operator defined as P^T. This can be understood geometrically by noticing that

$$d_j^c = x_j^c - x_{j-1}^c = x_{2j} - x_{2j-2} = x_{2j} - x_{2j-1} + x_{2j-1} - x_{2j-2} = d_{2j} + d_{2j-1},$$

where the superscript c means on a coarse level.

Note that for this interpolation operator, P^TDP is diagonal for any diagonal matrix D. From the discussion in Section 3.1.2, we know that the coarse-grid operator $\hat{A}_c = P^T\hat{A}[\text{diag}(\mathbf{u}^h)]P$ is a singular M-matrix with every column summing to zero. Therefore, we can apply the multigrid Algorithms 5 and 6 to solve this problem.

4.5.2 Numerical results

To test the performance of this algorithm, we record the number of iterations needed to reach a fixed tolerance for the system residual, using 2 pre- and post- weighted Jacobi iterations, and a pairwise aggregation operator, P^T. For the test case with mesh density function

$$\frac{1}{a(x)} = 1 + R\big(1 - \tanh^2(Rx)\big),$$

defined on the domain $x \in [-1, 1]$, we can find that the exact solution to Equation (4.4) is

$$u(x) = \frac{1 + x + \tanh(Rx) - \tanh(-R)}{2 + \tanh(R) - \tanh(-R)}.$$

We show the graphs of mesh density function, $\frac{1}{a(x)}$, and solution, $u(x)$, for comparison in Figure 4.1 when $R = 10$. We can see that the solution varies rapidly in the middle of the domain, and this corresponds to a rapid increase of the mesh density function. According to the equidistributing principle, the generated mesh will have a higher density in the region where the mesh density function is larger. Therefore, the resulting mesh will have a higher density in the middle of the domain and is better able to resolve the rapid change of the solution, which is just what we want.

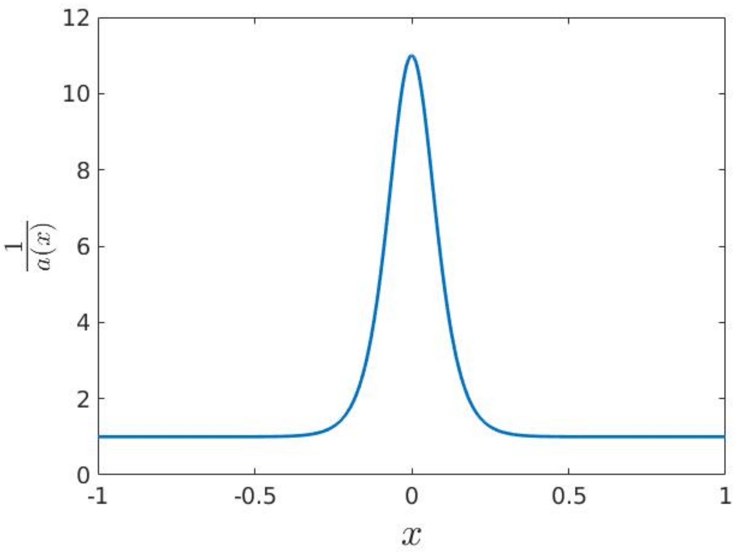
Now we compute the solution for the model problem with our positivity-preserving multigrid method using Algorithm 6. Tables 4.1 and 4.2 show the number of iterations required to reach convergence, $||\hat{A}\mathbf{d}||_2 < 10^{-9}$, when we choose $R = 10$ and $R = 50$ respectively. We can see that this algorithm gives mesh independent convergence, and that the convergence observed is typical of a normal multigrid algorithm for a diffusion equation. Also note that the number of iterations decreases with increasing N, as a result of the decrease in mesh size and the residual-norm stopping criteria we have chosen. It is important to note here that the computed solution $u = u(x)$ does not give node locations on the physical domain because we are using the inverse transformation. To show that the solution behaviour is correct, we plot the inverse function $x = x(u)$ from the solution $u = u(x)$ that we have computed using $N = 1024$. We can see from Figure 4.2 that a uniform mesh on the computational domain gives an adaptive mesh on the physical domain, where mesh points are concentrated in the region where the mesh density function is large. Figure 4.3 depicts the evolution of the mesh during the first 6 V-cycles on a mesh with 32 grid points for the case when $R = 10$. We can see that the mesh points quickly concentrate in the middle.

N	32	64	128	256	512	1024	2048
2G	14	11	8	6	4	3	3
3G	16	15	13	9	6	4	3
4G	17	16	16	14	9	6	4
5G		17	16	15	13	9	6
6G			16	15	14	12	8
7G				15	14	13	11
8G					14	13	12
9G						13	12

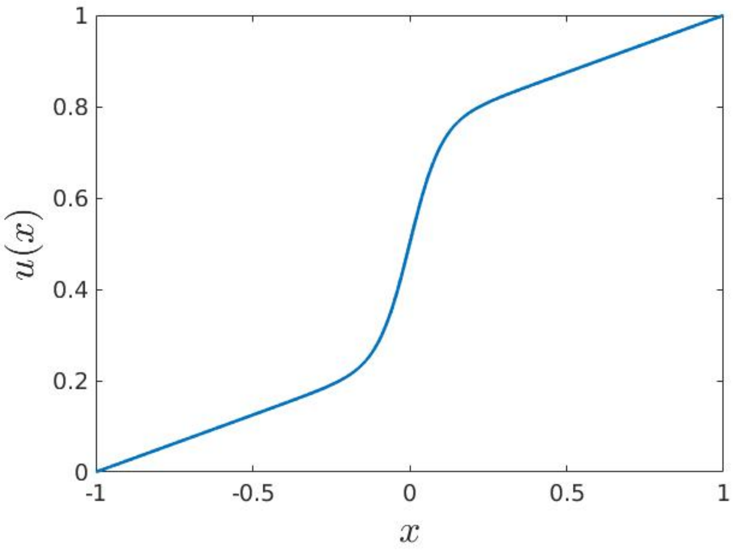
Table 4.1: Number of iterations needed to reach $||\hat{A}\mathbf{d}||_2 < 10^{-9}$ with $N + 1$ mesh points for different N. The constant R in the weighting function $a(x)$ is chosen to be $R = 10$. The labels 2G, 3G, ..., indicate the results for 2-grid algorithm, 3-grid algorithm, etc.

N	32	64	128	256	512	1024	2048
2G	13	13	13	11	11	5	4
3G	15	15	15	15	15	9	6
4G	16	15	15	15	15	13	9
5G		15	15	15	15	14	12
6G			15	15	15	14	13
7G				15	15	14	13
8G					15	14	13
9G						14	13

Table 4.2: Number of iterations needed to reach $||\hat{A}\mathbf{d}||_2 < 10^{-9}$ with $N + 1$ mesh points for different N. The constant R in the weighting function $a(x)$ is chosen to be $R = 50$. The labels 2G, 3G, ..., indicate the results for 2-grid algorithm, 3-grid algorithm, etc.



(a) Mesh density function $\frac{1}{a(x)}$



(b) Solution $u(x)$

Figure 4.1: Graphs of mesh density function $\frac{1}{a(x)} = 1 + 10(1 - \tanh^2(10x))$, and the corresponding exact solution of Equation (4.4).

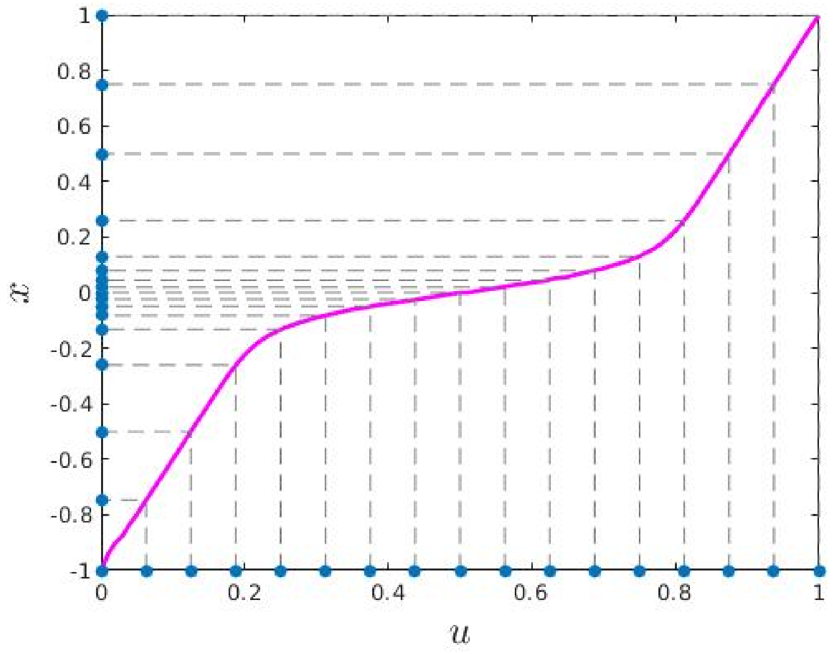


Figure 4.2: Positions of mesh points on the physical domain [-1,1] computed from an uniform mesh on the computational domain [0, 1] using the mapping $x = x(u)$. For visual clarity, only 17 mesh points are taken.

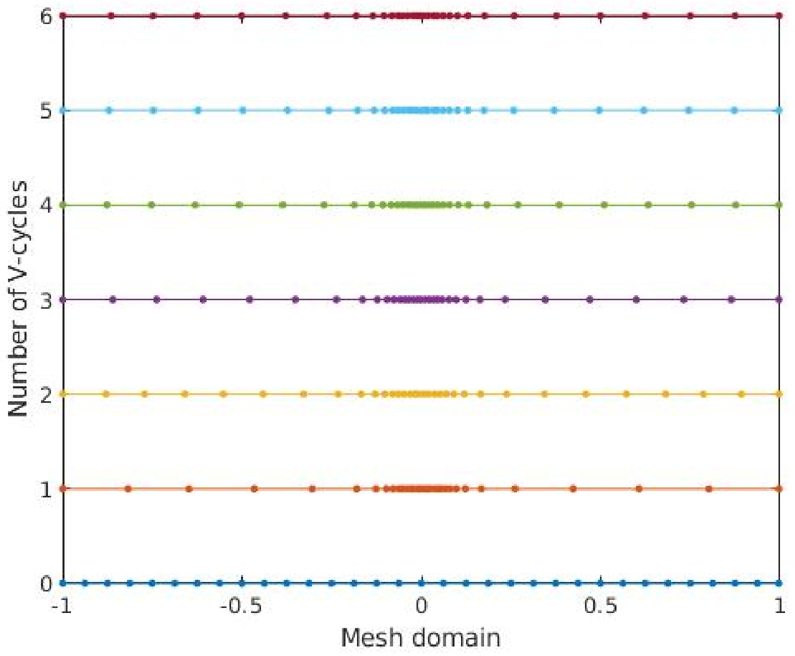


Figure 4.3: Mesh evolution with respect to V-cycle iterations for 32 mesh points

Chapter 5

Application to Nonlinear Diffusion Equations

In this chapter we discuss the application of positivity-preserving multigrid methods to nonlinear diffusion equations, which is a nonlinear generalization of the elliptic mesh generator we saw in the last chapter. We will only look at the steady state case in 1D:

$$-\frac{d}{dx}\left(a(u)\frac{du}{dx}\right) = f(x,u),$$

with Dirichlet boundary conditions

$$u(0) = g_0,\ u(1) = g_1,$$

where $u = u(x), a(u) > 0, f(u) \geq 0$, and $g_0, g_1 \geq 0$ are some constant numbers. The non-negative restriction on $f(x,u)$ and g_0, g_1 is to ensure that the solution, $u(x)$, is pointwise positive. Such equations arise in a wide range of applications, such as heat transfer, biological processes, chemical reactions, porous media and ground water modelling.

