COLLABORATIVE PDE SOLVERS: THEORY AND PRACTICE

PANAGIOTA TSOMPANOPOULOU

Ph.D. Thesis

UNIVERSITY OF CRETE, DEPARTMENT OF MATHEMATICS
HERAKLIO, GREECE

AUGUST 2000

To my family.

Acknowledgements

At this point I would like to thank all the people that helped and supported me during the years of my study.

First, I would like to thank my supervisor, Manolis Vavalis, for giving me the opportunity to work in such an exiting area, for his encouragement and for the knowledge he has taught me. His enthusiastic and tireless guidance, his confidence in me and his belief in my ability to complete my Ph.D. thesis along with our endless discussions and our friendship have helped keep my interest and motivation high throughout all the years of my study. Without his help it would be impossible to finish my thesis.

I thank the members of my defense committee, Professors Apostolos Hadjidimos, Elias Houstis, Charalambos Makridakis, Dimitrios Noutsos, Ioannis Papadakis and Alkis Tersenov for tolerating my late draft submissions without complaints, for their understanding and for all their questions and suggestions that helped me mold this dissertation. In particular, I would like to thank Professors Hadjidimos and Houstis for the many discussions we had and their enlightening collaboration during all my graduate years. Thanks also go to Professor Tersenov for the discussions on the theoretical PDE aspects of my thesis.

Also, I would like to specially thank Professor John R. Rice for his guidance and collaboration during all the years I stayed in Purdue. It is a great honor for me, that I have met and worked with him.

I thank all past and current members I met in the SoftLab research group of Computer Science Department, of Purdue University, for their help in my research. Specially, I would like to thank Ann Christine Catlin for her assistance and also for being such a good collaborator.

I would like to acknowledge all the members of Dep. of Mathematics of University of Crete, and specially Professor Suzana Papadopoulou and Professor Nikolaos Tzanakis for their patience, understanding and their help in all the problems I faced during the years of my studies and specially the last years that I was a "student in absentia". Also, I would like to thank Professor Georgios Akrivis and Professor Vassilios Dougalis for the knowledge they have taught me during the years of my undergraduate and graduate studies.

A great thank to the members of the recently founded Department of Applied Mathematics of University of Crete, and specially to Professor Ioannis Athanasopoulos, for the

software and hardware support in the last critical months prior to my defense.

My studies and research were balanced with enjoyable games, gatherings and discussions with many friends, who throughout the past years have helped and encouraged me in many ways. They include, Anna, Chara, Fotini, Giorgos, Katerina, Maria, Panagiotis, Sanghamitra and many more. A special thank to my friend Christina for her cheerful encouragement during these years, for her friendship and the lasting memories.

My educational achievements are mainly the result of the unbounded love, support, encouragement, advice and guidance I have received from my family. I am specially grateful to my mother, Georgia, for her encouraging letters to me while I was in Purdue and my father, Manolis, for being such a good and understanding father. A great thank to my sister Vana, for her listening to me and her love.

Finally, I would like to mention that my work was supported though various grants by the Department of Mathematics, University of Crete, by the Institute of Applied and Computational Mathematics, FORTH, and by the Computer Science Department, Purdue University.

Abstract

The simulation and modeling of complex physical systems often involves many components because

1. the physical system itself has components of differing natures,
2. parallel computing strategies require many (somewhat independent) components, and
3. existing simulation software applies only to simpler geometrical shapes and physical situations.

This dissertation focuses on elliptic partial differential equation (PDE) models as an instance of the approach we are proposing, and discusses how PDE agent based networks are applied to such multi-component applications. The network of PDE agents are used to

control the execution of existing solvers on sub-components,

mediate between sub-components, and

coordinate the execution of the ensemble.

Specifically a *Collaborative PDE solving system* is formulated (Chapter 2 and Appendix A), analyzed (Chapters 3 and 4), implemented (Chapter 5 and Appendix B) and evaluated (Chapter 6).

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 General

There are three computational approaches to simulating large scientific problems. The first and most common approach is to discretize the geometrical domain using grids or meshes to create a large discrete problem. These grids or meshes are then partitioned to create a set of inter-connected discrete problems. This is simple *domain decomposition* (also known as *sub-structuring*) [49] and the coupling between components (discrete problems) is rather tight as the mathematical model along interface points or elements is discretized into equations that involve details from both neighboring components. The second and oldest approach is Schwarz splitting [62] which decomposes the geometrical domain into components with a small overlap. The mathematical models on each component can then be solved independently in some way and the *Schwarz alternating method* is applied iteratively to compute the global solution. Of course, some discretization method is applied to the solution process on each individual component. The overlapping creates a serious complication in the Schwarz method [40] even when the global problem has a simple geometry. The method has become more feasible with the discovery of non-overlapping domain versions. The third and newest approach is *interface relaxation* [53] where the geometrical domain is decomposed into sub-domains, each with its own mathematical model. Along the interfaces between sub-domains one must satisfy interface conditions derived from the physical phenomena (e.g., continuity of mass or temperature, conservation of momentum). The models on each sub-domain are solved in the inner loop of the interface relaxation iteration method to compute the global solution. These methods use one of a variety of "smoothing" formulas to reduce the error in satisfying the interface conditions.

The goals of handling different physical models, using parallel computers and reusing existing software all lead to the need for high flexibility and loose coupling between components in the computation. These three approaches have similar goals but are quite different in their generality and flexibility. The tight coupling of domain decomposition requires that neighboring components have a lot of shared information about their discretizations. Further, this approach is quite awkward when the models are different on neighboring components. The *mortar method* creates specialized refinements of the models and meshes along the interfaces to accommodate changes in models across interfaces. Overlapping Schwarz methods are similarly constrained to a single physical model and also create a tight coupling between neighboring sub-domains. The non-overlapping Schwarz methods are restricted to a single mathematical model for neighboring sub-domains. The interface relaxation approach imposes no coupling conditions, except those inherent in the mathematical models, and it provides maximum generality and flexibility.

The power of decomposing a PDE problem into a collection of PDE subproblems was pointed out for the first time in [61]. It passed almost a century though before researchers in the Scientific Computing/Numerical Analysis area utilize this idea, first in [44]. Since then it has gained the interest of a still growing research community. The most appealing features of this approach are to be appreciated in terms of its elegant mathematical model as well as of its power as a new computational tool. The driving force of this new approach, which we name Collaborative PDE solvers, is the simulating of multi-component and multi-physics problems (see Section 1.2) where it applies in a natural way.

The main purpose of this thesis is to formulate, analyze, implement and evaluate the general framework of a Collaborative PDE solving system that enjoys several very desirable properties like fast convergence, wide applicability, increased adaptivity, high efficiency, inherent parallelism and software reuse by integrating advances in different scientific areas like mathematical analysis, numerical analysis, approximation, scientific computing, distributed computing and agent computing. A large part of this Thesis is devoted to the Interface Relaxation methods that consist the core of the whole system.

Although the Collaborative PDE solving framework can be applied with great success to any type of PDEs, we consider only the second order elliptic case. It is our believe that our research efforts presented here can be rather easily extended to, at least, certain parabolic PDEs (see section 7.3).

The basic idea that this Thesis builds on was proposed in a systematic way first by J.R. Rice [53] in 1987. Two of his students [43, 22] have designed and implemented primitive versions of our Collaborative environment. M. Mu also contributed to this idea [46, 45]. The first two researchers that proposed and analyzed particular Interface Relaxation methods were P.L. Lions [40] and A. Quarteroni [50, 49].

The rest of the Thesis is organized as follows. To complete the motivation of our Thesis we briefly describe in Section 1.2 few multi-component and multi-physics problems. In Section 1.3 we first describe the general PDE collaborative model we use throughout, and introduce some basic terminology and definitions. We then give in Chapter 2, a brief overview of a collection of interface relaxation mechanisms, which consist the core of the system. Chapters 3 and 4 are dedicated to the theoretical and experimental study of three such interface relaxers. In particular, we carry out their convergence analysis, and determine regions of convergence and "optimum" values for the relaxation parameters involved. We also validate our theoretical results with extensive numerical data. Next we present in Chapter 5 the implementation of the abstract computational model on a network of heterogeneous workstations using Agent software technology. We also briefly present certain computational issues that are crucial for the full understanding of our collaborative

methodology. Chapter 6 contains preliminary results from a long set of numerical experiments and focus on the practical implications of the Interface Relaxation schemes analyzed. These results also show that our theory can be of significant practical use for solving more general problems than those considered in theory. Most of the above Chapters end with our conclusions associated with the material each one of them contains. In Chapter 7, we derive our general conclusions, present several of our on-going research efforts and list some additional open problems.

The brief abstracts that are included in the beginning of each of the following Chapters and the three ppendices, contribute to the general view of our computational framework and analysis.

1.2 Examples of Composite PDE problems

Many of the real world problems are composite in the sense that they consist of components that have their own physical properties. Their coupling comes either due to the inherent continuity (or jump) conditions or due to additional imposed conditions that represent conservation or smoothness of other quantities. Below we briefly describe few of such problems.

We start with the simplest composite problem known as Window Josephson Junction [7] and depicted in Figure 1.1. It is a simple device of two different superconducting films (gray regions in the figure) separated by a thin oxide layer that is thin enough in the middle area to allow tunneling of electron pairs. Let Ω_{in} be a region of the Josephson junction window (associated with the middle area in the figure) which is imbedded in a global domain Ω, where $\Omega_{out} = \Omega \setminus \Omega_{in}$ is the idle region without superconductivity. The phase difference $u(x, y)$ of the order parameter in the superconducting films satisfies the semi-linear indefinite sine-Gordon equation inside Ω_{in} and is harmonic outside:

$$\nabla^2 u_{in} = sin u_{in} \quad \text{in} \quad \Omega_{in}, \tag{1.2.1}$$

$$\nabla^2 u_{out} = 0 \quad \text{in} \quad \Omega_{out}. \tag{1.2.2}$$

The local solutions are subject to the interface and boundary constrains:

$$u_{in} = u_{out} \quad on \quad \partial\Omega_{in} \tag{1.2.3}$$

$$\frac{1}{L_{in}} \frac{\partial u_{in}}{\partial n} = \frac{1}{L_{out}} \frac{\partial u_{out}}{\partial n} \quad on \quad \partial\Omega_{in} \tag{1.2.4}$$

$$\frac{\partial u_{out}}{\partial n} = g \quad on \quad \partial\Omega \tag{1.2.5}$$

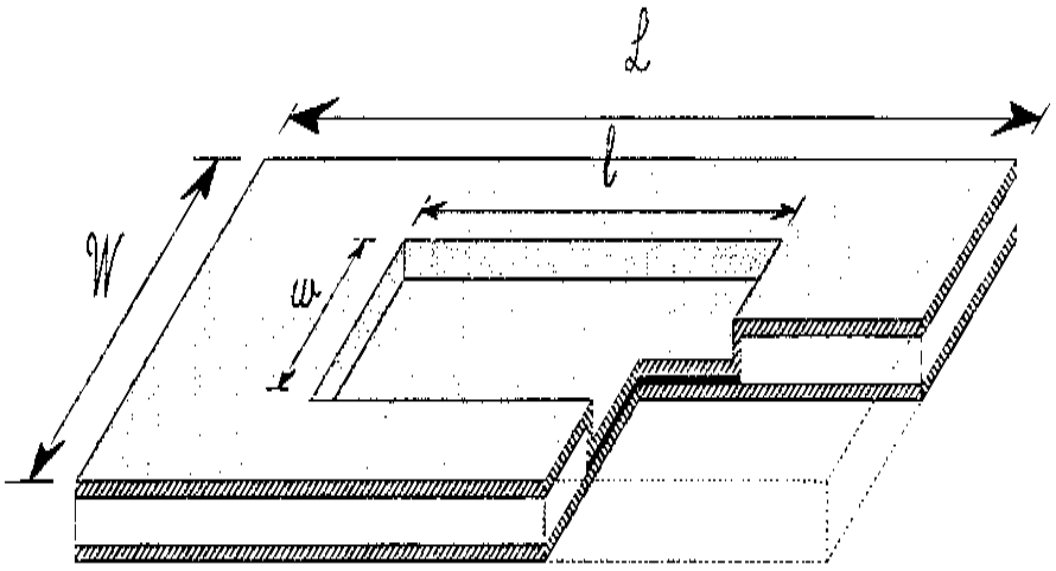


Figure 1.1: A Window Josephson Junction

where in general $L_{in} \neq L_{out}$, and Ω_{in} may consist of more than one disjoint subdomains if several junction windows are used.

A more complicated example is the one whose partition, along with the physical and mathematical description is presented in Figure 1.2 where a heat flow system consists of five parts with seven interfaces. The radiation conditions allow heat to leave on the left and the bottom while the temperatures U is zero on all the other boundaries. The mounting regions have heat dissipated. The interface conditions are continuity of temperature U and its derivative.

One might continue with many more examples of composite problems but we believe that it is worth to mention one more that is of particular interest and show the necessity of our proposed collaborative approach.

Aircraft design and simulations require coupling of several physical problems and components at several levels. Specifically, at the highest level one needs to couple aeroelastic, aerodynamic and aeroacoustic phenomena. Each one of these phenomena are obviously governed by different mathematical models (Navier–Stokes, Euler, ...) and each one of them is practically a synthesis of other models. It is worth to mention that experts in aeronautics have been trying to couple PDE problems for many years.

For example in the aeroacoustic case, a three-level zonal methodology is being developed. In this methodology a Computational Fluid Dynamics model (Navier Stokes with viscous and turbulence terms) is used for the near-field (the area around the airframe). Since in

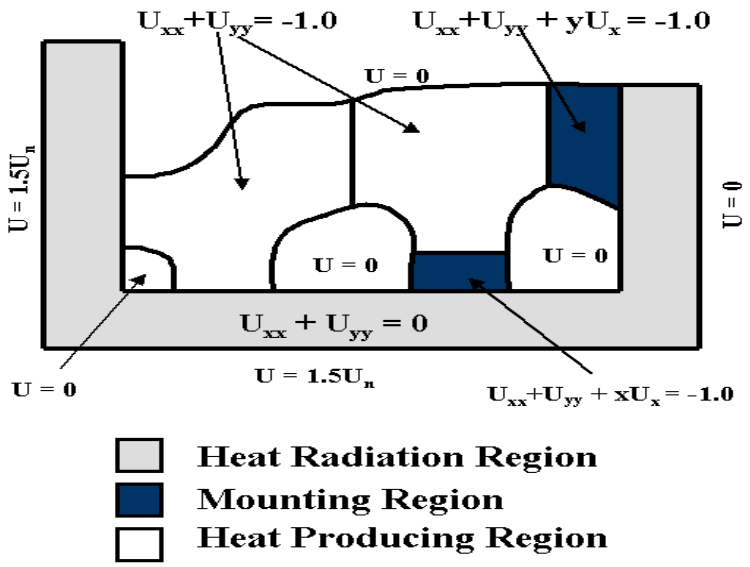


Figure 1.2: A heat flow problem

the intermediate region the speed of sound is not necessarily constant, and nonlinear effects can be ignored, a linearised Euler solver is used for the mid-field, and a Kirchhoff/FWH for the far-field [38]. Coupling methodologies of these models are being developed while the placement of the boundaries between the various methods is being investigated.

Besides the aeroacoustic case described above, we mention as another example the coupling of the structural dynamics solver and aerodynamics solver where people have realized that the naive coupling they have been using for years is inadequate from both the efficiency and accurate view points. So they have been very recently investigating new coupling methodologies known as "loose aeroelastic coupling" and "tight aeroelastic coupling" which are essential simplified Interface Relaxers [63].

To add to the whole picture, we mention the need of different type of grids, namely sliding grids (following the plane movements), rotating grids (for the rotors) and static grids (for the structural analysis of the fuselage).

1.3 The general definition of the collaborative elliptic PDE framework

The method assumes that one can solve exactly any single PDE on any simple domain or, more realistically, that given such a PDE problem, we can select a highly accurate solver for it from a library. The interface relaxation method uses a library of "single, simple-domain,

exact" PDE solvers to solve composite PDE problems. It is an iterative method of the classical type, based on relaxation, as follows:

1. *Guess solution values (and derivatives if needed) on all sub-domain interfaces.*
2. *Solve all single PDEs exactly and independently on all the sub-domains with these values as boundary conditions.*
3. *Compare and improve the values on all interfaces using a relaxer (discussed below).*
4. *Return to Step 2 until satisfactory accuracy is achieved.*

The simplest relaxers do some sort of "smoothing" of values on the interfaces and averaging is a good mental model for a relaxation formula.

The attraction of interface relaxation is threefold. First, it allows the accurate coupling of independent models and the reuse of PDE software that handles single-phenomenon models. Second, it uncouples the parallelism of the computation somewhat from that of the machines used. Finally, it is intuitively consistent with a person's view of the geometry and physical models of a composite PDE problem.

Interface relaxation is an iteration scheme defined at the continuum (mathematical) level; its convergence properties are a question of mathematical analysis, not of numerical analysis. The convergence analysis of interface relaxation presents formidable mathematical challenges; almost any question asked will be both hard and open. Even for the single-PDE case – one global PDE or domain decomposition – work on convergence analysis has appeared rarely, starting about 10 years ago [40] and then more recently in 1999 [49].

Given that theoretical analysis is intractable for the moment, we use experiments to provide guidance and insight for interface relaxation. Numerous experiments done in recent years indicate that interface relaxation converges for a wide variety of problems and relaxers. The convergence is sometimes very fast, other times not. The results in [45] (and in the references therein) show (for a single problem) that the rate of convergence is relatively independent of the number of sub-domains and this was verified by experiments using up to 500 sub-domains. There is reason to be hopeful that, as we better understand interface relaxation, it can become a very useful method for solving composite PDEs. A crude form of interface relaxation already in fairly widespread use simply involves "trading" current values across interfaces without any relaxation. This method makes the most sense in time-varying problems, but we are not aware of any attempts to analyze the effects of the errors involved.

With the above picture in mind the general mathematical description of the Interface Relaxation methods for second order elliptic PDE problems in $\mathbb{R}^2$ can be viewed in the following way. Denote the local PDEs by

$$L_i u_i = f_i \quad in \quad \Omega_i \quad for \quad i = 1, \ldots, p, \tag{1.3.1}$$

and assume that Ω_i do not overlap and that the interface conditions are given in the implicit form

$$G_{i,j}\left(u_i, \frac{\partial u_i}{\partial \eta_{i,j}}; u_j, \frac{\partial u_j}{\partial \eta_{j,i}}; J_1, J_2\right) = 0 \quad on \quad \Gamma_{i,j} \equiv \Omega_i \bigcap \Omega_j, \tag{1.3.2}$$

where η denotes the normal direction and J_1, J_2 the jump quantities associated with u or its derivative. We assume that $G_{i,j}$ can be a function mapping on the interface or even a functional. We also assume certain boundary conditions (not shown here) and the existence of the solution of the PDE problems.

Note that the fact that the Interface Relaxation involves coupled sequences of continuous approximations as well as discrete approximations leads to error analysis with unique characteristics. The approximation analysis of the Interface Relaxation methods, defined by the above two equations, essentially consist of two steps. First, we define a sequence of iterants $u_i^{(k)}$ for each subdomain on the PDE level. Then, using a standard discretization method, we solve each PDE problem locally to obtain the corresponding discrete solution $U_{i,h_i}^{(k)}$. The convergence of $u_i^{(k)}$ to $u|_{\Omega_i} \equiv u_i$ as k tends to infinity, is proved for a class of relaxers in [40] for a weak norm. There exist also the well known estimates for the discretization error, like

$$||U_{i,h_i}^{(k)} - u_i^{(k)}|| \leq C(u_i^{(k)}) h_i^{\alpha} \tag{1.3.3}$$

where the constant $C(u_i^{(k)})$ depends only on the smoothness of $u_i^{(k)}$. These two imply that

$$\lim_{k\to\infty} \lim_{h_i \to 0} U_{i,h_i}^{(k)} = u_i. \tag{1.3.4}$$

Since, solution from the interface relaxation scheme is computed on a finite grid, we have to study carefully the convergence of the following quantity: $\lim_{h_i\to 0} \lim_{k\to\infty} U_{i,h_i}^{(k)}$.

In the study of the discrete iteration procedure, there exist convergence results as the $\lim_{k\to\infty} ||U_{i,h_i}^{(k)} - U_{i,h_i}^{\star}|| = 0$, where $U_{i,h_i}^{\star}$ denotes the limit of iterative relaxation procedure using the corresponding discretization.

The above results, are necessary but not sufficient to prove convergence over both k and h_i at any order. The equation

$$\lim_{h_i\to 0} \lim_{k\to\infty} U_{i,h_i}^{(k)} = \lim_{k\to\infty} \lim_{h_i\to 0} U_{i,h_i}^{(k)} = u_i \tag{1.3.5}$$

holds only if we establish/prove strong and uniform convergence. Alternatively, one might examine the term $U^{\star}_{i,h_i} - u_i$. Such an error analysis for a model problem and a class of interface relaxers is given in [45].

Chapter 2

Overview of Interface Relaxation Methods

Abstract

A population of seven non-overlapping domain decomposition methods for solving elliptic differential equations are viewed and formulated as iterated interface relaxation procedures. A comprehensive review of the underlying mathematical ideas and the computational characteristics is given. The existing theoretical results are also reviewed and high level descriptions of the various algorithms are presented. The effectiveness of these methods on various differential problems is investigated by presenting and discussing preliminary performance evaluation data.

2.1 Introduction

The various domain decomposition methods that have been recently developed for the efficient solution of elliptic differential equations can be easily classified into two categories –overlapping and non–overlapping. Both approaches already have been used to effectively model large scale, industrial, ill-conditioned problems. Nevertheless it is believed that further theoretical and experimental analysis is required before such methods will become practical and useful tools for non-experts.

Overlapping (Schwartz) schemes have received in the past a great deal of attention. Articles that review and compare various such schemes [33] and survey the associated preconditioning strategies [13, 9] have already appeared in the literature. It is relatively recent that a number of studies have shown that non–overlapping schemes can compete well and can possibly free the user from certain complications in their formulation and implementation. The comparison of the main characteristics of these two classes of methods and the existence of equivalence relations between them have already received a great deal of study [8, 3, 11].

Interface relaxation methods are taking us a step beyond non–overlapping domain decomposition [53]. In an effort to mimic the physics in the real world, they split a complicated partial differential equation (PDE) that acts on a large and/or complex domain into a set of PDE problems with different but simple, operators acting on different smaller and "easy" subdomains. This Multi–PDE, Multi–domain system is properly coupled using smoothing operators on the inter–domain boundaries. The present work reviews and evaluates a class of interface relaxation methods for solving elliptic PDE problems. Although these methods can be considered from the preconditioning viewpoint, here we follow Southwell's relaxation [64, 65] of the 1930's – but at the PDE level instead of the linear algebra level – to formulate them as iterated interface smoothing procedures. We believe that such a formalism has certain theoretical and algorithmic advantages.

From the interface relaxation viewpoint these methods consist of partitioning the domain on a set of non–overlapping subdomains and of imposing some boundary conditions on the interface boundaries defined by this partition. Then, using initial guesses on the interfaces, the set of the resulting PDE problems is solved. The solutions obtained do not satisfy the interface boundary conditions and interface relaxation is applied to obtain new interface boundary values, which satisfy the conditions better, and we solve the PDEs with these new values. We repeat the above steps until convergence.

For our study we have collected most of the known interface relaxation methods and proposed two new ones. Specifically, we consider the methods listed below in alphabetical

order with respect to their acronyms. These acronyms are used in the sequel to refer to associated methods.

AVE A simple method of averaging the solution and its normal derivative along the interfaces.

GEO A method based on a simple geometric contraction.

NEW A scheme based on Newton's method to "correct" the interface values.

ROB An algorithm that uses Robin interface conditions for smoothing.

SCO A scheme that is based (but not formulated) on a Schur complement approach.

SHO A method based on the concept of the shooting method for solving Ordinary Differential Equations (ODEs).

SPO A method originated from the use of Steklov–Poincaré operator which involves alternating boundary condition types.

To the best of our knowledge **GEO** and **NEW** has not been considered in any previous studies. The analysis of the **GEO** method is presented in Chapter 4, while the analysis of the **NEW** method is beyond of the scope of this thesis. We should point out that in order to preserve some uniformity in our study we have not experimented with a class of interesting interdomain smoothing methods which use a few modes of the expansion [12] of certain interface operators (i.e., Lagrange multipliers [20, 21] or Steklov–Poincaré operators [47, 48]). We will only briefly describe these techniques.

The rest of this Chapter is organized as follows. In Section 2, we present the general framework for decomposing a multi–PDE problem into a collaborative pool of single–PDE problems and discuss the implications on simulating complicated physical problems. The interface relaxation methods we consider for this study are presented in Section 3, where we give their formulation and list the known theoretical results. In Section 4, we present our performance data and in Section 5 we summarize the contributions of our study.

2.2 Domain decomposition with iterated interface relaxation

Currently the domain decomposition world consists of two parts –overlapping and non–overlapping– both living in prosperity. Overlapping, known also as Schwartz, methods were the first considered and have already proved themselves as very efficient numerical procedures enjoying certain very desirable convergence properties. Nevertheless, it has been also

observed that they might have several serious drawbacks which will prohibit their use for certain applications. For example, almost all of the many proposed domain decomposition methods for solving wave propagation models (that consist of the Helmholtz equation coupled with various absorbing or reflecting boundary conditions) are non–overlapping and of interface relaxation type [1, 18, 34, 55].

Non–overlapping methods exhibit certain advantages compared to overlapping ones. Specifically:

- They are not sensitive to jumps on the operator coefficients. Their convergence behavior and theoretical error estimates remain the same even if the differential operator includes discontinuous coefficients provided that the jumps occur along the interface lines [76].

- They have smaller communication overhead in a parallel implementation on distributed memory multiprocessor systems. Their communication overhead is proportional to the length of the interface lines while it is proportional to the overlapping area in the case of overlapping methods [22].

- The bookkeeping is rather easy for the decomposition and manipulations of the associated data structures compared to the more complicated and costly bookkeeping of the overlapping methods [22].

There are two principal viewpoints of non-overlapping methods, preconditioning and interface relaxation. For an in depth and up-to-date survey of non–overlapping domain decomposition methods considered and analyzed from the preconditioning viewpoint the reader is referred to [74] and for a general formulation and analysis of interface relaxation methods to [45]. We give a brief presentation of the interface relaxation method philosophy and practice, in order to identify its main characteristics.

Interface relaxation is a step beyond non–overlapping domain decomposition; it follows Southwell's relaxation of the 1930's – but at the PDE instead of the linear algebra level – to formulate relaxation as iterated interface smoothing procedures. A complex physical phenomenon consists of a collection of simple parts with each one of them obeying a single physical law locally and adjusting its interface conditions with neighbors. Interface relaxation partitions the domain on a set of non–overlapping subdomains, imposes some boundary conditions on the interface among subdomains lines. Given an initial guess, it imitates the physics of the real world by solving the local problems exactly on each subdomain and relaxing boundary values to get better estimates of correct interface conditions. This is illustrated in Figure 2.1 where the generic relaxation formula $G_{i,j}$ (based on the

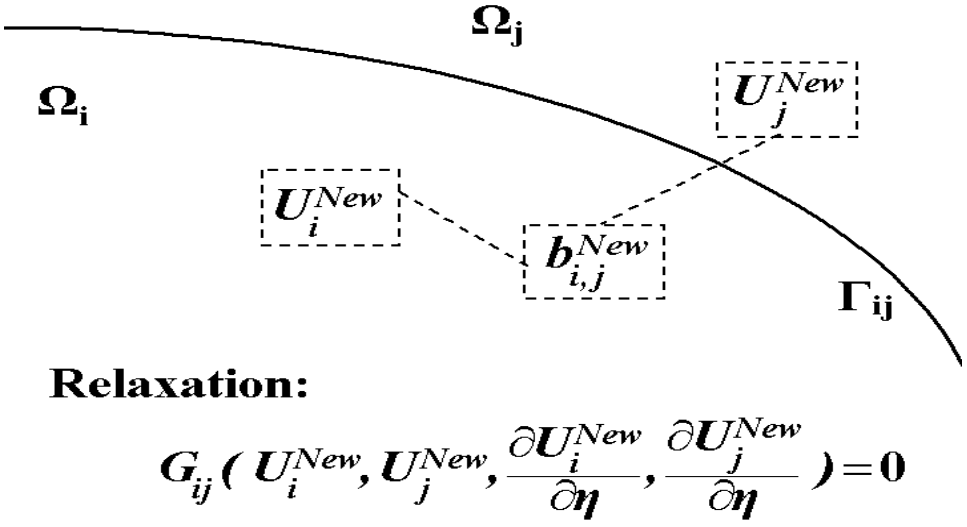


Figure 2.1: The interface relaxation mechanism

current solutions U_i^{New} and U_j^{New} of the two local to the neighboring subdomains Ω_i and Ω_j) calculates successive approximations $b_{i,j}^{New}$ to the solution on the interface $\Gamma_{i,j}$ between them.

To formally describe this method we consider the differential problem

$$Du = f \ in \ \Omega, \qquad Bu = c \ on \ \partial\Omega \tag{2.2.1}$$

where D is an elliptic, non–linear in general, differential operator and B a condition operator defined on the boundary $\partial\Omega$ of an open domain $\Omega \in R^d, d = 1, 2, \ldots$. This domain is partitioned into p open subdomains $\Omega_i, i = 1, \ldots, p$ such that $\Omega = \cup_{i=1}^{p} \overline{\Omega_i} \backslash \partial\Omega$ and $\cap_{i=1}^{p} \Omega_i = \emptyset$. For reasons related either to the physical characteristics of this problem or to the computing resources available, one would like to replace (2.2.1) with the following system of loosely coupled differential problems

$$\begin{gathered} D_i u = f_i \ in \ \Omega_i, \\ G_{ij} u = 0 \ on \ (\partial\Omega_i \cap \partial\Omega_j) \backslash \partial\Omega \quad \forall j \neq i, \qquad B_i u = c_i \ on \ \partial\Omega_i \cap \partial\Omega \end{gathered} \tag{2.2.2}$$

where $i = 1, \ldots, p$. These differential problems are coupled through the interface conditions $G_{ij}u = 0$ and involve the restrictions D_i and B_i of the global differential and boundary operators, D and B, respectively, on each subdomain with some of them possibly linear and some others nonlinear. The functions f_i and c_i are similar restrictions of functions f and c. The local interface operator G_{ij} is associated with the interface relaxation method

and different selections for the G_{ij}'s lead to different relaxation schemes. In this study we consider several interface relaxation methods that have the following characteristics:

- They first decompose the problem (2.2.1) at differential level and then discretize the resulting differential subproblems (2.2.2).
- They have the versatility to use the most appropriate discretization scheme for each subproblem.
- They do not overlap the subdomains Ω_i.
- Using good relaxation parameters in G_i, they are fast enough so no preconditioning is needed.
- They simplify the geometry and physics of the computation by considering the subproblems (2.2.2) instead of the global differential problem (2.2.1).
- They can utilize software parts technology by reusing existing "legacy" software parts for solving the individual subproblems (2.2.2).
- They are general and robust.

There are several challenging questions concerning practical applications of such methods (e.g. find the most suitable relaxer for a particular problem of application, determine what is the domain of applicability of each one of them, explain the interaction between the mathematical iteration and the numerical solving method, select "good" or "optimal" values for the relaxation parameters involved, ...). It is worth to point out that since all the methods decompose and relax interface values at continuum level the convergence analysis of these methods need to be carried out at PDE (continuum) level and therefore is a mathematical analysis and not a numerical analysis problem (see [45] for a discussion).

2.3 Interface relaxation methods

Due to the inherent abstraction, it is relatively easy to describe the various interface smoothing methods at both the conceptual and algorithmic level. Next we present the seven methods, give their high level algorithmic description and briefly present the known theoretical results. Detailed algorithms to define all schemes are given in the Appendix. For simplicity in the presentation of algorithms, we consider only one–way (along the x–axis) partition of the domain. Therefore each subdomain has two interface lines with the two neighboring subdomains. The basic building block for our algorithms is the procedure `u`

`= solve_pde(ui,dui)` which calculates the solution `u` of the local to a subdomain PDE problem with Dirichlet, Neumann or Robin boundary conditions on the interface using as the interface values `ui` and its gradient `dui`. The subscripts R and L denote left and right subdomains or interfaces respectively and u_i denotes the solution of the problem associated with subdomain Ω_i.

The Dirichlet/Neumann Averaging (AVE) Method

We start by presenting one of the simplest schemes which consists of two PDE solving sweeps coupled with two smoothing interface relaxation steps. In the first sweep, the Dirichlet problem is solved on all subdomains. Then the relaxation procedure smoothes the derivatives along all interfaces by estimating the normal derivative as a convex combination of the previously computed normal derivatives of the two adjacent subdomains. These estimates are then used as boundary conditions in the second PDE solving sweep where the Neumann problem is solved on all subdomains. The second relaxation step follows and computes estimates of the unknown function on the interfaces taking a convex combination of the previously computed solutions on the adjacent subdomains. These estimates are to be passed to the next iteration's Dirichlet sweep. This method, which we classify as a *two–step* method, can be algorithmically described by

---

- for k = 0,1,2, ...
  - $u^{(k+\frac{1}{2})}$ = solve_pde(ui) in each subdomain
  - $dui = \beta \frac{\partial u_R^{(k+\frac{1}{2})}}{\partial x} + (1-\beta)\frac{\partial u_L^{(k+\frac{1}{2})}}{\partial x}$ on each interface
  - $u^{(k+1)}$ = solve_pde(dui) in each subdomain
  - $ui = \alpha u_R^{(k+1)} + (1-\alpha) u_L^{(k+1)}$ on each interface.

---

where $\alpha, \beta \in (0,1)$ are relaxation parameters. There have been a few theoretical studies on the convergence of the above scheme which are discussed in [46]. In particular in [75] a convergence analysis of the method is carried out at a differential level using Hilbert space techniques. In [77] the Galerkin finite element method and the hybrid mixed finite element method are employed to give discrete versions of this method. Fourier analysis is used in [60] to obtain sharp convergence results and to estimate optimum values for the relaxation parameters involved for simple model problems.