The model boundary value problem that we will use in the numerical results is

$$-\frac{d}{dx}\left((1+u^2)\frac{du}{dx}\right) = 1, \quad x \in (0,1), \tag{5.1}$$

with $a(u) = 1 + u^2$ and $f(u) = 1$. Suppose the Dirichlet boundary conditions are homogeneous

$$u(0) = u(1) = 0,$$

so that the exact solution satisfies $u + \frac{1}{3}u^3 = \frac{1}{2}(x - x^2)$. The analytical solution of this model problem is positive.

5.1 Discretization

As in the linear case, the centered finite-difference method on a uniform mesh gives

$$\begin{aligned} &-\frac{d}{dx}\left(a(u)\frac{du}{dx}\right)_j \\ \approx &\frac{1}{h^2}\Big(- a(u_{j-1/2})u_{j-1} + \big(a(u_{j-1/2}) + a(u_{j+1/2})\big)u_j - a(u_{j+1/2})u_{j+1}\Big), \end{aligned}$$

where $u_j = u(x_j)$ and so on. The coefficient values can be approximated by [15]

$$a(u_{j-1/2}) = a\big((u_{j-1} + u_j)/2\big), \quad a(u_{j+1/2}) = a\big((u_j + u_{j+1})/2\big).$$

Then, we end up with a nonlinear algebraic system $A(\mathbf{u})\mathbf{u} - \mathbf{f}(\mathbf{u}) = \mathbf{0}$. The discretization with homogeneous Dirichlet boundary conditions on both sides gives the system matrix

$$A(\mathbf{u}) = \frac{1}{h^2}\begin{bmatrix} a_{1/2} + a_{3/2} & -a_{3/2} & 0 & \cdots & 0 \\ -a_{3/2} & a_{3/2} + a_{5/2} & -a_{5/2} & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -a_{n-5/2} & a_{n-5/2} + a_{n-3/2} & -a_{n-3/2} \\ 0 & \cdots & 0 & -a_{n-3/2} & a_{n-3/2} + a_{n-1/2} \end{bmatrix}. \tag{5.2}$$

where $a_{j-1/2} = a\big((u_{j-1} + u_j)/2\big)$ for $1 \leq j \leq n$,

$$\mathbf{u} = [u_1, u_2, ..., u_{n-1}]^T, \ \mathbf{f}(\mathbf{u}) = [f(u_1), f(u_2), ..., f(u_{n-1})]^T.$$

5.2 Linearization

Several possible approaches can be used for the linearization, including a Picard iteration or Newton's method. If we use Newton's method and write

$$\mathbf{F}(\mathbf{u}) = A(\mathbf{u})\mathbf{u} - \mathbf{f}(\mathbf{u}) = \mathbf{0},$$

the Newton's iteration with initial guess $\mathbf{u}^{(0)}$ is given by

$$\mathbf{u}^{(k+1)} = \mathbf{u}^{(k)} + \mathbf{s}^{(k)}, \text{ where } \mathbf{F}'(\mathbf{u}^{(k)})\mathbf{s}^{(k)} = \mathbf{F}(\mathbf{u}^{(k)}),$$

where $\mathbf{F}'(\mathbf{u}^{(k)})$ is the Jacobian matrix, which not only depends on the system matrix $A(\mathbf{u})$, but also the the right hand side $\mathbf{f}(\mathbf{u})$ and the derivative of $a(\mathbf{u})$. Therefore, it is not guaranteed to be a Z-matrix. In addition, it is not easy to control the sequence of steps $\mathbf{s}^{(k)}$ such that $\mathbf{u}^{(k)} + \mathbf{s}^{(k)}$ is positive given an initial approximation $\mathbf{u}^{(0)}$.

Thus, to have a chance at a positivity-preserving method, it seems more appropriate to use a Picard iteration for linearization, which means to replace the unknown $\mathbf{u}$ in $A(\mathbf{u})$ and $f(\mathbf{u})$ with values from the previous iteration, i.e.,

$$A(\mathbf{u}^{(k-1)})\mathbf{u}^{(k)} = \mathbf{f}(\mathbf{u}^{(k-1)}) = \mathbf{b}^{(k-1)}. \tag{5.3}$$

Note that $A(\mathbf{u}^{(k-1)})$ is a nonsingular irreducible M-matrix, so the iteration is well-posed. From Theorem 3.2.1, we can see that the exact solution is positive when the nonzero right-hand side satisfies $\mathbf{b}^{(k-1)} \geq 0$.

For our test problem, the forcing function $\mathbf{f}$ is simply a vector of all ones. That is, the system we are trying to solve at every Picard iteration is

$$A(\mathbf{u}^{(k-1)})\mathbf{u}^{(k)} = \mathbf{1}. \tag{5.4}$$

Therefore, we can apply the unigrid Algorithms 7 and 8 developed in Chapter 3 to solve the linearized system at each Picard iteration and ensure that the resulting approximate solution is always positive.

Algorithm 9 gives the pseudocode for a Picard iteration, in which the solution step inside the while loop is the main topic of discussion in this chapter. We will call the Picard iteration the outer loop, and the loop to solve for the next iterative approximation $\mathbf{u}_1$ the inner loop.

Algorithm 9: Picard Iteration

```
PICARD($A, \mathbf{u}_0, f$) :
while $||A(\mathbf{u}_0)\mathbf{u}_0 - \mathbf{f}(\mathbf{u}_0)|| > \tau$ do
    Solve for $\mathbf{u}_1$: $A(\mathbf{u}_0)\mathbf{u}_1 = \mathbf{f}(\mathbf{u}_0)$;
    Relaxation: $\mathbf{u}_1 = \omega\mathbf{u}_1 + (1-\omega)\mathbf{u}_0$;
    Update: $\mathbf{u}_0 = \mathbf{u}_1$;
end
Return $\mathbf{u}_1$;
```

5.3 A two-grid method

Before we give our numerical results using the unigrid Algorithms 7 and 8, it is worth to give a separate discussion of a two-grid method that we discovered from the special structure of the system matrix $A(\mathbf{u})$. We show a situation where the two-grid algorithm always gives the exact solution on the coarse-grid points. From this, we construct the interpolation operators that will give a positive approximate solution. We will first introduce the detailed formulation of the method, and then provide numerical results for the model problem.

5.3.1 Algorithm formulation

In this section, we introduce an approach that is effective as a two-grid method. The generalization of this algorithm to a V-cycle over more grids is not in the scope of this thesis and is left to be studied in the future. A two-grid algorithm consists of a coarse-grid solve and a correction step:

$$\begin{aligned} A^{2h}\mathbf{e}^{2h} &= R(\mathbf{b} - A\mathbf{u}^h), \\ \mathbf{u}^h &= \mathbf{u}^h + P\mathbf{e}^{2h}, \end{aligned}$$

along with one or more relaxation steps. When using a coarse-grid correction, we can see that

$$RA(\mathbf{u}^h + P\mathbf{e}^{2h}) = R\mathbf{b},$$

and the resulting approximate solution is in the solution space of $RA\mathbf{x} = R\mathbf{b}$, because

$$\mathbf{x} = \mathbf{u} + N(RA),$$

where $N(RA)$ is the nullspace of RA, and $\mathbf{u}$ is the true solution, i.e., $A\mathbf{u} = \mathbf{b}$.

We now choose a restriction operator and look to understand the corresponding nullspace of RA. For the variable-coefficient diffusion equation we are considering, the system matrix is given by Equation (5.2). We choose the standard operator-induced restriction operator,

$$R = \begin{bmatrix} q_1 & & & & \\ 1 & & & & \\ p_3 & q_3 & & & \\ & 1 & & & \\ & p_5 & q_5 & & \\ & & 1 & & \\ & & p_7 & \ddots & \\ & & & \ddots & \\ & & & \ddots & q_{N-3} \\ & & & & 1 \\ & & & & p_{N-1} \end{bmatrix}^T = \begin{bmatrix} \frac{a_{\frac{3}{2}}}{a_{\frac{1}{2}}+a_{\frac{3}{2}}} & & & & \\ 1 & & & & \\ \frac{a_{\frac{5}{2}}}{a_{\frac{5}{2}}+a_{\frac{7}{2}}} & \frac{a_{\frac{7}{2}}}{a_{\frac{5}{2}}+a_{\frac{7}{2}}} & & & \\ & 1 & & & \\ & \frac{a_{\frac{9}{2}}}{a_{\frac{9}{2}}+a_{\frac{11}{2}}} & \frac{a_{\frac{11}{2}}}{a_{\frac{9}{2}}+a_{\frac{11}{2}}} & & \\ & & 1 & & \\ & & \frac{a_{\frac{13}{2}}}{a_{\frac{13}{2}}+a_{\frac{15}{2}}} & \ddots & \\ & & & \ddots & \\ & & & \ddots & \frac{a_{N-\frac{5}{2}}}{a_{N-\frac{7}{2}}+a_{N-\frac{5}{2}}} \\ & & & & 1 \\ & & & & \frac{a_{N-\frac{3}{2}}}{a_{N-\frac{3}{2}}+a_{N-\frac{1}{2}}} \end{bmatrix}^T, \tag{5.5}$$

with $p_j = \frac{a_{j-\frac{1}{2}}}{a_{j-\frac{1}{2}}+a_{j+\frac{1}{2}}}$, $q_j = \frac{a_{j+\frac{1}{2}}}{a_{j-\frac{1}{2}}+a_{j+\frac{1}{2}}}$. From these, we get that

$$RA^h = \begin{bmatrix} 0 & t_1+t_3 & 0 & -t_3 & 0 & & \\ 0 & -t_3 & 0 & t_3+t_5 & 0 & -t_5 & 0 \\ & & & \ddots & & & \end{bmatrix},$$

where $t_j = \frac{a_{j-1/2}a_{j+1/2}}{a_{j-1/2}+a_{j+1/2}}$. This gives the nullspace of RA as

$$N(RA) = \left\{[v_1, 0, v_3, 0, \cdots, 0, v_{2j-1}, 0, \cdots]^T \mid v_{2j-1} \in \mathbb{R} \text{ for } 1 \leq j \leq N/2\right\} \tag{5.6}$$

.

Because

$$\mathbf{u}^h + P\mathbf{e}^{2h} = \mathbf{u} - \mathbf{E},\ \mathbf{E} \in N(RA^h), \tag{5.7}$$

we see that $\mathbf{E}$ is the error in the resulting approximate solution and, as long as the true solution is positive (which is reasonable because we want positive approximations of a positive true solution), we can easily choose the vector $\mathbf{E}$ to guarantee the positivity of the multigrid approximation $\mathbf{u}^h + P\mathbf{e}^{2h}$. Also notice that the above equation is also equivalent to requiring

$$P\mathbf{e}^{2h} = \mathbf{e}^h - \mathbf{E},\ \mathbf{E} \in N(RA^h). \tag{5.8}$$

We see that $\mathbf{E}$ is, in fact, also the discrepancy of the interpolated error with the exact error. For example, if $\mathbf{E} = \mathbf{0}$, we want to use P such that the error interpolation is exact, i.e., $P\mathbf{e}^{2h} = \mathbf{e}^h$, so that the corrected solution is also the exact solution. Of course this ideal case is not achievable in practice.

Therefore, we wish to find an interpolation operator P such that $P\mathbf{e}^{2h}$ is a good approximation of $\mathbf{e}^h$ and, at the same time, $\mathbf{u}^h + P\mathbf{e}^{2h}$ is positive given a positive approximation $\mathbf{u}^h$. To construct such an operator P, we start by assuming that P has the form

$$P = \begin{bmatrix} \beta_1 & & & \\ 1 & & & \\ \alpha_1 & \beta_2 & & \\ & 1 & & \\ & \alpha_2 & \beta_3 & \\ & & 1 & \\ & & \alpha_3 & \\ & & & \ddots \end{bmatrix}, \tag{5.9}$$

where $\alpha_j, \beta_j \geq 0$ for all j.

Theorem 5.3.1. *The solution, $\mathbf{e}^{2h}$, on the coarse grid of a two-level multigrid method for our model problem gives the exact error of the initial guess at coarse-grid points, i.e. $e^{2h}_j = e^h_{2j}$.*

Proof. This is easy to see by substituting P from Equation (5.9) to Equation (5.8) and comparing the left- and right-hand sides. □

Theorem 5.3.2. *A two-level multigrid method for our model problem with interpolation operator P given by Equation (5.9) and restriction operator R given by Equation (5.5) always gives the exact solution on coarse-grid points.*

Proof. By Equation (5.7), we have

$$\mathbf{u}^h + P\mathbf{e}^{2h} = \begin{bmatrix} u_1^h \\ u_2^h \\ u_3^h \\ u_4^h \\ u_5^h \\ \vdots \end{bmatrix} + \begin{bmatrix} \beta_1 e_1^{2h} \\ e_1^{2h} \\ \alpha_1 e_1^{2h} + \beta_2 e_2^{2h} \\ e_2^{2h} \\ \alpha_2 e_2^{2h} + \beta_3 e_3^{2h} \\ \vdots \end{bmatrix} = \begin{bmatrix} u_1^h + \delta_1 \\ u_2^h + e_1^{2h} \\ u_3^h + \delta_3 \\ u_4^h + e_2^{2h} \\ u_5^h + \delta_5 \\ \vdots \end{bmatrix} = \begin{bmatrix} u_1 - E_1^h \\ u_2 \\ u_3 - E_3^h \\ u_4 \\ u_5 - E_5^h \\ \vdots \end{bmatrix}.$$

From this, we see that $u_{2j}^h + e_j^{2h} = u_{2j}$, consistent with Theorem 5.3.1. In addition, we see that the corrected multigrid solution is

$$\mathbf{u}^h + P\mathbf{e}^{2h} = \begin{bmatrix} u_1^h + \delta_1 \\ u_2 \\ u_3^h + \delta_3 \\ u_4 \\ u_5^h + \delta_5 \\ \vdots \end{bmatrix}.$$

This proves the theorem. □

In order to guarantee the positivity of the multigrid solution, we only need to make sure that the approximate solution on the fine-grid points is positive. That is, we want

$$u_{2j-1}^h + \delta_{2j-1} > 0, \tag{5.10}$$

where

$$\begin{aligned} &\delta_1 = \beta_1 e_2^h, \\ &\delta_{2j-1} = \alpha_{j-1} e_{2j-2}^h + \beta_j e_{2j}^h, \quad j = 2, \cdots, N-1, \\ &\delta_{2N-1} = \alpha_{N-1} e_{2N-2}^h. \end{aligned} \tag{5.11}$$

Trivially, if we take $\delta_1 = \delta_3 = \cdots = \delta_{2N-1} = 0$, namely choose $\alpha_j = \beta_j = 0$, then the interpolation operator becomes

$$P = \begin{bmatrix} 0 & & & \\ 1 & & & \\ 0 & 0 & & \\ & 1 & & \\ & 0 & 0 & \\ & & 1 & \\ & & 0 & \\ & & & \ddots \end{bmatrix}.$$

and

$$\mathbf{u}^h + P\mathbf{e}^{2h} = \begin{bmatrix} u_1^h & u_2 & u_3^h & u_4 & u_5^h & \cdots \end{bmatrix}^T > \mathbf{0},$$

given any positive initial approximation. The injection interpolation operator is, however, well-known to be too simplistic to yield the best-possible performance from a multigrid method.

Hence, we try to construct an interpolation operator of the form given in Equation (5.9) using the constraint condition in Equation (5.10). Because $\mathbf{u}^h$ is given, the sign of u_{2j-1}^h is dependent on the values e_{2j-2}^h and e_{2j}^h for the general case. It is easier for analysis if we know the signs of e_{2j-2}^h and e_{2j}^h. Therefore, we consider 4 cases below, taking $N = 6$ for simplicity of discussion.

- Case 1: $e_2^h > 0, e_4^h > 0$. Then, for any $\alpha_j, \beta_j > 0$

  $$\delta_1 = \beta_1 e_2^h > 0,\ \delta_3 = \alpha_1 e_2^h + \beta_2 e_4^h > 0,\ \delta_5 = \alpha_2 e_4^h > 0.$$

  Therefore, fine-grid solutions are always all positive after correction,

  $$u_1^h + \delta_1 > 0,\ u_2^h + \delta_2 > 0,\ u_3^h + \delta_3 > 0.$$

  Thus, no violation of positivity can occur in this case.

- Case 2: $e_2^h < 0, e_4^h < 0$. Then,

  $$\delta_1 = \beta_1 e_2^h < 0,\ \delta_3 = \alpha_1 e_2^h + \beta_2 e_4^h < 0,\ \delta_5 = \alpha_2 e_4^h < 0.$$

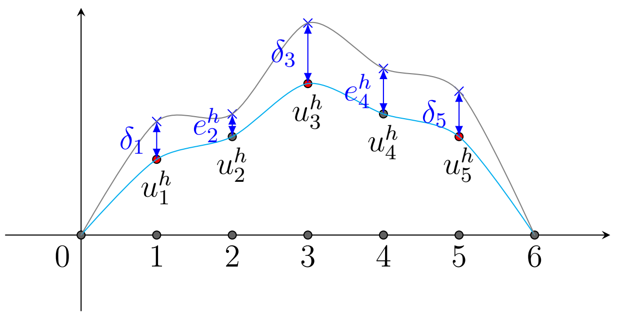


Figure 5.1: Case 1: $e_2^h > 0, e_4^h > 0$. An example of the approximate solution before and after coarse-grid correction. $e_2^h = e_1^{2h}$ and $e_4^h = e_2^{2h}$ are solved on the coarse grid. δ_1, δ_3 and δ_5 are interpolated from $\mathbf{e}^{2h}$ using $P\mathbf{e}^{2h}$. This figure depicts the situation when $e_2^h > 0,\ e_4^h > 0$.

For now, ignore the boundary points at 1 and 5. For point 3, the extreme situation occurs when $u_3^h + \delta_3 = 0$, i.e.

$$u_3^h + \alpha_1 e_2^h + \beta_2 e_4^h = 0. \tag{5.12}$$

As shown in Figure 5.2b, treating α_1 and β_2 as variables, then Equation (5.12) forms a line in the plane and all the points below this line are suitable pairs of parameters such that $u_3^h + \delta_3 > 0$. For points 1 and 5, $u_1^h + \beta_1 e_2^h$ and $u_5^h + \alpha_2 e_4^h$ are guaranteed to be positive in the shaded region. Therefore, the resulting corrected solution is positive when α_1 and β_2 are chosen to be in the shaded region in Figure 5.2b.

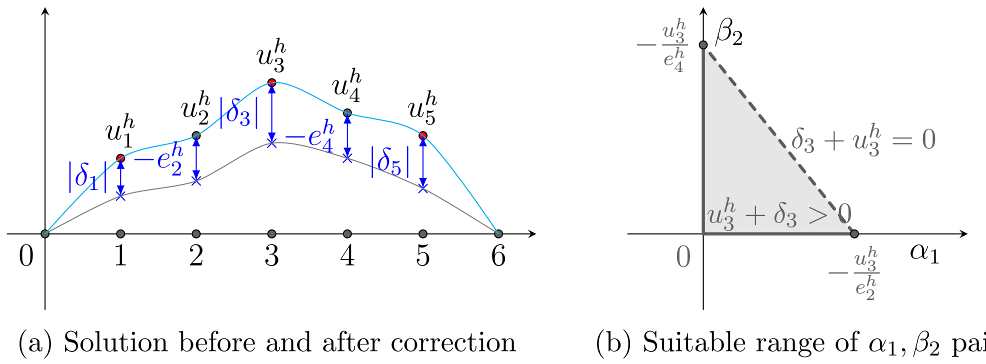
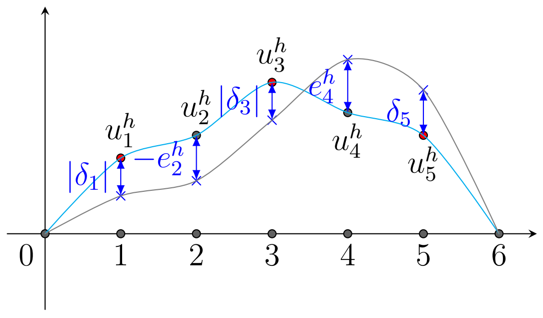
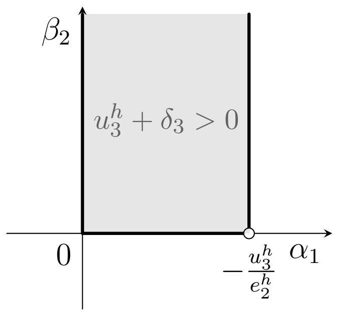


Figure 5.2: Case 2: $e_2^h < 0, e_4^h < 0$. An example of the approximate solution before and after coarse-grid correction. $e_2^h = e_1^{2h}$ and $e_4^h = e_2^{2h}$ are solved on the coarse grid. δ_1, δ_3 and δ_5 are interpolated from $\mathbf{e}^{2h}$ using $P\mathbf{e}^{2h}$. This figure depicts the situation when $e_2^h < 0,\ e_4^h < 0$.



(a) Solution before and after correction



(b) Suitable range of α, β pair

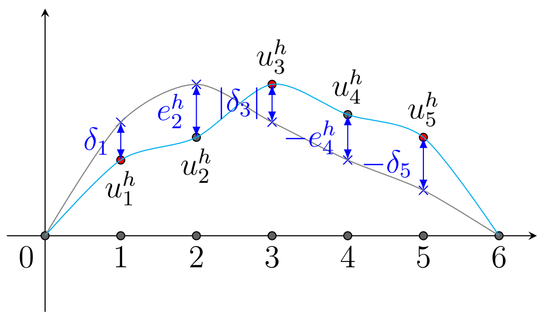
Figure 5.3: Case 3: $e_2^h < 0, e_4^h > 0$. An example of the approximate solution before and after coarse-grid correction. $e_2^h = e_1^{2h}$ and $e_4^h = e_2^{2h}$ are solved on the coarse grid. δ_1, δ_3 and δ_5 are interpolated from $\mathbf{e}^{2h}$ using $P\mathbf{e}^{2h}$. This figure depicts the situation when $e_2^h < 0,\ e_4^h > 0$.