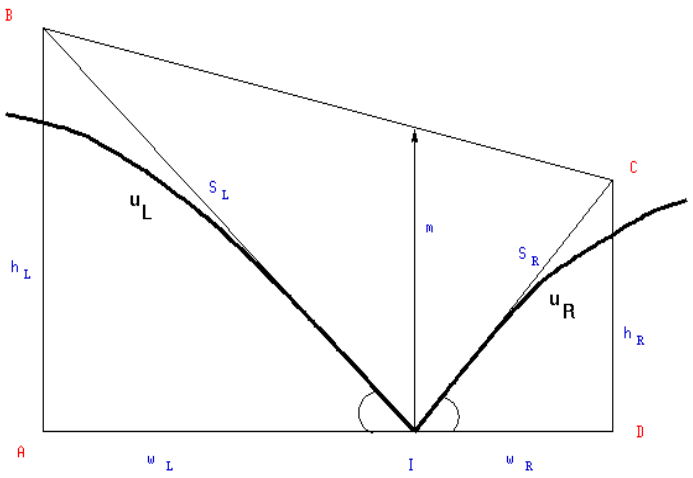


Figure 2.2: Cross section perpendicular to the interface where u_L and u_R have slopes S_L and S_R at the interface point I. Changing the values of u_L and u_R by a quantity m makes these slopes equal in magnitude.

The Geometric (GEO) Contraction Based Method

GEO estimates the new solution for each subdomain by solving a Dirichlet problem and is classified as an *one–step* method. The values on the interfaces are obtained by adding to the old ones, a geometrically weighted combination of the normal boundary derivatives of the adjacent subdomains. Specifically, we assume in Figure 2.2 that u_L and u_R are the solutions of the PDE problems associated with the left and right subdomains, respectively, of the interface point I. They are equal along I and we denote by S_L and S_R their slopes at I. As it can be easily seen geometrically, m is the correction needed to be added to u_L and u_R so as to match the normal derivatives at I. To calculate m we consider the two right triangles IAB and CDI whose heights are given multiplying the corresponding tangent with the base of the triangle, or equivalently multiplying the normal derivative with the base. The bases w_L and w_R are the widths assumed for the validity of the slope values; these can be arbitrarily selected and play the role of the relaxation parameters. The new interface values are now given by adding the weighted average of the heights to the old interface values u_L and u_R. One can intuitively view this as grabbing the function u at I and stretching it up by m until its derivative becomes continuous. Numerical experiments show that the convergence rate does not seem to depend much on the widths w_L and w_R. In case that $u_R \neq u_L$ on I we simply use their average. **GEO** is given algorithmically by

---

- for k = 0,1,2, ...
  - $ui^{(k+1)} = \frac{u_L^{(k)}+u_R^{(k)}}{2} - \frac{w_L w_R}{w_L+w_R}(\frac{\partial u_L^{(k)}}{\partial x} - \frac{\partial u_R^{(k)}}{\partial x})$ on each interface
  - $u^{(k+1)} = \text{solve\_pde}(ui^{(k+1)})$ in each subdomain

---

To the best of our knowledge this method has not been considered in any previous studies.

The Newton's (NEW) Method

Another new idea is to use discrete Newton's method to update the values at the interface according to the following procedure,

Step 0: For $i = 1, 2$, guess $u_R^{(i)}$, $u_L^{(i)}$ on interfaces and compute $\frac{\partial u_R^{(i)}}{\partial x}$, $\frac{\partial u_L^{(i)}}{\partial x}$.

Step 1: Consider finding for δ_L and δ_R so that on all interfaces we have

$$\begin{aligned}(u_L^{(2)} + \delta_L) - (u_R^{(2)} + \delta_R) &= 0\\ \frac{\partial u_L^{(2)}}{\partial x}(u_L^{(2)} + \delta_L) - \frac{\partial u_R^{(2)}}{\partial x}(u_R^{(2)} + \delta_R) &= 0\end{aligned}$$

Step 2: Apply linearization to solve the equations approximately

$$\begin{aligned}(u_L^{(2)} + \delta_L) - (u_R^{(2)} + \delta_R) &= 0\\ \frac{\partial u_L^{(2)}}{\partial x}(u_L^{(2)}) \left[1 + \frac{\partial}{\partial u_L}\left(\frac{\partial u_L}{\partial x}\right)\delta_L\right] - \frac{\partial u_R^{(2)}}{\partial x}(u_R^{(2)}) \left[1 + \frac{\partial}{\partial u_R}\left(\frac{\partial u_R}{\partial x}\right)\delta_R\right] &= 0\end{aligned}$$

Step 3: Approximate the unknown derivatives by differences

$$\begin{aligned}\frac{\partial}{\partial u_L}\left(\frac{\partial u_L}{\partial x}\right) &= \frac{\frac{\partial u_L^{(2)}}{\partial x} - \frac{\partial u_L^{(1)}}{\partial x}}{u_L^{(2)} - u_L^{(1)}} = A_L\\ \frac{\partial}{\partial u_R}\left(\frac{\partial u_R}{\partial x}\right) &= \frac{\frac{\partial u_R^{(2)}}{\partial x} - \frac{\partial u_R^{(1)}}{\partial x}}{u_R^{(2)} - u_R^{(1)}} = A_R\end{aligned}$$

Step 4: Solve for δ_R and δ_L

$$\begin{aligned}\delta_L - \delta_R = u_R^{(2)} - u_L^{(2)} &= 0\\ A_L\delta_L - A_R\delta_R = \frac{\partial u_R^{(2)}}{\partial x} - \frac{\partial u_L^{(2)}}{\partial x}&.\end{aligned}$$

so the $\delta_L = \delta_R = \delta = [\frac{\partial u_R^{(2)}}{\partial x} - \frac{\partial u_L^{(2)}}{\partial x}]/(A_L - A_R)$. The values of $\frac{\partial u_L}{\partial x}$ and $\frac{\partial u_R}{\partial x}$ depend on u_L and u_R throughout the differential equations given above. In the spirit of the above procedure, **NEW** can be described as follows

---

- for k = 2,3,4 ...
  - Solve for corrections $\delta_L = \delta_R = \delta$ from the interface system as above:
    - $\left(u_L^{(k)} + \delta_L\right) - \left(u_R^{(k)} + \delta_R\right) = 0$
    - $\frac{\partial u_L^{(k)}}{\partial x}\left(u_L^{(k)} + \delta_L\right) - \frac{\partial u_R^{(k)}}{\partial x}\left(u_R^{(k)} + \delta_R\right) = 0$
  - $ui^{(k+1)} = ui^{(k)} + \delta$ on each interface
  - $u^{(k+1)} = \text{solve\_pde}(ui^{(k+1)})$ in each subdomain

---

There is no general convergence analysis for this new single step scheme which does not involve any relaxation parameters. Like most applications of Newton's method, it should converge very rapidly in some neighborhood of the true solution.

The Robin Relaxation (ROB) Method

An even simpler interface relaxation is the one based on Robin interface conditions to transmit information across subdomain boundaries. It was first proposed in [39] and analyzed later in [19, 32]. One solves the local PDE on the subdomains using Robin conditions on the interface lines by matching a combination of Dirichlet and Neumann data from the neighboring subdomains.

- for k = 0,1,2, ...
  - On each sub-domain solve:
    - $Lu^{(k+1)} = f \in \Omega$ with
    - $-\frac{\partial u^{(k+1)}}{\partial x} + \lambda u^{(k+1)} = -\frac{\partial u_L^{(k)}}{\partial x} + \lambda u_L^{(k)}$ on subdomain's left interface.
    - $\frac{\partial u^{(k+1)}}{\partial x} + \lambda u^{(k+1)} = \frac{\partial u_R^{(k)}}{\partial x} + \lambda u_R^{(k)}$ on subdomain's right interface.

Here λ is a relaxation parameter. The convergence of this method was analyzed in [39] at the differential level, assuming arbitrary decompositions and using "energy" estimates. The determination of effective choices for λ was marked as "*by large an open problem*".

Variations of the above described method have appeared in the literature lately. Specifically in [31] an ADI–based modification for accelerating the convergence of the **ROB** scheme is proposed and analyzed. A modification of **ROB** that extents its applicability and frees it from the cross–point trouble is formulated and analyzed in [17]. Another variation that uses the tangential derivatives in addition to the normal derivative for smoothing is given in [68] where optimal values for the relaxation parameters are obtained for a model problem.

The Schur complement (SCO) Method

Among the first interface relaxation procedures that captured the attention of researchers is the one analyzed in [24] (see also the references therein). It alternates Dirichlet and Neumann interface conditions in space and can be described by

---

- for k = 0,1,2, ...
- $u1_L = u$, $u1_R = \theta_1 u_2^{(k)} + (1-\theta_1)u_1^{(k)}$
- $u_1^{(k+1)} = solve\_pde(u1_L, u1_R)$
- for i = 2,...,p-1
  - $dui_L = \frac{\partial u_{i-1}^{(k+1)}}{\partial x}$
  - $ui_R = \theta_i u_{i+1}^{(k)} + (1-\theta_i)u_i^{(k)}$
  - $u_i^{(k+1)} = solve\_pde(ui_R, dui_L)$
- $up_R = u$, $dup_L = \frac{\partial u_{p-1}^{(k+1)}}{\partial x}$
- $u_p^{(k+1)} = solve\_pde(up_R, dup_L)$

Here $\theta \in (0,1)$ is a relaxation parameter. The convergence analysis at the differential level, for the case of Helmholtz equation in two variables and 1–dimensional decompositions at differential level, is given in [24] together with expressions that lead to optimum values for θ. A method for dynamically determine, at each iteration, values for θ for the spectral collocation approximation of the differential problems is also given. To the best of our knowledge, **SCO** is the only interface relaxation technique that has so far been successfully extended and applied to fourth order elliptic problems [26].

The Shooting (SHO) Method

This method is proposed in [36] where it is formulated primarily for 1–dimensional boundary value problems. A convergence analysis was carried out and optimum values for the relaxation parameters were obtained for model problems. The basic idea is to couple the problems on the subdomains by solving the defect equation $D(ui^{(k)}) \equiv \frac{\partial u_L^{(k)}}{\partial x} - \frac{\partial u_R^{(k)}}{\partial x} = 0$ on the interfaces using a fixed point (Picard) iteration scheme to obtain new values.

- for k = 0,1,2, ...
  - $\alpha^{(k+1)} = \frac{\alpha^{(k)} \left| D(ui^{(k+1)}) \right|}{\left| D(ui^{(k)}) - D(ui^{(k+1)}) \right|}$ on each interface
  - $ui^{(k+2)} = ui^{(k)} - \alpha^{(k+1)} D(ui^{(k)})$ on each interface
  - $u^{(k+2)} = \text{solve\_pde}(ui^{(k+2)})$ in each subdomain

---

The Steklov–Poincaré operator (SPO) Method

This method was first mentioned in [37] but analyzed from the preconditioning viewpoint only. It uses the Steklov–Poincaré operator to carry the procedure of smoothing the normal derivatives at the interfaces, it is a *two-step* method described by the following algorithm

---

- for k = 0,1,2, ...
  - $u^{(k+\frac{1}{2})} = \text{solve\_pde}(ui)$ in each subdomain
  - $dui = \frac{1}{2} \left(\frac{\partial u_R^{(k+\frac{1}{2})}}{\partial x} + \frac{\partial u_L^{(k+\frac{1}{2})}}{\partial x} \right)$ on each interface
  - $u^{(k+1)} = \text{solve\_pde0}(dui)$ $(Lu = 0)$ in each subdomain
  - $ui = ui - \frac{\rho}{2} \left(u_R^{(k+1)} + u_L^{(k+1)} \right)$ on each interface

---

No theoretical results, from the interface relaxation viewpoint, are available for **SPO**.

2.3.1 Methods not considered

As mentioned in the introduction, powerful interface relaxation methods can be constructed using spectral expansions of a trace operator. In this approach operators like Lagrange multipliers or Steklov–Poincaré operator, which can be interpreted as the interface flux, are solved to determine an improved value of the unknown function on the interface. Specifically, in [47] and [48] an independent low dimensional set of interfacial basis functions are used to meet interdomain continuity requirements on the solution. These functions are derived locally in each subdomain by solving an eigenvalue problem of the Steklov–Poincaré operator on the complementary region. An idea similar to the above approach is used in [20] and [21] where a different set of basis functions is used to smooth across interfaces. It is shown that a relatively small basis set for the Lagrange multipliers has certain significant advantages. In particular trigonometric functions, orthogonal polynomials, and one–dimensional Lagrange finite elements have been suggested as approximating basis set on the interface.

We have also investigated a new method which simply makes a least squares fit to approximately satisfy the overdetermined interface conditions at each iteration. This method seems to be very much slower than any other interface relaxation method so we do not present our data for it.

2.4 Numerical Experiments

An extensive and systematic performance evaluation study of all the interface relaxation schemes presented above is under way for general two dimensional decompositions using the SciAgents system (see Chapter 5). Parts of this study will be presented in Chapter 6. In this section we present and discuss numerical performance data mainly for one dimensional problems. These problems might be too simple to be of practical importance, but the experimentation with them can be very illuminating for understanding the nature of the interface relaxation methods. They might be useful to show the physical meaning and importance of the various characteristics and parameters involved in the relaxers in particular for general unstructured decompositions.

We consider the differential equation $-u'' + \gamma^2 u = f \ in \ [0,1]$, where f is selected such that $u(x) = e^{x+4}x(x-1)(x-.7)$ and we assume Dirichlet boundary conditions. All interface relaxation schemes are implemented in a unified way using MATLAB on a SUN workstation. The MATLAB code for the algorithms given in the Appendix can be obtained from our web page[1]. Central differences are used to discretize the differential equation. The interval $[0,1]$

[1] `http://www.cs.purdue.edu/homes/giwta/dom-dec/1_dim/matlab/index.html`

		2nd iteration		4th iteration		6th iteration		8th iteration		10th iteration	
		error	ϕ_k	error	ϕ_k	error	ϕ_k	error	ϕ_k	error	ϕ_k
AVE	N=80	6.27E-1	.11	4.89E-2	.33	7.53E-3	.48	4.56E-3	.58	4.3E-3	.64
	N=160	6.24E-1	.11	4.57E-2	.33	4.23E-3	.48	1.25E-3	.58	1.0E-3	.64
GEO	N=80	2.74E-0	.08	8.42E-1	.32	2.99E-1	.48	1.06E-1	.58	3.7E-2	.64
	N=160	2.74E-0	.08	8.40E-1	.32	2.99E-1	.48	1.06E-1	.58	3.7E-2	.64
NEW	N=80	6.10E-0	.03	7.20E-1	.33	1.57E-1	.48	1.97E-1	.58	1.7E-1	.64
	N=160	6.10E-0	.02	7.22E-1	.33	1.58E-1	.48	1.08E-1	.58	7.8E-2	.64
ROB	N=80	5.43E-0	.15	3.21E-0	.36	1.96E-0	.50	1.23E-0	.59	7.9E-1	.64
	N=160	5.45E-0	.15	3.24E-0	.36	2.00E-0	.50	1.27E-0	.59	8.1E-1	.64
SCO	N=80	2.09E-0	.13	4.83E-1	.34	8.22E-2	.48	2.04E-2	.58	1.1E-2	.64
	N=160	2.09E-0	.13	4.90E-1	.34	8.41E-2	.48	1.49E-2	.58	3.4E-3	.64
SHO	N=80	5.40E-0	.04	3.82E-1	.33	1.12E-1	.48	6.96E-2	.58	4.5E-3	.64
	N=160	5.40E-0	.04	3.83E-1	.33	1.13E-1	.48	6.79E-2	.58	3.9E-3	.64
SPO	N=80	2.31E-0	.09	3.74E-1	.33	6.59E-2	.48	1.25E-2	.58	2.3E-3	.64
	N=160	2.31E-0	.09	3.73E-1	.33	6.57E-2	.48	1.26E-2	.58	2.6E-3	.64

Table 2.1: The convergence factor and the error for several iterations of seven relaxers in a 4 sub-domain uniform decomposition using a total of 80 or 160 grid points in [0,1].

is partitioned into subdomains with interface conditions taken to be continuous value and derivative. Unless otherwise stated, we start all iterations from a zero initial guess and we select the values for the various parameters involved in the relaxation scheme in a straight forward and naive way. In particular we set $\alpha = \beta = 1/2$ in **AVE**, $w_L = w_R =$ half the length of the associated subdomain in **GEO**, $\lambda = 1/2$ in **ROB** and **SPO**, and $\theta = 1/2$ in **SCO**. We also set $\gamma^2 = 20$ for all data except Figure 2.8. We select this, not very common, value of γ^2 in order to increase the experimental data (see the discussion of Figures 2.3-2.4) that can be fitted into the plots (in particular in Figures 2.3 and 2.8) so a clear qualitative comparison picture can be easily drawn.

We have verified, by experimentation, that the convergence rate of all methods is independent of the local grid size. A very representative set of data is given in Table 2.1 where we present the convergence factor

$$\phi_k = \sqrt[k]{\frac{\| Du^{(k)} - f \|_\infty}{\| Du^{(0)} - f \|_\infty}} \quad \text{for} \quad k = 2, 4, 6, 8 \quad \text{and} \quad 10 \tag{2.4.1}$$

and the associated error norm $\| u^{(k)} - u \|_\infty$ for all methods and for the cases of 80 and 160 equally distributed in $[0, 1]$ discretization points. It is easily seen that ϕ_k does not depend on the number of grids.

For the rest of the experiments we use 160 equally distributed grid points to discretize the domain $\Omega \equiv (0, 1)$.

We start with Figure 2.3 where the convergence rate of all relaxers is presented for 2, 4, 5 and 8 subdomains. We plot the logarithm of the max–norm of the error (on the y–axis) of the computed solution at the first 20 iterations versus the iteration number. For 8 subdomains

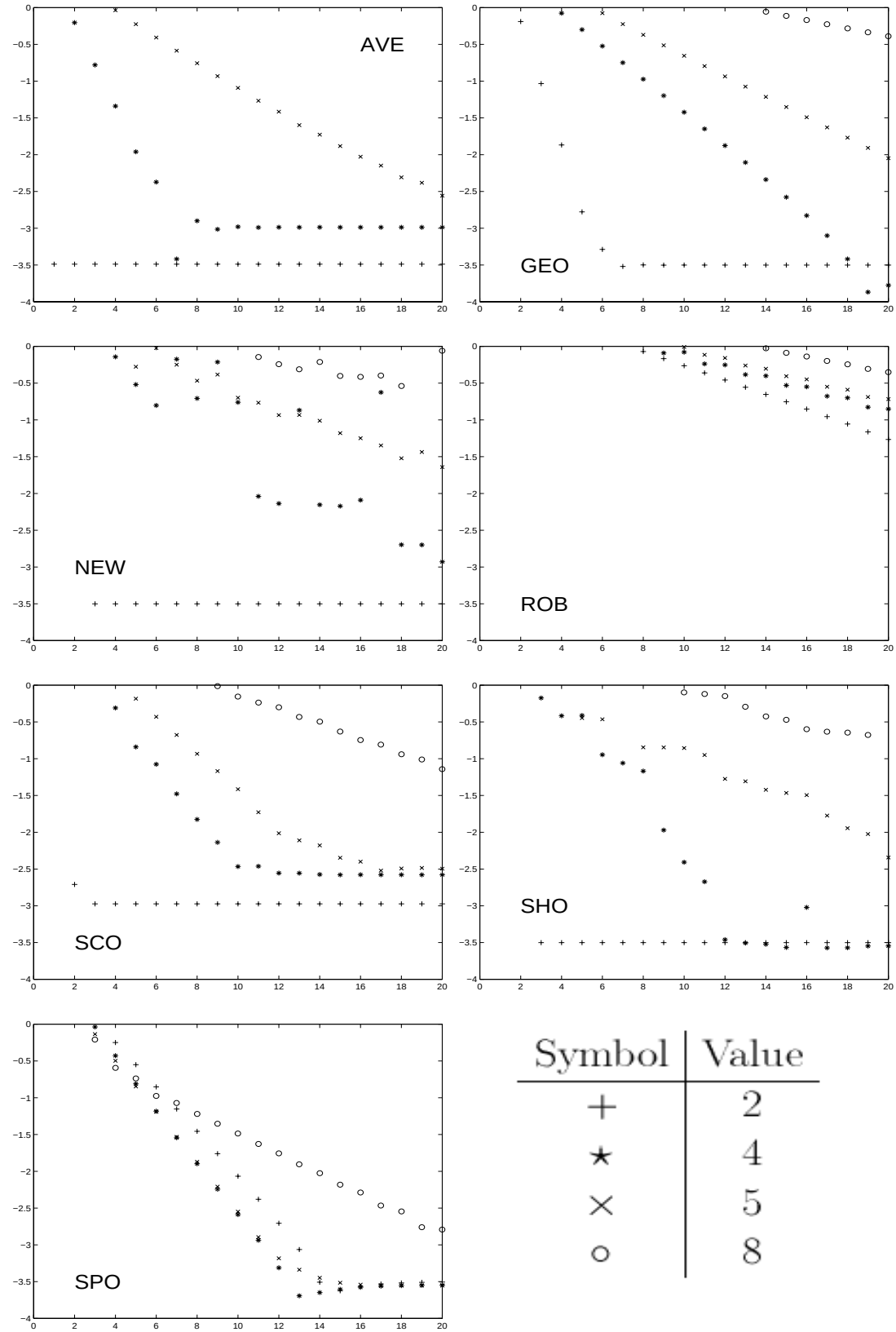
SPO is the fastest and **AVE** the second slowest (not seen in Figure 2.3). Nevertheless **AVE** is the fastest for 2 and 4 subdomains. **NEW** and **SHO** behave in a similar and rather erratic way. We also plot in Figure 2.5 the convergence factor ϕ_k (formula (2.4.1)) versus k for 2,4,5 and 8 subdomains. **AVE** clearly diverge for 8 subdomains while the rest of the methods exhibit the same pattern of convergence.

As it was previously mentioned, we decided to fix $\gamma^2 = 20$ for most of our experiments. The main reason for this choice was the fact that at least two of the methods considered (**AVE** and **SPO**) are expected to behave poorly for small values of γ^2. This is due to the fact that during their Neumann sweep, they both need to solve on the internal subdomains the Helmholtz operator with Neumann boundary conditions on both end points. In such a case the PDE becomes singular as γ^2 approximates zero. This is clearly reflected in Figure 2.4, where we present the convergence rate data, as in Figure 2.3 but now for $\gamma^2 = 1$. Besides the break down of **AVE** and **SPO** for more than two subdomains, it is also seen a general decrease, relatively to the $\gamma^2 = 20$ case, on the rates of convergence.

We believe that the specific convergence pattern might give important information about the convergence characteristics. To explore this, Figure 2.6 shows, for all relaxers and for a uniform decomposition of Ω into 4 subdomains, the exact solution and the computed solutions associated with the first three iterations. Three of the schemes (**GEO**, **SHO** and **SPO**) approach the exact solution in a monotonic (or nearly so) and smooth way while the rest do not seem to exhibit a specific pattern.

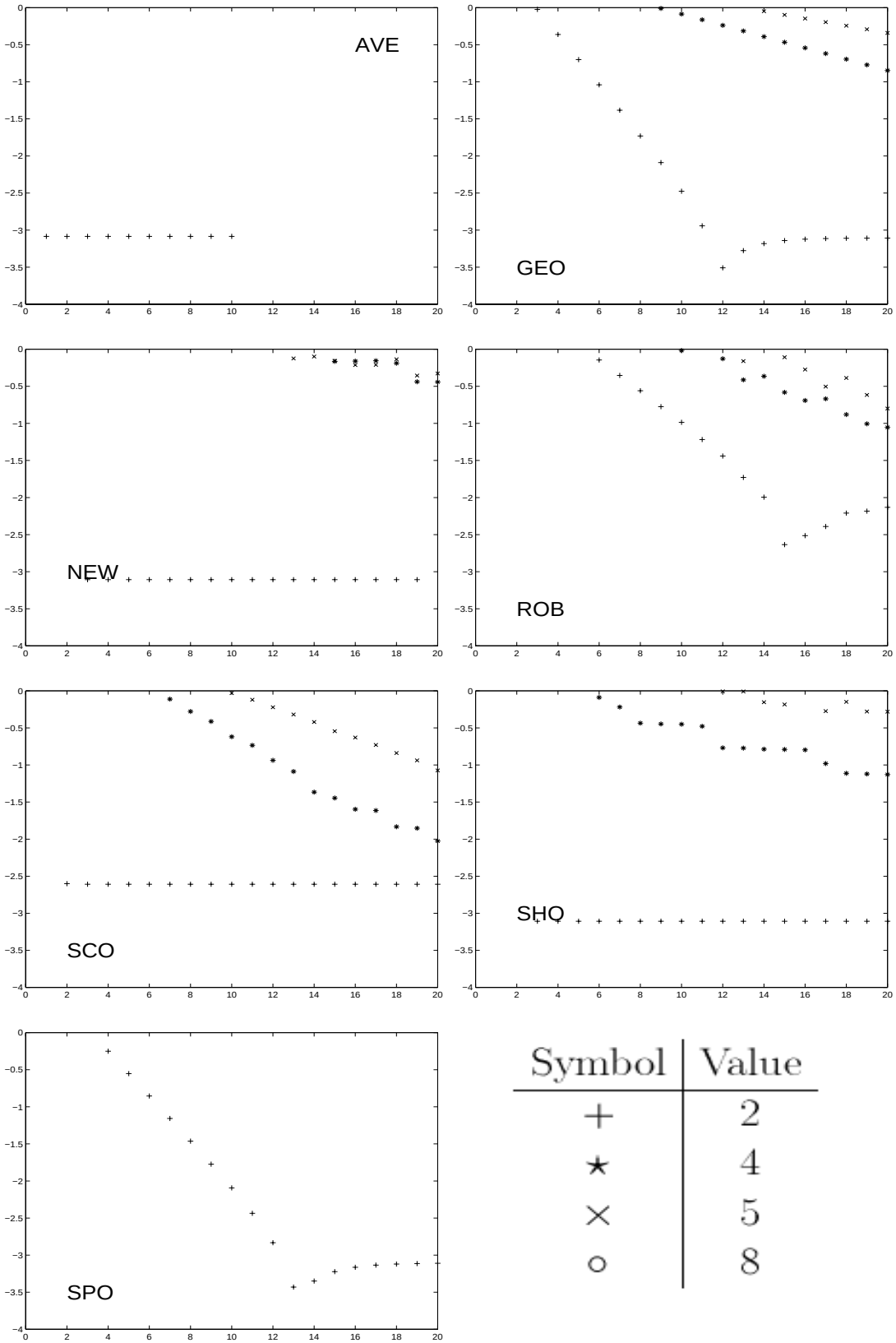
We next examine the convergence history of the *two step schemes* in more detail. The plots associated with the two–step methods (**AVE** and **SPO**) in Figure 2.6 correspond to their Dirichlet steps. In Figure 2.7 we present, in the same way, the history for their Neumann steps as well. We therefore see that some methods (**GEO**, **SPO**) converge in a monotonic and systematic way. This suggests that their convergence could be accelerated by some extrapolation procedure. Other methods exhibit oscillatory convergence so averaging might improve the convergence. Still others show no obvious patterns of convergence.

As it is obviously expected, and already seen in Figures 2.3 and 2.4, the convergence of the interface relaxation depends on the differential operator. To obtain a preliminary idea about this dependence we systematically vary the coefficient $\gamma^2 (= 10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 20,$ 30) in the operator and observe the convergence for a 4 subdomain uniform partition of Ω. Our data are presented in Figure 2.8 which uses the same axes as in Figure 2.3. As γ^2 becomes larger there is a general trend for the convergence rate to become faster (**AVE**, **GEO**, **NEW**, **SCO**, **SHO**) or to be nearly unchanged (**ROB**, **SPO**). Note that **AVE** and **SPO** diverge for $\gamma^2 \leq 1$ while the data for the rest methods are split in two groups, one for $\gamma^2 = 10, 20$ and 30 and one for $\gamma^2 = .1, .01$ and .001 with the case of $\gamma^2 = 1$ in the later



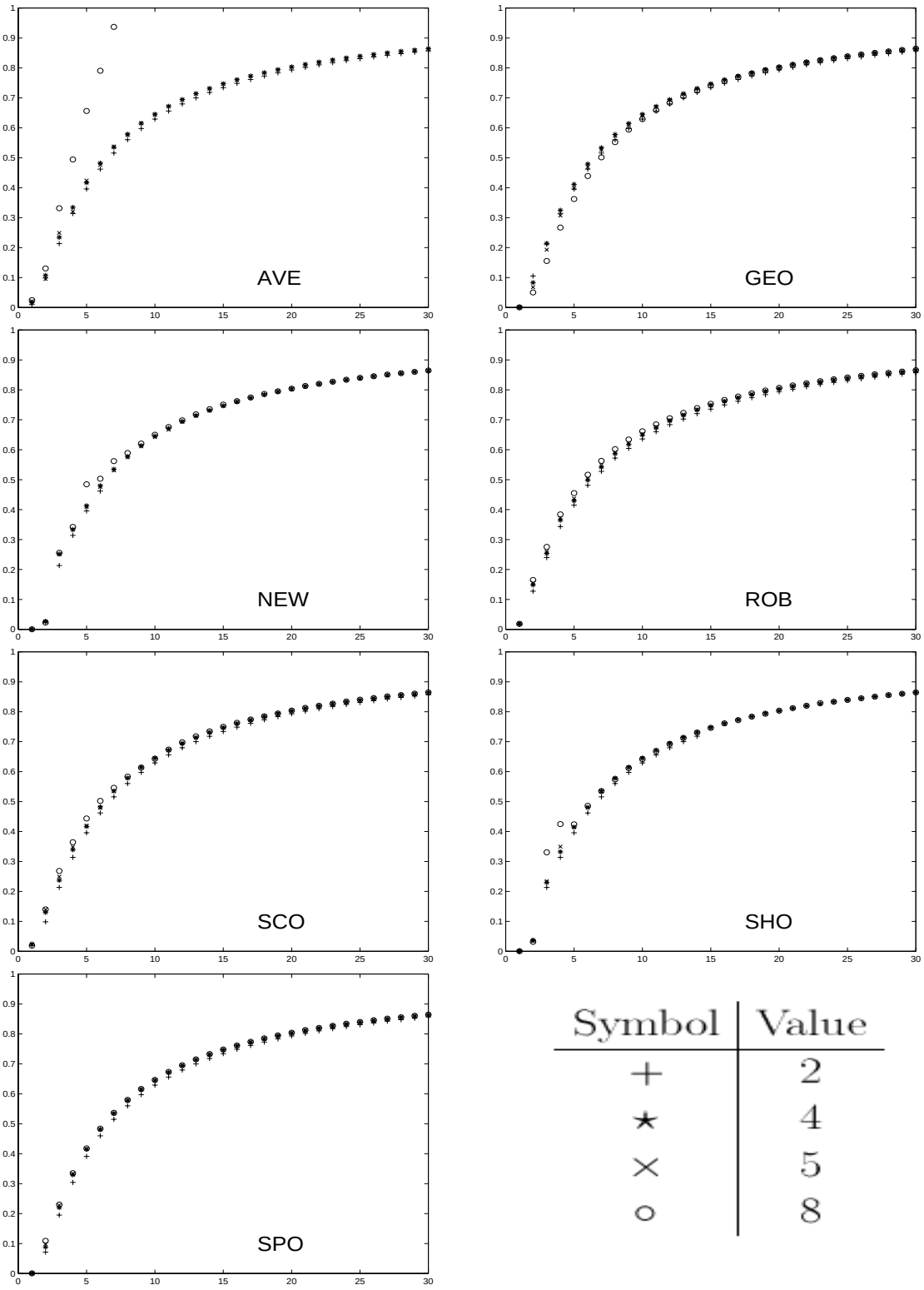
Symbol	Value
+	2
⋆	4
×	5
○	8

Figure 2.3: Convergence plots for 2 (+), 4 (*), 5 (x) and 8 (o) subdomains, for $\gamma^2 = 20$. On the x-axis we have the iteration number and on the y–axis the logarithm of the max–norm of the error.



Symbol	Value
+	2
⋆	4
×	5
○	8

Figure 2.4: Convergence plots for 2 (+), 4 (*), 5 (x) and 8 (o) subdomains, for $\gamma^2 = 1$. On the x-axis we have the iteration number and on the y–axis the logarithm of the max–norm of the error.



Symbol	Value
+	2
⋆	4
×	5
∘	8

Figure 2.5: The convergence factor ϕ_k versus iterations for all methods, for 2 (+), 4 (*), 5 (x) and 8 (o) subdomains.