- Case 3: $e_2^h < 0, e_4^h > 0$. Then the sign of $\delta_3 = \alpha_1 e_2^h + \beta_2 e_4^h$ is indefinite, but the extreme case is still when

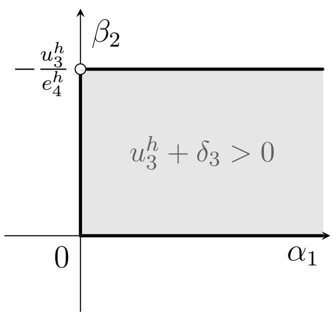
$$u_3^h + \delta_3 = u_3^h + \alpha_1 e_2^h + \beta_2 e_4^h = 0,$$

  which is indicated in Figure 5.3b by an open circle. Taking $\beta_2 = 0$ gives $\alpha_1 = -\frac{u_3^h}{e_2^h}$. If $\beta_2 > 0$ then, as long as $\alpha_1 < -\frac{u_3^h}{e_2^h}$, we can ensure that $u_3^h + \delta_3 > 0$. Hence, the shaded region in Figure 5.3b depicts the acceptable α_1, β_2 pairs, which also ensures that $u_1^h + \beta_1 e_2^h$ and $u_5^h + \alpha_2 e_4^h$ are positive for points 1 and 5. Therefore, the resulting corrected solution is guaranteed to be positive.

- Case 4: $e_2^h > 0, e_4^h < 0$. Similar to case 3, we have the acceptable region as shaded in Figure 5.4b.



(a) Solution before and after correction



(b) Suitable range of α, β pair

Figure 5.4: Case 4: $e_2^h > 0, e_4^h < 0$. An example of the approximate solution before and after coarse-grid correction. $e_2^h = e_1^{2h}$ and $e_4^h = e_2^{2h}$ are solved on the coarse grid. δ_1, δ_3 and δ_5 are interpolated from $\mathbf{e}^{2h}$ using $P\mathbf{e}^{2h}$. This figure depicts the situation when $e_2^h > 0$, $e_4^h < 0$.

The above analysis gives all possible cases of the coarse-grid solution and the corresponding range of α_{j-1} and β_j that we can use to make sure the resulting corrected fine-grid solution is positive. However, we do not know the signs of $e_{2j}^h = e_j^{2h}$ because they are what we are trying to solve for on the coarse grid, and we cannot even know the coarse-grid system to be solved without giving the interpolation operator whose entries are α_{j-1} and β_j.

Luckily, by drawing the shaded region of the 4 cases in one picture as shown in Figure 5.5, we see that the triangle region in case 2 is contained in the acceptable region of all 4 cases. This means that we can use points in this triangle without discriminating between the 4 cases. In order to avoid dependency on the coarse-grid solution $\mathbf{e}^{2h}$, suppose $e_{2j}^h < 0$, note that since

$$u_{2j}^h + e_{2j}^h = u_{2j} > 0,$$

we have $e_{2j}^h > -u_{2j}^h$. Hence

$$-\frac{u_{2j-1}}{e_{2j}} > \frac{u_{2j-1}}{u_{2j}}.$$

And similarly

$$-\frac{u_{2j-1}}{e_{2j-2}} > \frac{u_{2j-1}}{u_{2j-2}},$$

when $e_{2j-2} < 0$. Therefore, as shown in Figure 5.5, we can get a smaller triangular region (the shaded triangle OCD) that satisfies the conditions for all the 4 cases based

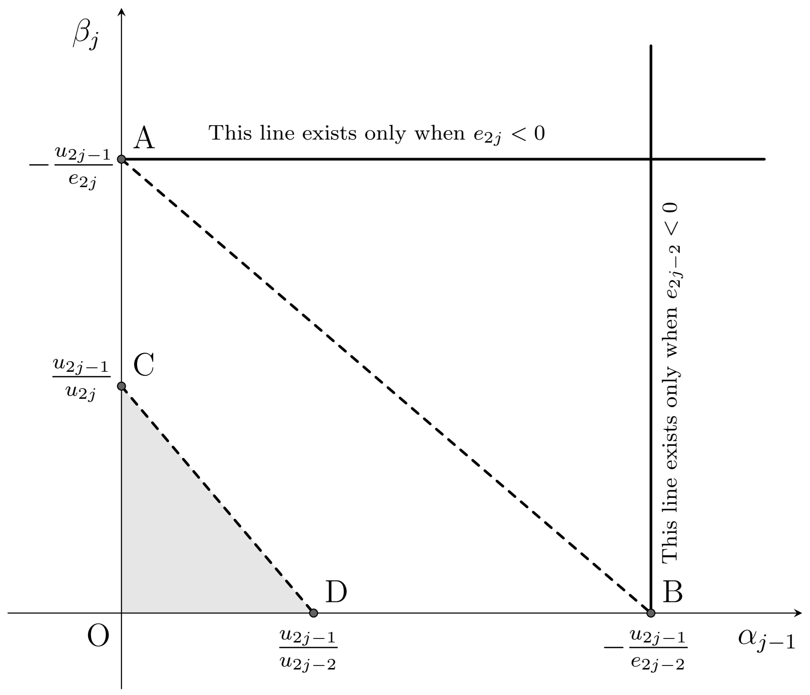

β_j
$-\frac{u_{2j-1}}{e_{2j}}$
A
This line exists only when $e_{2j} < 0$
$\frac{u_{2j-1}}{u_{2j}}$
C
This line exists only when $e_{2j-2} < 0$
D
B
O
$\frac{u_{2j-1}}{u_{2j-2}}$
$-\frac{u_{2j-1}}{e_{2j-2}}$
α_{j-1}

Figure 5.5: The 4 cases shown in one graph. This figure draws the acceptable regions of the 4 cases from Figure 5.2b to Figure 5.4b in one graph. We can see that the triangle from case 2 is the common area of all 4 cases. In addition, the shaded little triangle OCD is guaranteed to be contained by the bigger triangle OAB from case 2.

only on approximation values $\mathbf{u}^h$ known before coarse-grid correction.

Therefore, obvious possibilities for the weights would be to choose points on the line

$$\beta_j = -\frac{1}{u_{2j}}(u_{2j-2}\alpha_{j-1} - u_{2j-1}).$$

Possible examples are to take $\alpha_{j-1} = 0, \beta_j = \frac{u^h_{2j-1}}{u^h_{2j}}$ (option 1 below), or $\alpha_{j-1} = \frac{u^h_{2j-1}}{u^h_{2j-2}}, \beta_j = 0$ (option 2), or $\alpha_{j-1} = \beta_j = \frac{u^h_{2j-1}}{u^h_{2j-2}+u^h_{2j}}$ (option 3). Option 1, $\alpha_{j-1} = 0, \beta_j = \frac{u^h_{2j-1}}{u^h_{2j}}$ gives the first entry of P to be zero, so e^h_1 would not get corrected; however, to avoid this, we can choose $P_{11} = \frac{u^h_1}{u^h_2}$. Similarly for option 2, $\alpha_{j-1} = \frac{u^h_{2j-1}}{u^h_{2j-2}}, \beta_j = 0$, the last entry is nominally zero, so e^h_{N-1} would not get corrected. Again, we can choose this to be $\frac{u^h_{N-1}}{u^h_{N-2}}$. More concretely, the option 1 interpolation operator is given by

$$P_1 = \begin{bmatrix} \frac{u_1^h}{u_2^h} & & & & \\ 1 & & & & \\ 0 & \frac{u_3^h}{u_4^h} & & & \\ & 1 & & & \\ & 0 & \frac{u_5^h}{u_6^h} & & \\ & & 1 & & \\ & & 0 & & \\ & & & \ddots & \\ & & & & \frac{u_{N-3}^h}{u_{N-2}^h} \\ & & & & 1 \\ & & & & \frac{u_{N-1}^h}{u_{N-2}^h} \end{bmatrix}, \tag{5.13}$$

the option 2 interpolation operator is given by

$$P_2 = \begin{bmatrix} \frac{u_1^h}{u_2^h} & & & & \\ 1 & & & & \\ \frac{u_3^h}{u_4^h} & 0 & & & \\ & 1 & & & \\ & \frac{u_5^h}{u_4^h} & 0 & & \\ & & 1 & & \\ & & \frac{u_7^h}{u_6^h} & & \\ & & & \ddots & \\ & & & & 0 \\ & & & & 1 \\ & & & & \frac{u_{N-1}^h}{u_{N-2}^h} \end{bmatrix}, \tag{5.14}$$

and the option 3 interpolation operator is given by

$$P_3 = \begin{bmatrix} \frac{u_1^h}{u_2^h} & & & & \\ 1 & & & & \\ \frac{u_3^h}{u_2^h+u_4^h} & \frac{u_3^h}{u_2^h+u_4^h} & & & \\ & 1 & & & \\ & \frac{u_5^h}{u_4^h+u_6^h} & \frac{u_5^h}{u_4^h+u_6^h} & & \\ & & 1 & & \\ & & \frac{u_7^h}{u_6^h+u_8^h} & & \\ & & & \ddots & \\ & & & & \frac{u_{N-3}^h}{u_{N-4}^h+u_{N-2}^h} \\ & & & & 1 \\ & & & & \frac{u_{N-1}^h}{u_{N-2}^h} \end{bmatrix}. \tag{5.15}$$

Therefore, we get the desired nontrivial interpolation operators that are guaranteed to give a positive approximate solution using a two-grid method. It is interesting to note that these interpolation operators try to predict the behavior of the error according to the approximation. Taking P_1 for example,

$$\delta_{2j-1}^h = \frac{u_{2j-1}^h}{u_{2j}^h} e_j^{2h} = \frac{u_{2j-1}^h}{u_{2j}^h} e_{2j}^h,$$

where e_{2j}^h are the exact errors of the approximate solution at the coarse-grid points, and δ_{2j-1}^h are the interpolated corrections at the fine-grid points, this gives

$$\frac{\delta_{2j-1}^h}{e_{2j}^h} = \frac{u_{2j-1}^h}{u_{2j}^h}.$$

When u_{2j-1}^h is much bigger than u_{2j}^h, then the interpolated correction δ_{2j-1}^h would also be much bigger than the exact error e_{2j}^h, and vice versa. However, we do not know the error behaviour beforehand. Therefore, it seems more appropriate to use a random initial approximation when performing this algorithm, so as to avoid systematic biases in the initial corrections.

5.3.2 Numerical results

In this section, we show the numerical results of solving our model problem using Algorithm 1 with the restriction operators from Equation (5.5) and interpolation operators P_1, P_2 and P_3 formulated above.

In the numerical experiments, we chose a randomly generated initial guess, perform 2 pre- and post-relaxations using the weighted Jacobi method with weight parameter $\omega = 1/3$. The stopping criterion for the inner loop of Algorithm 9 is $||\mathbf{b} - A(\mathbf{u}^{(k)})\mathbf{u}^{(k+1)}||_2 < 10^{-8}$, and for the outer loop, the stopping criterion is $||\mathbf{b} - A(\mathbf{u}^{(k+1)})\mathbf{u}^{(k+1)}||_2 < 10^{-8}$. The number of iterations required for the convergence of the method with different mesh sizes are recorded in Tables 5.1 to 5.3.

N	1st. PIt.	2nd. PIt.	3rd. PIt.	4th. PIt.	5th. PIt.
2^{10}	24	19	11	5	4
2^{9}	23	18	11	5	3
2^{8}	22	17	11	5	3
2^{7}	20	17	10	5	3
2^{6}	19	15	10	5	3

Table 5.1: Number of inner multigrid iterations required for convergence at each outer Picard iteration of the 2-grid algorithm using the interpolation operator P_1. The "PIt." means Picard iteration.

N	1st. PIt.	2nd. PIt.	3rd. PIt.	4th. PIt.	5th. PIt.
2^{10}	24	19	11	5	4
2^{9}	23	18	11	5	3
2^{8}	22	17	11	5	3
2^{7}	20	16	10	5	3
2^{6}	19	15	10	5	3

Table 5.2: Number of inner multigrid iterations required for convergence at each outer Picard iteration of the 2-grid algorithm using option 2 interpolation operator P_2. The "PIt." means Picard iteration.

Note that P_1 and P_2 give the same convergence, while P_3 has slightly better convergence results in comparison, which is expected as it uses the information from both neighboring points. Note also that the number of iterations required for convergence does not increase dramatically when we double the size of the problem. A smaller

N	1st. PIt.	2nd. PIt.	3rd. PIt.	4th. PIt.	5th. PIt.
2^{10}	20	16	9	4	2
2^{9}	19	15	9	4	2
2^{8}	18	15	8	3	2
2^{7}	17	14	9	3	2
2^{6}	16	13	8	3	2

Table 5.3: Number of inner multigrid iterations required for convergence at each outer Picard iteration of the 2-grid algorithm using option 3 interpolation operator P_3. The "PIt." means Picard iteration.

number of iterations are required at later Picard iterations, because a better initial guess is achieved for the inner two-grid solver of later Picard iterations.

To test the performance of the algorithm with P constructed from other points in the shaded triangle OCD in Figure 5.5, we did further numerical experiments by choosing points on the $\beta_j = \alpha_{j-1}$ line, which can be achieved by letting

$$\beta_j = \alpha_{j-1} = \frac{m u^h_{2j-1}}{u^h_{2j-2} + u^h_{2j}}, \tag{5.16}$$

where m determines how far-away the point is from the origin. We can see when m is between 0.8 and 1.2, the algorithm is generally convergent (when $m > 1.0$, the algorithm is convergent but not positivity preserving) as shown in Table 5.4. The best convergence is achieved when $m = 1$, which is at the middle point of line CD in Figure 5.5.

Moreover, if we change the position of points on the bounding line CD in Figure 5.5 by letting

$$\alpha_{j-1} = \frac{u_{2j-1}}{u_{2j-2} + q u_{2j}}, \quad \beta_j = \frac{q u_{2j-1}}{u_{2j-2} + q u_{2j}}, \tag{5.17}$$

and varying the value of q, the numerical results in Table 5.5 show that the convergence does not change dramatically for different q. The best result is still at the middle point.

From these results, we can see that the two-grid algorithm developed above is effective. However, naively applying the algorithm recursively to form a multilevel method does not give us good convergence properties. This can be understood by recognizing that the solution at every coarse level gives the error in the approximation on their finer levels. Take a problem with 17 grid points, 0, 1, ..., 16, for example. For the two-grid method, the coarse-grid solution gives the exact error in the approximate

m	1st. PIt.	2nd. PIt.	3rd. PIt.	4th. PIt.	5th. PIt.
0.80	100	100	100	52	61
0.825	44	37	64	29	100
0.85	69	46	96	14	42
0.86	100	25	18	22	27
0.87	24	67	13	20	43
0.88	23	34	20	11	21
0.89	48	38	27	23	13
0.90	22	28	11	23	15
0.95	23	16	10	6	7
1.00	20	16	9	4	2
1.05	21	22	12	12	8
1.08	55	27	12	12	9
1.10	100	59	52	63	100
1.12	23	24	25	44	13
1.15	23	19	81	12	34
1.20	100	59	52	63	100
1.30	100	100	53	100	100
1.40	100	100	100	100	100

Table 5.4: Number of inner multigrid iterations required for convergence at each outer Picard iteration of the 2-grid algorithm for $N = 2^{10}$ when modifying m, where m is used in Equation (5.16) for determining how far the point in away from the origin in Figure 5.5. The "PIt." means Picard iteration. A maximum of 100 inner iterations were allowed for each Picard iteration, so the results reporte as "100" indicate a failure to converge.

solution of the fine level at coarse-grid points 2, 4, .., 14, while the error at fine-grid points 1, 3, ..., 15 are only interpolated approximately. If we add a third level, then the second level only gives the exact error at grid points 4, 8, because the error at grid points 2, 6, 12 are interpolated from the third level. Therefore, if the interpolation operator does not reflect the error distribution well, the convergence performance will suffer. Moreover, the explicit parameterization of $N(RA)$ given here is only generally known for simple matrices, A, so the technique is not readily generalizable to more settings.

q	1st. PIt.	2nd. PIt.	3rd. PIt.	4th. PIt.	5th. PIt.
100	24	19	11	5	4
50	23	18	11	5	4
2	20	16	9	5	3
1	20	16	9	4	2
0.5	20	16	9	5	3
0.01	24	19	11	5	4
0.001	24	19	11	5	4

Table 5.5: Number of inner multigrid iterations required for convergence at each outer Picard iteration of the 2-grid algorithm for $N = 2^{10}$ when modifying q in Figure 5.5. The "PIt." means Picard iteration.

5.4 Numerical results using the unigrid method

Due to the reasons discussed above, we are not able to generalize the previous two-grid method to a multigrid method. To be able to implement a positivity-preserving multigrid method, we apply the unigrid Algorithms 7 and 8 and provide numerical results in this section.

As before, the stopping criterion for the inner loop of Algorithm 9 using the unigrid method as the solver is $||\mathbf{b} - A(\mathbf{u}^{(k)})\mathbf{u}^{(k+1)}||_2 < 10^{-8}$, and for the outer loop, the stopping criterion is $||\mathbf{b} - A(\mathbf{u}^{(k+1)})\mathbf{u}^{(k+1)}||_2 < 10^{-8}$. Two sweeps of relaxation are performed at each level of the unigrid solver. We first implement Algorithm 7 with uniform thresholding as the solver in the inner loop of Algorithm 9, and get the convergence performance shown in Table 5.6 for solving our model nonlinear diffusion equation. We can see that the number of iterations required for convergence does not increase dramatically when we double the size of the problem. A smaller number of iterations are required at later Picard iterations, because a better initial guess is available for the inner unigrid solver of the later Picard iterations. Table 5.7 gives the numerical results when using Algorithm 8 with local correction as the solver in the inner loop. We can see the iterations required for convergence are exactly the same as in Table 5.6.

To further test this algorithm, we provide more numerical results by starting the iteration with an all-ones initial guess. Tables 5.8 and 5.9 show the number of unigrid iterations required for convergence at each Picard iteration. We can see a similar convergence as seen in Tables 5.6 and 5.7 where a random initial guess was used.

Again, the local correction scheme gives exactly the same results as the uniform thresholding scheme.

To explain the similar behaviour of the two correction schemes, one reasonable conjecture is that the additional correction step we add into the unigrid method to ensure the solution positivity does not have a big influence on the unigrid method itself. To test this idea, we applied the standard V-cycle multigrid method with 2 pre- and post-relaxations using Gauss-Seidel method in the inner loop as the solver, and give the numerical results in Table 5.10. We can see that this gives exactly the same convergence seen in Table 5.6 which showed results from the modified unigrid method, Algorithm 7. This verifies our conjecture. Therefore, we can conclude that our algorithms are effective methods that achieve the goal of positivity preserving.

N	1st. PIt.	2nd. PIt.	3rd. PIt.	4th. PIt.	5th. PIt.	6th.PIt.
2^{10}	19	15	13	10	7	4
2^{9}	18	14	12	9	6	3
2^{8}	17	14	11	9	6	3
2^{7}	16	13	11	8	5	
2^{6}	15	12	10	7	4	

Table 5.6: Number of inner unigrid iterations required for convergence at each outer Picard iteration with random initial guess for different N. The "PIt." means Picard iteration. Algorithm 7 with uniform thresholding is applied.

N	1st. PIt.	2nd. PIt.	3rd. PIt.	4th. PIt.	5th. PIt.	6th.PIt.
2^{10}	19	15	13	10	7	4
2^{9}	18	14	12	9	6	3
2^{8}	17	14	11	9	6	3
2^{7}	16	13	11	8	5	
2^{6}	15	12	10	7	4	

Table 5.7: Number of inner unigrid iterations required for convergence at each outer Picard iteration with random initial guess for different N. The "PIt." means Picard iteration. Algorithm 8 with local correction is applied.

N	1st. PIt.	2nd. PIt.	3rd. PIt.	4th. PIt.	5th. PIt.	6th.PIt.
2^{10}	18	15	13	10	7	4
2^9	17	15	12	10	7	4
2^8	17	14	12	9	6	3
2^7	16	14	11	8	6	2
2^6	15	13	10	8	5	2

Table 5.8: Number of inner unigrid iterations required for convergence at each outer Picard iteration with all-ones initial guess for different N. The "PIt." means Picard iteration. Algorithm 7 with uniform thresholding is applied.

N	1st. PIt.	2nd. PIt.	3rd. PIt.	4th. PIt.	5th. PIt.	6th.PIt.
2^{10}	18	15	13	10	7	4
2^9	17	15	12	10	7	4
2^8	17	14	12	9	6	3
2^7	16	14	11	8	6	2
2^6	15	13	10	8	5	2

Table 5.9: Number of inner unigrid iterations required for convergence at each outer Picard iteration with all-ones initial guess for different N. The "PIt." means Picard iteration. Algorithm 8 with local correction is applied.

N	1st. PIt.	2nd. PIt.	3rd. PIt.	4th. PIt.	5th. PIt.	6th.PIt.
2^{10}	19	15	13	10	7	4
2^9	18	14	12	9	6	3
2^8	17	14	11	9	6	3
2^7	16	13	11	8	5	
2^6	15	12	10	7	4	

Table 5.10: Number of inner multigrid iterations required for convergence at each outer Picard iteration with random initial guess for different N. The "PIt." means Picard iteration. The standard V-cycle multigrid method using Algorithm 2 is applied as the solver in the inner loop.

Chapter 6

Application to Singularly Perturbed Problems

In this chapter, we study the solution of singularly perturbed problems that further motivates the importance of sign preservation in certain applications, and are a good fit for the algorithms developed in the last chapter to solve nonlinear problems. We show that a Picard iteration combined with the positivity-preserving multilevel methods developed in the last chapter provides an alternative approach for nonlinear problems, and is also a feasible way to solve for some solutions that are not easily computed using Newton's method.