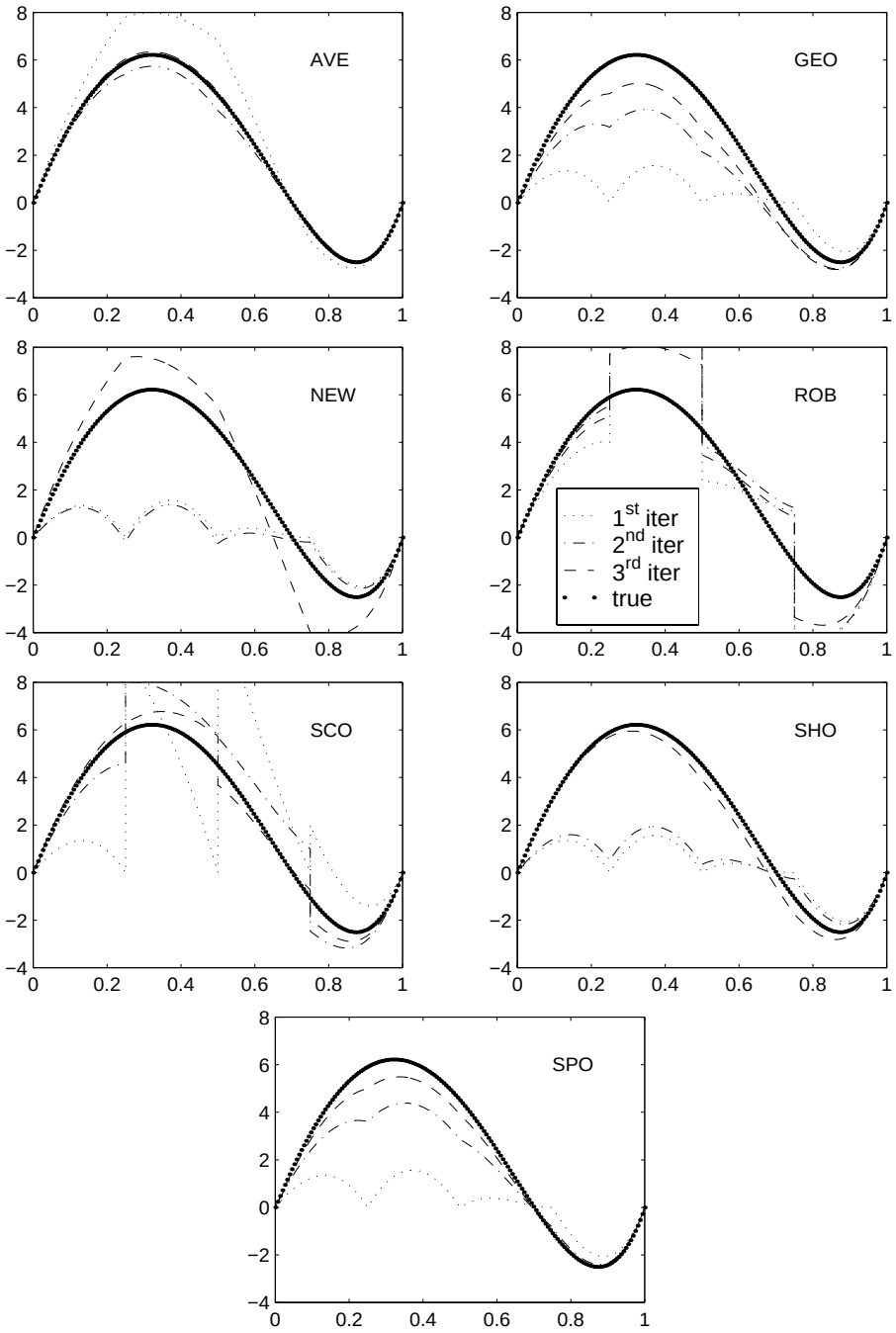


Figure 2.6: Convergence history of the seven relaxers in a 4 subdomain decomposition and $\gamma^2 = 20$. The true solution (solid line) is plotted along with the first (dotted line), second (dot–dashed line), and third (dashed line) computed solutions.

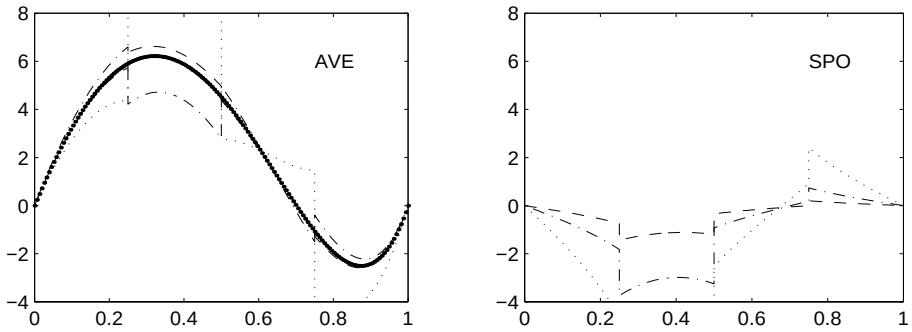


Figure 2.7: Convergence history of the two–step relaxers, **AVE** and **SPO**, during the Neumann sweep.

group except for the **ROB** method.

Recall that all the data presented so far correspond to uniform partitions of the interval $[0, 1]$. We next test the effect of non–uniform partitions by moving the interface point (denoted by *ip*) from .2 to .4, .6 and .8. In Figure 2.9, where we consider the four different 2 subdomain partitions, we clearly see that none of the schemes is very sensitive to this change. Again the axes are as in Figure 2.3.

Recall that two (**NEW**, **SHO**) of the seven methods are parameter-free. The rest involve parameters of various kind whose values were selected in a naive and straightforward way for the experiments described above. In Figure 2.10 we systematically vary the values of these parameters and present convergence plots for the 2 subdomain case with the interface point at .8. In those two methods (**AVE**, **GEO**) with two parameters, their values are made equal in this experiment. Note that for the **GEO** method, $w_L = \alpha * \ell_L$ and $w_R = \alpha * \ell_R$ where ℓ_L, ℓ_R are the lengths of the left and right subdomain respectively and α takes the values shown on the symbol legend. We see that the parameter choices have a strong affect on the convergence behavior. The best parameter choices for Figure 2.10 are: **AVE** ($\alpha = \beta = 0.3$), **GEO** ($w_L = w_R = 0.6$), **ROB** ($\lambda = 0.9$), **SCO** ($\theta = 0.5$) and **SPO** ($\rho = 0.9$). These data show clearly that there is an important open question for these methods: **How does one choose optimal (or good) parameter values?**

Among all the relaxation methods considered in this study only **SCO** appears to be sensitive on the order the various subdomains are processed during the interface relaxation process. We experimented with **SCO** as follows: Select a subdomain q as the first for an iteration and then process the others in sequence (left to right, wrapping around at the right end of the interval). In Table 2.2 we present the number of iterations required by this scheme to reduce the norm of the difference of two successive iterants below 10^{-5} (i.e.

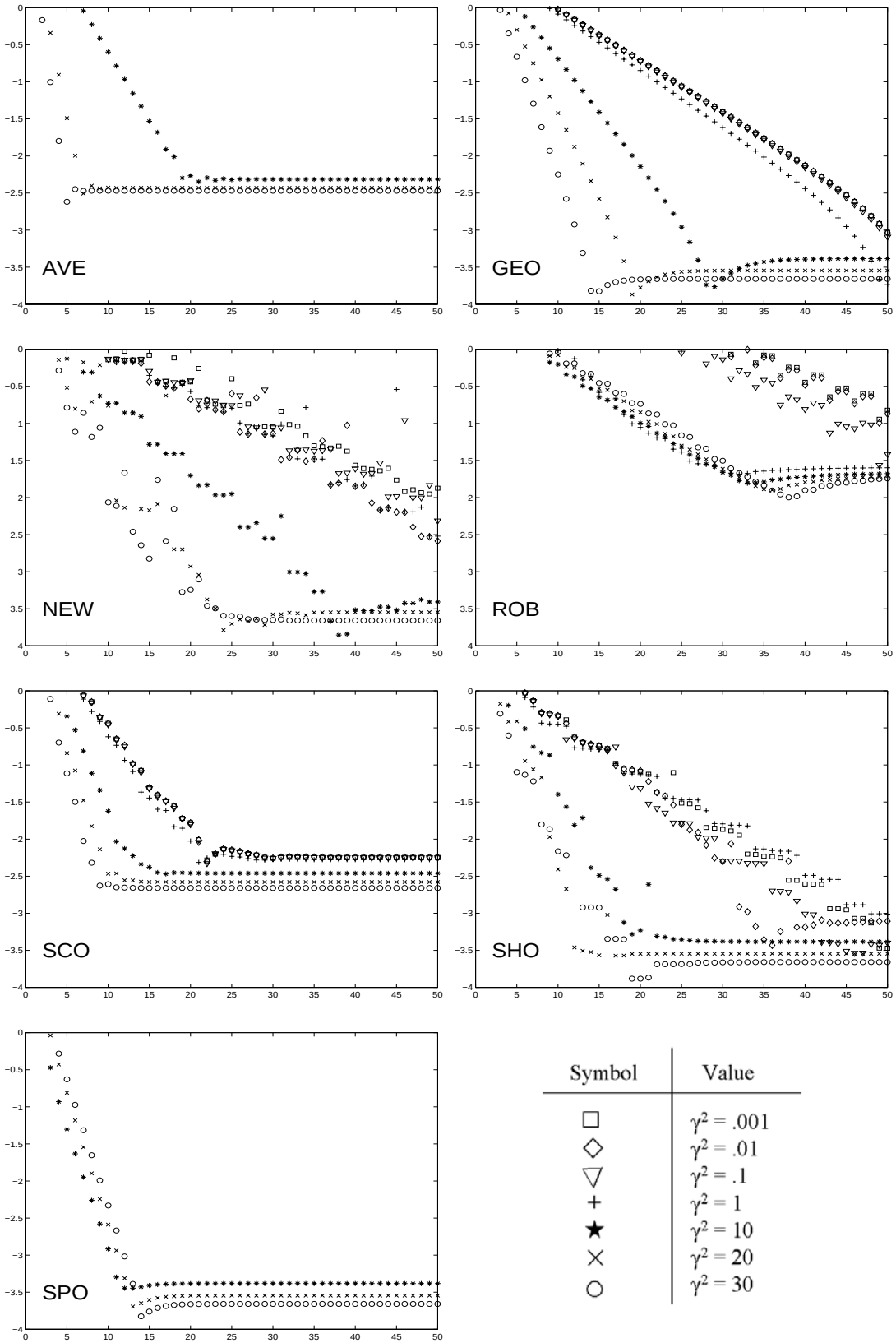
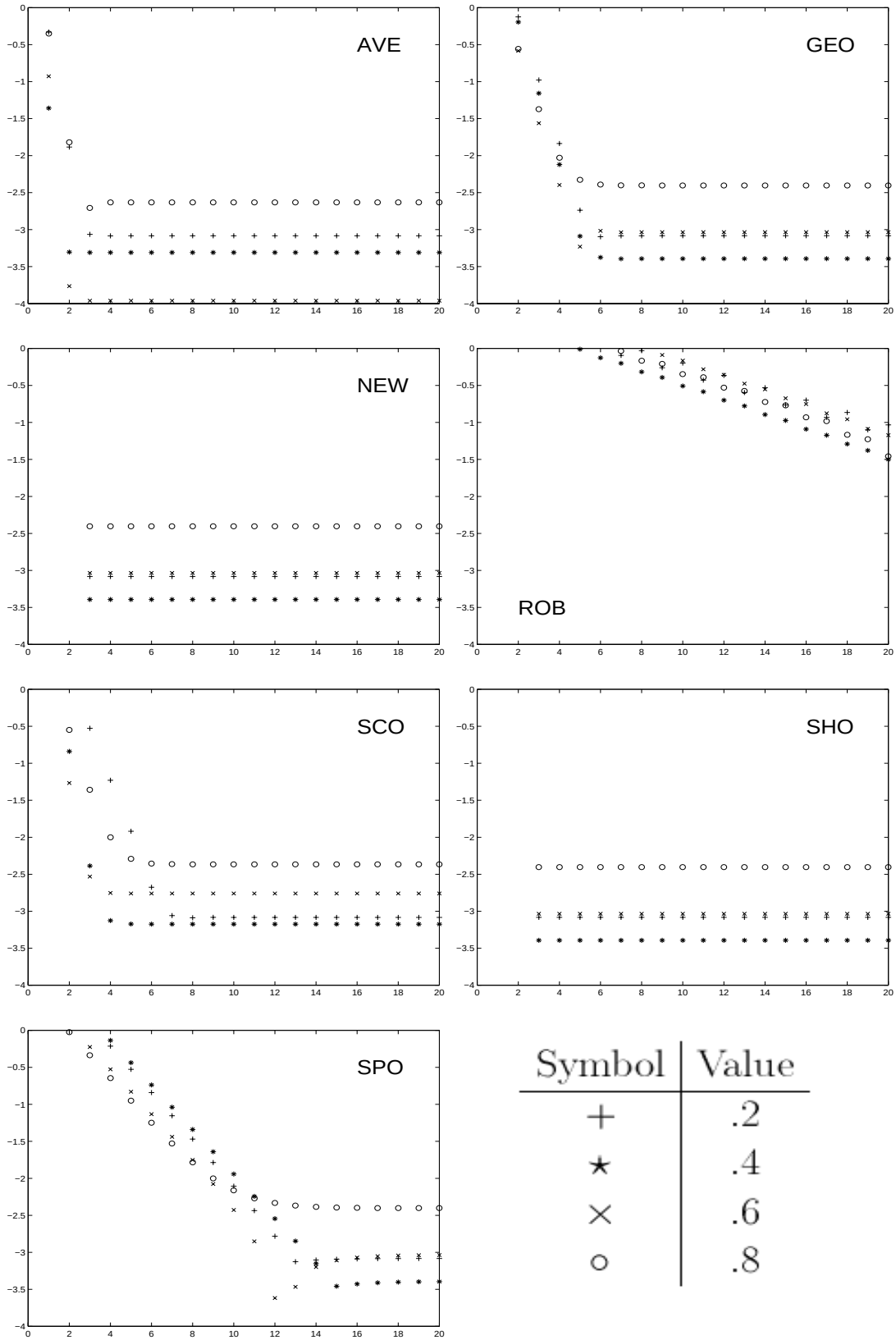


Figure 2.8: The effect of the coefficient γ^2 on the convergence rate of the relaxation schemes with a four subdomain partition of $[0, 1]$. The legends and the value of γ^2 are given in the lower right.



Symbol	Value
+	.2
⋆	.4
×	.6
○	.8

Figure 2.9: The convergence of the relaxation schemes for non-uniform 2 subdomain decompositions. The interface point *ip* is placed at 0.2 (+), 0.4 (*), 0.6 (x) and 0.8 (o).

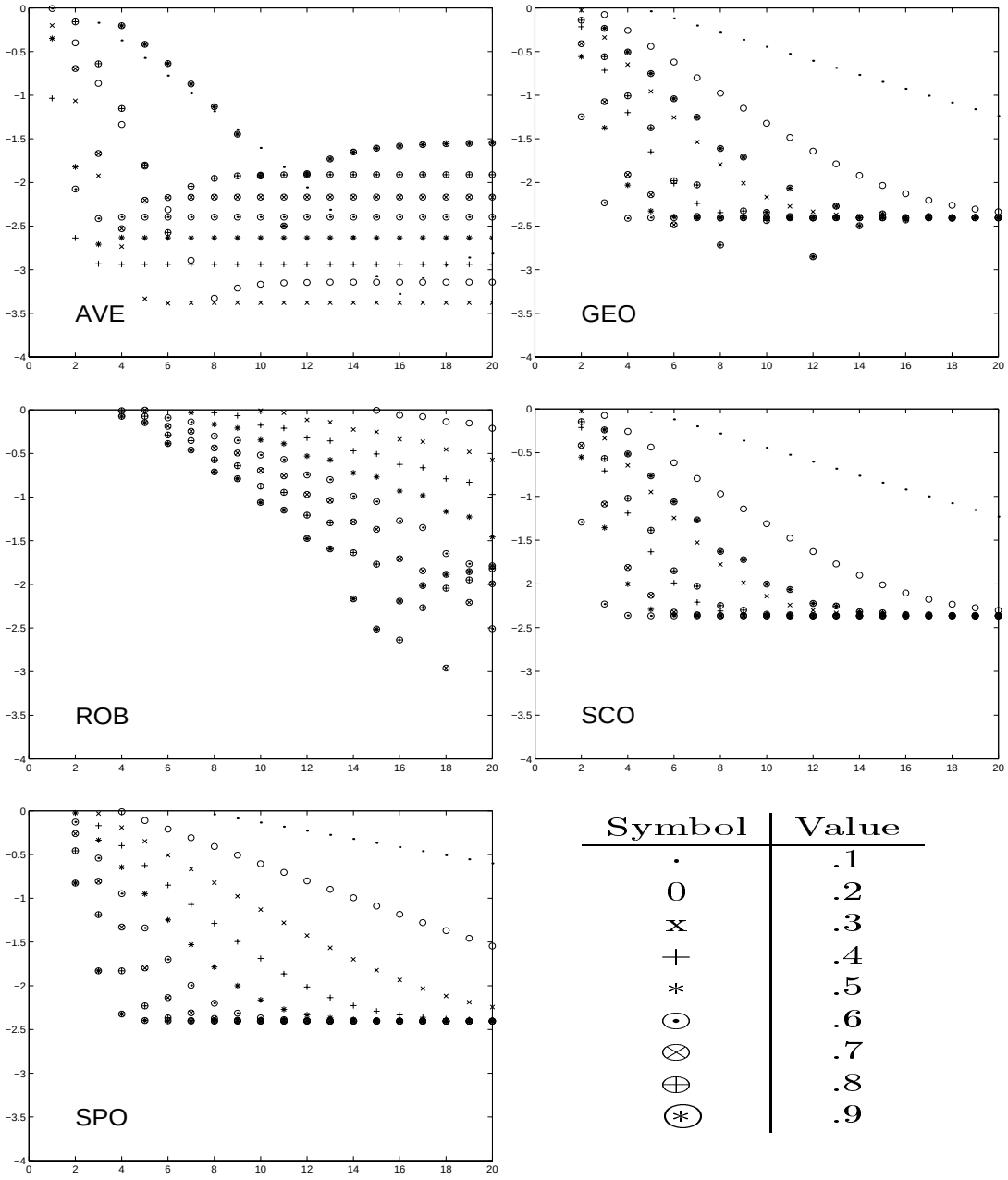


Figure 2.10: The effect of the parameter selection on the convergence behavior of five relaxation methods for $\gamma^2 = 1$. The legends and parameter values used are given in the lower right, the two parameters of **AVE** and **GEO** are both set equal to the value shown.

Starting Subdomain	1	2	3	4	5	6	7	8
Number of Iterations	48	40	32	24	25	34	40	42

Table 2.2: Number of iterations k to achieve $\|u^{(k+1)} - u^{(k)}\|_\infty < 10^{-5}$ for various starting subdomains in the **SCO** method.

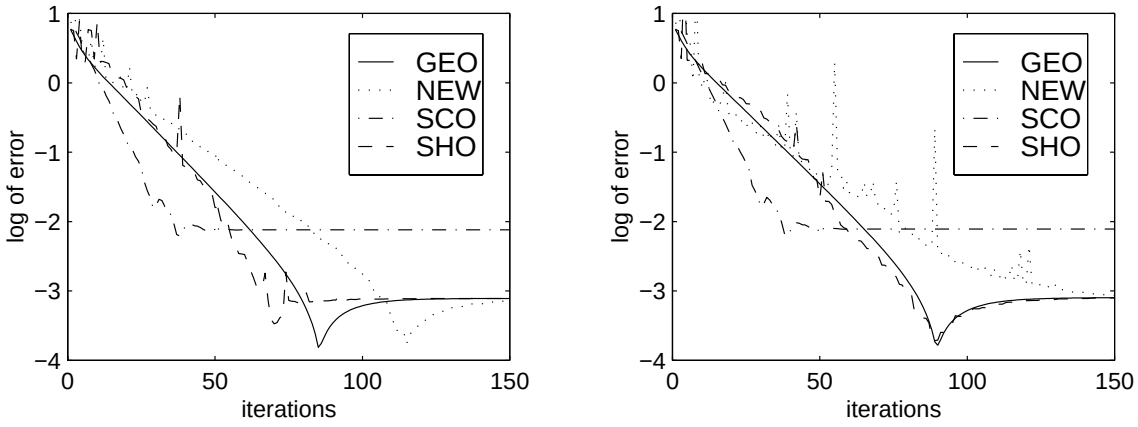


Figure 2.11: The history of convergence of four relaxation methods for $-u'' + \sin(2\pi x)u = f$ (on the left) and $-u'' + \cos(2\pi x)u = g$ (on the right) in the case of 5 sub-domains of equal size.

$\|u^{(k+1)} - u^{(k)}\|_\infty < 10^{-5}$) as a function of the starting subdomain q. Specifically, for Table 2.2 we use a uniform decomposition of 8 subdomains and we start the domain decomposition scheme from subdomain $q = 1, 2, \ldots, 8$. We see that the selection of the starting subdomain significantly affects the rate of convergence of the **SCO** interface relaxation method.

Finally we test if the convergence of the methods depends on the definiteness of the PDE operator. We decompose the domain uniformly into 5 sub-domains and consider the following two differential equations, $-u'' + \sin(2\pi x)u = f$ and $-u'' + \cos(2\pi x)u = g$, that do not satisfy the ellipticity condition and appear (in a two dimensional form) in practical applications. Only four methods (**GEO**, **NEW**, **SCO**, **SHO**) converge for these indefinite problems. Figure 2.11 shows the convergence behavior of these methods for both problems. We see that the convergence rate is comparable to that seen in Figure 2.3. The rest either diverge or oscillate. Such behavior has been already noticed for some of the methods [19]. We are unable to formally explain why the **SCO** convergence stagnates at 10^{-2}. One possibility is that the asymptotic error constant involved in this method [45] is greatly affected by the particular form of these problems.

We should add that we have implemented most of the relaxation schemes presented above for two dimensional problems using ELLPACK [54] assuming "skyline" domains

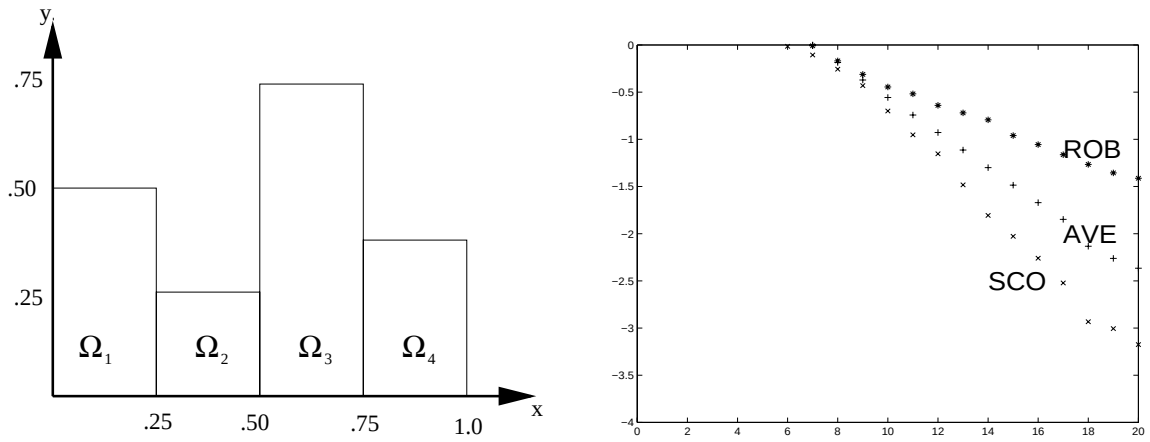


Figure 2.12: The history of convergence of **AVE** (+ symbols), **ROB** (*) and **SCO** (x) methods for $-\Delta u + 10u = f$ (on the right) assuming the PDE domain and its partition given on the left.

(a string of rectangles of different heights and widths [2]). This leads to one dimensional decompositions and we performed some selective experiments, all of these were in good agreement with both the quantitative and qualitative conclusions we draw from the one dimensional experiments presented above. The detailed presentation of our performance study is beyond the scope of this Chapter. We will provide more experimental data later on, in Chapter 6. Nevertheless, in the right plot of Figure 2.12 we give the history of convergence of the **AVE**, **SCO**, and **ROB** methods for the differential equation $-\Delta u + 10u = f \in \Omega$ where f is selected such that $u(x) = e^{y(x+4)}x(x-1)(x-.7)y(y-.5)$. We assume Dirichlet boundary conditions. The PDE domain and its one dimensional partition into 4 subdomains $\Omega \equiv \cap_{i=1}^{4}\Omega_i$ is depicted in left plot of Figure 2.12. The 5-point star ELLPACK discretization module was used. The similarity of the convergence behavior of the three methods in one dimension (Figure 2.8) and in two dimensions (Figure 2.12) is easily observed.

2.5 Conclusions

We present a wide class of non–overlapping domain decomposition, interface relaxation methods for elliptic differential equations. A set of experiments are described which explore the convergence properties of these methods in several directions. The qualitative conclusions are categorized in Figure 2.13. This figure also summarizes the existing mathematical and derived computational properties of the methods. It is seen that the speed of

[2] This ELLPACK code is also available from our web page.

convergence of the interface relaxation methods can be of high, moderate or low, that the iterates can approach the exact solution monotonically or not, and there can be two, one or no relaxation parameters to accelerate the convergence. Some single or two step interface relaxation methods use "history" (the new value on the interface is explicitly set to be the old one plus a correction term), some do not. It is natural to expect that the rate of convergence of all interface relaxation methods is affected, to some extend, by certain problem parameters. Some of the most important of these parameters are the fineness of the mesh or grid discretization of the domains (column "domain discretization" in Figure 2.13), the particular method (finite element, finite difference ...) used to discretize the PDE operators (column "PDE discretization") and the geometric characteristics (rectangular or not, holes, wide angles ...) of the domains (column "PDE domain"). The theory and the experimental data available to explain all these cases and phenomena is very limited. In Figure 2.13, we present our conclusions about the effect of these parameters drawn for either the existing theoretical studies or from our preliminary experiments with various PDE problems.

The principal conclusions of the study presented in this Chapter are: (1) There are many interface relaxation methods that seem to have the potential to work effectively. (2) There is still much to be learned about their behavior and about how to choose among them or to choose their parameters.

	Speed	Convergence Monotonicity	Parameters	Theory Available	History	Single Step	Effects of: No Subdomains	Domain Discretization	PDE Discretization	PDE Operator	PDE Domain	Interface Position
AVE	—		2	**Some**		No						
GEO	—	**Yes**	2		**Yes**	**Yes**						
NEW	—		**None**		**Yes**	**Yes**					High	
ROB	Low		1	**Some**		**Yes**	**Low**		**Low**	**Low**	**Low**	**Low**
SCO	—		1	**Some**	**Yes**	No						
SHO	—	**Yes**	**None**	**Some**	**Yes**	**Yes**						
SPO	**High**	**Yes**	1		**Yes**	No				**Low**		

Figure 2.13: Categorization of the properties of the seven interface relaxation methods. A blank entry indicates an "average" evaluation (for numerical properties) or absence of a property. Those evaluations considered to be positive are given in larger, bolder type.

Chapter 3

Fine Tuning Interface Relaxation Methods

Abstract

Two simple interface relaxation techniques for solving elliptic differential equations are considered. Their theoretical analysis at the differential level is carried out and "optimal" relaxation parameters are obtained for one dimensional model problems. An experimental numerical study is also presented.

3.1 Introduction

Domain decomposition has proven an effective means of partitioning the task of solving Differential Equation (DE) problems numerically. It is mainly an algebraic approach and works by splitting the discrete DE domain into subdomains which can be coupled in many ways. The well established additive and multiplicative Schwartz methods are examples of typical domain decomposition approaches that have been analyzed extensively. Interface Relaxation (IR) is a step beyond domain decomposition [53]. IR methods are defined and analyzed at the continuous level, yet they can be implemented by traditional numerical methods which can vary from subdomain to subdomain. They assume a splitting of the domain into a set of non-overlapping subdomains and consider the associated DE problem defined on them. These subproblems are coupled through relaxation mechanisms on the interfaces. IR methods naturally apply to multi-physics problems when the DE may change from one subdomain to another. For a general introduction to the IR methodology the reader is refereed to [53, 45, 46].

A review study of a large collection of IR methods can be found in Chapter 2. The convergence of these schemes depends, as expected, on the differential operator, the geometry of the original domain, and in addition on the geometry of the subdomains chosen. This makes the selection of "optimum" values for the relaxation parameters a hard and challenging problem. On the other hand, the local subdomain discretization scheme does not affect the convergence properties of the IR schemes which gives these methods great versatility; one can select the most appropriate discretization parameters or numerical method for the differential problem defined on each subdomain.

The development of an automated and adaptive procedure that dynamically estimates "good" relaxation parameters, using automatic differentiation techniques, for general differential operators and arbitrarily shaped subdomains is under way [56]. Nevertheless, in order for this parameter selection procedure to be effective, theoretical results for simple model problems are needed that provide the required reasonably good initial guess for the optimum values of the parameters and, more importantly, a better understanding of the convergence mechanisms involved.

The main objective of this Chapter is to better understand IR methods for one dimensional model problems where direct analysis can be made. In particular, we analytically estimate values for the parameters involved in two recently proposed and analyzed IR methods. Namely we consider an averaging scheme [60, 75, 77] (denoted by **AVE** in the sequel) and a Robin–type IR scheme [39] (denoted by **ROB**). Although both methods were considered in several previous studies more understanding is needed. Specifically excretions

that relate their rate of convergence to the characteristics of the differential problem and its partition in a clear way are needed.

We restrict ourselves to Helmholtz boundary value problems. In both schemes the error involved on each interface can be given analytically in terms of the error in the previous iteration. This leads us to a system of linear algebraic equations that represents the relation between the errors on all interfaces in two consequent iterations. Then, we minimize the spectral radius of the iteration matrix involved using a different approach for each method. For the **AVE** scheme we minimize the area of the associated Gerschgorin discs (which is equivalent of bounding the max norm of the iteration matrix) to derive, in Theorem 3.3.4, an important relation between the size of the subdomains and the coefficient of the differential equation that determines the domain of convergence of the method.

The iteration matrix associated with the **ROB** scheme is quite sparse, and so we were able to make its spectral radius zero by selecting appropriate values for the relaxation parameters involved. In particular, Theorem 3.3.3 gives optimum values of the relaxation parameters involved, which are proved to be independent both of the particular discretization of the differential operator and its original domain.

The rest of this Chapter is organized as following. In the next section we formulate the two Interface Relaxation methods whose theoretical convergence analysis is given in Section 3. Section 4 presents numerical results from an experimental study which confirm our theoretical results; they also show that these hold for more general problems, including two dimensional ones.

3.2 Two interface relaxation methods

We consider the Helmholtz boundary value problem

$$Lu \equiv -u''(x) + \gamma^2(x)u(x) = f(x), \qquad x \in \Omega \equiv (a,b) \tag{3.2.1}$$

with $a, b \in \mathbb{R}$, subject to boundary conditions on a and b which, for simplicity, are taken to be homogeneous Dirichlet. Assume that Ω is decomposed into the p non-overlapping subdomains $\Omega_i \equiv (x_{i-1}, x_i)$, $i = 1, \ldots, p$ with $x_0 = a$, $x_p = b$ and $x_{i-1} < x_i \in \Omega$ for $i = 1, \ldots, p-1$. We denote the size of a subdomain Ω_i by $\ell_i = x_i - x_{i-1}$ and the restrictions of L, f and γ in Ω_i by L_i, f_i, γ_i, respectively. We further assume that $\gamma(x) = \gamma_i$ for $x \in \Omega_i, i = 1, \ldots, p$, where the γ_i's are real constants.

3.2.1 The ROB method.

The **ROB** scheme is defined, for the model problem under consideration, by the following algorithm:

1. Define:

$$\left.\begin{aligned} g_i^i &= \left.\frac{du_{i+1}^{(k)}}{dx}\right|_{x=x_i} + \lambda_i \left. u_{i+1}^{(k)}\right|_{x=x_i} \\ g_i^{i+1} &= -\left.\frac{du_i^{(k)}}{dx}\right|_{x=x_i} + \lambda_i \left. u_i^{(k)}\right|_{x=x_i} \end{aligned}\right\} i = 1, \ldots, p-1.$$

2. Choose initial guesses $u_i^{(0)}(x)$ for the solutions on each subdomain $\Omega_i, i = 1, 2, \ldots, p$.

3. Define the sequence of subdomain solutions $u_i^{(k)}(x)$, $k = 1, 2, \ldots$ as follows:

$$\begin{aligned} & L_1 u_1^{(k+1)} = f_1 \quad \text{in} \quad \Omega_1 \\ & \left. u_1^{(k+1)}\right|_{x=x_0} = 0 \\ & \left.\frac{du_1^{(k+1)}}{dx}\right|_{x=x_1} + \lambda_1 \left. u_1^{(k+1)}\right|_{x=x_1} = g_1^1 \end{aligned} \quad \left| \quad \begin{aligned} & L_p u_p^{(k+1)} = f_p \quad \text{in} \quad \Omega_p \\ & -\left.\frac{du_p^{(k+1)}}{dx}\right|_{x=x_{p-1}} + \lambda_{p-1} \left. u_p^{(k+1)}\right|_{x=x_{p-1}} = g_{p-1}^p \\ & \left. u_p^{(k+1)}\right|_{x=x_p} = 0 \end{aligned} \right.$$

$$\left.\begin{aligned} & L_i u_i^{(k+1)} = f_i \quad in \quad \Omega_i \\ & -\left.\frac{du_i^{(k+1)}}{dx}\right|_{x=x_{i-1}} + \lambda_{i-1} \left. u_i^{(k+1)}\right|_{x=x_{i-1}} = g_{i-1}^i \\ & \left.\frac{du_i^{(k+1)}}{dx}\right|_{x=x_i} + \lambda_i \left. u_i^{(k+1)}\right|_{x=x_i} = g_i^i \end{aligned}\right\} i = 2, \ldots, p-1.$$

This scheme, first proposed in [39], is based on a simple relaxation technique that involves the Robin interface conditions shown above. The DE problem is solved in each subdomain where the boundary conditions are provided from the previously computed solution and its outward normal derivative from the adjacent subdomains. The *relaxation parameter* λ_i controls the influence of the value of the function and/or its normal derivative on the smoothing Robin interface conditions.

This method was first analyzed in [39] where, through energy estimates, the convergence of the method at differential level was established for arbitrary decompositions and elliptic operators. Later in [19, 32], this method was further analyzed at discrete level in a finite element framework. Several variations of this method have been also appeared. In [31] an ADI based modification is considered and analyzed at discrete level for model problems and decompositions. A second variation of **ROB** method that extends its applicability and frees it from the cross–point trouble is formulated and analyzed in [17]. In [68] the addition of

tangential derivatives in the smoothing procedure is proposed and analyzed and, recently, in [78] a finite difference variation is presented and analyzed. In some of these studies optimal values for the relaxation parameters have been obtained but only for model problems and only assuming a discrete formulation of the method (i.e., first discretize and then decompose the linear algebra problem). Therefore the determination of effective choices for λ_i's in the IR framework and for general domains and decompositions is, *in general, an open problem.*

3.2.2 The two step average AVE method.

The **AVE** [58, 60, 75, 77] IR method is a two-step iterative scheme described by the following algorithm:

1. Choose initial guesses $u_i^{(0)}(x)$ for the solution on each subdomain $\Omega_i, i = 1, 2, \ldots, p$.

2. Define the odd terms of the sequence of subdomain solutions $u_i^{(2k+1)}(x)$ as follows:

$$g_i^i = \beta_i \left.\frac{du_i^{(2k)}}{dx}\right|_{x=x_i} + (1-\beta_i) \left.\frac{du_{i+1}^{(2k)}}{dx}\right|_{x=x_i}, \quad i = 1, \ldots, p-1.$$

$$\begin{array}{l|l|l} L_1 u_1^{(2k+1)} = f_1 \text{ in } \Omega_1 & \text{for } i = 2, \ldots, p-1 & L_p u_p^{(2k+1)} = f_p \text{ in } \Omega_p \\ u_1^{(2k+1)}\Big|_{x=x_0} = 0 & L_i u_i^{(2k+1)} = f_i \text{ in } \Omega_i & \left.\frac{du_p^{(2k+1)}}{dx}\right|_{x=x_{p-1}} = g_{p-1}^{p-1} \\ \left.\frac{du_1^{(2k+1)}}{dx}\right|_{x=x_1} = g_1^1 & \left.\frac{du_i^{(2k+1)}}{dx}\right|_{x=x_{i-1}} = g_{i-1}^{i-1} & u_p^{(2k+1)}\Big|_{x=x_p} = 0 \\ & \left.\frac{du_i^{(2k+1)}}{dx}\right|_{x=x_i} = g_i^i & \end{array}$$

3. Define the even terms of the sequence of subdomain solution $u_i^{(2k+2)}(x)$ as follows:

$$h_i^i = \alpha_i \left. u_i^{(2k+1)}\right|_{x=x_i} + (1-\alpha_i) \left. u_{i+1}^{(2k+1)}\right|_{x=x_i}, \quad i = 1, \ldots, p-1.$$

$$\begin{array}{l|l|l} L_1 u_1^{(2k+2)} = f_1 \text{ in } \Omega_1 & \text{for } i = 2, \ldots, p-1 & L_p u_p^{(2k+2)} = f_p \text{ in } \Omega_p \\ u_1^{(2k+2)}\Big|_{x=x_0} = 0 & L_i u_i^{(2k+2)} = f_i \text{ in } \Omega_i & u_p^{(2k+2)}\Big|_{x=x_{p-1}} = h_{p-1}^{p-1} \\ u_1^{(2k+2)}\Big|_{x=x_1} = h_1^1 & u_i^{(2k+2)}\Big|_{x=x_{i-1}} = h_{i-1}^{i-1} & u_p^{(2k+2)}\Big|_{x=x_p} = 0 \\ & u_i^{(2k+2)}\Big|_{x=x_i} = h_i^i & \end{array}$$

The relaxation parameters α_i and β_i are to smooth the function and its normal derivative respectively and they both take values in $(0, 1)$. In the first step (odd terms), the Neumann problem is solved for each subdomain, using as estimates of the derivatives on the interface a convex combination of the normal derivatives of the initial guess (or previously computed solutions). Then a convex combination of the values of computed solutions on adjacent

domains is computed and used as boundary values to solve the Dirichlet problem in the second step (even terms). There are already several theoretical results concerning the **AVE** method. In [77], two finite element approaches (a Galerkin and a hybrid mixed) have been employed to analyze the convergence of the method at a discrete level setting both relaxation parameters equal to 1/2. A convergence analysis of the method at the differential level using Hilbert space techniques is given in [75]. A simple model problem with a two subdomain decomposition is considered in [60] where Fourier analysis at the differential level is used to obtain "good" values for the interface relaxation parameter β_1 while α_1 is set equal to 1/2.

It is worth pointing out the inherent parallelism in both the algorithms. In each one the DE solver or interface task steps can be executed on different processing elements. The only synchronization needed is a barrier at the end of each step and then only data on the interfaces need to be communicated to the processors handling neighboring subdomains. Note that parallelism and the number of subdomain are somewhat separate issues. One can apply IR to a problem with k subdomains using one, k or any number of processors in-between.

3.3 Selection of relaxation parameters

We start our analysis by stating the following simple lemma that can be easily verified.

Lemma 3.3.1. *The solution of the boundary value problem*

$$Lu = 0 \text{ in } (a,b), \quad c_1u'(a) + c_2u(a) = v_1 \text{ and } c_3u'(b) + c_4u(b) = v_2$$

with constants $c_i \in \mathbb{R}$, $i = 1,2,3,4$ *is given by*

$$\begin{aligned} u(x) = \big[&(-(c_3\gamma + c_4)e^{\gamma(b-x)} + (-c_3\gamma + c_4)e^{-\gamma(b-x)})v_1 + \\ &(-(-c_1\gamma + c_2)e^{\gamma(x-a)} + (c_1\gamma + c_2)e^{-\gamma(x-a)})v_2\big] \\ &\big[(c_1\gamma + c_2)(-c_3\gamma + c_4)e^{-\gamma(b-a)} - (c_3\gamma + c_4)(-c_1\gamma + c_2)e^{\gamma(b-a)}\big]^{-1}. \end{aligned} \tag{3.3.1}$$

Let us now introduce notation for the sequence of values of the solutions, their derivatives and their errors at the interface points: $u_{i,j}^{(k)} \equiv u_i^{(k)}(x_j)$, $du_{i,j}^{(k)} \equiv \left.\frac{du_i^{(k)}}{dx}\right|_{x=x_j}$, $\epsilon_i^{(k)}(x) \equiv u_i^{(k)}(x) - u(x)$, $\epsilon_{i,j}^{(k)} \equiv u_{i,j}^{(k)} - u(x_j)$ and $d\epsilon_{i,j}^{(k)} \equiv du_{i,j}^{(k)} - u'(x_j)$.

3.3.1 Optimum relaxation parameters for the ROB method

Consider the following differential problems associated with the error functions in each subdomain which can be easily obtained from the **ROB** algorithm given in the previous

section.

$$\begin{aligned} &L_1\epsilon_1^{(k+1)}(x) = 0, \quad x \in \Omega_1, \\ &\epsilon_{1,0}^{(k+1)} = 0, \quad d\epsilon_{1,1}^{(k+1)} + \lambda_1\epsilon_{1,1}^{(k+1)} = d\epsilon_{2,1}^{(k)} + \lambda_1\epsilon_{2,1}^{(k)}, \end{aligned} \tag{3.3.2}$$

for $i = 2, \ldots, p-1$,

$$\begin{aligned} &L_i\epsilon_i^{(k+1)}(x) = 0, \quad x \in \Omega_i, \\ &-d\epsilon_{i,i-1}^{(k+1)} + \lambda_{i-1}\epsilon_{i,i-1}^{(k+1)} = -d\epsilon_{i-1,i-1}^{(k)} + \lambda_{i-1}\epsilon_{i-1,i-1}^{(k)}, \\ &d\epsilon_{i,i}^{(k+1)} + \lambda_i\epsilon_{i,i}^{(k+1)} = d\epsilon_{i+1,i}^{(k)} + \lambda_i\epsilon_{i+1,i}^{(k)}, \end{aligned} \tag{3.3.3}$$

$$\begin{aligned} &L_p\epsilon_p^{(k+1)}(x) = 0, \quad x \in \Omega_p, \\ &-d\epsilon_{p,p-1}^{(k+1)} + \lambda_{p-1}\epsilon_{p,p-1}^{(k+1)} = -d\epsilon_{p-1,p-1}^{(k)} + \lambda_{p-1}\epsilon_{p-1,p-1}^{(k)}, \quad \epsilon_{p,p}^{(k+1)} = 0. \end{aligned} \tag{3.3.4}$$

Using (3.3.1) we observe that these error functions are given by

$$\epsilon_1^{(k+1)}(x) = \frac{-e^{\gamma_1(x-x_0)} + e^{-\gamma_1(x-x_0)}}{(-\gamma_1+\lambda_1)e^{-\gamma_1\ell_1} - (\gamma_1+\lambda_1)e^{\gamma_1\ell_1}} \left(d\epsilon_{21}^{(k)} + \lambda_1\epsilon_{21}^{(k)}\right), \tag{3.3.5}$$

for $i = 2, \ldots, p-1$,

$$\begin{aligned} \epsilon_i^{(k+1)}(x) = &\left[(-\gamma_i+\lambda_{i-1})(-\gamma_i+\lambda_i)e^{-\gamma_i\ell_i} - (\gamma_i+\lambda_i)(\gamma_i+\lambda_{i-1})e^{\gamma_i\ell_i}\right]^{-1} \\ &\Big[\left(-(\gamma_i+\lambda_i)e^{\gamma_i(x_i-x)} + (-\gamma_i+\lambda_i)e^{-\gamma_i(x_i-x)}\right)\left(-d\epsilon_{i-1,i-1}^{(k)} + \lambda_{i-1}\epsilon_{i-1,i-1}^{(k)}\right) + \\ &\left(-(\gamma_i+\lambda_{i-1})\,e^{\gamma_i(x-x_{i-1})} + \left(-\gamma_i+\lambda_{i-1}\right)e^{-\gamma_i(x-x_{i-1})}\right)\left(d\epsilon_{i+1,i}^{(k)} + \lambda_i\varepsilon_{i+1,i}^{(k)}\right)\Big] \end{aligned} \tag{3.3.6}$$

and

$$\epsilon_p^{(k+1)}(x) = \frac{-e^{\gamma_p(x_p-x)} + e^{-\gamma_p(x_p-x)}}{(-\gamma_p+\lambda_{p-1})e^{-\gamma_p\ell_p} - (\gamma_p+\lambda_{p-1})e^{\gamma_p\ell_p}} \left(-d\epsilon_{p-1,p-1}^{(k)} + \lambda_{p-1}\epsilon_{p-1,p-1}^{(k)}\right). \tag{3.3.7}$$

From these we obtain

$$\begin{aligned} \varepsilon_{1,1}^{(k+1)} &= \tfrac{m_1}{\gamma_1 n_1+\lambda_1 m_1}\left(d\varepsilon_{2,1}^{(k)} + \lambda_1\varepsilon_{2,1}^{(k)}\right), \\ \varepsilon_{i,i-1}^{(k+1)} &= \tfrac{1}{d_i}\left[(\gamma_i n_i + \lambda_i m_i)\left(-d\varepsilon_{i-1,i-1}^{(k)} + \lambda_{i-1}\varepsilon_{i-1,i-1}^{(k)}\right) + 2\gamma_i\left(d\varepsilon_{i+1,i}^{(k)} + \lambda_i\varepsilon_{i+1,i}^{(k)}\right)\right] \\ \varepsilon_{i,i}^{(k+1)} &= \tfrac{1}{d_i}\left[(\gamma_i n_i + \lambda_{i-1} m_i)\left(d\varepsilon_{i+1,i}^{(k)} + \lambda_i\varepsilon_{i+1,i}^{(k)}\right) + 2\gamma_i\left(-d\varepsilon_{i-1,i-1}^{(k)} + \lambda_{i-1}\varepsilon_{i-1,i-1}^{(k)}\right)\right], \\ \varepsilon_{p,p-1}^{(k+1)} &= \tfrac{m_p}{\gamma_p n_p+\lambda_{p-1} m_p}\left(-d\varepsilon_{p-1,p-1}^{(k)} + \lambda_{p-1}\varepsilon_{p-1,p-1}^{(k)}\right) \end{aligned}$$

where

$$\begin{aligned} &d_i = (\gamma_i^2 + \lambda_i\lambda_{i-1})m_i + \gamma_i(\lambda_i+\lambda_{i-1})n_i, \quad i = 2, \ldots, p-1, \\ &n_i = e^{\gamma_i\ell_i} - e^{-\gamma_i\ell_i} \quad \text{and} \quad m_i = e^{\gamma_i\ell_i} + e^{-\gamma_i\ell_i}, \quad i = 1, \ldots, p. \end{aligned}$$

By differentiating equations (3.3.5)–(3.3.7) we obtain expressions similar to the above that relate $d\varepsilon_{1,1}^{(k+1)}$, $d\varepsilon_{i,i-1}^{(k+1)}$, $d\varepsilon_{i,i}^{(k+1)}$, $i = 2, \ldots, p-1$, and $d\varepsilon_{p,p-1}^{(k+1)}$ with associated values from the iteration (k); these are

$$\begin{aligned}
d\varepsilon_{1,1}^{(k+1)} &= \tfrac{\gamma_1 n_1}{\gamma_1 n_1 + \lambda_1 m_1} \left(d\varepsilon_{2,1}^{(k)} + \lambda_1 \varepsilon_{2,1}^{(k)} \right), \\
d\varepsilon_{i,i-1}^{(k+1)} &= \tfrac{\gamma_i}{d_i} \left[(\gamma_i m_i + \lambda_i n_i) \left(d\varepsilon_{i-1,i-1}^{(k)} - \lambda_{i-1} \varepsilon_{i-1,i-1}^{(k)} \right) + 2\lambda_{i-1} \left(d\varepsilon_{i+1,i}^{(k)} + \lambda_i \varepsilon_{i+1,i}^{(k)} \right) \right], \\
d\varepsilon_{i,i}^{(k+1)} &= \tfrac{\gamma_i}{d_i} \left[(\gamma_i m_i + \lambda_{i-1} n_i) \left(d\varepsilon_{i+1,i}^{(k)} + \lambda_i \varepsilon_{i+1,i}^{(k)} \right) + 2\lambda_i \left(d\varepsilon_{i-1,i-1}^{(k)} - \lambda_{i-1} \varepsilon_{i-1,i-1}^{(k)} \right) \right], \\
d\varepsilon_{p,p-1}^{(k+1)} &= \tfrac{\gamma_p n_p}{\gamma_p n_p + \lambda_{p-1} m_p} \left(d\varepsilon_{p-1,p-1}^{(k)} - \lambda_{p-1} \varepsilon_{p-1,p-1}^{(k)} \right).
\end{aligned}$$

Now we order the errors on the interface points to create a sequence of error vectors as follows, for $k = 0, 1, 2, \ldots,$ we set

$$\begin{aligned}
\underline{\varepsilon}^{(k)} \equiv \Big[& d\varepsilon_{1,1}^{(k)}, \varepsilon_{1,1}^{(k)}, \varepsilon_{2,1}^{(k)}, d\varepsilon_{2,1}^{(k)}, d\varepsilon_{2,2}^{(k)}, \varepsilon_{2,2}^{(k)}, \varepsilon_{3,2}^{(k)}, d\varepsilon_{3,2}^{(k)}, \ldots, \\
& d\varepsilon_{i,i}^{(k)}, \varepsilon_{i,i}^{(k)}, \varepsilon_{i+1,i}^{(k)}, d\varepsilon_{i+1,i}^{(k)}, \ldots, d\varepsilon_{p-1,p-1}^{(k)}, \varepsilon_{p-1,p-1}^{(k)}, \varepsilon_{p,p-1}^{(k)}, d\varepsilon_{p,p-1}^{(k)} \Big]^T
\end{aligned}$$

We obtain the following relation between the vectors of interface errors in the two consecutive iteration steps (k) and $(k+1)$

$$\underline{\epsilon}^{(k+1)} = M \underline{\epsilon}^{(k)}, \quad k = 0, 1, \ldots, \tag{3.3.8}$$

where the iteration matrix $M \in \mathbb{R}^{4(p-1)\times 4(p-1)}$ has the form

$$M = \begin{bmatrix}
0 & M_{1,2} & 0 & 0 & 0 & 0 & \cdots & 0 \\
M_{2,1} & 0 & 0 & M_{2,4} & 0 & 0 & \cdots & 0 \\
M_{3,1} & 0 & 0 & M_{3,4} & 0 & 0 & \cdots & 0 \\
0 & 0 & M_{4,3} & 0 & 0 & M_{4,6} & \cdots & 0 \\
0 & 0 & M_{5,3} & 0 & 0 & M_{5,6} & \cdots & 0 \\
\vdots & \vdots & & \ddots & \ddots & \ddots & & \vdots \\
0 & 0 & \cdots & 0 & M_{2(p-1)-2,2(p-1)-3} & 0 & 0 & M_{2(p-1)-2,2(p-1)} \\
0 & 0 & \cdots & 0 & M_{2(p-1)-1,2(p-1)-3} & 0 & 0 & M_{2(p-1)-1,2(p-1)} \\
0 & 0 & \cdots & 0 & 0 & 0 & M_{2(p-1),2(p-1)-1} & 0
\end{bmatrix} \tag{3.3.9}$$

The submatrices of M are as follows:

$$M_{1,2} = \frac{1}{\gamma_1 n_1 + \lambda_1 m_1} \begin{bmatrix} \gamma_1 n_1 \lambda_1 & \gamma n_1 \\ m_1 \lambda_1 & m_1 \end{bmatrix},$$

for $i = 2, \ldots, p-1$,

$$M_{2(i-1),2(i-1)-1} = \frac{1}{d_i} \begin{bmatrix} -(\gamma_i n_i + \lambda_i m_i) & \lambda_{i-1}(\gamma_i n_i + \lambda_i m_i) \\ \gamma_i(\gamma_i m_i + \lambda_i n_i) & -\gamma_i \lambda_{i-1}(\gamma_i m_i + \lambda_i n_i) \end{bmatrix},$$

$$M_{2(i-1)+1,2(i-1)+2} = \frac{1}{d_i}\begin{bmatrix} \gamma_i\lambda_i(\gamma_i m_i + \lambda_{i-1}n_i) & \gamma_i(\gamma_i m_i + \lambda_{i-1}n_i) \\ \lambda_i(\gamma_i n_i + \lambda_{i-1}m_i) & (\gamma_i n_i + \lambda_{i-1}m_i) \end{bmatrix},$$

$$M_{2(i-1),2(i-1)+2} = \frac{2\gamma_i}{d_i}\begin{bmatrix} \lambda_i & 1 \\ \lambda_i\lambda_{i-1} & \lambda_{i-1} \end{bmatrix}, \quad M_{2(i-1)+1,2(i-1)-1} = \frac{2\gamma_i}{d_i}\begin{bmatrix} \lambda_i & -\lambda_i\lambda_{i-1} \\ -1 & \lambda_{i-1} \end{bmatrix},$$

and

$$M_{2(p-1),2(p-1)-1} = \frac{1}{\gamma_p n_p + \lambda_{p-1}m_p}\begin{bmatrix} -m_p & \lambda_{p-1}m_p \\ \gamma_p n_p & -\gamma_p n_p\lambda_{p-1} \end{bmatrix}.$$

For the rest of the analysis in this section we use a methodology similar to the one found in [35]. In the following lemma we construct a matrix $\tilde{M} \in \mathbb{R}^{2(p-1)\times 2(p-1)}$ of reduced size which is spectrally equivalent to the iteration matrix M and whose special non-zero structure lets us select optimum values for the relaxation parameters λ_i.

Lemma 3.3.2. *The two matrices M and $\tilde{M}$ have the same non-zero eigenvalues, i.e.,*

$$\sigma(M)\backslash\{0\} = \sigma(\tilde{M})\backslash\{0\} \tag{3.3.10}$$

where

$$\tilde{M} = \begin{bmatrix} 0 & \tilde{M}_{1,2} & 0 & 0 & 0 & 0 & \cdots & 0 \\ \tilde{M}_{2,1} & 0 & 0 & \tilde{M}_{2,4} & 0 & 0 & \cdots & 0 \\ \tilde{M}_{3,1} & 0 & 0 & \tilde{M}_{3,4} & 0 & 0 & \cdots & 0 \\ 0 & 0 & \tilde{M}_{4,3} & 0 & 0 & \tilde{M}_{4,6} & \cdots & 0 \\ 0 & 0 & \tilde{M}_{5,3} & 0 & 0 & \tilde{M}_{5,6} & \cdots & 0 \\ \vdots & \vdots & & \ddots & \ddots & \ddots & & \vdots \\ 0 & 0 & \cdots & 0 & \tilde{M}_{2(p-1)-2,2(p-1)-3} & 0 & 0 & \tilde{M}_{2(p-1)-2,2(p-1)} \\ 0 & 0 & \cdots & 0 & \tilde{M}_{2(p-1)-1,2(p-1)-3} & 0 & 0 & \tilde{M}_{2(p-1)-1,2(p-1)} \\ 0 & 0 & \cdots & 0 & 0 & 0 & \tilde{M}_{2(p-1),2(p-1)-1} & 0 \end{bmatrix}. \tag{3.3.11}$$

The elements of $\tilde{M}$ are defined as follows:

$$\tilde{M}_{1,2} = \frac{-\gamma_1 n_1 + \lambda_1 m_1}{\gamma_1 n_1 + \lambda_1 m_1},$$

for $i = 2, \ldots, p-1$,

$$\tilde{M}_{2(i-1),2(i-1)-1} = \frac{\lambda_{i-1}(\gamma_i n_i + \lambda_i m_i) - \gamma_i(\gamma_i m_i + \lambda_i n_i)}{d_i},$$

$$\tilde{M}_{2(i-1)+1,2(i-1)+2} = \frac{\lambda_i(\gamma_i n_i + \lambda_{i-1} m_i) - \gamma_i(\gamma_i m_i + \lambda_{i-1} n_i)}{d_i},$$

$$\tilde{M}_{2(i-1),2(i-1)+2} = \frac{4\gamma_i\lambda_{i-1}}{d_i}, \quad \tilde{M}_{2(i-1)+1,2(i-1)-1} = \frac{4\gamma_i\lambda_i}{d_i},$$

and

$$\tilde{M}_{2(p-1),2(p-1)-1} = \frac{-\gamma_p n_p + \lambda_{p-1} m_p}{\gamma_p n_p + \lambda_{p-1} m_p}.$$

Proof: We define the non-singular matrix

$$Q = diag(\, Q_1, Q_1^T, Q_2, Q_2^T, \cdots, Q_{p-1}, Q_{p-1}^T \,),$$

where

$$Q_i = Q_i^{-1} = \begin{bmatrix} 1 & -\lambda_i \\ 0 & -1 \end{bmatrix}, \quad Q_i^T = Q_i^{-T} = \begin{bmatrix} 1 & 0 \\ -\lambda_i & -1 \end{bmatrix}, \quad i = 1, \ldots, p-1,$$

and consider the similarity transformation matrix $Q^{-1}MQ$ whose submatrices are specified by the following relations.

$$Q_1^{-1} M_{1,2} Q_1^T = \begin{bmatrix} 0 & \frac{-\gamma_1 n_1 + \lambda_1 m_1}{\gamma_1 n_1 + \lambda_1 m_1} \\ 0 & \frac{m_1}{\gamma_1 n_1 + \lambda_1 m_1} \end{bmatrix},$$

for $i = 2, \ldots, p-1$,

$$Q_{i-1}^{-T} M_{2(i-1),2(i-1)-1} Q_{i-1} = \frac{1}{d_i} \begin{bmatrix} -(\gamma_i n_i + \lambda_i m_i) & 0 \\ \lambda_{i-1}(\gamma_i n_i + \lambda_i m_i) - \gamma_i(\gamma_i m_i + \lambda_i n_i) & 0 \end{bmatrix},$$

$$Q_{i-1}^{-T} M_{2(i-1),2(i-1)+2} Q_i^T = \frac{2\gamma_i}{d_i} \begin{bmatrix} 0 & -1 \\ 0 & 2\lambda_{i-1} \end{bmatrix},$$

$$Q_i^{-1} M_{2(i-1)+1,2(i-1)-1} Q_{i-1} = \frac{2\gamma_i}{d_i} \begin{bmatrix} 2\lambda_i & 0 \\ 1 & 0 \end{bmatrix},$$

$$Q_i^{-1} M_{2(i-1)+1,2(i-1)+2} Q_i^T = \frac{1}{d_i} \begin{bmatrix} 0 & -\gamma_i(\gamma_i m_i + \lambda_{i-1} n_i) + \lambda_i(\gamma_i n_i + \lambda_{i-1} m_i) \\ 0 & \gamma_i p_i + \lambda_{i-1} m_i \end{bmatrix},$$

and

$$Q_{p-1}^{-T} M_{2(p-1),2(p-1)-1} Q_{p-1} = \frac{1}{\gamma_p n_p + \lambda_{p-1} m_p} \begin{bmatrix} -m_p & 0 \\ m_p \lambda_{p-1} - \gamma_p n_p & 0 \end{bmatrix}.$$

A simple comparison of the above relations with the elements of the matrix $\tilde{M}$ and the fact that there exists (Lemma 3.2 in [35]) a permutation matrix P such that

$$P^T Q^T M Q P = \begin{bmatrix} 0 & * \\ 0 & \tilde{M} \end{bmatrix},$$

complete the proof of the lemma. □

We conclude this section with the main theorem that presents analytic expressions for the relaxation parameters.

Theorem 3.3.3. *Consider the model problem (1) and a non-overlapping decomposition of Ω into p subdomains Ω_i of length ℓ_i, $i = 1, \ldots, p$. If the parameters λ_i involved in the **ROB** interface relaxation method are selected as*

$$\lambda_{p-1} = \frac{\gamma_p n_p}{m_p}, \qquad \lambda_{i-1} = \frac{\gamma_i(\gamma_i m_i + \lambda_i n_i)}{\gamma_i n_i + \lambda_i m_i}, \quad i = p-1, \ldots, 2, \tag{3.3.12}$$

then the spectral radius of the iteration matrix M is zero.

Proof: It can been seen (Lemma 3.2 in [35]) that if we set $\tilde{M}_{2(i-1),2(i-1)-1} = 0, i = 2, \ldots, p$, then we obtain that $\sigma(\tilde{M}) = 0$. This leads to the following equations.

$$\lambda_{p-1} m_p - \gamma_p n_p = 0$$

and

$$\lambda_{i-1}(\gamma_i n_i + \lambda_i m_i) - \gamma_i(\gamma_i m_i + \lambda_i n_i) = 0, \quad i = 2, \ldots, p-1. \tag{3.3.13}$$

To conclude the proof, we back solve for $\lambda_i, i = p-1, \ldots, 1$ and use the previous Lemma. □

3.3.2 "Optimum" relaxation parameters for the AVE method

Using the notation adopted in the previous section and the **AVE** algorithm given in Section 2 we easily see that the error functions involved satisfy the following differential equations: For the odd steps the equation for the first subdomain is

$$\begin{aligned} L_1 \epsilon_1^{(2k+1)}(x) &= 0 \; x \in \Omega_1, \\ \epsilon_{1,0}^{(2k+1)} &= 0, \quad d\epsilon_{1,1}^{(2k+1)} = \beta_1 d\epsilon_{1,1}^{(2k)} + (1-\beta_1) d\epsilon_{2,1}^{(2k)}, \end{aligned} \tag{3.3.14}$$

for the ith interior subdomain, $i = 2, \ldots, p-1$, the equation is

$$\begin{aligned} L_i \epsilon_i^{(2k+1)}(x) &= 0 \; x \in \Omega_i, \\ d\epsilon_{i,i-1}^{(2k+1)} &= \beta_{i-1} d\epsilon_{i-1,i-1}^{(2k)} + (1-\beta_{i-1}) d\epsilon_{i,i-1}^{(2k)}, \\ d\epsilon_{i,i}^{(2k+1)} &= \beta_i d\epsilon_{i,i}^{(2k)} + (1-\beta_i) d\epsilon_{i+1,i}^{(2k)}, \end{aligned} \tag{3.3.15}$$

and for the last subdomain the equation is

$$\begin{aligned} L_p \epsilon_p^{(2k+1)}(x) &= 0 \; x \in \Omega_p, \\ d\epsilon_{p,p-1}^{(2k+1)} &= \beta_{p-1} d\epsilon_{p-1,p-1}^{(2k)} + (1-\beta_{p-1}) d\epsilon_{p,p-1}^{(2k)}, \\ \epsilon_{p,p}^{(2k+1)} &= 0. \end{aligned} \tag{3.3.16}$$

For the even steps the equation for the first subdomain is

$$\begin{aligned} L_1\epsilon_1^{(2k+2)}(x) &= 0 \;\; x \in \Omega_1, \\ \epsilon_{1,0}^{(2k+2)} &= 0, \quad \epsilon_{1,1}^{(2k+2)} = \alpha_1\epsilon_{1,1}^{(2k+1)} + (1-\alpha_1)\epsilon_{2,1}^{(2k+1)}, \end{aligned} \tag{3.3.17}$$

for the ith interior subdomain, $i = 2, \ldots, p-1$, the equation is

$$\begin{aligned} L_i\epsilon_i^{(2k+2)}(x) &= 0 \;\; x \in \Omega_i, \\ \epsilon_{i,i-1}^{(2k+2)} &= \alpha_{i-1}\epsilon_{i-1,i-1}^{(2k+1)} + (1-\alpha_{i-1})\epsilon_{i,i-1}^{(2k+1)}, \\ \epsilon_{i,i}^{(2k+2)} &= \alpha_i\epsilon_{i,i}^{(2k+1)} + (1-\alpha_i)\epsilon_{i+1,i}^{(2k+1)}, \end{aligned} \tag{3.3.18}$$

and for the last subdomain the equation is

$$\begin{aligned} L_p\epsilon_p^{(2k+2)}(x) &= 0 \;\; x \in \Omega_p, \\ \epsilon_{p,p-1}^{(2k+2)} &= \alpha_{p-1}\epsilon_{p-1,p-1}^{(2k+1)} + (1-\alpha_{p-1})\epsilon_{p,p-1}^{(2k+1)}, \quad \epsilon_{p,p}^{(2k+2)} = 0. \end{aligned} \tag{3.3.19}$$

The solutions to the Neumann problems (3.3.14)–(3.3.16) are given by (see Lemma 3.3.1)

$$\begin{aligned} \epsilon_1^{(2k+1)}(x) &= \tfrac{1}{\gamma_1 m_1}\left(e^{\gamma_1(x-x_0)} - e^{-\gamma_1(x-x_0)}\right)\left(\beta_1 d\epsilon_{1,1}^{(2k)} + (1-\beta_1)d\epsilon_{2,1}^{(2k)}\right), \\ \epsilon_i^{(2k+1)}(x) &= \tfrac{1}{\gamma_i n_i}\left\{\left(-e^{\gamma_i(x_i-x)} - e^{-\gamma_i(x_i-x)}\right)\left(\beta_{i-1}d\epsilon_{i-1,i-1}^{(2k)} + (1-\beta_{i-1})d\epsilon_{i,i-1}^{(2k)}\right)\right. \\ &\quad + \left.\left(e^{\gamma_i(x-x_{i-1})} + e^{-\gamma_i(x-x_{i-1})}\right)\left(\beta_i d\epsilon_{i,i}^{(2k)} + (1-\beta_i)d\epsilon_{i+1,i}^{(2k)}\right)\right\}, \quad i = 2, \ldots, p-1 \\ \epsilon_p^{(2k+1)}(x) &= \tfrac{1}{\gamma_p m_p}\left(-e^{\gamma_p(x_p-x)} + e^{-\gamma_p(x_p-x)}\right)\left(\beta_{p-1}d\epsilon_{p-1,p-1}^{(2k)} + (1-\beta_{p-1})d\epsilon_{p,p-1}^{(2k)}\right). \end{aligned}$$

The solutions to the Dirichlet problems (3.3.17)–(3.3.19) are given by

$$\begin{aligned} \epsilon_1^{(2k+2)}(x) &= \tfrac{1}{n_i}\left(e^{\gamma_1(x-x_0)} - e^{-\gamma_1(x-x_0)}\right)\left(\alpha_1\epsilon_{1,1}^{(2k+1)} + (1-\alpha_1)\epsilon_{2,1}^{(2k+1)}\right), \\ \epsilon_i^{(2k+2)}(x) &= \tfrac{1}{n_i}\left\{\left(e^{\gamma_i(x_i-x)} - e^{-\gamma_i(x_i-x)}\right)\left(\alpha_{i-1}\epsilon_{i-1,i-1}^{(2k+1)} + (1-\alpha_{i-1})\epsilon_{i,i-1}^{(2k+1)}\right) + \right. \\ &\quad \left.\left(e^{\gamma_i(x-x_{i-1})} - e^{-\gamma_i(x-x_{i-1})}\right)\left(\alpha_i\epsilon_{i,i}^{(2k+1)} + (1-\alpha_i)\epsilon_{i+1,i}^{(2k+1)}\right)\right\}, \quad i = 2, \ldots, p-1 \\ \epsilon_p^{(2k+2)}(x) &= \tfrac{1}{n_p}\left(e^{\gamma_p(x_p-x)} - e^{-\gamma_p(x_p-x)}\right)\left(\alpha_{p-1}\epsilon_{p-1,p-1}^{(2k+1)} + (1-\alpha_{p-1})\epsilon_{p,p-1}^{(2k+1)}\right). \end{aligned}$$

If, for $k = 0, 1, \ldots$, we define the vectors

$$\underline{\varepsilon}^{(k)} \equiv \left[\varepsilon_{1,1}^{(k)}, \varepsilon_{2,2}^{(k)}, \ldots, \varepsilon_{p-1,p-1}^{(k)}\right]^T \quad \text{and} \quad \underline{d\varepsilon}^{(k)} \equiv \left[d\varepsilon_{1,1}^{(k)}, d\varepsilon_{2,2}^{(k)}, \ldots, d\varepsilon_{p-1,p-1}^{(k)}\right]^T \tag{3.3.20}$$

then we get from the above that

$$\underline{\epsilon}^{(2k+2)} = M^D \underline{d\epsilon}^{(2k+1)} \tag{3.3.21}$$

$$\underline{d\epsilon}^{(2k+1)} = M^N \underline{\epsilon}^{(2k)} \tag{3.3.22}$$

where the Dirichlet and Neumann iteration matrices $M^D, M^N \in \mathbb{R}^{(p-1)\times(p-1)}$ are tridiagonal with elements

$$\begin{aligned}
&M^D_{1,1} = \tfrac{\alpha_1 m_1}{n_1\gamma_1} - \tfrac{(1-\alpha_1)n_2}{m_2\gamma_2}, \quad M^D_{p-1,p-1} = \tfrac{\alpha_{p-1}n_{p-1}}{m_{p-1}\gamma_{p-1}} - \tfrac{(1-\alpha_{p-1})m_p}{n_p\gamma_p},\\
&M^D_{i,i} = \tfrac{\alpha_i n_i}{m_i\gamma_i} - \tfrac{(1-\alpha_i)n_{i+1}}{m_{i+1}\gamma_{i+1}}, \quad i = 2,\ldots,p-2,\\
&M^D_{i,i+1} = \tfrac{2(1-\alpha_i)}{m_{i+1}\gamma_{i+1}}, \quad i = 2,\ldots,p-1,\\
&M^D_{i+1,i} = -\tfrac{2\alpha_i}{m_i\gamma_i}, \quad i = 1,\ldots,p-2,
\end{aligned} \tag{3.3.23}$$

$$\begin{aligned}
&M^N_{1,1} = \tfrac{\beta_1 n_1\gamma_1}{m_1} - \tfrac{(1-\beta_1)n_2\gamma_2}{m_2}, \quad M^N_{p-1,p-1} = \tfrac{\beta_{p-1}n_{p-1}\gamma_{p-1}}{m_{p-1}} - \tfrac{(1-\beta_{p-1})n_p\gamma_p}{m_p},\\
&M^N_{i,i} = \tfrac{\beta_i n_i\gamma_i}{m_i} - \tfrac{(1-\beta_i)n_{i+1}\gamma_{i+1}}{m_{i+1}}, \quad i = 2,\ldots,p-2,\\
&M^N_{i,i+1} = \tfrac{2(1-\beta_i)\gamma_{i+1}}{m_{i+1}}, \quad i = 2,\ldots,p-1,\\
&M^N_{i+1,i} = -\tfrac{2\beta_i\gamma_i}{m_i}, \quad i = 1,\ldots,p-2.
\end{aligned} \tag{3.3.24}$$

For $p = 2$ it is easy to see (make the roots of the characteristic polynomial of M^D or M^N be zero) that $\alpha_1 = \frac{m_2 n_1\gamma_1}{m_2 n_1\gamma_1 + m_1 n_2\gamma_2}$ or $\beta_1 = \frac{m_1 n_2\gamma_2}{m_1 n_2\gamma_2 + m_2 n_1\gamma_1}$ are optimum values and achieve immediate convergence. For $p > 2$ we have been unable to derive optimum values for the relaxation parameters. Instead we obtain values for them that are optimum in the max-norm (see Appendix C).