6.1 Introduction to the problem

The second-order singularly perturbed reaction diffusion equation with boundary conditions is

$$\begin{aligned} &F_\varepsilon u(x) = -\varepsilon^2 u''(x) + b(x,u) = 0,\ x \in [0,1], \\ &u(0) = g_0,\ u(1) = g_1. \end{aligned} \tag{6.1}$$

where ε is a small positive parameter, $b \in C^\infty([0,1] \times \mathbb{R}^1)$, and g_0, g_1 are some given constants. The solutions to Equation (6.1) usually exhibit boundary and/or interior layers, which are narrow regions where the solutions change rapidly. The boundary layers are often caused by the contradiction between the boundary conditions imposed and the solution of the reduced problem

$$b(x,u) = 0,\ x \in [0,1], \tag{6.2}$$

which is obtained by setting $\varepsilon = 0$ in Equation (6.1). The solution of Equation (6.2) does not in general satisfy either of the boundary conditions in Equation (6.1). Under certain hypotheses [12] on the function b, the reduced problem in Equation (6.2) will have at least one solution. Following [12], we call the solution, $u_0(x) \in C^\infty$, a stable reduced solution of Equation (6.2) if there exists a constant γ such that

$$b_u(x,u_0) > \gamma^2 > 0,\ \forall x \in [0,1]. \tag{6.3}$$

Otherwise, it is called an unstable reduced solution. We will call the solution to Equation (6.1) near its stable reduced solution a stable solution, and an unstable solution otherwise, if it exists.

One common approach for solving these singularly perturbed problems is to apply iterative methods, such as Newton's method for nonlinear problems, and use the solutions to the reduced problem as the initial guesses. While this method is usually effective to solve for stable solutions to Equation (6.1), the existence and computation of the solutions near its unstable reduced solutions are often not clear.

In the following discussion, we present several numerical examples and show that Newton's method predominantly converges to only stable solutions. Therefore, it is preferable to use adaptively damped Picard iterations to directly control the behaviour of the approximate solution.

6.2 Discretization

Because the solution of the problem often has boundary and/or interior layers, we will perform the discretization of Equation (6.1) on a Shishkin mesh, which is a piece-wise equidistant mesh. Suppose the mesh is defined by

$$0 = x_0 < x_1 < \cdots < x_{N-1} < x_N = 1,$$

with $h_j = x_j - x_{j-1}$ for $j = 1, 2, ..., N$, and $\bar{h}_j = (h_j + h_{j+1})/2$ for $j = 1, 2, ..., N-1$. Then, the standard second-order finite-difference approximation of $u''(x_j)$ is

$$\begin{aligned} u''(x_j) &\approx \frac{u'(x_{j+1/2}) - u'(x_{j-1/2})}{\bar{h}_j} \approx \frac{1}{\bar{h}_j}\left(\frac{u_{j+1}-u_j}{h_{j+1}} - \frac{u_j - u_{j-1}}{h_j}\right) \\ &= \frac{1}{\bar{h}_j}\left[\frac{1}{h_j}u_{j-1} - (\frac{1}{h_j} + \frac{1}{h_{j+1}})u_j + \frac{1}{h_{j+1}}u_{j+1}\right], \end{aligned}$$

where $x_{j+1/2} = (x_j + x_{j+1})/2$. If we introduce the quantities

$$D^2 = -\begin{bmatrix} \frac{1}{h_1}+\frac{1}{h_2} & -\frac{1}{h_2} & & & \\ -\frac{1}{h_2} & \frac{1}{h_2}+\frac{1}{h_3} & -\frac{1}{h_3} & & \\ & \ddots & \ddots & \ddots & \\ & & -\frac{1}{h_{N-2}} & \frac{1}{h_{N-2}}+\frac{1}{h_{N-1}} & -\frac{1}{h_{N-1}} \\ & & & -\frac{1}{h_{N-1}} & \frac{1}{h_{N-1}}+\frac{1}{h_N} \end{bmatrix},$$

$$\bar{H} = \begin{bmatrix} \bar{h}_1 & & & \\ & \bar{h}_2 & & \\ & & \ddots & \\ & & & \bar{h}_{N-1} \end{bmatrix}, \ \mathbf{y} = \begin{bmatrix} b(x_1, u_1) \\ b(x_2, u_2) \\ \vdots \\ b(x_{N-1}, u_{N-1}) \end{bmatrix}, \ \mathbf{bc} = \varepsilon^2 \begin{bmatrix} g_0/h_1 \\ 0 \\ \vdots \\ g_1/h_N \end{bmatrix},$$

then, the discrete system corresponding to Equation (6.1) may be written (after rescaling by $\bar{H}$) as

$$-\varepsilon^2 D^2 \mathbf{u} + \bar{H}\mathbf{y} + \mathbf{bc} = -\varepsilon^2 D^2 \mathbf{u} + \mathbf{f} = 0. \tag{6.4}$$

6.3 First example problem

We first study a problem with two reduced solutions, one of which is a stable reduced solution, and the other one is an unstable reduced solution. Consider the following differential equation with boundary conditions

$$-\varepsilon^2 u'' - u^2 - u + 2 = 0, \ u(0) = u(1) = 0. \tag{6.5}$$

This problem has two (constant) reduced solutions, $u = -2$ and $u = 1$. Here, $b_u(x, u) = -2u - 1$. From Equation (6.3), we can see that $u = -2$ is a stable reduced solution, and $u = 1$ is an unstable reduced solution.

The solution of this problem will have boundary layers. We will use Shishkin meshes to better resolve these boundary layers. To construct the Shishkin mesh for the equation, set

$$\tau = \min\{1/4, (2.2\varepsilon/\beta)\ln N\}. \tag{6.6}$$

Then, divide the intervals $[0,\ \tau]$, $[\tau,\ 1\text{-}\tau]$ and $[1\text{-}\tau,\ 1]$ into $N/4$, $N/2$, and $N/4$ equidistant sub-intervals respectively. The mesh on $[0,\ \tau]$ and $[1\text{-}\tau,\ 1]$ is usually finer due to a small τ, so that the boundary layers can be properly resolved.

6.3.1 Numerical results with Newton's method

We show that Newton's method can converge to the stable solution, and behaves poorly for the unstable solution. Let

$$\mathbf{F}(\mathbf{u}) = -\varepsilon^2 D^2\mathbf{u} + \mathbf{f} = 0.$$

Here, the j-th entry of $\mathbf{f}$ is $f_j = \bar{h}_j(-u_j^2 - u_j + 2)$. The Jacobian matrix of this system is

$$\mathbf{F}'(\mathbf{u}) = -\varepsilon^2 D^2 + \text{diag}(\mathbf{f}),$$

where $\text{diag}(\mathbf{f})$ is a diagonal matrix with $[\text{diag}(\mathbf{f})]_{jj} = f_j$. Newton's method with an initial guess $\mathbf{u}^{(0)}$ computes the iterations

$$\mathbf{u}^{(k+1)} = \mathbf{u}^{(k)} + \mathbf{s}^{(k)},\ \text{where } \mathbf{F}'(\mathbf{u}^{(k)})\mathbf{s}^{(k)} = \mathbf{F}(\mathbf{u}^{(k)}),$$

until the convergence condition $||\mathbf{F}(\mathbf{u}^{(k)})||_\infty < 10^{-9}$ is satisfied.

By setting the initial guess of Newton's method to $\mathbf{u}^{(0)} = -2$, and choosing $N = 2^{12}$ so that the boundary layers are well resolved for the values of ε that we are testing, we can get the convergent solution of Equation (6.5) that is close to $\mathbf{u} = -2$ as shown in Figure 6.1. However, Newton's method fails to converge when we set the initial guess to $\mathbf{u}^{(0)} = 1$, as can be seem from Table 6.1.

ε	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}
$\|\|\mathbf{F}(\mathbf{u}^{(100)})\|\|_\infty$	1.49e4	2.9e13	1.2e9	2.33e12	6.79e10

Table 6.1: The ∞-norm of the residual with respect to ε after 100 Newton iterations.

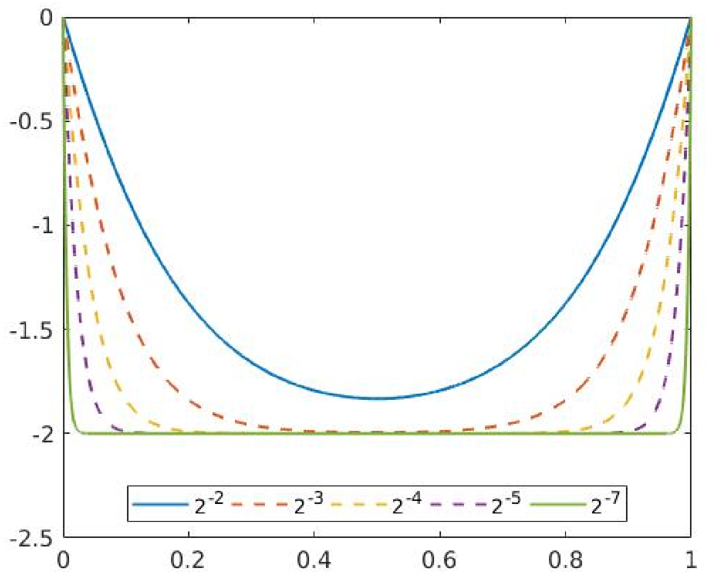


Figure 6.1: Convergent solution of Equation (6.5) on a mesh of size $N = 2^{12}$ computed with Newton's method by setting the initial guess to $\mathbf{u}^{(0)} = -2$. The legends $2^{-2}, 2^{-3}, ...,$ indicate the values of ε for each solution.

6.3.2 Numerical results with adaptively damped Picard iteration

In this section, we try to solve for the unstable solution (if it exists) using an adaptively damped Picard iteration. By writing

$$\frac{\mathbf{u}^{(k+1)} - \mathbf{u}^{(k)}}{\omega} - \varepsilon^2 \bar{H}^{-1} D^2 \mathbf{u}^{(k+1)} - \mathbf{u}^{(k+1)} = -2 + (\mathbf{u}^{(k)})^2,$$

and thinking of this as a time-stepping, where $0 < \omega < 1$, we get the iterative scheme

$$-\epsilon^2 D^2 \mathbf{u}^{(k+1)} + \left(\frac{1}{\omega} - 1\right) \bar{H} \mathbf{u}^{(k+1)} = \bar{H}\big(-2 + (\mathbf{u}^{(k)})^2 + \frac{1}{\omega}\mathbf{u}^{(k)}\big), \tag{6.7}$$

where $0 < \omega < 1$ is a damping parameter chosen adaptively according to the current approximation, and $(\mathbf{u}^{(k)})^2$ means element-wise square of $\mathbf{u}^{(k)}$. This gives the linear system to be solved at each Picard iteration as $A\mathbf{u}^{(k+1)} = \mathbf{b}$ with

$$A = -\epsilon^2 D^2 + \left(\frac{1}{\omega} - 1\right) \bar{H}, \ \mathbf{b} = \bar{H}\big(-2 + (\mathbf{u}^{(k)})^2 + \frac{1}{\omega}\mathbf{u}^{(k)}\big).$$

To obtain the stable solution, we only need to choose an initial guess that is close to the reduced stable solution $\mathbf{u} = -2$ with appropriate damping parameter such as $\omega = 0.01$, and perform the iteration until convergence.