Theorem 3.3.4. *Consider the iteration matrix $M \equiv M^N M^D$ of the **AVE** method associated with the model problem (3.2.1) and a non-overlapping decomposition of Ω into p subdomains Ω_i of length $\ell_i, i = 1,\ldots,p$ with $\gamma_i = \gamma$ in $i = 1,\ldots,p$. For the values the relaxation parameters given below the max-norms of M^N and M^D are minimized and the matrix M is a contraction mapping, with respect to the max-norm.*

$$\alpha_1 = \tfrac{n_1 n_2}{n_1 n_2 + m_1 m_2}, \qquad \alpha_{p-1} = \tfrac{m_p m_{p-1}}{m_p m_{p-1} + n_p n_{p-1}}, \tag{3.3.25}$$

$$\alpha_i = \tfrac{m_i n_{i+1}}{m_i n_{i+1} + m_{i+1} n_i}, \qquad i = 2,\ldots,p-2 \tag{3.3.26}$$

and

$$\beta_1 = \tfrac{m_1 n_2}{m_1 n_2 + m_2 n_1}, \qquad \beta_{p-1} = \tfrac{m_{p-1} n_p}{m_{p-1} n_p + m_p n_{p-1}} \tag{3.3.27}$$

$$\beta_i = \tfrac{m_i n_{i+1}}{m_i n_{i+1} + m_{i+1} n_i}, i = 2,\ldots,p-2, \tag{3.3.28}$$

provided that $\ell_i > \frac{ln(1+\sqrt{2})}{\gamma}, \quad i = 1,\ldots,p.$

Proof: To minimize the max-norm of the iteration matrix, it is sufficient to minimize the quantity

$$
\begin{aligned}
&f(\alpha_i,\beta_i,\beta_{i-1},\beta_{i+1}) = 4\tfrac{\alpha_i\beta_{i-1}}{m_i m_{i-1}} + \\
&\tfrac{2}{m_i}\left|\alpha_i\tfrac{\beta_{i-1}(m_i n_{i-1}+m_{i-1}n_i)-m_{i-1}n_i}{m_{i-1}m_i} + \beta_i\tfrac{\alpha_i(m_{i+1}n_i+m_i n_{i+1})-m_i n_{i+1}}{m_i m_{i+1}}\right| + \\
&\left|-4\tfrac{\alpha_i(1-\beta_{i-1})}{m_i^2} + \tfrac{\alpha_i(m_{i+1}n_i+m_i n_{i+1})-m_i n_{i+1}}{m_i m_{i+1}}\tfrac{\beta_i(m_{i+1}n_i+m_i n_{i+1})-m_i n_{i+1}}{m_i m_{i+1}} - \right. \\
&\qquad \left. 4\tfrac{(1-\alpha_i)\beta_{i+1}}{m_{i+1}^2}\right| + \\
&\tfrac{2}{m_{i+1}}\left|\tfrac{\alpha_i(m_{i+1}n_i+m_i n_{i+1})-m_i n_{i+1}}{m_i m_{i+1}}(1-\beta_i) + \tfrac{\beta_{i+1}(m_{i+2}n_{i+1}+m_{i+1}n_{i+2})-m_{i+1}n_{i+2}}{m_{i+1}m_{i+2}}(1-\alpha_i)\right| + \\
&4\tfrac{(1-\alpha_i)(1-\beta_{i+1})}{m_i m_{i+1}}.
\end{aligned}
\tag{3.3.29}
$$

One can determine values for $\alpha_i, \beta_i, \beta_{i-1}$ and β_{i+1} that minimize f by an elementary but very lengthy and tedious analysis which involves splitting the absolute values and considering several different cases. (This analysis is presented in Appendix C.) Here instead, we give an indication why this theorem is true.

Set $\alpha_i = \alpha_i^* \equiv \frac{m_i n_{i+1}}{m_{i+1}n_i+m_i n_{i+1}}$, $\beta_i = \beta_i^* \equiv \frac{m_i n_{i+1}}{m_{i+1}n_i+m_i n_{i+1}}$, $\beta_{i-1} = \beta_{i-1}^* \equiv \frac{m_{i-1}n_i}{m_{i-1}n_i+m_i n_{i-1}}$ and $\beta_{i+1} = \beta_{i+1}^* \equiv \frac{m_{i+1}n_{i+2}}{m_{i+1}n_{i+2}+m_{i+2}n_{i+1}}$. Then the expressions in the absolute values of (3.3.29) become zero and so we have

$$
f(\alpha_i^*,\beta_i^*,\beta_{i-1}^*,\beta_{i+1}^*) = \frac{4}{m_{i+1}n_i+m_i n_{i+1}}\left(\frac{m_{i+1}(m_i+m_{i-1})}{m_{i-1}n_i+m_i n_{i-1}} + \frac{m_i(m_{i+1}+m_{i+2})}{m_{i+1}n_{i+2}+m_{i+2}n_{i+1}}\right).
$$

Under the constraint that $\ell_j > \frac{\ln(1+\sqrt{2})}{\gamma}$ we have that $n_j > 2, \quad j = i-1, i, i+1$, and therefore we have

$$
f(\alpha_i^*,\beta_i^*,\beta_{i-1}^*,\beta_{i+1}^*) < \frac{4}{2(m_i+m_{i+1})}\left(\frac{m_{i+1}(m_i+m_{i-1})}{2(m_i+m_{i-1})} + \frac{m_i(m_{i+1}+m_{i+2})}{2(m_{i+2}+m_{i+1})}\right) \quad = 1.
$$

Continuing in the same way for the $1^{st}, 2^{nd}, (p-2)^{th}$ and $(p-1)^{th}$ rows of the iteration matrix, we get optimum values for the relaxation parameters for all the interface points. □

3.4 Numerical experiments

The purpose of the numerical experiments performed in this study is two-fold. First to verify and elucidate our theoretically determined relaxation parameter values on a class of one dimensional problems and then to examine how effective these parameters values are for two dimensional PDEs. All experiments reported here are performed in single precision on SUN workstations.

3.4.1 One dimensional case

We have implemented the two IR methods considered in this Chapter using MATLAB. All MATLAB files we use to produce the one dimensional data in this section are available through our web page.[1] Implementations of several other relaxation schemes also can be found there. We use zero as initial guess and consider the following model problem:

$$Lu \equiv -u''(x) + \gamma^2 u(x) = f(x),\ x \in (0,1), \quad u(0) = 0,\ u(1) = 0, \tag{3.4.1}$$

where the right hand side function f is selected such that the true solution $u(x)$ is either

DP1 $u(x) = \cosh(2\,x - 1) - \cosh(1.0)$, or

DP2 $u(x) = e^{x+4}x(x-1)(x-.7)$.

In Table 3.1 we present the max norm of the error $||u^{(k)} - u||_\infty$ and the computed convergence factor

$$r_k = \sqrt[k]{||Lu^{(k)} - f||_\infty / ||Lu^{(0)} - f||_\infty}, k = 1, 2, \ldots$$

of the **ROB** method applied to the model problem (3.4.1) with $\gamma^2 = 2$ and solution DP1. We assume that the domain is decomposed into $p = 2, 4, 10, 20$ domains of equal size. We use the 5-point star difference approximation with two different global discretization steps $h = .01$ and $h = .005$ to solve the DE. Similarly in Table 3.2 we consider the **AVE** method and set $\gamma^2 = 10$. The rapid rate of convergence is easily observed as one moves down along any column. Note that this convergence is not immediate (1 iteration) as our theory might indicate. It can be shown [69] that this is mainly due to the particular block structure of the Jordan form of the iteration matrices which may require from 1 to $2(p-1)$ iteration steps instead of one.

It can be also observed that, as the computed convergence factors indicate, the rate of convergence of both methods does not seem to depend on the fineness of the domain discretization. Nevertheless, the order h^2 finite difference discretization convergence rate is preserved. The rate of convergence does depend, as expected, on both the number of subdomains and the coefficient γ^2. Extensive numerical experiments (some of them presented in Figure 3.3 below, and some others that are not included in this Chapter) show that the rate of convergence increases as γ^2 increases for both methods but much more rapidly in the **AVE** case. The **AVE** method diverges for $\gamma^2 = 10$ and $p = 20$. This is in good agreement with the restriction $\ell_i > \frac{ln(1+\sqrt{2})}{\gamma}$, $i = 1, \ldots, p$ imposed by Theorem 3.3.4. This restriction seems to be necessary as well as sufficient (see also our discussion below of the figures).

	h=.01				h=.005			
iter	p=2	p=4	p=10	p=20	p=2	p=4	p=10	p=20
2	3.08E-5 (.2966)	1.48E-1 (.2997)	2.74E-1 (.2424)	3.87E-1 (.1811)	7.83E-6 (.2966)	1.48E-1 (.2992)	2.74E-1 (.2421)	3.87E-1 (.1809)
3	1.19E-5 (.4447)	7.18E-2 (.4635)	1.72E-1 (.4409)	3.00E-1 (.3723)	3.07E-6 (.4447)	7.18E-2 (.4635)	1.72E-1 (.4408)	3.00E-1 (.3721
4	1.19E-5 (.5446)	3.18E-2 (.5524)	1.42E-1 (.5651)	2.27E-1 (.5106)	3.07E-6 (.5446)	3.18E-2 (.5524)	1.42E-1 (.5651)	2.27E-1 (.5105)
5	1.19E-5 (.6150)	1.41E-2 (.6167)	1.53E-1 (.6425)	1.75E-1 (.6063)	3.07E-6 (.6150)	1.41E-2 (.6166)	1.53E-1 (.6425)	1.75E-1 (.6063)
8	1.19E-5 (.7379)	6.11E-5 (.7379)	7.25E-2 (.7494)	1.64E-1 (.7576)	3.07E-6 (.7379)	1.52E-5 (.7379)	7.24E-2 (.7494)	1.64E-1 (.7576)
16	1.19E-5 (.8590)	6.14E-5 (.8590)	2.04E-3 (.8590)	7.43E-2 (.8658)	3.07E-6 (.8590)	1.52E-5 (.8590)	2.05E-3 (.8590)	7.41E-2 (.8658)
20	1.19E-5 (.8855)	6.14E-5 (.8855)	2.13E-4 (.8855)	3.99E-2 (.8883)	3.07E-6 (.8855)	1.52E-5 (.8855)	5.28E-5 (.8855)	3.98E-2 (.8883)
32	1.19E-5 (.9268)	6.14E-5 (.9268)	2.14E-4 (.9268)	2.23E-3 (.9268)	3.07E-6 (.9268)	1.52E-5 (.9268)	5.32E-5 (.9268)	2.18E-3 (.9268)
36	1.19E-5 (.9347)	6.14E-5 (.9347)	2.14E-4 (.9347)	4.72E-4 (.9347)	3.07E-6 (.9347)	1.52E-5 (.9347)	5.32E-5 (.9347)	2.54E-4 (.9347)

Table 3.1: The max norm of the error and the computed values of the convergence factor of the **ROB** method applied to model problem (3.4.1)-DP1 ($\gamma^2 = 2$). In the first column we have the iteration number, in the first row the discretization step-size and in the second row the number of equal subdomains.

	h=.01				h=.005			
iter	p=2	p=4	p=10	p=20	p=2	p=4	p=10	p=20
2	1.39E-6 (.0965)	2.32E-4 (.0965)	1.43E-2 (.0966)	1.03E-1 (.0966)	3.48E-7 (.0965)	2.34E-4 (.0965)	1.43E-2 (.0966)	1.04E-1 (.0966)
3	1.39E-6 (.2103)	5.23E-6 (.2104)	7.18E-3 (.2097)	2.23E-1 (.2104)	3.48E-7 (.2103)	3.71E-6 (.2104)	7.23E-3 (.2097)	2.25E-1 (.2104)
4	1.39E-6 (.3106)	1.99E-6 (.3106)	4.58E-3 (.3113)	5.39E-1 (.3419)	3.48E-7 (.3106)	4.65E-7 (.3106)	4.62E-3 (.3113)	5.44E-1 (.3423)
5	1.39E-6 (.3924)	2.04E-6 (.3924)	3.15E-3 (.3920)	1.41E+0 (.4934)	3.48E-7 (.3924)	5.10E-7 (.3924)	3.19E-3 (.3920)	1.43E+0 (.4945)
8	1.39E-6 (.5573)	2.04E-6 (.5573)	8.61E-4 (.5574)	4.22E+1 (.9617)	3.48E-7 (.5573)	5.09E-7 (.5573)	8.76E-4 (.5574)	4.32E+1 (.9644)
16	1.39E-6 (.7465)	2.04E-6 (.7465)	2.93E-5 (.7465)	5.18E+5 (1.830)	3.48E-7 (.7465)	5.09E-7 (.7465)	2.65E-5 (.7465)	5.43E+5 (1.770)
20	1.39E-6 (.7915)	2.04E-6 (.7915)	9.18E-6 (.7915)	1.68E+6 (1.993)	3.48E-7 (.7915)	5.09E-7 (.7915)	5.58E-6 (.7915)	6.19E+7 (2.001)
32	1.39E-6 (.8640)	2.04E-6 (.8640)	7.44E-6 (.8640)	2.68E+14 (2.487)	3.48E-7 (.8640)	5.09E-7 (.8640)	1.86E-6 (.8640)	2.95E+14 (2.491)
36	1.39E-6 (.8781)	2.04E-6 (.8781)	7.44E-6 (.8812)	4.41E+16 (2.586)	3.48E-7 (.8781)	5.09E-7 (.8781)	1.86E-6 (.8749)	6.29E+16 (2.597)

Table 3.2: The max norm of the error and the computed values of the convergence factor of the **AVE** method applied to model problem (3.4.1)-DP1 ($\gamma^2 = 10$). In the first column we have the iteration number, in the first row the discretization step-size and in the second row the number of equal subdomains.

In Figures 3.1, 3.2 and 3.3 we consider the model problem (3.4.1)-DP1, with a splitting of the domain Ω into three subdomains. We present the contour plots of the experimentally determined number of iterations required to reduce the max norm of the difference of two successive iterants smaller than 10^{-5} as a function of the various relaxation parameters involved. The stars in these plots indicate the theoretically optimum relaxation parameters computed by using the formulas (3.3.12) and (3.3.25) of the **ROB** and **AVE** methods respectively. In all plots associated with the **AVE** method, we use $\gamma^2 = 2$. The Neumann relaxation parameters β_1 and β_2 are computed by formula (3.3.27) while we systematically vary the Dirichlet parameters α_1 and α_2 in $(0, 1)$. For the **ROB** method, we set $\gamma^2 = 2$ while the relaxation parameters vary in a larger interval since there are no bounds for them. For this method we see that there is a curve in the $\lambda_1\lambda_2$ plane with optimum values for the relaxation parameters. The stars in the **ROB** plots represent the optimum values computed using formula (3.3.12), which is located at the intersection of the above curve and the solution of equation (3.3.13) for $p = 3$, i.e.,

$$\lambda_1(\gamma_2 n_2 + \lambda_2 m_2) = \gamma_2(\gamma_2 m_2 + \lambda_2 n_2).$$

We note that, at the points indicated by stars in all the following graphs, the experimentally observed number of iterations were always in the range of 5 to 8. This confirms the theoretical optimality of the parameter values. It is also interesting to observe that this optimality seems to be independent of the uniformity of the decomposition and of changes in the value of γ^2 in the subdomains.

In particular, in Figure 3.2, we have the same non-uniform decomposition as in the bottom two plots in Figure 3.1, but here the coefficient of u in the DE is discontinuous at the interface points. Specifically, in the first subdomain $\gamma^2 = 2$, in the second $\gamma^2 = 10$ and in the third $\gamma^2 = 4$. The right plot for **AVE** is made using, as before, Neumann relaxation parameters (β_1, β_2) computed by formula 3.3.27 and letting α_1 and α_2 vary in $(0, 1)$.

In general, the **AVE** method seems to converge faster than **ROB** but Theorem 3.3.4 imposes a restriction on its convergence region. In Figure 3.3 we experimentally verify the results in Theorem 3.3.4 and we clearly see that the restriction on the size of the subdomains imposed is not only sufficient but required too. The restriction $\gamma\ell_i > \ln(1+\sqrt{2})$ of Theorem 3.3.4 is, for the six cases in Figure 3.3, top row (γ/p = 2.24, 1.19, .89 > .881) and bottom (γ/p = .913, 1.05, 1.17 > .881). The convergence region (the area where the spectral radius of the iteration matrix is less than 1) shrinks as one either increases the number of subdomains keeping γ^2 constant or decreases γ^2 assuming a constant number of subdomains. The imposed bound on the size of subdomains seems to be a sharp one

[1] `http://www.cs.purdue.edu/homes/giwta/dom-dec/1_dim/matlab/index.html`

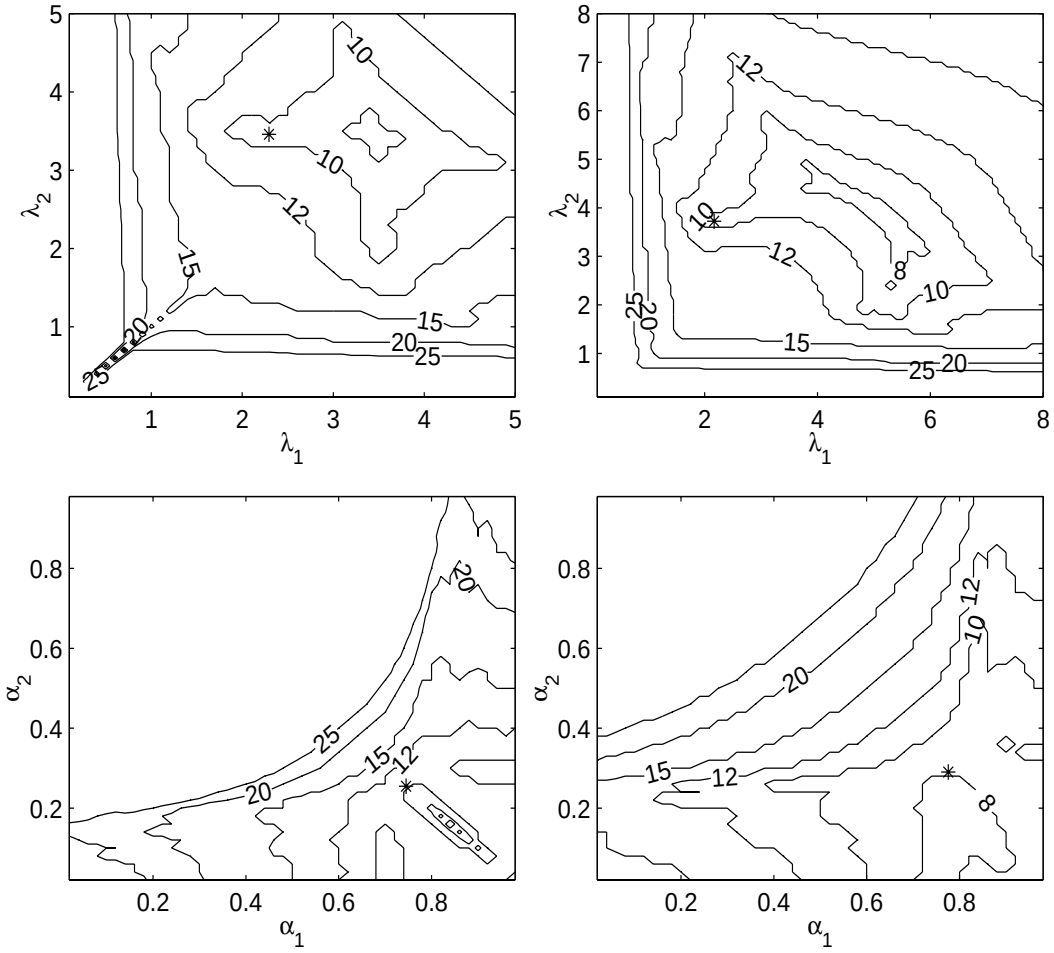


Figure 3.1: Contour plots for case DP1 of the number of iterations required by the **ROB** (top two plots) and **AVE** (bottom two plots) methods to make the max norm of the difference of two successive iterants smaller than 10^{-5} as a function of associated relaxation parameters. We assume a uniform 3 subdomain partition in the graphs on the left and non-uniform partition with $x_1 = .2$ and $x_2 = .7$ on the right ($\gamma^2 = 2$). The stars point the theoretically determined optimum values of the parameters.

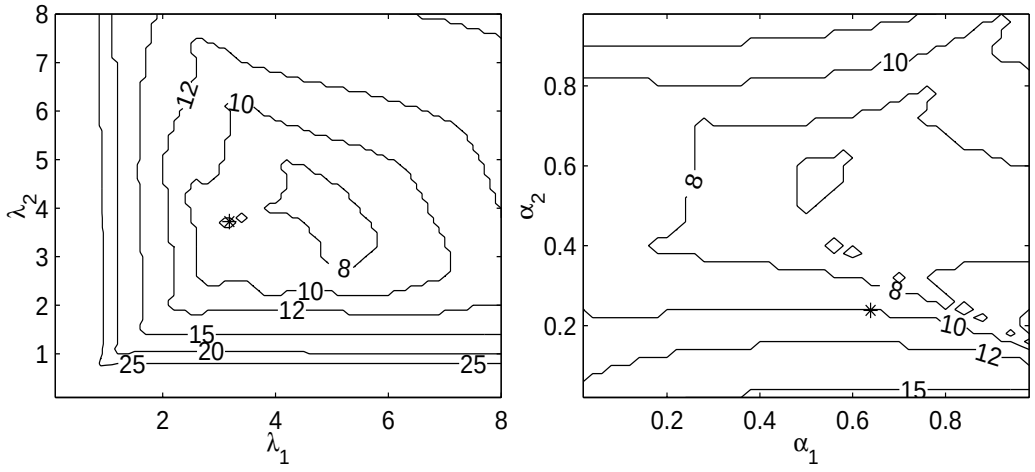


Figure 3.2: Contour plots for case DP1 of the number of iterations required by the **ROB** (left) and the **AVE** (right) methods to make the max norm of the difference of two successive iterants smaller than 10^{-5} as a function of the associated relaxation parameters. We assume a uniform 3 subdomain partition in the graph on the left and non-uniform partition with $x_1 = .2$ and $x_2 = .7$ on the right. Here the coefficient of u is $\gamma^2 = 2$ for the first subdomain, $\gamma^2 = 10$ for the second and $\gamma^2 = 4$ for the third subdomain. The stars represent the theoretical optimum values.

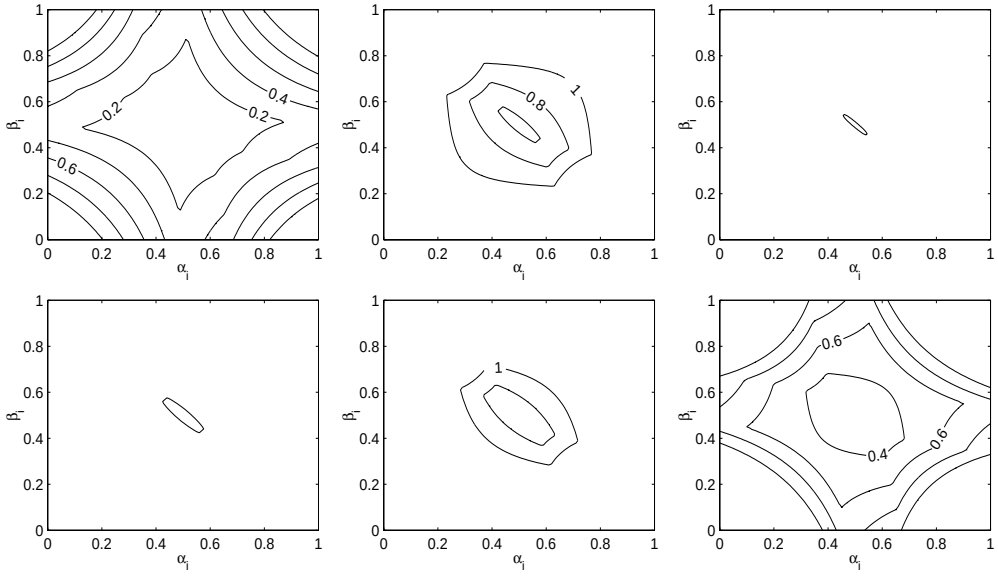


Figure 3.3: Contour plots for case DP1 of the upper bounds of the spectral radius for the uniform case for the **AVE** method. In the top three plots $\gamma^2 = 20$ while the number of subdomains is equal to 2 (left), 4 (middle) and 5 (right). In the bottom three figures we fix the number of subdomains at $p = 6$ and γ^2 is equal to 30 (left), 40 (middle) and 80 (right).

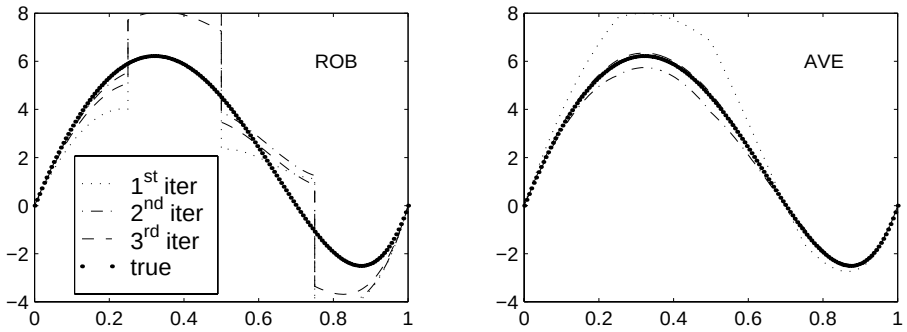


Figure 3.4: Convergence history for case DP2 with $\gamma^2 = 20$ and a 4 subdomain uniform decomposition. The graph shows the true solution and the first three iterants for the **ROB** (on the left plot) and the **AVE** (on the right) methods.

since in all our experiments we observe divergence every time we make $\ell_i\gamma$ slightly less than $\ln(1+\sqrt{2})$ while we always obtained convergence otherwise.

To obtain additional information on the convergence behavior of the two methods we now switch to the model problem (3.4.1)-DP2. The data for Figures 3.4 and 3.5 have been extracted from Chapter 2 and are presented here for completeness. In Figure 3.4 we set $\gamma^2 = 20$ and plot the true solution and the first three iterants. We observe that both methods converge in a non-monotone way, but **AVE** follows a much smoother path.

In Figure 3.5 we consider the model problem (3.4.1)-DP2, with a two subdomain partition. We set all relaxation parameters equal to .5 and experimentally measure the effect that the size of γ^2 and the location of the interface point have on the convergence rates for the two methods. We plot the logarithm of the max norm of the error (on the y-axis) versus the number of iterations (on the x-axis). The interface point is fixed at .5 for the two plots on the left of the figure while $\gamma^2 = 20$ for the two on the right. We observe that the **AVE** method is significantly affected by both parameters while the **ROB** method converges in a smoother but slower way.

3.4.2 Two dimensional case

We have implemented [2] the **AVE** and **ROB** methods for two dimensional problems using ELLPACK [54] assuming "skyline" domains (a string of rectangles of different heights and widths). This leads to a one dimensional decomposition of two dimensional rectangles. The detailed presentation of this two dimensional performance analysis is beyond the scope of this section but, we give an example in Figure 3.6 of the convergence rate of the **AVE**, and

[2] See `http://www.cs.purdue.edu/homes/giwta/dom-dec/1_dim/matlab/index.html`

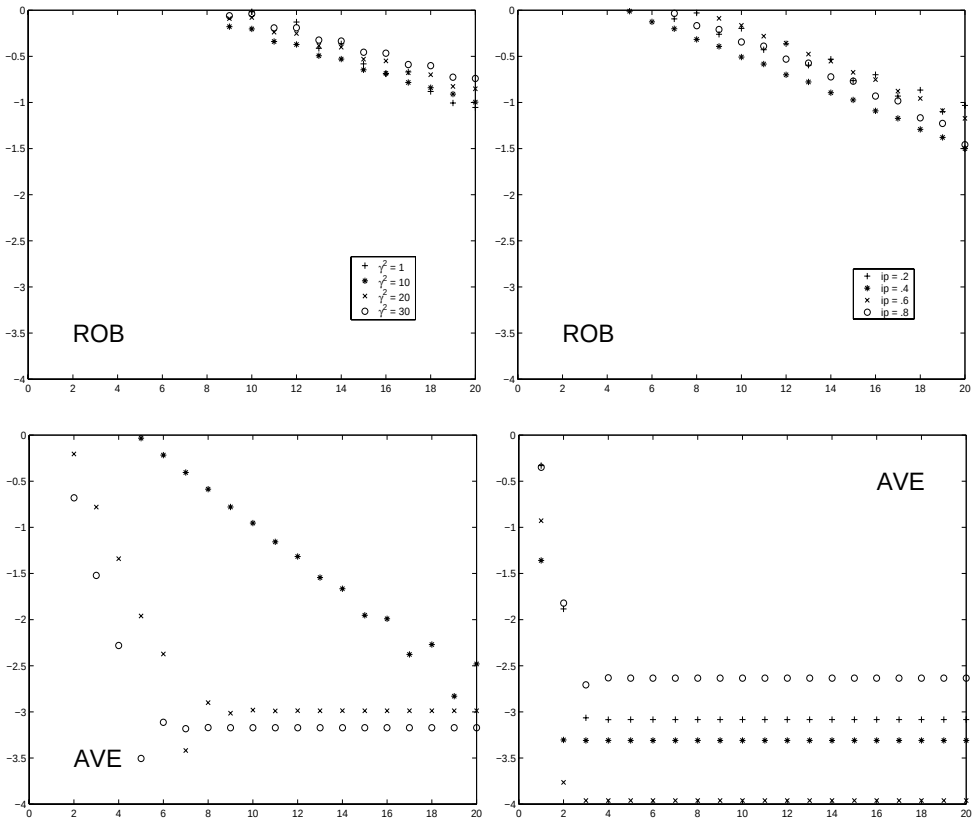


Figure 3.5: The effect of the coefficient γ^2 (left graph $\gamma^2 = 1, 10, 20, 30$) and of the location of the interface point (right, $x = .2, .4, .6, .8$) on the convergence rates for the **ROB** (top) and **AVE** (bottom) applied to case DP2. The y-axis is the max norm of the difference of successive solutions and the x-axis is the number of iterations.

ROB methods. The Helmholtz differential equation is $-\Delta u + \gamma^2 u = f \in \Omega$ with Dirichlet boundary conditions where f is selected such that $u(x) = e^{y(x+4)}x(x-1)(x-.7)y(y-.5)$. The PDE domain and its one dimensional partition into 3 subdomains is as follows:

$$\Omega_1 = (0, x_1) \times (0, 2), \quad \Omega_2 = (x_1, x_2) \times (0, .5), \quad \Omega_3 = (x_2, 1) \times (0, 1), \quad \Omega \equiv \cap_{i=1}^{3} \Omega_i.$$

where $0 < x_1 < x_2 < x_3 < 1$. We have numerically verified that either the discretization scheme or the grid size has very little effect on the convergence rate of both the IR methods considered. In all the experiments associated with the present study the 5-point star ELLPACK discretization module was used and the domain is discretized with a uniform grid in both directions using $h = .01$.

The similarity of the convergence behavior between the one dimensional and the two

dimensional problems is easily observed by comparing Figures 3.1 and 3.6. We performed many other experiments (some of them are given in Chapter 6), all of these were in reasonably good agreement with both the quantitative and qualitative conclusions we draw from the one dimensional experiments presented above, provided that the subdomain are not very narrow in the y-direction. Specifically, it is apparent that the region of convergence shrinks down as the subdomain become large in the y-direction. This is consistent with the similar behavior observed or even proved in other conventional domain decomposition studies like [10].