To obtain the unstable solution, setting the initial guess close to $\mathbf{u} = 1$ does not work. Because we expect the solution to be positive, we try to restrict our approximation at every iteration step, requiring it to be always positive. The way we achieve this is to take advantage of the damping parameter ω in the iterative scheme. We want to choose ω such that system matrix A is positive definite and the right hand side $\mathbf{b}$ is positive. A is always positive definite for this iterative scheme because $1/\omega - 1 > 0$, which implies that the matrix A is an irreducibly strictly diagonal dominant Z-matrix. Therefore, the parameter ω is adaptively chosen according to the right-hand side $\mathbf{b}$. At iteration k, if $-2 + \mathbf{u^{(k)}}^2 > 0$, we pick a fixed value of ω such as $\omega = 0.01$ (in our implementation); otherwise, we pick

$$\omega = \gamma \min \left\{ \frac{\min\{\mathbf{u}^{(k)}\}}{2 - \min\{\mathbf{u}^{(k)}\}^2}, 1 \right\},$$

where $0 < \gamma < 1$ is a constant chosen appropriately ($\gamma = 0.999$ in the numerical results), guaranteeing that the right-hand side $\mathbf{b}$ is entry-wise positive. Then, by the properties of M-matrices, we can guarantee that the approximate solution at iteration $k+1$ is positive. This is also where the sign-preserving property of the linear solver is important in this application.

By setting the initial guess to $\mathbf{u} = 1$ and performing this algorithm, we get the solution after 100 iterations for different choices of ε as shown in Figure 6.2. Note, $N = 2^{15}$ is chosen to be large enough to resolve the boundary layers. To measure how accurate the solution is, we also record the norms of the residual

$$\mathbf{r}^{(k)} = -\varepsilon^2 D^2 \mathbf{u}^{(k)} - (\mathbf{u}^{(k)})^2 - \mathbf{u}^{(k)} + 2,$$

with different choices of N for the case when $\varepsilon = 2^{-12}$ as shown in Table 6.2. We see that the norms of residual decrease when N increases. From the pattern in the table, it is reasonable to argue that the norms of residual can reach a 10^{-8} tolerance when N is big enough. However, this residual does not represent well the actual error in the solution.

N	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}
$\|\|\mathbf{r}\|\|_2$	9.8e − 4	6.9e − 4	4.8e − 4	3.3e − 4	2.3e − 4
$\|\|\mathbf{r}\|\|_\infty$	5.4e − 5	2.7e − 5	1.3e − 5	6.6e − 6	3.3e − 6

Table 6.2: The 2-norm and ∞-norm of the residual after 100 Picard iterations with respect to N.

From Equation (6.5), we can see that

$$\varepsilon^2 u'' = -u^2 - u + 2.$$

Therefore, when $u < 1$, we have $u'' = -u^2 - u + 2 > 0$, which means the true solution u should be concave upward; When $u > 1$, we have $u'' = -u^2 - u + 2 < 0$, which means the true solution u should be concave downward. The only inflection point is at $u = 1$. The numerical results shown in Figure 6.2 do not agree with this expected behaviour of the solution.

Due to these observations, we are led to the belief that another solution of Equation (6.5) close to its reduced unstable solution does not exist.

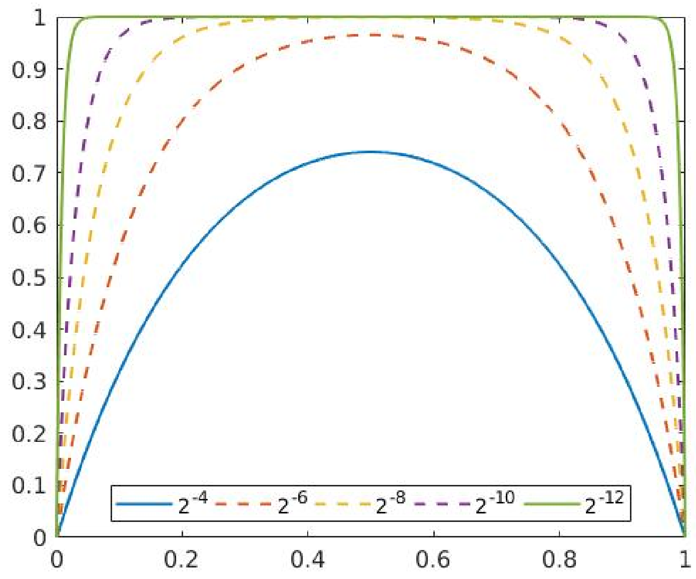


Figure 6.2: Computed solution of Equation (6.5) on a mesh of size $N = 2^{15}$ after 100 iterations using Picard iteration by setting the initial guess to $\mathbf{u}^{(0)} = 1$. The legends $2^{-4}, 2^{-6}, ...,$ indicate values of ε for each solution.

6.4 Herceg problem

The second problem we study is due to Herceg [9]. The differential equation and boundary conditions for this problem are

$$-\varepsilon^2 u'' + (u^2 + u - 0.75)(u^2 + u - 3.75) = 0, \quad u(0) = u(1) = 0. \tag{6.8}$$

This problem has 4 reduced solutions, $u = -2.5$, $u = -1.5$, $u = 0.5$ and $u = 1.5$, of which $u = -2.5$ and $u = 0.5$ are unstable, and $u = -1.5$ and $u = 1.5$ are stable reduced solutions. Six distinct solutions are found in [2] for this problem using a deflation technique. Of all the six solutions, there are three solutions that are either positive or negative in the whole domain $[0, 1]$. Using Newton's method, by setting the initial guess close to $u = -1.5$ and $u = 1.5$, we are able to find two distinct stable solutions. However, Newton's method cannot find solutions that are close to the unstable reduced solutions. We will compare the performance of Newton's method and the damped Picard iteration with sign-preserving inner loop, and show that the later algorithm is more robust to the initial guess than Newton's method.

The implementation of Newton's method is the same as in the previous example, so we do not repeat the discussion here. For the damped Picard iteration, we use the linearization

$$\frac{\mathbf{u}^{(k+1)} - \mathbf{u}^{(k)}}{\omega} - \varepsilon^2 \bar{H}^{-1} D^2 \mathbf{u}^{(k+1)} = -((\mathbf{u}^{(k)})^2 + \mathbf{u}^{(k)} - 0.75)((\mathbf{u}^{(k)})^2 + \mathbf{u}^{(k)} - 3.75).$$

This gives the linear system to be solved at each Picard iteration as $A\mathbf{u}^{(k+1)} = \mathbf{b}$ with

$$A = -\varepsilon^2 D^2 + \frac{1}{\omega}\bar{H}, \ \mathbf{b} = \bar{H}\left(\frac{1}{\omega}\mathbf{u}^{(k)} - ((\mathbf{u}^{(k)})^2 + \mathbf{u}^{(k)} - 0.75)((\mathbf{u}^{(k)})^2 + \mathbf{u}^{(k)} - 3.75)\right).$$

It is easy to see that the system matrix A is positive definite and an M-matrix. Therefore, when solving for the negative solution, we need to make sure that the right-hand side $\mathbf{b}$ is entry-wise negative; when solving for the positive solution, we need to make sure that the right-hand side $\mathbf{b}$ is entry-wise positive. These conditions are restricted through the damping parameter ω. In this way, we can guarantee that the approximate solution at every step is sign-preserving.

To be more concrete, when computing the positive solution close to $u = 1.5$, if

$$-((\mathbf{u}^{(k)})^2 + \mathbf{u}^{(k)} - 0.75)((\mathbf{u}^{(k)})^2 + \mathbf{u}^{(k)} - 3.75) \leq 0,$$

which means $\min\{\mathbf{u}^{(k)}\} \geq 0.5$ and $\max\{\mathbf{u}^{(k)}\} \leq 1.5$, then we pick a fixed value of ω such as $\omega = 0.01$ (in our implementation); otherwise, we pick

$$\omega = \gamma \min\left\{\min\left\{\frac{\mathbf{u_p}^{(k)}}{((\mathbf{u_p}^{(k)})^2 + \mathbf{u_p}^{(k)} - 0.75)((\mathbf{u_p}^{(k)})^2 + \mathbf{u_p}^{(k)} - 3.75)}\right\}, 1\right\},$$

where $\mathbf{u_p}^{(k)} = \mathbf{u}^{(k)}(\mathbf{u}^{(k)} > 1.5 \mid \mathbf{u}^{(k)} < 0.5)$ is a vector of the entries of $\mathbf{u}^{(k)}$ that are greater 1.5 or less than 0.5, and $0 < \gamma < 1$ is an appropriate constant ($\gamma = 0.999$ in the numerical results) and the products and quotients in the definition of ω are interpreted entrywise. When computing the negative solution close to $u = -1.5$, if

$$-((\mathbf{u}^{(k)})^2 + \mathbf{u}^{(k)} - 0.75)((\mathbf{u}^{(k)})^2 + \mathbf{u}^{(k)} - 3.75) \geq 0,$$

which means $\min\{\mathbf{u}^{(k)}\} \geq -1.5$ or $\max\{\mathbf{u}^{(k)}\} \leq -2.5$, then we pick a fixed value of ω such as $\omega = 0.01$ (in our implementation); otherwise, we pick

$$\omega = \gamma \min\left\{\min\left\{\frac{\mathbf{u_n}^{(k)}}{((\mathbf{u_n}^{(k)})^2 + \mathbf{u_n}^{(k)} - 0.75)((\mathbf{u_n}^{(k)})^2 + \mathbf{u_n}^{(k)} - 3.75)}\right\}, 1\right\},$$

where $\mathbf{u_n}^{(k)} = \mathbf{u}^{(k)}(-2.5 < \mathbf{u}^{(k)} < -1.5)$ is a vector of the entries of $\mathbf{u}^{(k)}$ that are between -1.5 and -2.5, and $0 < \gamma < 1$ is an appropriately chosen constant ($\gamma = 0.1$ in the numerical results) and the products and quotients in the definition of ω are interpreted entrywise.

The graphs of the convergent solutions close to $u = -1.5$ and $u = 1.5$ are plotted in Figures 6.3 and 6.4 for several choices of ε. Note that when $\varepsilon = 2^{-2}, 2^{-3}, 2^{-4}$, the parameter $\tau = 0.25$ in Equation (6.6), so the Shishikin mesh is reduced to a uniform mesh; when $\varepsilon = 2^{-5}, 2^{-7}$, we have $\tau < 0.25$, so the mesh is denser at the boundaries.

Although we set the initial approximations equal to the reduced solution in the computation of these two figures, this algorithm is robust to the initial guess. Numerical experiments for the case when $\varepsilon = 2^{-6}$ show that in the computation of the negative solution as shown in Figure 6.4, when the initial guess is chosen to be any negative constant vector greater than -2.5, i.e. $-2.5 < \mathbf{u}^{(0)} < 0$, this algorithm is convergent. We note Newton's method is convergent only for initial guesses between -0.9 and -1.9. In the computation of the positive solution as shown in Figure 6.3, when the initial guess is chosen to be any positive constant vector between 0.7 and 4, i.e. $0.7 \leq \mathbf{u}^{(0)} \leq 4.0$, this algorithm is convergent. Newton's method is convergent only when the initial guess is greater than 1. Also note that in order to achieve negativity-preservation when computing the negative solution, we need to add some minor changes to the Algorithms 7 and 8 that were developed for positivity-preserving algorithms. That is, we need to check negativity instead of positivity of corrected solutions, and correspondingly add restrictions to ensure the corrected solutions are negative.

In summary, we can see that this adaptively damped Picard iteration combined with sign-preserving linear-system solver can be considered as an alternative way to solve nonlinear systems, and can often be more robust to initial guesses compared to Newton's method. However, we are not able to get the fourth solution in [2] using this algorithm. The approximation always tends to converge to the stable solution shown in Figure 6.4 given a properly chosen negative initial guess.

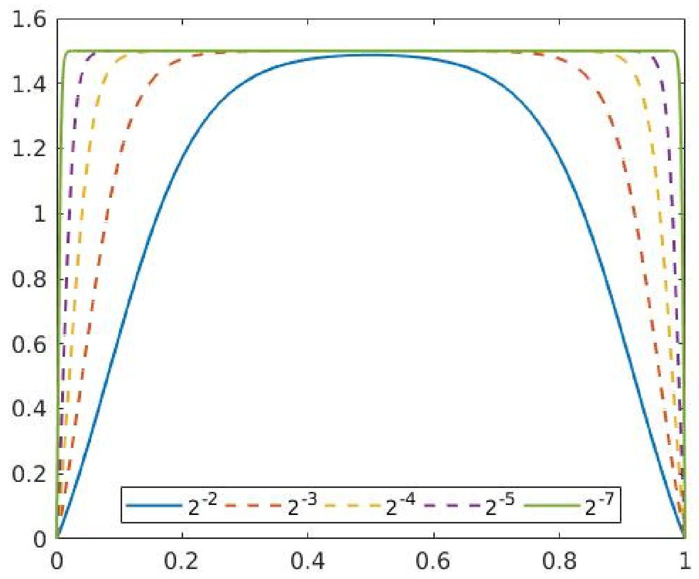


Figure 6.3: Convergent solution of Equation (6.8) on a mesh of size $N = 2^{10}$ computed with the adaptively damped Picard iteration by setting the initial guess to $\mathbf{u}^{(0)} = 1.5$. The legends $2^{-2}, 2^{-3}, \ldots$, indicate the values of ε for each solution.

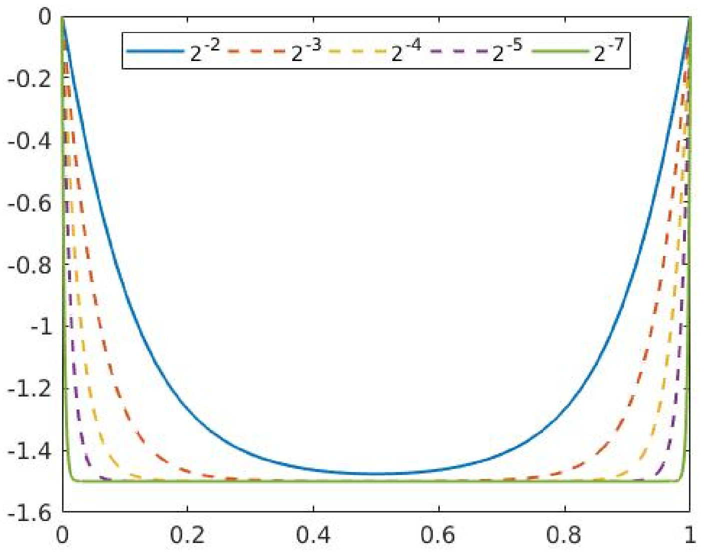


Figure 6.4: Convergent solution of Equation (6.8) on a mesh of size $N = 2^{10}$ computed with the adaptively damped Picard iteration by setting the initial guess to $\mathbf{u}^{(0)} = -1.5$. The legends $2^{-2}, 2^{-3}, ...,$ indicate the values of ε for each solution.

Chapter 7

Conclusions and Future Work

7.1 Conclusion

The positivity-preserving multigrid and multilevel methods developed in this thesis can be applied to certain types of singular and nonsingular linear systems effectively. Originally used in applications to Markov chains, the multiplicative-error multigrid method replaces the additive error correction in the standard multigrid methods with multiplicative-form error correction. We extended the application of this method to other general problems where a positivity-preserving property is required on the solution. We applied this method to solve 1D equidistributing meshes and developed a new multigrid algorithm that is monotonicity-preserving so that the generated meshes are not tangled, and also demonstrated the mesh size-independent convergence property applied to this problem. As we have seen, while rooted in aggregation methods for solving Markov chain problems, the choice of the pair-wise aggregation operator in the algorithm is reasonable in the sense that it also has a physical meaning for that particular problem. The system matrices are singular M-matrices at all levels in this application so the approximate solution is always positive given a positive initial guess. In general, *lumping* is required for the system matrices to remain singular M-matrices at coarse levels.

Directly applying this method to other problems requires a careful design of the interpolation operator. To solve nonsingular linear systems, the two new algorithms developed in Section 3.2, unigrid methods with uniform thresholding and local corrections, give similar convergence speed (as compared to Galerkin multigrid methods) while preserving solution positivity. The two algorithms show convergence properties comparable to the standard V-cycle multigrid method when applied to nonlinear diffusion equations. In the discussion of nonlinear diffusion equations, we also developed a new two-grid method that is efficient for this application.

The study of singularly perturbed differential equations introduced in Chapter 6 provides another application of these positivity-preserving unigrid methods, and further motivates the importance of sign preservation in certain problems. Combining these algorithms with Picard iterations provides more robust approaches to find some solutions of these nonlinear problems that Newton's method cannot easily find. By constraining the signs of the approximate solutions, we get better robustness to the initial guess than Newton's method. In addition, we have a hope of finding some unstable solutions, if they exist, to singularly perturbed problems, or provide evidence that they do not exist.

7.2 Future work

Future work from this thesis includes algorithmic improvements and application extensions. In terms of algorithms, this thesis considers the pair-wise aggregation operator in the multiplicative-error multigrid method, but only provides a brief discussion of the problems with other interpolation operators. This is also the reason we did not apply this method to nonsingular systems. Numerical experiments show that naively applying this method does not get us good convergence properties. The reason is that using the pair-wise aggregation operator as an interpolation operator loses its physical meaning here and hence does not work well. Better interpolation operators need to be developed using ideas from algebraic multigrid methods [4].

In the positivity-preserving unigrid methods we have developed, the order we perform relaxations is to start from the finest level to progress to the coarsest level. While this is already efficient, such a fixed order can be replaced by other strategies. For example, we can greedily choose the direction with the largest (or at least relatively large) residual norm for the next update step. This is also known as the Gauss-Southwell method. We can even choose the next iteration direction randomly, which can be shown to have similar estimates for the error reduction, as discussed in [16, 7]. Further investigation of the most efficient approach to unigrid in this setting could be interesting.

On the application side, mesh nontangling in 2D equidistributing problems is an important topic. We have only looked at the simple 1 dimensional case, since the condition of a nontangled mesh in 1D can be easily expressed in terms of monotonicity. However, it is not easy to come up with a similar quantity that can be used to decide the quality of a 2D mesh.

We also expect that our methods can be applied efficiently to 2D nonlinear diffusion equations because the system matrix in 2D is also an M-matrix. The 2D problem satisfies all the conditions we require for our methods to be applicable. A finite-element discretization would be more suitable for such problems on complex domains.

In addition, Algorithm 9 applies the Picard iteration straightforwardly. A better iterative scheme is to use the idea of nested iteration on a hierarchy of levels so as to reduce the total amount of work required for convergence. Similarly to the nested iteration scheme in a multigrid method, we can also combine Picard iterations with nested iteration and start the iteration from the coarsest level, which will provide a better initial guess for the Picard iteration on the finer levels. To be able to do this, a good interpolation operator needs to be developed so that we can use the approximate solution on the coarse levels to construct system matrices on finer levels. Our numerical experiments show that linear interpolation is not a good choice for this problem. Therefore, better interpolation such as quadratic or even cubic interpolations need to be developed and tested.

Bibliography

[1] A. Berman and R. J. Plemmons. *Nonnegative matrices in the mathematical sciences*, volume 9 of *Classics in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1994. Revised reprint of the 1979 original.

[2] A. Birkisson and P. E. Farrell. Computing distinct solutions of singularly perturbed problems via deflation. Technical report, Mathematical Institute, University of Oxford, Oxford, UK, 2015.

[3] W. L. Briggs, V. E. Henson, and S. F. McCormick. *A multigrid tutorial*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, second edition, 2000.

[4] H. De Sterck, T. A. Manteuffel, S. F. McCormick, K. Miller, J. Ruge, and G. Sanders. Algebraic multigrid for Markov chains. *SIAM J. Sci. Comput.*, 32(2):544–562, 2010.

[5] H. De Sterck, T. A. Manteuffel, S. F. McCormick, Q. Nguyen, and J. Ruge. Multilevel adaptive aggregation for Markov chains, with application to web ranking. *SIAM J. Sci. Comput.*, 30(5):2235–2262, 2008.

[6] H. De Sterck, K. Miller, E. Treister, and I. Yavneh. Fast multilevel methods for Markov chains. *Numer. Linear Algebra Appl.*, 18(6):961–980, 2011.

[7] M. Griebel and P. Oswald. Greedy and randomized versions of the multiplicative Schwarz method. *Linear Algebra Appl.*, 437(7):1596–1610, 2012.

[8] B. R. Haverkort et al. *Performance of computer communication systems: a model-based approach*. Wiley New York, 1998.

[9] D. Herceg. Uniform fourth order difference scheme for a singular perturbation problem. *Numer. Math.*, 56(7):675–693, 1990.

[10] W. Huang and R. D. Russell. *Adaptive moving mesh methods*, volume 174 of *Applied Mathematical Sciences*. Springer, New York, 2011.

[11] C. Isensee and G. Horton. A multi-level method for the steady state solution of Markov chains. In *SimVis*, pages 191–202, 2004.

[12] N. Kopteva and M. Stynes. Numerical analysis of a singularly perturbed nonlinear reaction-diffusion problem with multiple solutions. *Appl. Numer. Math.*, 51(2-3):273–288, 2004.

[13] U. R. Krieger. Numerical solution of large finite markov chains by algebraic multigrid techniques. In *Computations with Markov chains*, pages 403–424. Springer, 1995.

[14] U. R. Krieger, B. Muller-Clostermann, and M. Sczittnick. Modeling and analysis of communication systems based on computational methods for Markov chains. *IEEE Journal on selected areas in communications*, 8(9):1630–1648, 1990.

[15] H. P. Langtangen. *Solving nonlinear ODE and PDE problems.* Center for Biomedical Computing, Simula Research Laboratory and Department of Informatics, University of Oslo, 2016.

[16] T. A. Manteuffel, S. F. McCormick, O. Röhrle, and J. Ruge. Projection multilevel methods for quasilinear elliptic partial differential equations: numerical results. *SIAM J. Numer. Anal.*, 44(1):120–138, 2006.

[17] S. F. McCormick and J. W. Ruge. Unigrid for multigrid simulation. *Math. Comp.*, 41(163):43–62, 1983.

[18] Y. Saad. *Iterative methods for sparse linear systems.* Society for Industrial and Applied Mathematics, Philadelphia, PA, second edition, 2003.

[19] W. J. Stewart. *Introduction to the numerical solution of Markov chains.* Princeton University Press, 1994.

[20] Y. Takahashi. *A Lumping Method for Numerical Calculations of Stationary Distributions of Markov Chains.* Research Report B-18, Department of Information Sciences, Tokyo Institute of Technology, 1975.

[21] R. S. Varga. *Matrix iterative analysis*, volume 27 of *Springer Series in Computational Mathematics.* Springer-Verlag, Berlin, expanded edition, 2000.

[22] A. M. Winslow. Numerical solution of the quasilinear Poisson equation in a nonuniform triangle mesh. *Journal of computational physics*, 1(2):149–172, 1966.