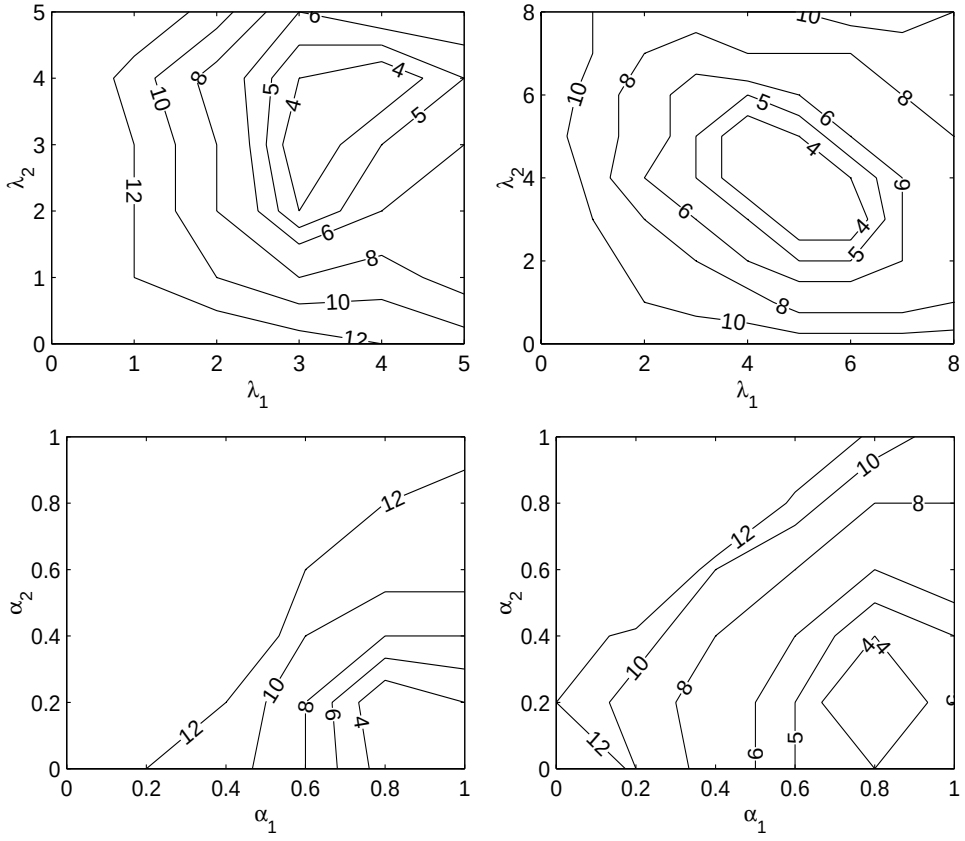


Figure 3.6: Contour plots of number of iterations required for PDE problem defined in 3.4.2 by the **ROB** (top two plots) and **AVE** (bottom two plots) methods to make the max norm of the difference of two successive iterants smaller than 10^{-5} for the two dimensional Dirichlet problem $-\Delta u + 2u = f$ as a function of associated relaxation parameters. We assume a uniform 3 subdomain partition in the graphs on the left and non-uniform partition with $x_1 = .2$ and $x_2 = .7$ on the right the PDE domain and its partition given on the left.

Chapter 4

Analysis of a New Interface Relaxation Method

Abstract

The new Interface Relaxation method is theoretically analyzed for one dimensional model problems. The analysis is carried out at both continuous and discrete level. Regions of convergence are determined and optimum parameters are obtained.

4.1 Introduction

The main objective of this Chapter is to theoretically analyze the new Interface Relaxation method, named **GEO** in Chapter 2, for solving elliptic differential equations. This method is based on a simple geometric contraction mechanism and iterates to relax the values on the interfaces by adding to the old ones a geometrically weighted combination of the normal boundary derivatives of the adjacent subdomains. A variation of this method has been also considered in [45] where a convergence analysis has been carried out and numerical data are presented. Unfortunately, the rate of convergence of this variation of the method strongly depends on the discretization parameter h and as such violates one of the main principles of Interface Relaxation and converges so slow that needs preconditioning. In this Chapter we analyze the **GEO** scheme and prove for one dimensional model problems that the convergence is independent of h.

We consider the same model problem defined in 3.2.1. The following algorithm defines the **GEO** iterative scheme:

1. Define:

$$g_i^i = \left.\frac{u_i^{(k)}+u_{i+1}^{(k)}}{2}\right|_{x=x_i} + \frac{w_i^i w_i^{i+1}}{w_i^i+w_i^{i+1}}\left(-\left.\frac{du_i^{(k)}}{dx}\right. + \left.\frac{du_{i+1}^{(k)}}{dx}\right)\right|_{x=x_i} \Bigg\} i = 1,\ldots,p-1.$$

2. Choose initial guesses $u_i^{(0)}(x)$ for the solutions on each subdomain Ω_i, $i = 1, 2, \ldots, p$.

3. Define the sequence of subdomain solutions $u_i^{(k)}(x)$, $k = 1, 2, \ldots$ as follows:

$$\left.\begin{array}{l} Lu_1^{(k+1)} = f \quad \text{in} \quad \Omega_1 \\ \left.u_1^{(k+1)}\right|_{x=x_0} = 0 \\ \left.u_1^{(k+1)}\right|_{x=x_1} = g_1^1 \end{array}\right| \quad \left|\begin{array}{l} Lu_p^{(k+1)} = f \quad \text{in} \quad \Omega_p \\ \left.u_p^{(k+1)}\right|_{x=x_{p-1}} = g_{p-1}^{p-1} \\ \left.u_p^{(k+1)}\right|_{x=x_p} = 0 \end{array}\right.$$

$$\left.\begin{array}{l} Lu_i^{(k+1)} = f \quad in \quad \Omega_i \\ \left.u_i^{(k+1)}\right|_{x=x_{i-1}} = g_{i-1}^{i-1} \\ \left.u_i^{(k+1)}\right|_{x=x_i} = g_i^i \end{array}\right\} \quad i = 2,\ldots,p-1.$$

The rest of this Chapter contains the convergence analysis at PDE level, similar to the one contained in the previous Chapter, for the three subdomain case in Section 4.2. In Section 4.3 we present an analysis for the three subdomain case with Laplace operator at the discrete (finite difference) level, while in section 4.4 numerical experiments verify some of the theoretical results.

4.2 Convergence analysis at PDE level

The analysis at the PDE level is carried out as in the case of **ROB** and **AVE** methods and therefore we will use the same notation as in Chapter 3.

The differential equations that are satisfied by the error functions, as they are defined in the previous Chapter, can be easily obtained from the corresponding equations that define the iterative relaxation scheme given in the previous section, and they can be written as follows

$$\begin{aligned} &L_1\epsilon_1^{(k+1)}(x) = 0, \quad x \in \Omega_1, \\ &\epsilon_{1,0}^{(k+1)} = 0, \quad \epsilon_{1,1}^{(k+1)} = \frac{\epsilon_{1,1}^{(k)}+\epsilon_{2,1}^{(k)}}{2} + \rho_1 \left(-d\epsilon_{1,1}^{(k)} + d\epsilon_{2,1}^{(k)}\right), \end{aligned} \tag{4.2.1}$$

for $i = 2, \ldots, p-1$,

$$\begin{aligned} &L_i\epsilon_i^{(k+1)}(x) = 0, \quad x \in \Omega_i, \\ &\epsilon_{i,i-1}^{(k+1)} = \frac{\epsilon_{i-1,i-1}^{(k)}+\epsilon_{i,i-1}^{(k)}}{2} + \rho_{i-1} \left(-d\epsilon_{i-1,i-1}^{(k)} + d\epsilon_{i,i-1}^{(k)}\right), \\ &\epsilon_{i,i}^{(k+1)} = \frac{\epsilon_{i,i}^{(k)}+\epsilon_{i+1,i}^{(k)}}{2} + \rho_i \left(-d\epsilon_{i,i}^{(k)} + d\epsilon_{i+1,i}^{(k)}\right), \end{aligned} \tag{4.2.2}$$

$$\begin{aligned} &L_p\epsilon_p^{(k+1)}(x) = 0, \quad x \in \Omega_p, \\ &\epsilon_{p,p}^{(k+1)} = 0, \quad \epsilon_{p,p-1}^{(k+1)} = \frac{\epsilon_{p-1,p-1}^{(k)}+\epsilon_{p,p-1}^{(k)}}{2} + \rho_{p-1} \left(-d\epsilon_{p-1,p-1}^{(k)} + d\epsilon_{p,p-1}^{(k)}\right), \end{aligned} \tag{4.2.3}$$

where $\rho_i = \frac{w_i^i w_i^{i+1}}{w_i^i + w_i^{i+1}} > 0, i = 1, \ldots, p.$
According to 3.3.1, the error functions are given by

$$\epsilon_1^{(k+1)}(x) = \frac{e^{\gamma_1(x-x_0)} - e^{-\gamma_1(x-x_0)}}{e^{\gamma_1\ell_1} - e^{-\gamma_1\ell_1}} \left(\frac{\epsilon_{1,1}^{(k)} + \epsilon_{2,1}^{(k)}}{2} + \rho_1 \left(-d\epsilon_{1,1}^{(k)} + d\epsilon_{2,1}^{(k)}\right)\right), \tag{4.2.4}$$

for $i = 2, \ldots, p-1$,

$$\begin{aligned} &\epsilon_i^{(k+1)}(x) = \left[e^{\gamma_i\ell_i} - e^{-\gamma_i\ell_i}\right]^{-1} \\ &\left[\left(e^{\gamma_i(x_i-x)} - e^{-\gamma_i(x_i-x)}\right)\left(\frac{\epsilon_{i-1,i-1}^{(k)}+\epsilon_{i,i-1}^{(k)}}{2} + \rho_{i-1}\left(-d\epsilon_{i-1,i-1}^{(k)} + d\epsilon_{i,i-1}^{(k)}\right)\right) + \right. \\ &\left.\left(e^{\gamma_i(x-x_{i-1})} - e^{-\gamma_i(x-x_{i-1})}\right)\left(\frac{\epsilon_{i,i}^{(k)}+\epsilon_{i+1,i}^{(k)}}{2} + \rho_i\left(-d\epsilon_{i,i}^{(k)} + d\epsilon_{i+1,i}^{(k)}\right)\right)\right] \end{aligned} \tag{4.2.5}$$

and

$$\epsilon_p^{(k+1)}(x) = \frac{e^{\gamma_p(x_p-x)} - e^{-\gamma_p(x_p-x)}}{e^{\gamma_p\ell_p} - e^{-\gamma_p\ell_p}} \left(\frac{\epsilon_{p-1,p-1}^{(k)} + \epsilon_{p,p-1}^{(k)}}{2} + \rho_{p-1}\left(-d\epsilon_{p-1,p-1}^{(k)} + d\epsilon_{p,p-1}^{(k)}\right)\right). \tag{4.2.6}$$

Differentiating the above functions, we get the following expressions for the derivatives on the interface points,

$$
\begin{aligned}
d\epsilon_{1,1}^{(k+1)} &= \frac{e^{\gamma_1\ell_1}+e^{-\gamma_1\ell_1}\gamma_1}{e^{\gamma_1\ell_1}-e^{-\gamma_1\ell_1}}\left(\frac{\epsilon_{1,1}^{(k)}+\epsilon_{2,1}^{(k)}}{2}+\rho_1\left(-d\epsilon_{1,1}^{(k)}+d\epsilon_{2,1}^{(k)}\right)\right),\\
d\epsilon_{i,i-1}^{(k+1)} &= -\frac{(e^{\gamma_i\ell_i}+e^{-\gamma_i\ell_i})\gamma_i}{e^{\gamma_i\ell_i}-e^{-\gamma_i\ell_i}}\left(\frac{\epsilon_{i-1,i-1}^{(k)}+\epsilon_{i,i-1}^{(k)}}{2}+\rho_{i-1}\left(-d\epsilon_{i-1,i-1}^{(k)}+d\epsilon_{i,i-1}^{(k)}\right)\right)+\\
&\quad \frac{2\gamma_i}{e^{\gamma_i\ell_i}-e^{-\gamma_i\ell_i}}\left(\frac{\epsilon_{i,i}^{(k)}+\epsilon_{i+1,i}^{(k)}}{2}+\rho_i\left(-d\epsilon_{i,i}^{(k)}+d\epsilon_{i+1,i}^{(k)}\right)\right),\quad i=2,\ldots,p-1\\
d\epsilon_{i,i}^{(k+1)} &= \frac{2\gamma_i}{e^{\gamma_i\ell_i}-e^{-\gamma_i\ell_i}}\left(\frac{\epsilon_{i-1,i-1}^{(k)}+\epsilon_{i,i-1}^{(k)}}{2}+\rho_{i-1}\left(-d\epsilon_{i-1,i-1}^{(k)}+d\epsilon_{i,i-1}^{(k)}\right)\right)+\\
&\quad \frac{(e^{\gamma_i\ell_i}+e^{-\gamma_i\ell_i})\gamma_i}{e^{\gamma_i\ell_i}-e^{-\gamma_i\ell_i}}\left(\frac{\epsilon_{i,i}^{(k)}+\epsilon_{i+1,i}^{(k)}}{2}+\rho_i\left(-d\epsilon_{i,i}^{(k)}+d\epsilon_{i+1,i}^{(k)}\right)\right),\quad i=2,\ldots,p-1\\
d\epsilon_{p,p-1}^{(k+1)} &= -\frac{(e^{\gamma_p\ell_p}+e^{-\gamma_p\ell_p})\gamma_p}{e^{\gamma_p\ell_p}-e^{-\gamma_p\ell_p}}\left(\frac{\epsilon_{p-1,p-1}^{(k)}+\epsilon_{p,p-1}^{(k)}}{2}+\rho_{p-1}\left(-d\epsilon_{p-1,p-1}^{(k)}+d\epsilon_{p,p-1}^{(k)}\right)\right).
\end{aligned}
$$

For simplicity, in the rest of this Chapter, we set $p = 3$. This analysis seems to be extended easily for an arbitrary number of subdomains, similarly to the analysis of **AVE** and **ROB** . Also, we define the error vectors by ordering the individual errors and their derivatives on the interface points, for $k = 0, 1, 2, \ldots$, as follows

$$
\underline{\epsilon}^{(k)} \equiv \left[\epsilon_{1,1}^{(k)}, \epsilon_{2,1}^{(k)}, \epsilon_{2,2}^{(k)}, \epsilon_{3,2}^{(k)}, d\epsilon_{1,1}^{(k)}, d\epsilon_{2,1}^{(k)}, d\epsilon_{2,2}^{(k)}, d\epsilon_{3,2}^{(k)}\right]^T.
$$

The relation that holds between the error vectors of two successive iterations, (k) and $(k+1)$, can be written as

$$
\underline{\epsilon}^{(k+1)} = M\underline{\epsilon}^{(k)}, \quad k = 0, 1, \ldots, \tag{4.2.7}
$$

where the iteration matrix $M \in \mathbb{R}^{8\times 8}$ has the form

$$
M = \begin{bmatrix}
1/2 & 1/2 & 0 & 0 & -\rho_1 & \rho_1 & 0 & 0\\
1/2 & 1/2 & 0 & 0 & -\rho_1 & \rho_1 & 0 & 0\\
0 & 0 & 1/2 & 1/2 & 0 & 0 & -\rho_2 & \rho_2\\
0 & 0 & 1/2 & 1/2 & 0 & 0 & -\rho_2 & \rho_2\\
A_1\gamma_1/2 & A_1\gamma_1/2 & 0 & 0 & -\rho_1A_1\gamma_1 & \rho_1A_1\gamma_1 & 0 & 0\\
-A_2\gamma_2/2 & -A_2\gamma_2/2 & B_2\gamma_2/2 & B_2\gamma_2/2 & \rho_1A_2\gamma_2 & -\rho_1A_2\gamma_2 & -\rho_2B_2\gamma_2 & \rho_2B_2\gamma_2\\
-B_2\gamma_2/2 & -B_2\gamma_2/2 & A_2\gamma_2/2 & A_2\gamma_2/2 & \rho_1B_2\gamma_2 & -\rho_1B_2\gamma_2 & -\rho_2A_2\gamma_2 & \rho_2A_2\gamma_2\\
0 & 0 & -A_3\gamma_3/2 & -A_3\gamma_3/2 & 0 & 0 & \rho_2A_3\gamma_3 & -\rho_2A_3\gamma_3
\end{bmatrix}, \tag{4.2.8}
$$

with $A_i = tanh^{-1}(\gamma_i\ell_i)$ and $B_i = \frac{1}{sinh(\gamma_i\ell_i)}$ for $i = 1, 2$.

The following Lemma proves that the iteration matrix has the same spectral radius with a simpler matrix of reduced size 2×2.

Lemma 4.2.1. *The non-identically zero eigenvalues of matrix M (4.2.8) are equal to the non-identically zero eigenvalues of matrix $\tilde{M}$, where*

$$
\tilde{M} = \begin{bmatrix}
1-\rho_1(\gamma_1A_1+\gamma_2A_2) & \rho_2\gamma_2B_2\\
\rho_1\gamma_2B_2 & 1-\rho_2(\gamma_2A_2+\gamma_3A_3)
\end{bmatrix} \tag{4.2.9}
$$

Proof: Using basic properties of the determinants we obtain the following equalities.

$$det(M - \lambda I_8) =$$

$$det\begin{bmatrix} -\lambda & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1/2 & 1-\lambda & 0 & 0 & -\rho_1 & \rho_1 & 0 & 0 \\ 0 & 0 & -\lambda & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & 1-\lambda & 0 & 0 & -\rho_2 & \rho_2 \\ A_1\gamma_1/2 & A_1\gamma_1 & 0 & 0 & -\rho_1 A_1\gamma_1 - \lambda & \rho_1 A_1\gamma_1 & 0 & 0 \\ -A_2\gamma_2/2 & -A_2\gamma_2 & B_2\gamma_2/2 & B_2\gamma_2 & \rho_1 A_2\gamma_2 & -\rho_1 A_2\gamma_2 - \lambda & -\rho_2 B_2\gamma_2/2 & \rho_2 B_2\gamma_2/2 \\ -B_2\gamma_2/2 & -B_2\gamma_2 & A_2\gamma_2/2 & B_2\gamma_2 & \rho_1 B_2\gamma_2 & -\rho_1 B_2\gamma_2 & -\rho_2 A_2\gamma_2/2 - \lambda & \rho_2 A_2\gamma_2/2 \\ 0 & 0 & -A_3\gamma_3/2 & -A_3\gamma_3 & 0 & 0 & \rho_2 A_3\gamma_3 & -\rho_2 A_3\gamma_3 - \lambda \end{bmatrix} =$$

$$\lambda^2 det\begin{bmatrix} 1-\lambda & 0 & -\rho_1 & \rho_1 & 0 & 0 \\ 0 & 1-\lambda & 0 & 0 & -\rho_2 & \rho_2 \\ A_1\gamma_1 & 0 & -\rho_1 A_1\gamma_1 - \lambda & \rho_1 A_1\gamma_1 & 0 & 0 \\ -A_2\gamma_2 & B_2\gamma_2 & \rho_1 A_2\gamma_2 & -\rho_1 A_2\gamma_2 - \lambda & -\rho_2 B_2\gamma_2/2 & \rho_2 B_2\gamma_2/2 \\ -B_2\gamma_2 & B_2\gamma_2 & \rho_1 B_2\gamma_2 & -\rho_1 B_2\gamma_2 & -\rho_2 A_2\gamma_2/2 - \lambda & \rho_2 A_2\gamma_2/2 \\ 0 & -A_3\gamma_3 & 0 & 0 & \rho_2 A_3\gamma_3 & -\rho_2 A_3\gamma_3 - \lambda \end{bmatrix} =$$

$$= \lambda^4 det\begin{bmatrix} -\rho_1 A_1\gamma_1 - \lambda + 1 & \rho_1 A_1\gamma_1 - 1 & 0 & 0 \\ \rho_1 A_2\gamma_2 & -\rho_1 A_2\gamma_2 - \lambda & -\rho_2 B_2\gamma_2/2 & \rho_2 B_2\gamma_2/2 \\ \rho_1 B_2\gamma_2 & -\rho_1 B_2\gamma_2 & -\rho_2 A_2\gamma_2/2 - \lambda + 1 & \rho_2 A_2\gamma_2/2 - 1 \\ 0 & 0 & \rho_2 A_3\gamma_3 & -\rho_2 A_3\gamma_3 - \lambda \end{bmatrix}$$

$$= \lambda^4 det\begin{bmatrix} -\rho_1(A_1\gamma_1 + A_2\gamma_2) - \lambda + 1 & 0 & \rho_2 B_2\gamma_2/2 & 0 \\ \rho_1 A_2\gamma_2 & -\lambda & -\rho_2 B_2\gamma_2/2 & 0 \\ \rho_1 B_2\gamma_2 & 0 & -\rho_2(A_2\gamma_2/2 + A_3\gamma_3) - \lambda + 1 & 0 \\ 0 & 0 & \rho_2 A_3\gamma_3 & -\lambda \end{bmatrix}$$

$$= \lambda^6 det(\tilde{M} - \lambda I_2). \quad \square$$

Theorem 4.2.2. *Consider the model problem 3.2.1 and a non-overlapping decomposition of* Ω *into three subdomains* Ω_i *of length* ℓ_i, $i = 1, 2, 3$. *The* ***GEO*** *interface relaxation method converges to the solution of the original problem, if and only if its relaxation parameters satisfy the following inequalities:*

$$0 < \rho_1 < \frac{2}{C_1}, \quad 0 < \rho_2 < 2\frac{2 - \rho_1 C_1}{2C_2 - \rho_1(C_1 C_2 - \gamma_2^2 B_2^2)},$$

where $C_i = \gamma_i A_i + \gamma_{i+1} A_{i+1}, i = 1, 2.$

Proof : For the convergence of the method, we'll show that the spectral radius of the iteration matrix is less than one. According to the previous Lemma, this is equivalent to proving that the spectral radius of $\tilde{M}$ is less than one. But,

$$det(\tilde{M} - \lambda I_2) = 0 \Leftrightarrow \lambda^2 - \lambda(D_1 + D_2) + D_1 D_2 - \rho_1 \rho_2 \gamma_2^2 B_2^2 = 0,$$

where $D_i = 1 - \rho_i C_i,\ i = 1, 2$. The roots of the above polynomial are given by the analytical expression

$$\lambda_{1,2} = \frac{D_1 + D_2 \pm \sqrt{(D_1 - D_2)^2 + 4\rho_1 \rho_2 (\gamma_2 B_2)^2}}{2}. \tag{4.2.10}$$

To determine the region of convergence, we force the inequality $|\lambda_1|, |\lambda_2| < 1$ and solve with respect to ρ_1, ρ_2. We notice that the same conditions would have been obtained if we had applied the Schur-Cohn algorithm [27].

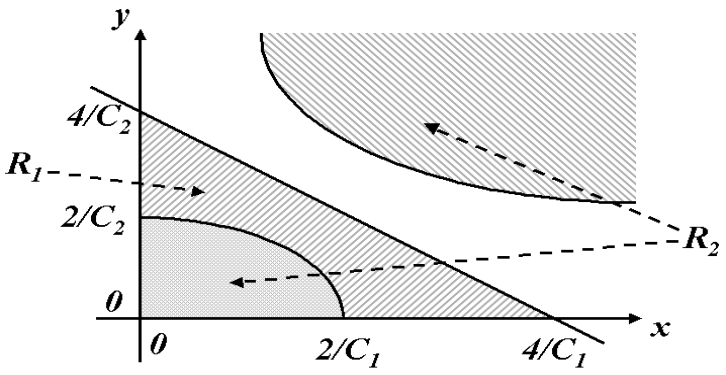


Figure 4.1: Region of convergence for the relaxation parameters of the **GEO** method

It is clear that the quantity under the square root is positive which means that $\lambda_1, \lambda_2 \in \mathbb{R}$. The inequality $\lambda_{1,2} < 1$ leads to the fact that $\rho_1, \rho_2 > 0$, which holds true. Also,

$$\lambda_{1,2} > -1 \Leftrightarrow -4 + \rho_1 C_1 + \rho_2 C_2 < \pm\sqrt{(C_1 - C_2)^2 + 4\rho_1\rho_2(\gamma_2 B_2)^2}$$

which forces $-4 + \rho_1 C_1 + \rho_2 C_2 < 0$. This holds if

$$(\rho_1, \rho_2) \in R_1 = \{\, (x, y) \in \mathbb{R}^2 \big| \quad x < \frac{4}{C_1}, \quad y < \frac{4 - \rho_1 C_1}{C_2} \}. \tag{4.2.11}$$

Taking $\rho_i, \ i = 1, 2$ in the above set, it is sufficient to solve only the inequality

$$-4 + \rho_1 C_1 + \rho_2 C_2 < -\sqrt{(C_1 - C_2)^2 + 4\rho_1\rho_2(\gamma_2 B_2)^2}$$

which is equivalent to the one below

$$4 - 2\rho_1 C_1 - 2\rho_2 C_2 + \rho_1\rho_2(C_1 C_2 - \gamma_2^2 B_2^2) > 0.$$

This is true for

$$(\rho_1, \rho_2) \in R_2 = \{\, (x, y) \in \mathbb{R}^2 \big| \quad x < \frac{2}{C_1}, \quad y < 2\frac{2 - xC_1}{2C_2 - x(C_1 C_2 - \gamma_2^2 B_2^2)} \}. \tag{4.2.12}$$

We conclude the proof by combining the regions in 4.2.11 and 4.2.12 (see Figure 4.1) and the fact that $\rho_i > 0, \quad i = 1, 2.$ $\square$

Working for the corresponding Laplace PDE problem in the same way as above, we derive the region of convergence of the **GEO** method. Specifically, we give the following corollary.

Corollary 4.2.3. *If we replace in the model problem and its decomposition considered in Theorem 4.2.2 the Helmholtz operator with the Laplace one, the region of convergence is given by*

$$0 < \rho_1 < \frac{2}{\Gamma_1}, \qquad 0 < \rho_2 < 2\frac{2 - \rho_1\Gamma_1}{2\Gamma_2 - \rho_1(\Gamma_1\Gamma_2 - 1/\ell_2^2)}$$

, where $\Gamma_i = \frac{\ell_i + \ell_{i+1}}{\ell_i \ell_{i+1}}, i = 1, 2.$

Finally, we would like to mention that the analysis for the 2 domain case, for both Laplace and Helhmoltz operators, is quite simple and the region of convergence of the method under consideration and the optimum values for the relaxation parameters derived, are given in a simpler form in the following corollary.

Corollary 4.2.4. *Assume that we have a two subdomain partition of the domain* Ω*. Then for the Laplace operator, the relaxation method converges if and only if* $0 < \rho_1 < 2\ell_1\ell_2$*, while the optimum value,* $\rho_1 = \ell_1\ell_2$*, makes the method to converge immediately. The region of convergence for the Helmholtz case is the interval* $(0, 2/(\gamma_1 tanh^{-1}(\gamma_1\ell_1) + \gamma_2 tanh^{-1}(\gamma_2\ell_2)))$*, while for* $\rho_1 = 1/(\gamma_1 tanh^{-1}(\gamma_1\ell_1) + \gamma_2 tanh^{-1}(\gamma_2\ell_2))$ *the method converges immediately. With both operators, the immediate convergence is meant as explained in Section 3.4.*

4.3 Convergence analysis at discrete level

In this section we give the analysis at the discrete (Linear Algebra) level. We consider the Laplace version of the model problem 3.2.1 and split Ω into three domains Ω_i, $i = 1, 2, 3$. We denote the interface points as I_1 and I_2 and we discretize each subdomain using $n_i + 1$, $i = 1, 2, 3$ uniformly distributed points. To discretize the Laplace operator we use the standard 3-point star discretization scheme. Denoting by $I_0 = a$, $I_3 = b$ and $h_i = \frac{I_i - I_{i-1}}{n_i}$, $i = 1, 2, 3$, we define the discretization points in Ω_i as $x_{j_i}^{(i)} = I_{i-1} + h_i j_i$ with $j_i = 0, \ldots, n_i$ for $i = 1, 2, 3$. For an approximation of the derivatives on the interface points we use Taylor series expansion to obtain

$$\left.\frac{du}{dx}\right|_{x=x^\star} = \frac{1}{h}\left(\frac{3}{2}u(x^\star) - 2u(x^\star - h) + \frac{1}{2}u(x^\star - 2h)\right) + O(h^2),$$

and

$$\left.\frac{du}{dx}\right|_{x=x^\star} = \frac{1}{h}\left(-\frac{3}{2}u(x^\star) + 2u(x^\star + h) - \frac{1}{2}u(x^\star + 2h)\right) + O(h^2).$$

In this section, we denote as $\epsilon_{j_i}^{(i,k)} \equiv u_{i,h_i}^{(k)}(x_{j_i}^{(i)}) - u_{i,h_i}(x_{j_i}^{(i)})$ the error on point $x_{j_i}^{(i)}$ at the k^{th} iteration, and define the iteration error vectors as

$$\underline{\epsilon}^{(k)} \equiv \left[\epsilon_1^{(1,k)}, \epsilon_2^{(1,k)}, \ldots, \epsilon_{n_1}^{(1,k)}, \epsilon_0^{(2,k)}, \epsilon_1^{(2,k)}, \ldots, \epsilon_{n_2}^{(2,k)}, \epsilon_0^{(3,k)}, \epsilon_1^{(3,k)}, \ldots, \epsilon_{n_3-1}^{(3,k)}\right]^T,$$

where u_{i,h_i} the discrete solution of the original problem in subdomain Ω_i. Considering all the above, we derive the following relation that holds between the error vectors of iterations (k) and $(k+1)$

$$\underline{\epsilon}^{(k+1)} = M\underline{\epsilon}^{(k)}, \quad k = 0, 1, \ldots, \tag{4.3.1}$$

where the matrix M is

$$M = T^{-1}N, \tag{4.3.2}$$

and

$$T = \begin{bmatrix} T_2^{(n_1-1)} & 0 & 0 & 0 & 0 \\ 0 & I_2 & 0 & 0 & 0 \\ 0 & 0 & T_2^{(n_2-1)} & 0 & 0 \\ 0 & 0 & 0 & I_2 & 0 \\ 0 & 0 & 0 & 0 & T_2^{(n_3-1)} \end{bmatrix}, N = \begin{bmatrix} 0_{n_1-2,n_1-3} & 0 & 0 & 0 & 0 \\ 0 & N_1 & 0 & 0 & 0 \\ 0 & 0 & 0_{n_2-3,n_2-5} & 0 & 0 \\ 0 & 0 & 0 & N_2 & 0 \\ 0 & 0 & 0 & 0 & 0_{n_3-2,n_3-3} \end{bmatrix},$$

and $T_2^n \equiv tridiag\{-1, 2, -1\} \in \mathbb{R}^{n\times n}$, I_2 is the identity matrix of dimension two, $N_i \in \mathbb{R}^{4\times 6}, i = 1, 2$, while $0_{n,m} \in \mathbb{R}^{n\times m}$ with all its elements equal to 0 and

$$N_i = \begin{bmatrix} a_i & b_i & c_i & d_i & e_i & f_i \\ a_i & b_i & c_i & d_i & e_i & f_i \\ a_i & b_i & c_i & d_i & e_i & f_i \\ a_i & b_i & c_i & d_i & e_i & f_i \end{bmatrix},$$

with $a_i = -\rho_i/(2h_i)$, $b_i = 2\rho_i/h_i$, $c_i = 1/2 - 3\rho_i/(2h_i)$, $d_i = 1/2 - 3\rho_i/(2h_{i+1})$, $e_i = 2\rho_i/h_{i+1}$, $f_i = -\rho_i/h_{i+1}$, $i = 1, 2$.

Lemma 4.3.1. *The first and the last column of the inverse matrix of $T_2^{(n)}$ are equal to the following vectors correspondingly,*

$$[1/n, \ldots, (n-1)/n]^T, \qquad [(n-1)/n, \ldots, 1/n]^T.$$

Proof: The above formulas for the vectors are well known and readily derived using elementary analysis of difference equations. □

Lemma 4.3.2. *The matrix M, defined above, is spectrally equivalent, in the spirit of Lemma 4.2.1, to $\tilde{M}$, where*

$$\tilde{M} = \begin{bmatrix} 1-\rho_1\Gamma_1 & \frac{\rho_1}{h_2} & 0 & 0 \\ 0 & 0 & \frac{\rho_2}{\ell_2} & \frac{1-\rho_2\Gamma_2}{n_2} \\ \frac{1-\rho_1\Gamma_1}{n_2} & \frac{\rho_1}{\ell_2} & 0 & 0 \\ 0 & 0 & \frac{\rho_2}{h_2} & 1-\rho_2\Gamma_2 \end{bmatrix}, \tag{4.3.3}$$

and Γ_i, $i = 1, 2$, as defined in previous section.

Proof: Using properties of the determinant, we can see that,

$$det(M - \lambda I_{n_1+n_2+n_3+1}) = \lambda^{n_1+n_2+n_3-11} det \begin{bmatrix} N_{1,1} & N_{1,2} \\ N_{2,1} & N_{2,2} \end{bmatrix},$$

where

$$N_{i,i} = \begin{bmatrix} a_i g_i - \lambda & b_i g_i & c_i g_i & d_i g_i & e_i g_i & f_i g_i \\ a_i h_i & b_i h_i - \lambda & c_i h_i & d_i h_i & e_i h_i & f_i h_i \\ a_i & b_i & c_i - \lambda & d_i & e_i & f_i \\ a_i & b_i & c_i & d_i - \lambda & e_i & f_i \\ a_i h_{i+1} & b_i h_{i+1} & c_i h_{i+1} & d_i h_{i+1} & e_i h_{i+1} - \lambda & f_i h_{i+1} \\ a_i g_{i+1} & b_i g_{i+1} & c_i g_{i+1} & d_i g_{i+1} & e_i g_{i+1} & f_i g_{i+1} - \lambda \end{bmatrix}, \quad i = 1, 2$$

and $g_i = \frac{n_i - 2}{n_i}$, $h_i = \frac{n_i - 1}{n_i}$ and

$$N_{1,2} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ a_1/n_2 & b_1/n_2 & c_1/n_2 & d_1/n_2 & e_1/n_2 & f_1/n_2 \\ 2a_1/n_2 & 2b_1/n_2 & 2c_1/n_2 & 2d_1/n_2 & 2e_1/n_2 & 2f_1/n_2 \end{bmatrix},$$

$$N_{2,1} = \begin{bmatrix} 2a_1/n_2 & 2b_1/n_2 & 2c_1/n_2 & 2d_1/n_2 & 2e_1/n_2 & 2f_1/n_2 \\ a_1/n_2 & b_1/n_2 & c_1/n_2 & d_1/n_2 & e_1/n_2 & f_1/n_2 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Also,

$$det(M - \lambda I_{n_1+n_2+n_3+1}) =$$

$$\lambda^{n_1+n_2+n_3-7} det \begin{bmatrix} \frac{1}{2} - \frac{\rho_1}{\ell_1} - \lambda & \frac{1}{2} - \frac{\rho_1}{\ell_2} & e_1 & f_1 & 0 & 0 & 0 & 0 \\ \frac{1}{2} - \frac{\rho_1}{\ell_1} & \frac{1}{2} - \frac{\rho_1}{\ell_2} - \lambda & e_1 & f_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\lambda & 0 & \frac{a_2}{n_2} & \frac{b_2}{n_2} & \frac{1}{2n_2} - \frac{\rho_2}{n_2 \ell_2} & \frac{1}{2n_2} - \frac{\rho_2}{n_2 \ell_3} \\ 0 & 0 & 0 & -\lambda & \frac{2a_2}{n_2} & \frac{2b_2}{n_2} & \frac{1}{n_2} - \frac{2\rho_2}{n_2 \ell_2} & \frac{1}{n_2} - \frac{2\rho_2}{2\ell_3} \\ \frac{1}{n_2} - \frac{2\rho_1}{\ell_1 n_2} & \frac{1}{n_2} - \frac{2\rho_1}{\ell_2 n_2} & \frac{2e_1}{n_2} & \frac{2f_1}{n_2} & -\lambda & 0 & 0 & 0 \\ \frac{1}{2n_2} - \frac{\rho_1}{\ell_1 n_2} & \frac{1}{2n_2} - \frac{\rho_1}{\ell_2 n_2} & \frac{e_1}{n_2} & \frac{f_1}{n_2} & 0 & -\lambda & 0 & 0 \\ 0 & 0 & 0 & 0 & a_2 & b_2 & \frac{1}{2} - \frac{\rho_2}{\ell_2} - \lambda & \frac{1}{2} - \frac{\rho_2}{\ell_3} \\ 0 & 0 & 0 & 0 & a_2 & b_2 & \frac{1}{2} - \frac{\rho_2}{\ell_2} & \frac{1}{2} - \frac{\rho_2}{\ell_3} - \lambda \end{bmatrix}$$

$$= \lambda^{n_1+n_2+n_3-5} det \begin{bmatrix} 1 - \rho_1\Gamma_1 - \lambda & f_1 & 0 & \frac{\rho_1}{h_2} & 0 & 0 \\ 0 & -\lambda & a_2/n_2 & 0 & \frac{\rho_2}{\ell_2} & \frac{1-\rho_2\Gamma_2}{n_2} \\ 0 & 0 & -\lambda & 0 & 0 & 0 \\ 0 & 0 & 0 & -\lambda & 0 & 0 \\ \frac{1-\rho_1\Gamma_1}{n_2} & f_1/n_2 & 0 & \frac{\rho_1}{\ell_2} & -\lambda & 0 \\ 0 & 0 & 0 & a_2 & \frac{\rho_2}{h_2} & 1 - \rho_2\Gamma_2 - \lambda \end{bmatrix} =$$

$$\lambda^{n_1+n_2+n_3-3} det(\tilde{M} - \lambda I_4). \quad \square$$

Theorem 4.3.3. *Consider the model problem 3.2.1 and a non-overlapping decomposition of Ω into three subdomains Ω_i of length ℓ_i, $i = 1, 2, 3$. Consider also the discretization described at the beginning of the section. The convergence of* ***GEO*** *interface relaxation*

method is independent of h_i, $i = 1, 2, 3$ *and the region of convergence is the same as in Corollary 4.2.3, therefore*

$$0 < \rho_1 < \frac{2}{\Gamma_1}, \qquad 0 < \rho_2 < 2\frac{2 - \rho_1\Gamma_1}{2\Gamma_2 - \rho_1(\Gamma_1\Gamma_2 - 1/\ell_2^2)},$$

where $\Gamma_i = \frac{\ell_i + \ell_{i+1}}{\ell_i \ell_{i+1}}$, $i = 1, 2$.

Proof: According to the previous Lemma, it is sufficient to prove that the spectral radius of $\tilde{M}$ is less than one . But

$$det(\tilde{M} - \lambda I_4) = det \begin{bmatrix} 1 - \rho_1\Gamma_1 - \lambda & \frac{\rho_1}{h_2} & 0 & 0 \\ 0 & -\lambda & 0 & \lambda/n_2 \\ \lambda/n_2 & 0 & -\lambda & 0 \\ 0 & 0 & \frac{\rho_2}{h_2} & 1 - \rho_2\Gamma_2 - \lambda \end{bmatrix} =$$

$$(1 - \rho_1\Gamma_1 - \lambda) det \begin{bmatrix} \lambda & 0 & -\lambda/n_2 \\ 0 & -\lambda & 0 \\ 0 & \frac{\rho_2}{h_2} & 1 - \rho_2\Gamma_2 - \lambda \end{bmatrix} + \frac{\lambda}{n_2} det \begin{bmatrix} \frac{\rho_1}{h_2} & 0 & 0 \\ -\lambda & 0 & \lambda/n_2 \\ 0 & \frac{\rho_2}{h_2} & 1 - \rho_2\Gamma_2 - \lambda \end{bmatrix} = 0.$$

Therefore, to compute the non-identically zero eigenvalues we have to solve the equation

$$(1 - \rho_1\Gamma_1 - \lambda)(1 - \rho_2\Gamma_2 - \lambda) - \frac{\rho_1}{h_2 n_2}\frac{\rho_2}{h_2 n_2} = 0$$

which is equivalent to

$$\lambda^2 - \lambda(2 - \rho_1\Gamma_1 - \rho_2\Gamma_2) + (1 - \rho_1\Gamma_1)(1 - \rho_2\Gamma_2) - \frac{\rho_1\rho_2}{\ell_2^2} = 0.$$

It is clear that the coefficients of the above polynomial are independent of h_i and n_i, and depend only on the decomposition of Ω. Working in the same way as in theorem 4.2.2, we prove that the region of convergence is

$$0 < \rho_1 < \frac{2}{\Gamma_1}, \qquad 0 < \rho_2 < 2\frac{2 - \rho_1\Gamma_1}{2\Gamma_2 - \rho_1(\Gamma_1\Gamma_2 - 1/\ell_2^2)}. \quad \square$$

The above theorem proves that $\lim_{k\to\infty} \underline{\epsilon}^{(k+1)} = 0$ which means that $u_{i,h_i}^{(k)} \to u_{i,h_i}$, as $k \to \infty$. Since $\rho(M) < 1$ there exists a natural norm $|||.|||$ in $\mathbb{R}^{(\sum_{i=1}^3 n_i+1)\times(\sum_{i=1}^3 n_i+1)}$, such that $c^\star \equiv |||M||| < 1$. Then, if we denote by

$$\begin{aligned} \underline{u}_\Delta^{(k)} \equiv & \Big[u_{1,h_1}^{(k)}(x_1^{(1)}), u_{1,h_1}^{(k)}(x_2^{(1)}), \ldots, u_{1,h_1}^{(k)}(x_{n_1}^{(1)}), u_{2,h_2}^{(k)}(x_0^{(2)}), u_{2,h_2}^{(k)}(x_1^{(2)}), \ldots, u_{2,h_2}^{(k)}(x_{n_2}^{(2)}), \\ & u_{3,h_3}^{(k)}(x_0^{(3)}), u_{3,h_3}^{(k)}(x_1^{(3)}), \ldots, u_{3,h_3}^{(k)}(x_{n_3-1}^{(3)})\Big]^T, \end{aligned}$$

$$\begin{aligned} \underline{u}_\Delta \equiv & \Big[u_{1,h_1}(x_1^{(1)}), u_{1,h_1}(x_2^{(1)}), \ldots, u_{1,h_1}(x_{n_1}^{(1)}), u_{2,h_2}(x_0^{(2)}), u_{2,h_2}(x_1^{(2)}), \ldots, u_{2,h_2}(x_{n_2}^{(2)}), \\ & u_{3,h_3}(x_0^{(3)}), u_{3,h_3}(x_1^{(3)}), \ldots, u_{3,h_3}(x_{n_3-1}^{(3)})\Big]^T \end{aligned}$$

and

$$\underline{u} \equiv \Big[u(x_1^{(1)}), u(x_2^{(1)}), \ldots, u(x_{n_1}^{(1)}), u(x_0^{(2)}), u(x_1^{(2)}), \ldots, u(x_{n_2}^{(2)}), u(x_0^{(3)}), u(x_1^{(3)}), \ldots, u(x_{n_3-1}^{(3)})\Big]^T$$

we have that

$$|||\underline{u}_\Delta^{(k+1)} - \underline{u}||| \leq |||\underline{\epsilon}^{(k+1)}||| + |||\underline{u}_\Delta - \underline{u}||| \leq c^*|||\underline{\epsilon}^{(k)}||| + c_\star H^2 \leq$$

$$\ldots \leq (c^\star)^{k+1}|||\epsilon^{(0)}||| + c_\star H^2,$$

where $H = \max_{1\leq i\leq 3}\{h_i\}$. Note that, $c^\star$ is independent of $h_i, n_i, i = 1, 2, 3$ while $c_\star$ depends only on the smoothness of u.

Finally, we would like to present the results for the trivial two-domain case for both Laplace and Helmholtz PDE operators.

Corollary 4.3.4. *The region of convergence and the optimum value of the relaxation parameter is as in the section 4.2 for the two-domain case with Laplace PDE operator. For the corresponding Helmholtz PDE problem the region of convergence is the interval* $(0, \frac{2}{A^\star+B^\star})$, *while the optimum value is equal to* $(A^\star + B^\star)^{-1}$, *where* $A^\star = \frac{1}{2h_1}(3 + \frac{\sinh((n_1-2)\theta_1)}{\sinh(n_1\theta_1)} - 4\frac{\sinh((n_1-1)\theta_1)}{\sinh(n_1\theta_1)})$ *and* $B^\star = \frac{1}{2h_2}(3 + \frac{\sinh((n_2-2)\theta_2)}{\sinh(n_2\theta_2)} - 4\frac{\sinh((n_2-1)\theta_2)}{\sinh(n_2\theta_2)})$ *and* θ_i *satisfies the equation* $2\cosh(\theta_i) = 2 + \gamma_i^2 h_i^2, \quad i = 1, 2$. *In both cases using the optimum value for the relaxation parameter, leads to immediate convergence(in the sense of Corollary 4.2.4) to the solution of the initial problem.*

4.4 Numerical Experiments

To experimentally verify the above given theoretical results, and in particular of the two theorems, we give in Figure 4.2 the numerically determined number of required iterations and in Figure 4.3 the theoretically estimated values of the spectral radius for the model problem (3.4.1)-DP1 considered in section 3.4 by assuming a splitting into three subdomains.

Specifically, in Figure 4.2 we present the contour plots of the experimentally determined number of iterations required to reduce the max norm of the difference of two successive iterants below 10^{-5} as a function of the various relaxation parameters involved. The stars in these plots indicate the case where $\rho_i = \frac{1}{C_i}$, $i = 1, 2$ which seems to be a reasonable choice for "good" values since they zero the centers of the Gerschgorin disks of matrix 4.2.9 while they keep the spectral radius less than one. We mention that for those "optimum" values convergence was achieved in about 6 iterations for all cases. In the graph on the left we have set $\gamma^2 = 4$ assume a uniform partition of Ω and systematically vary the values for the relaxation parameters. For the graph in the middle, we keep γ^2 the same, but the

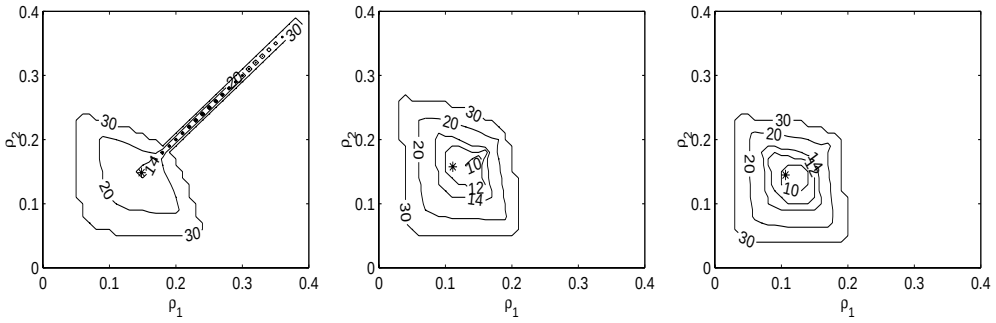


Figure 4.2: Contour plots for DP1 of the number of iterations required by the **GEO** method to reduce the max norm of the difference of two successive iterants below 10^{-5} as a function of the associated relaxation parameters. We assume a uniform three subdomain partition in the graph on the left, non-uniform partition with $x_1 = .2$ and $x_2 = .7$ on the middle and the right graph. $\gamma^2 = 4$ for the left and middle graphs, while on the right the coefficient of u is $\gamma^2 = 2$ for the first subdomain, $\gamma^2 = 10$ for the second and $\gamma^2 = 4$ for the third subdomain on the same partition as in the middle graph.

partition is a non-uniform one, with interface points set at $x_1 = .2$ and $x_2 = .7$. For the right graph, we keep the partition as in middle graph, and set the coefficient of u is $\gamma^2 = 2$ for the first subdomain, $\gamma^2 = 10$ for the second and $\gamma^2 = 4$ for the third subdomain.

In Figure 4.3 we plot the contours of the theoretically determined upper bound of the spectral radius of the **GEO** method for the configurations considered in Figure 4.3 above. Specifically, we plot $\max |\lambda_1|, |\lambda_2|$ with the λ_i's given in equation 4.2.10.

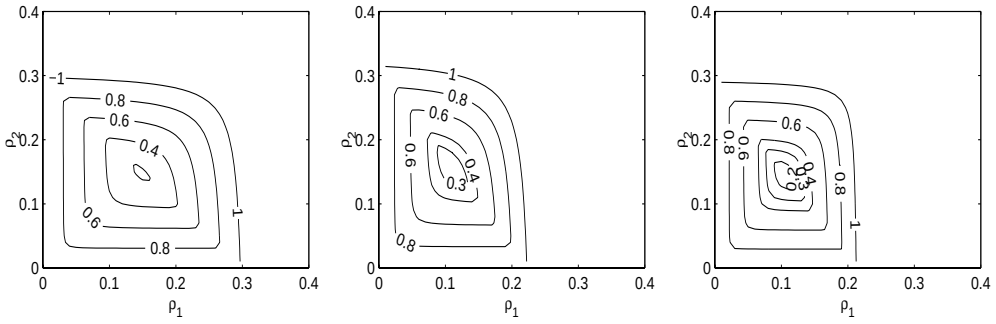


Figure 4.3: Contour plots for DP1 of the upper bound of the spectral radius for the **GEO** method. In the left graph the partition is uniform and $\gamma^2 = 4$. In the middle graph we keep $\gamma^2 = 4$ and the interface points are at $x_1 = .2$ and $x_2 = .7$. In the right one the partition is the same as in the middle, while the coefficient of u is $\gamma^2 = 2$ for the first subdomain, $\gamma^2 = 10$ for the second and $\gamma^2 = 4$ for the third subdomain.

One can clearly see the matching of the theoretical with the experimental data. We also

see that the variations on the subdomain splitting and the different γ's do not drastically change the region of convergence. It is also seen that the values of ρ's used need to have at least two correct significant digits to achieve reasonably fast convergence.

Chapter 5

Implementation and Computational Model Issues

Abstract

The implementation of a Collaborative PDE system, named SciAgent, for truly heterogeneous distributed computer systems is briefly presented. In particular the architecture and the main software components are described and the Agent technology used is introduced.

5.1 Introduction

Given the fact that the proposed collaborative PDE methodology is relatively new we argue the necessity of an implementation (denoted by *SciAgents* in the sequel) that is general enough to prove its concept and exploit its characteristics and in particular the convergence properties of the various Interface Relaxation methods. A first naïve of such a prototype implementation [42] goes back to 1991. It was solely based on core TCP/IP routines to implement the collaboration among the co-workers. It did not use software parts technology but rather developed from scratch one local solvers and implemented just one relaxer (**AVE** with all parameters set to .5). The second primitive implementation [22] differs from the first one mainly on the fact that it exploited, based on plain KQML messages, the Agent approach (to be presented next) to integrate the ELLPACK PSE. Both were very unstable, were used through text based user interfaces, did not complied with standard technology and were very limited for our purposes. Nevertheless, the later implementation have provided us with a good starting point.

A complete description of our implementation and the functionality of SciAgents is beyond the scope of this dissertation. The reader is referred to [5] and the Appendix B for some details. Instead, a high level architectural view of our SciAgent system is presented in Figure 5.1 and in the rest of this Chapter we will briefly comment only on those components and implementation issues that are closely related to the rest of the Thesis. Specifically, in Section 5.2 we present the incorporation of the legacy PDE Problem Solving Environment (PSE) integration through Agents and present the inherent parallelism, and in Section 5.3 we discuss the implementation of the relaxation mechanisms and we list the additional numerical libraries incorporated. In other words Section 5.2 concerns with the *Wrapper* and the *Agent Support* components in Figure 5.1 and section 5.3 with the *Interface Relaxation Mechanisms* and *Numerical Libraries* components. The *Graphical User Interface (GUI)* of the *SciAgents PSE* is described in appendix B while the *Run Time Support* is not so relevant to this Thesis and is not presented.

5.2 Software reuse and Agent computing

For the SciAgent implementation we need to transform the physical problem into a network of collaborative solvers and relaxers. This network for the composite problem depicted in Figure 1.2 is graphically depicted in Figure 5.2 where the local solvers are depicted with parallelepiped and the computing procedures that relax the interface values (named *mediators in the sequel*) with rectangular boxes.

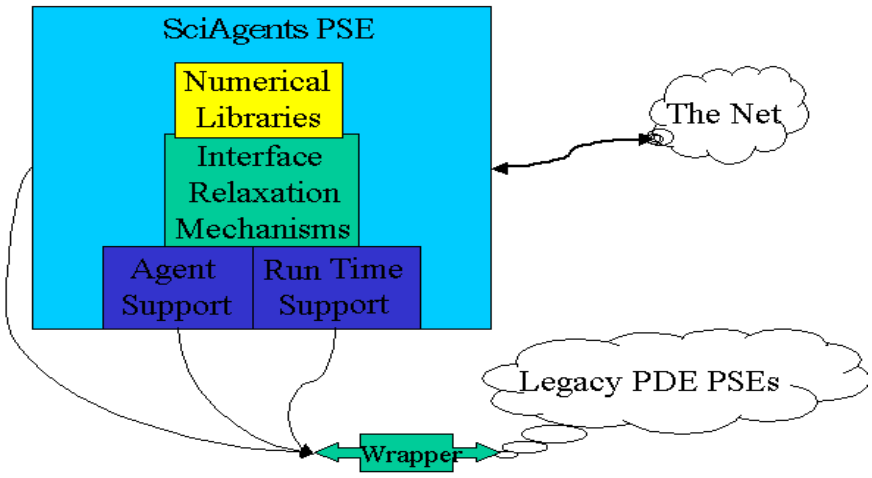


Figure 5.1: The components of the SciAgent system

Specifically, Solver 1 is associated with the heat radiation region, Solvers 2 and 3 with the heat producing regions and Solvers 4 and 5 with the mounting regions. So, one has built such a system as the above then needs to map it onto a collection of networked computer and finally needs to control the execution of its components according to the iteration workflow. SciAgents have done these using the Agent technology and in particular the Bond[1] Agent system [5].

Agent computing is a step beyond object oriented computing and may provide an answer to the increased complexity of the software systems. Different groups have radically different views of what software agents are [23] and what applications could benefit from the agent technology. Our view of an agent is described in [4] and its primary functions are: planning, scheduling and control, management of local resources, use-level resource management. To the best of our knowledge there are no other systematic attempts to bring the Agent technology to the Numerical Analysis/Scientific Computing area besides those mentioned in [53].

Details of the actual implementation of the network of PDE solvers are provided in [67, 5]. Here we outline the component agents and their functionality. The basic functionality

[1] http://bond.cs.purdue.edu

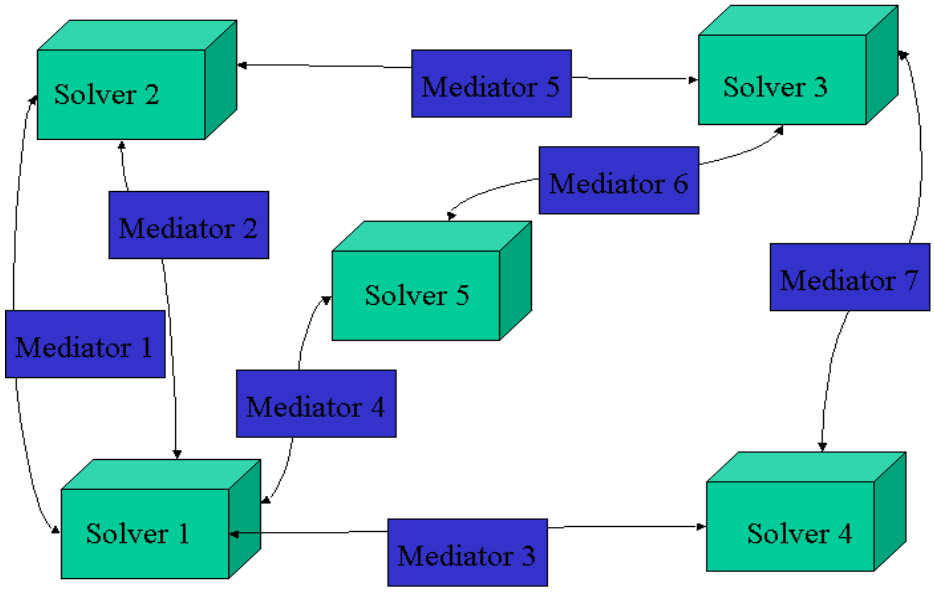


Figure 5.2: The network of solvers and mediators for the composite problem depicted in Figure 1.2

of individual agents involved in a network of PDE solvers is presented in the SciAgents system [22]. Thus we were able to identify with relative ease the functions expected from each agent and write new strategies in Java. The actual design and implementation of the network of PDE solving agents took a couple of months.

Three types of agents are involved: one `PDECoordinator` agent, several `PDESolver` and `PDEMediator` agents. The `PDECoordinator` is responsible with the control of the entire application, a `PDEMediator` arbitrates between the two solvers sharing a boundary between two domains, and a `PDESolver` is a wrapper for the legacy application.

Almost all execution agents have been built by properly wrapping existing legacy codes. These wrappers act as an interface layer generating agents out of monolithic codes. They are relatively inexpensive to build. Mediator agents needed to be created from scratch but their complexity is minimal comparing to the overall problem. They are usually build on top of existing interpolation libraries. It is worth to mention here that the whole SciAgents system which involves more than 1.5 million lines of mainly C and Fortran code contains less than two thousand lines of "wrapping" and "mediator" code.

It is rather difficult to install legacy software on a new system and in our implementation we assume that the software is already installed and the paths to executables on all systems

are known. The operation of the network of agents is presented next.

A `PDECoordinator` agent is started by means of a GUI. Once started, the agent reads and parses a problem description file and writes the information into its model. This input file contains information about the number of solvers and mediators, the characteristics of the interfaces, the initial guesses on the interfaces, the relaxation methods and the names of the machines that will be used to solve the global problem. The next step is the creation and the configuration of `PDESolver` and `PDEMediator` agents. Since the agents are alive, the `PDECoordinator` uses their addresses to setup the communication among them. Then the coordinator waits for messages from the mediators, regarding the status of the convergence to the solution of the problem, or from the user. The messages from the user are to change the values of specific variables of the input file, such as the convergence tolerance, or to force the execution to stop.

The `PDESolver` agent is created, configured and started by the `PDECoordinator`. Its model contains addresses of the legacy programs used to solve the problem locally, paths the input/output files, addresses of the visualization programs, etc. In the first state, the `PDESolver` starts-up the `Pelltool` which compiles the `.e` file that describes the local PDE problem, and creates the executable that will be used later on by the `ExecuteTool`. These tools and file designations are all part of the **PELLPACK** system [28]. In the next state, the solver extracts the points on the interfaces from the file that contains the mesh/grid points, and writes them into a file. Then the solver notifies the mediators that the files are ready. The `PDESolver` agent remains idle until being notified by the mediators that the list with all the points and their initial guesses are stored in a file at a specific location. Then the solver uses these files of points and initial guesses to run the `ExecuteTool` to solve the problem. When the execution is finished, the solver sends a message to the mediators that new values are computed, and then waits for their response. Depending on the message from the mediators, the solver will solve the problem again, remain idle waiting for the other solvers to reach convergence, or plot the local solution. The `PDECoordinator` is able to terminate the `PDESolver` by sending an appropriate message.

The mediator agent, `PDEMediator`, is created and configured by the coordinator agent. The mediator agent has a complete description of the interface, the relaxation method used, the solvers to collaborate with, the location of the input/output files, the location of the legacy programs, the tolerance used to decide convergence, and the initial guess function. This information is provided by the coordinator agent. After being started, the mediator waits for the boundary points from the two neighboring solvers. In the next state, the mediator combines the two point lists and then uses the initial guess to compute values at these points. Afterwards, the mediator sends a message to the two solvers that the files

with the points and their values are ready. The mediator remains idle, waiting for new values from the two solvers. When it receives new values it moves to the next state, reads the new data and compares them with current data. Then the mediator agent uses the relaxation method to calculate the new values for the boundary conditions. If convergence is reached on this interface then the mediator sends messages to the solvers and informs the `PDECoordinator` about the local convergence so it will be able to decide on global convergence. A message from the coordinator will cause the mediator to (1) finish, in case of global convergence or (2) wait for new data from the two solvers. In the latter case the procedure is repeated until convergence.

5.3 Additional software components

Besides building the components mentioned in the previous section additional software is needed. In particular SciAgents require strong interpolation support, procedures for estimating initial guesses, mechanisms for determining "good" values for the relaxation parameters and criteria to control the iterative procedure.

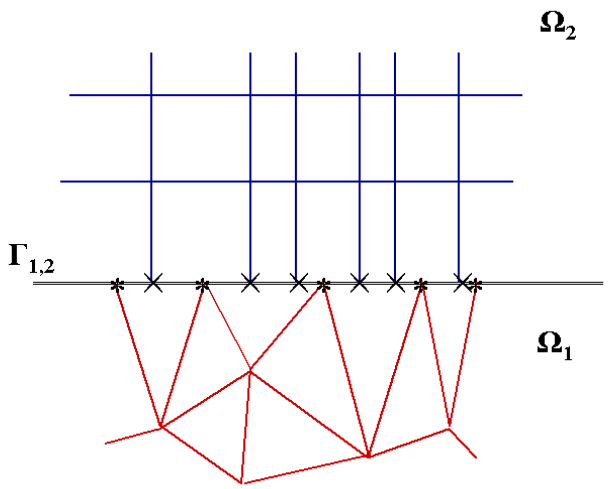


Figure 5.3: Two subdomains (Ω_1 and Ω_2) meeting at un-matching grids on their interface $\Gamma_{1,2}$.

Since in SciAgents the grids/meshes do not necessarily match on the interface (see Figure 5.3 for an example) we have implemented the following procedure:

1. The mediator sends $*$ points to Ω_2 and $\times$ points to Ω_1 and requests the k "closest" points and values.

2. The solvers respond to this request.

3. The mediator constructs an interpolant to obtain values from the top side of $*$'s and from the bottom side of $\times$'s.

4. The mediator proceeds with the relaxation.

For our SciAgent implementation interpolation is done using the ELLPACK's intrigued functions.

During our experimentation with the SciAgent system we have realized that special attention is needed for the estimation of the initial guesses on the interfaces. We have observed that the Neumman and the mixed boundary conditions are sensitive to their initial guesses. Although we were able to provide converging initial guesses for all the problems we have solved so far using naïve methods we believe that as the PDE problems get more complicated better initial guesses will be needed.

We have equipped our SciAgent system with default values for the relaxation parameters. In the same partitioning cases we have implemented naïve procedures to estimate "good" values for this parameters. These procedures (described in some detail in Chapter 6) are based on naive ways to utilize the theoretically estimated optimum parameters obtained in Chapters 3 and 4. An alternative way that seems to work for any problem and decomposition is proposed in section 7.3.

Finally, we should mark that software reuse was not problem free and additional code had to be written. As an example, we mention the fact that ELLPACK assigns the corner discretization points on the boundary, to boundary segments in an arbitrary way. This has no effect on the local solvers but complicates significantly the relaxation mechanisms.

Chapter 6

Further Numerical Experiments in 2 Dimensions

Abstract

An experimental study of the behaviour of two Interface Relaxation methods is presented. Three linear and one non-linear elliptic two-dimensional PDE problems are considered coupled with both cartesian and general decompositions. Some of the general characteristics and the effectiveness of both the Interface Relaxation methods and the collaborative PDE solving framework explained in Chapter 5 are shown. In particular, certain very desirable properties are clearly observed.

6.1 Introduction

Numerical experiments concerning the various Interface Relaxation procedures have been already presented at the end of Chapters 2 and 3. The main purpose of the experiments in Chapter 2 was to get a first idea on the behavior of several Interface Relaxation methods on model one-dimensional problems, while the numerical data in Chapter 3 was to confirm and elucidate the theoretical results obtained for model problems and to investigate if those results were also valid for more general problems. Furthermore, most of the presented numerical data were for one-dimensional problems.

We have implemented [1] **AVE** and **ROB** in the SciAgents framework, described in Chapter 5, for general two-dimensional decompositions of linear and non–linear Elliptic PDE problems. This implementation of the interface relaxers and the one considered for two-dimensional problems in Section 3.4 differ mainly on the fact that the first exploits the parallelism inherent in the Interface Relaxation methods using the Agents computing paradigm over a network of heterogeneous workstations. A comprehensive experimental study of all known interface relaxation schemes based on our SciAgent implementation is under way. Here we present the numerical study of typical PDE problems previously considered in other Interface Relaxation studies [53].

In all experiments we have calculated an initial guess for the solution by using the appropriate interpolant on each interface segment. For initial guesses of the normal derivatives we simply impose the correct sign (direction), by setting them equal to the unit outward normal vector. To calculate the required by the mediators derivatives on the interfaces the associated build-in to ELLPACK procedure was used as it is described in Chapter 5.

All experiments presented in this Chapter were run in parallel using single precision arithmetic on SUN workstations connected through an ethernet line. Each subdomain was assigned to a different machine and all the interfaces were assigned to an additional machine.

The rest of this Chapter includes the definition of the population of the PDE problems in Section 6.2 and the sections that follow present numerical data for these PDE problems.

6.2 The PDE Problems Considered

For the experimental analysis we consider four similar domains, two on the left and two on the right of Figure 6.1 together with the associated boundary conditions and with the decompositions on each one of them as follows:

[1] See `http://www.cs.purdue.edu/homes/giwta/dom-dec/1_dim/matlab/index.html`

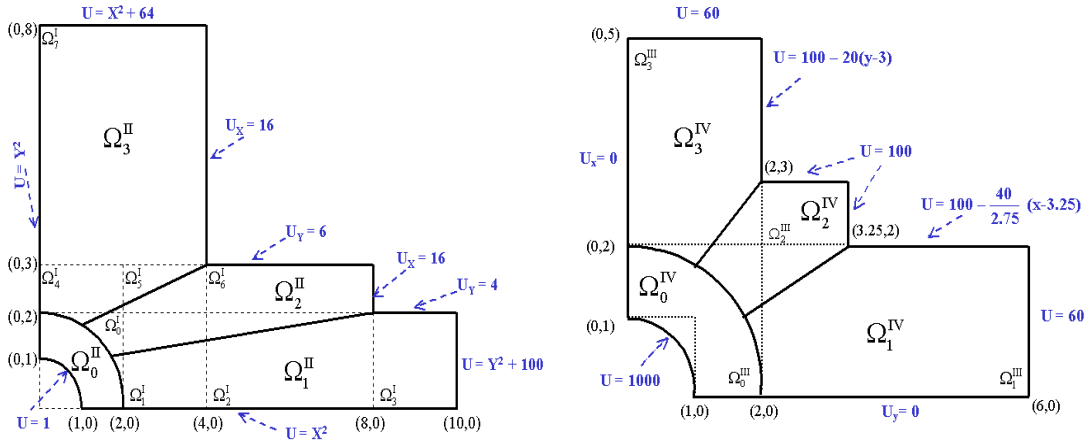


Figure 6.1: Two domains, Ω^I and Ω^{II}, on the left and two Ω^{III} and Ω^{IV} on the right, with the associated boundary conditions, their decompositions and the numbering of these subdomains and their interface segments.

Ω^I is the domain on the left of the figure, and it consists of the eight rectangular subdomains shown with dotted lines.

Ω^{II} is the one on the left of the figure, and it consists of the four subdomains shown with solid lines.

Ω^{III} is the domain on the right of the figure, and it consists of the four rectangular subdomains shown with dotted lines.

Ω^{IV} is the one on the right of the figure, and it consists of the four subdomains shown with solid lines.

Figure 6.1 also defines the numbering of the subdomains.

We can now define the four PDE problems we consider in this Chapter as follows:

PDE1

$$
\begin{aligned}
&-\Delta u + \gamma_i^2 u = f_i \quad \text{on} \quad \Omega_i^I \quad i = 0, \ldots, 7 \quad \text{with} \\
&\gamma_0^2 = \tfrac{1}{2}\exp(x+y), \quad \gamma_1^2 = 10, \quad \gamma_2^2 = 16, \quad \gamma_3^2 = 20, \\
&\gamma_4^2 = 2(\sin((x+y)\pi) + 4), \quad \gamma_5^2 = 15, \quad \gamma_6^2 = 25, \quad \gamma_7^2 = 5\exp(\tfrac{x+y}{8})
\end{aligned}
\tag{6.2.1}
$$

The f_i's have been selected so that the true solution is $x^2 + y^2$. We use second order accurate local discretization schemes.

PDE2

$$\begin{aligned} &-\Delta u + \gamma_i^2 u = f_i \quad \text{on} \quad \Omega_i^{II} \quad i = 0,1,2,3 \quad \text{with} \\ &\gamma_1 = \tfrac{1}{2}\exp(x+y), \gamma_2 = 16, \gamma_3 = 25 \quad \text{and} \\ &\gamma_4 = \sin((x+y)\pi) + \tfrac{5}{2}\exp(\tfrac{x+y}{8}) + 4 \end{aligned} \tag{6.2.2}$$

The right hand sides (f_i's) have been selected so that the true solution is $x^2 + y^2$.

PDE3

$$\begin{aligned} &\Delta u + 0.2u + 60(x^2+y^2+2) = 0, \quad \text{on} \quad \Omega_0^{IV}, \\ &\Delta u + 0.4u = 0, \quad \text{on} \quad \Omega_1^{IV} \quad \text{and} \quad \Omega_3^{IV}, \\ &\Delta u - 10\left(\tfrac{\partial u}{\partial x} + \tfrac{\partial u}{\partial y}\right) + 0.3u = 0 \quad \text{on} \quad \Omega_2^{IV} \end{aligned} \tag{6.2.3}$$

PDE4

$$\begin{aligned} &\Delta u + 0.2u\left(1 + \tfrac{u}{1000}\right) + 60(x^2+y^2+2) = 0, \quad \text{on} \quad \Omega_0^{IV}, \\ &\Delta u + 0.4u = 0, \quad \text{on} \quad \Omega_1^{IV} \quad \text{and} \quad \Omega_3^{IV}, \\ &\tfrac{\partial^2 u}{\partial x^2} + \left(1 + \tfrac{u}{1000}\right)\tfrac{\partial^2 u}{\partial y^2} + \left(\tfrac{1}{500}\tfrac{\partial u}{\partial x} + 3\right)\tfrac{\partial u}{\partial y} + 0.3u = 0, \quad \text{on} \quad \Omega_2^{IV} \end{aligned} \tag{6.2.4}$$

For each one of these four decompositions we denote the interface segments by $I^d_{s_1,s_2}$, where d denote the corresponding decomposition, and s_1, s_2 the indices of the two subdomains which are adjacent to this particular interface. For example, by $I^I_{2,3}$, we denote the interface that lays between the subdomains Ω^I_2 and Ω^I_3 of the Ω^I (Figure 6.1).

The above PDE problems are selected so that one might compare the effect of

- the size of the domains (as moving from domain Ω^{II} to Ω^{IV})
- the type of the decomposition (by comparing results associated with domains Ω^I, Ω^{III} to the ones associated with domains Ω^{II}, Ω^{IV})
- the nonlinearity (by considering PDE4)

Furthermore, we note that one might consider the decompositions of subdomains Ω^I and Ω^{III} as "cartesian approximations" to the ones of subdomains Ω^{II} and Ω^{IV} respectively and as such they might be used to calculate "good" relaxation parameters for the non-cartesian decompositions (see Section 6.4).

Interface segment	**ROB** relaxation parameter	**AVE** relaxation parameters	
		a_i	β_i
1	3.162	0.626	0.374
2	4.000	0.442	0.559
3	4.472	0.472	0.528
4	3.164	0.627	0.375
5	3.873	0.508	0.492
6	5.000	0.444	0.556
7	3.873	0.721	0.179
8	5.000	0.437	0.564
9	4.734	0.398	0.600
10	4.734	0.448	0.552

Table 6.1: The theoretically determined "optimal" values of the interface relaxation parameters used to obtain the data associated with the dotted-dashed lines in Figures 6.2-6.5.

6.3 PDE1: A Linear PDE with a Cartesian Decomposition

For the results in this section we discretize each of the subdomains using rectangular meshes with common discretization parameter $h = 0.1$. Note that for this discretization the subdomain grids match on the interfaces. Then the local PDE operators were discretized using the ELLPACK' s [54] 5-point-star module, which is an $O(h^2)$ finite difference scheme, on all subdomains. This resulted into linear systems with $N = 441, 441, 861, 441, 231, 231, 451, 451, 2051$ equations and unknowns on subdomains $0, 1, \ldots, 7$ respectively. All these systems were solved using the ELLPACK's Gauss Elimination module for banded matrices.

To calculate values for the relaxation parameters we tried to utilize our one-dimensional theoretical results presented in Chapter 3. For that we collapse appropriate rows or columns of rectangles by considering them as lines and their interface lines as interface points. For example to obtain the relaxation parameters on the interface segments $I^I_{0,4}, I^I_{4,7}$ and $I^I_{1,5}, I^I_{5,7}$ we consider the one-dimensional decomposition $[0,2],[2,3]$ and $[3,8]$, and on the segments $I^I_{0,1}, I^I_{1,2}$ and $I^I_{2,3}$ the $[0,2],[2,4],[4,8]$ and $[8,10]$ while for the segments $I^I_{4,5}, I^I_{5,6}$ the $[0,2],[2,4]$ and $[4,8]$ and finally for the segment $I^I_{2,6}$ we consider the decomposition $[0,2],[2,3]$. Furthermore, the γ_i's involved in the theoretical expressions of the one-dimensional analysis of Chapter 3 we set equal to the average of the coefficient of u_i in each subdomain. The new one-dimensional PDE problems are first scaled to $[0,1]$ by multiplying the γ_i's with the square of the length of the interval and then the optimum parameters are computed using the formulas obtained in Chapter 3. The values estimated for PDE1 by using the above procedure are given in Table 6.1.

To examine the basic convergence properties we first plot the following two measures:

$$S = \log \|u_i^{(k)} - u_i^{(k-1)}\|_1, \tag{6.3.1}$$

$$E = \log \|u_i^{(k)} - u_i\|_\infty. \tag{6.3.2}$$

The subscript i denotes, in 6.3.1 the interface segment and in 6.3.2 the subdomain as these are depicted in Figure 6.1. The superscripts denote, as usual, the iteration, u_i is the restriction of the true solution u in Ω_i and $u_i^{(k)}$ is the computed, at the k iteration, solution of the problem in subdomain i.

Both Figures 6.2 and 6.3 clearly show that the estimated by the above procedure values of the relaxation parameters for the **ROB** scheme are really close to "optimum" and result in convergence that is significant faster than the one associated with $\lambda_i = 2$ or 4. In the **AVE** case though as seen from Figures 6.4 and 6.5 the theoretical determined relaxation parameters do not show such optimality . Nevertheless, it is seen that the rate of convergence, depicted by the slopes of the convergence lines, does not differ significantly in the case of the experimentally "best" relaxation parameters.

By comparing Figures 6.3 and 6.5 one sees that "optimum" **ROB** seems to be faster that "optimum" **AVE** while both schemes seem to be rather effective, achieving 3 significant digits in less that 4 iterations and 5 digits in about 10 iterations.

We conclude this section by investigating the effect of the particular PDE discretization scheme one might use to solve the individual PDE subproblems. For this we plot in Figure 6.6 the history of convergence for the **ROB** scheme applied to PDE1 using three different discretization modules from ELLPACK. Specifically, we have performed three experiments, one (depicted by dash-dotted lines in the graphs), using the finite element module on all subdomains, another using the 5-point-star finite difference module (solid lines) and another using the collocation module (dotted lines). Note that the finite element and the finite difference schemes are of order $O(h^2)$ while the collocation is of order $O(h^4)$. For all methods we used $h = .10$. These lead to linear systems of significant difference on size and mathematical properties. For example, the collocation matrix has 1764, 1764, 3444, 1764, 924, 924, 1804, 8364 equations and unknowns, the finite element has 304, 306, 601, 306, 167, 167, 319, 1463 and the finite difference has 399, 418, 818, 399, 218, 227, 450, 2000 in subproblems $i = 0, \ldots, 7$ respectively. It is very clearly seen that the convergence behaviour of the Interface Relaxation method is identical to all three cases indicating the natural expectation one might have drawn from the formulation and the preliminary analysis of the previous chapters.

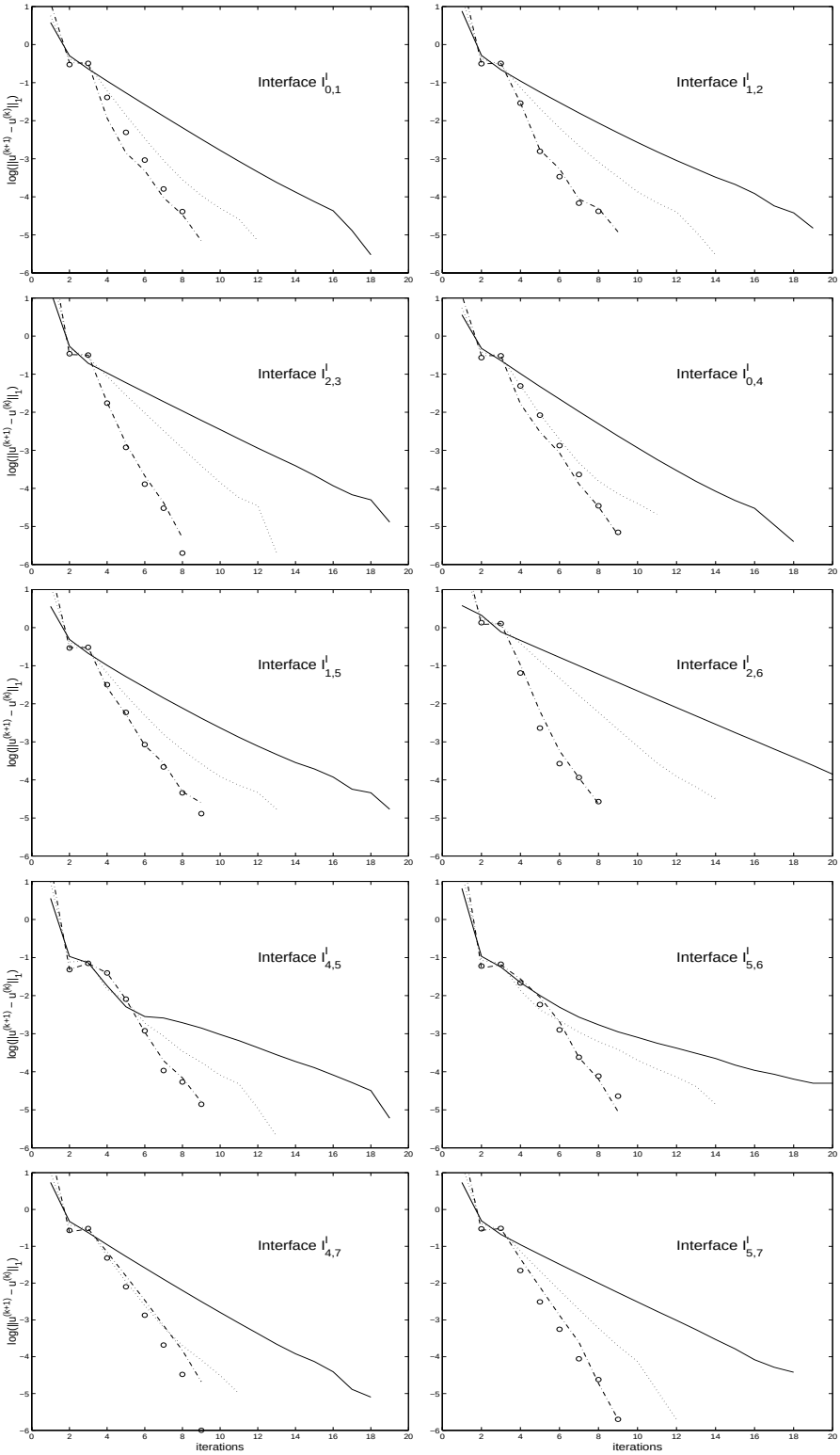


Figure 6.2: Reduction of the L_1 norm of the difference of two successive iterants on each interface using the **ROB** scheme for PDE1. Solid lines, dotted lines and circles denote data using $\lambda_i = 1$, $\lambda_i = 2$ and $\lambda_i = 4$ for $i = 0, \ldots, 8$ respectively. The dash-dotted lines denote data using the theoretically determined optimum values for λ_i's shown in Table 6.1.

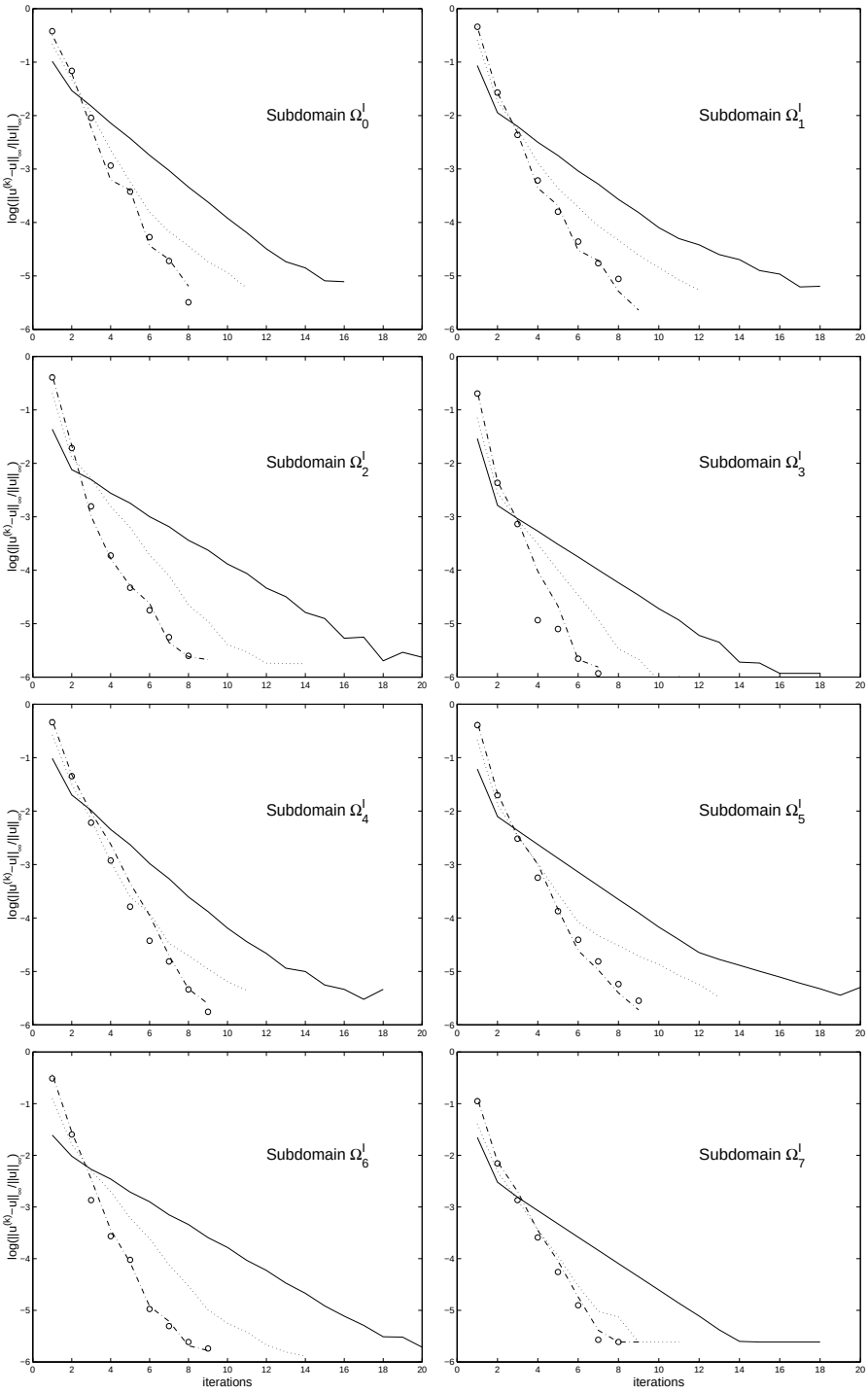


Figure 6.3: Reduction of the L_∞ norm of the relative errors in each subdomain using the **ROB** scheme for PDE1. Solid lines, dotted lines and circles denote data using $\lambda_i = 1$, $\lambda_i = 2$ and $\lambda_i = 4$ for $i = 0, \ldots, 7$ respectively. The dash-dotted lines denote data using the theoretically determined optimum values for λ_i's shown in Table 6.1.

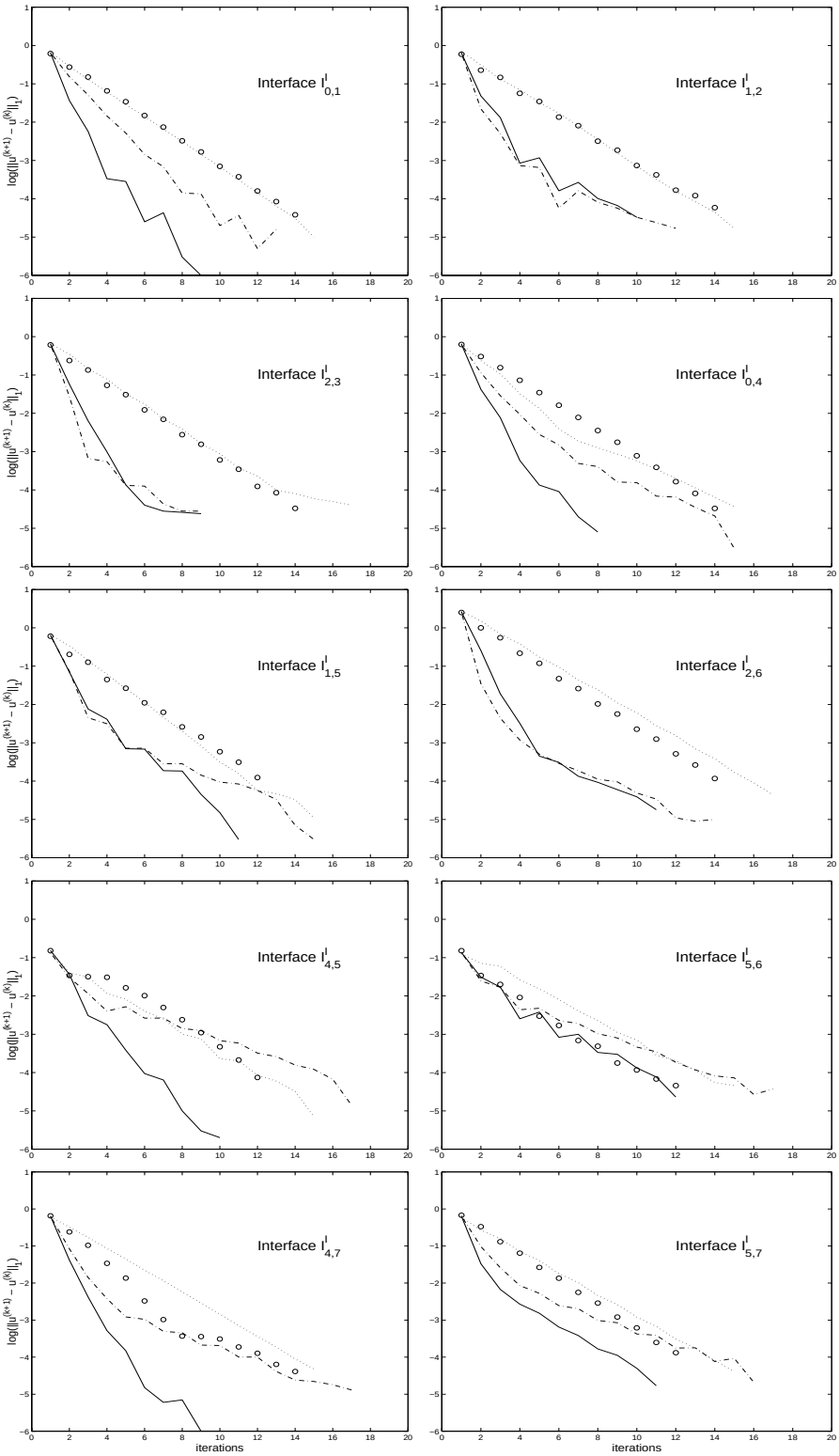


Figure 6.4: Reduction of the L_1 norm of the difference of two successive iterants on each interface using the **AVE** scheme for PDE1. Solid lines, dotted lines and circles denote data using $\alpha_i = \beta_i = 0.5$, $\alpha_i = \beta_i = 0.25$, and $\alpha_i = \beta_i = 0.75$, for $i = 0, \ldots, 7$ respectively. The dash-dotted lines denote data using the theoretically determined optimum values for the α_i's and β_i's shown in Table 6.1.

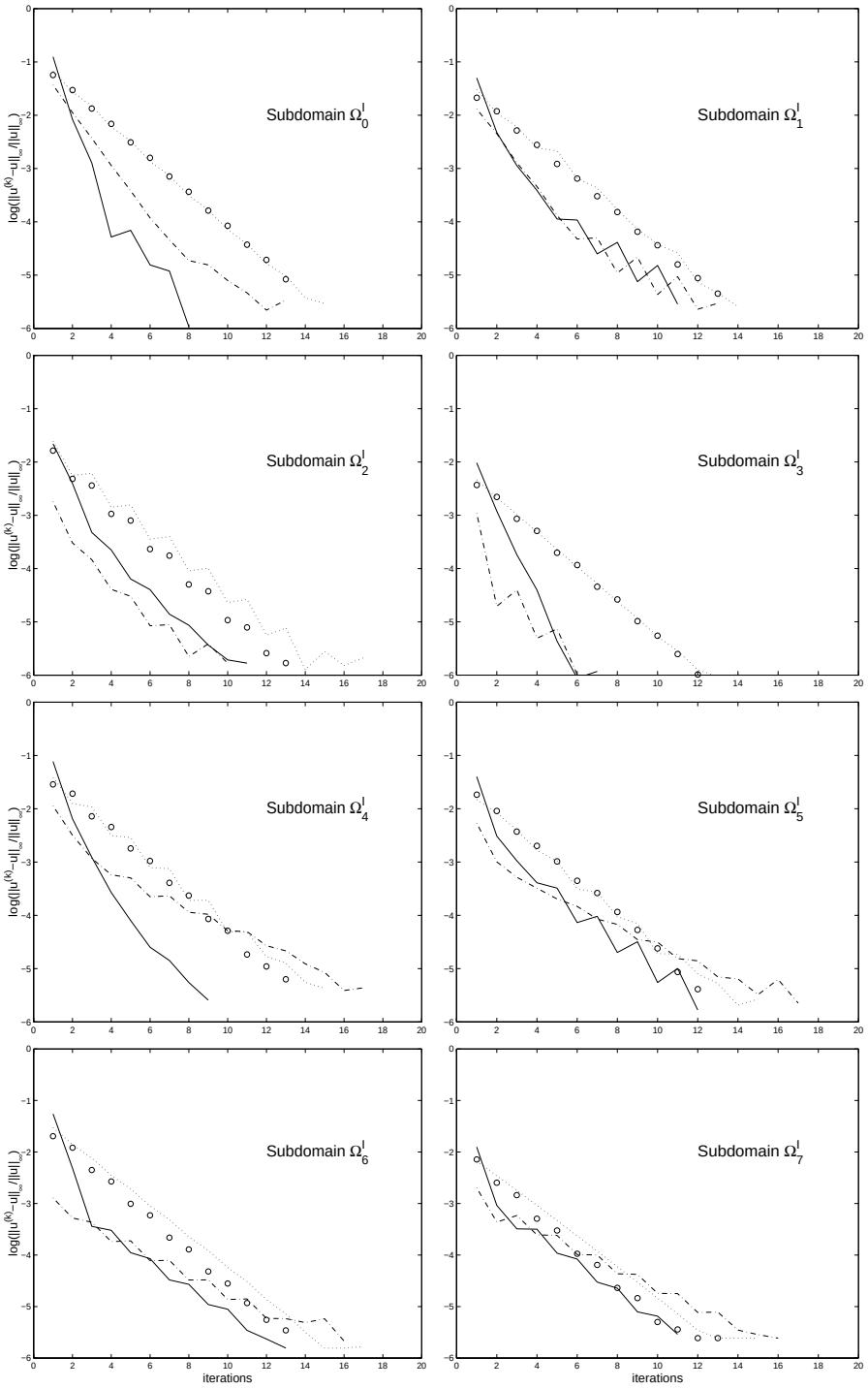


Figure 6.5: Reduction of the L_∞ norm of the relative errors in each subdomain using the **AVE** scheme for PDE1. Solid lines, dotted lines and circles denote data using $\alpha_i = \beta_i = 0.5$, $\alpha_i = \beta_i = 0.25$, and $\alpha_i = \beta_i = 0.75$, for $i = 0, \ldots, 7$ respectively. The dash-dotted lines denote data using the theoretically determined optimum values for the α_i's and β_i's shown in Table 6.1.

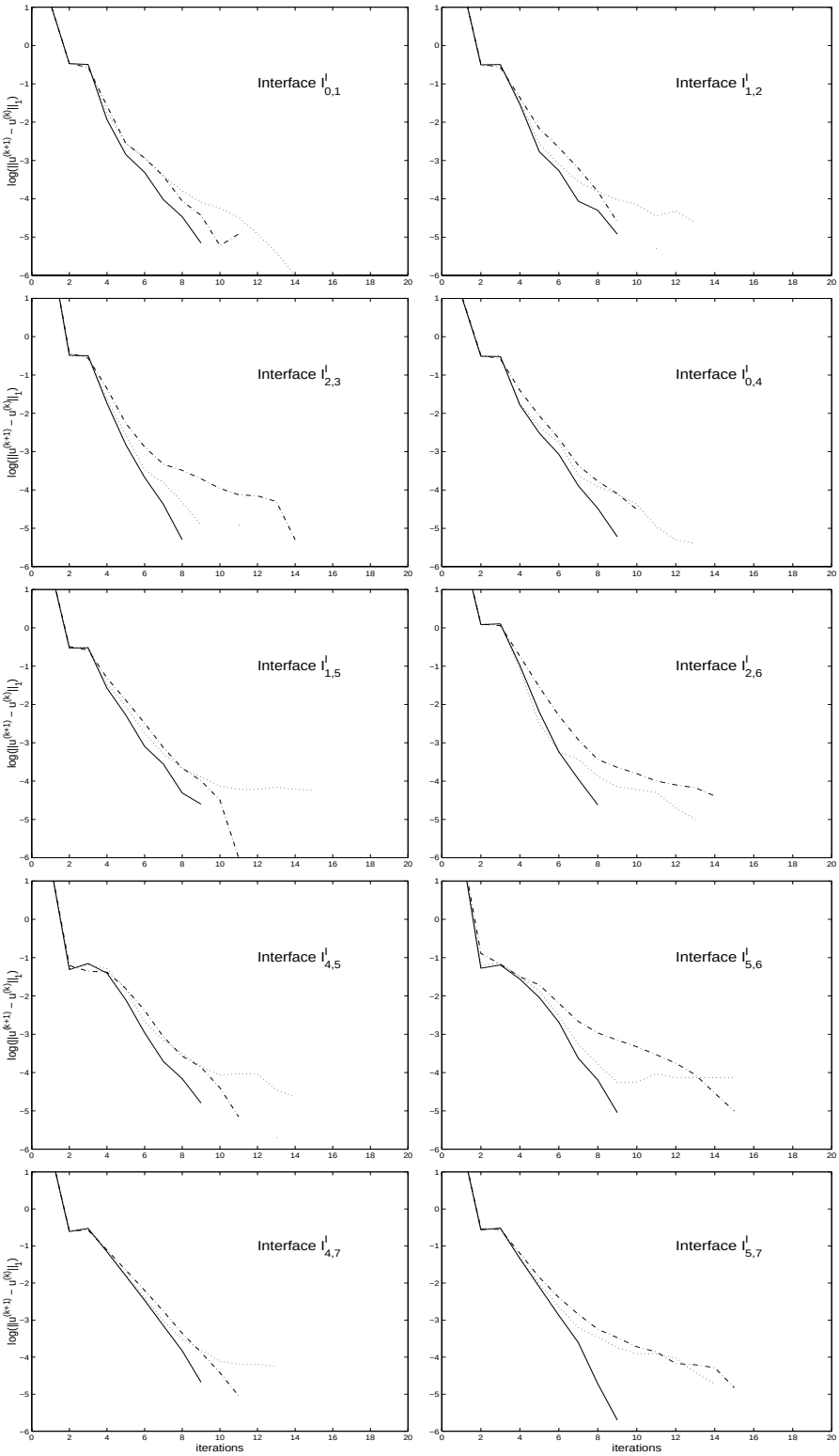


Figure 6.6: Reduction of the L_1 norm of the difference of two successive iterants on each interface using the **ROB** scheme with the optimum values for the λ_i's for PDE1. Solid lines, dotted lines and dash-dotted denote data using the 5-point-star finite difference scheme, the collocation and the finite element respectively.

Interface segment	**ROB** relaxation parameter	**AVE** relaxation parameters	
	λ_i	a_i	β_i
1	5.2248	0.5695	0.4305
2	4.2360	0.5557	0.4443
3	4.4721	0.5419	0.4581
4	5.0000	0.4721	0.5279
5	5.0003	0.4444	0.5556

Table 6.2: The theoretically determined "optimal" values of the interface relaxation parameters used to obtain the data associated with the dotted-dashed lines in Figures 6.7-6.10.

6.4 PDE2 and PDE3: Two Linear PDEs with General Decompositions

We now move into PDE2 which was solved using the same configuration as the one described for PDE1 above with only the following two differences:

First we discretize the domain Ω^{II} using triangular elements with $h = 0.1$ on all subdomains. On this we used the ELLPACK's Finite Element discretization module which is also an $O(h^2)$ scheme.

Secondly to estimate the "optimal" values for the relaxation parameters we use the "approximate" cartesian decomposition associated with Ω_I by utilizing our one-dimensional theoretical results in the following way: First we map the interface segments of the general decomposition that are not straight lines parallel to x- or y-axis to their "closest" interface lines on the "approximate" decomposition. This map can be done using geometry or computational geometry tools and procedures coupled with appropriate objective functions. Here we have done it using a straightforward naïve way. Specifically, we use the corresponding PDE operators on each subdomain and for interface $I^{II}_{0,1}$ the one-dimensional partitioning [0,2][2,10], for interfaces $I^{II}_{0,3}$ and $I^{II}_{2,3}$ the decomposition [0,2][2,8], for interface $I^{II}_{1,2}$ the decomposition [0,3], while for interface $I^{II}_{0,2}$ we just took the average of the values from interfaces $I^{II}_{0,1}$ and $I^{II}_{0,3}$. The values determined by the above procedure are shown in Table 6.2.

Similarly as for PDE1 we present in Figures 6.7–6.8 the convergence histories of the norms of the successive iterants and the relative errors for PDE2 during the first 20 iterations for both the **ROB** and **AVE** schemes and different values for the parameters λ. We first note that by comparing Figures 6.2-6.5 with 6.7-6.10 one sees that both the rate of convergence and the effectiveness of the parameters of the Interface Relaxation methods are very similar for both the PDE1 and PDE2 regardless of the fact that these two problems differ on many

			ROB relaxation					
	Single Domain		$\lambda = 1$		$\lambda = 2$		$\lambda = opt$	
	L_∞	L_2	L_∞	L_2	L_∞	L_2	L_∞	L_2
Ω_0^{II}	.1550E-02	.7472E-03	.4242E-02	.1630E-02	.3792E-02	.1634E-02	.3425E-02	.1661E-02
Ω_1^{II}	.8455E-04	.7321E-04	.1449E-03	.7212E-04	.1281E-03	.7041E-04	.1008E-03	.6789E-04
Ω_2^{II}	.1186E-03	.1307E-03	.2269E-03	.1249E-03	.2143E-03	.1228E-03	.1989E-03	.1218E-03
Ω_3^{II}	.1161E-03	.1093E-03	.2193E-03	.1123E-03	.1899E-03	.1076E-03	.1869E-03	.1085E-03

Table 6.3: The values of the relative error for PDE2 in the L_∞ and L_2 norms of the computed solutions in each subdomain by using the ELLPACK's Finite Element module on each single subdomain (in the second and third columns) and by using the **ROB** interface relaxation method with different relaxation parameters (in the subsequent columns).

parameters (e.g. different decomposition, different type and number of subdomains). We also note that different local PDE discretizations were also used. Furthermore, the common horizontal leveling of the lines in Figures 6.8 and 6.10 represents the PDE discretization error which, in contrast to the PDE1 case, is due to the geometry of the non-rectangular subdomains for PDE2. Both methods converge very fast achieving the discretization error level in about 5 iterations.

To examine the effect of the **ROB** (similar results, not presented here, were obtained for **AVE**) Interface Relaxation procedure on the accuracy of the computed solution for PDE2 we give in Table 6.3 the two norms of the relative errors of the computed solutions obtained by two different ways. Specifically, in the second and third column we list the errors obtained by first imposing Dirichlet boundary conditions with exact right hand sides on all interfaces and then solving the individual **uncoupled** PDE subproblems defined on each subdomain by using the Finite Element discretization with $h = .10$ on each one of them. In the subsequent columns we give the same errors obtained by the experiment with which we obtained the data in Figure 6.8. As is seen the relaxation slightly reduces the accuracy indicating that the constant involved in the convergence rate of **ROB** is significantly small.

To investigate the effect of the space discretization parameter h on the rate of convergence we present in Figures 6.11 and 6.12 the history of the norm of the successive iterants and the norms of the relative errors, respectively, for PDE2 using the **ROB** scheme with $\lambda_i = 4$ for $i = 1, \ldots, p-1$. The dotted lines represent data with $h = .20$ and the solid lines data with $h = .10$. It is seen that even though we approximately double the number of elements this has virtually no effect on the convergence.

To conclude the presentation of our experimental data for PDE2 we plot in Figure 6.13 the contours of the computed solution after the first four iterations.

We switch now to PDE3 and present in Figure 6.14 the reduction of the norm of the

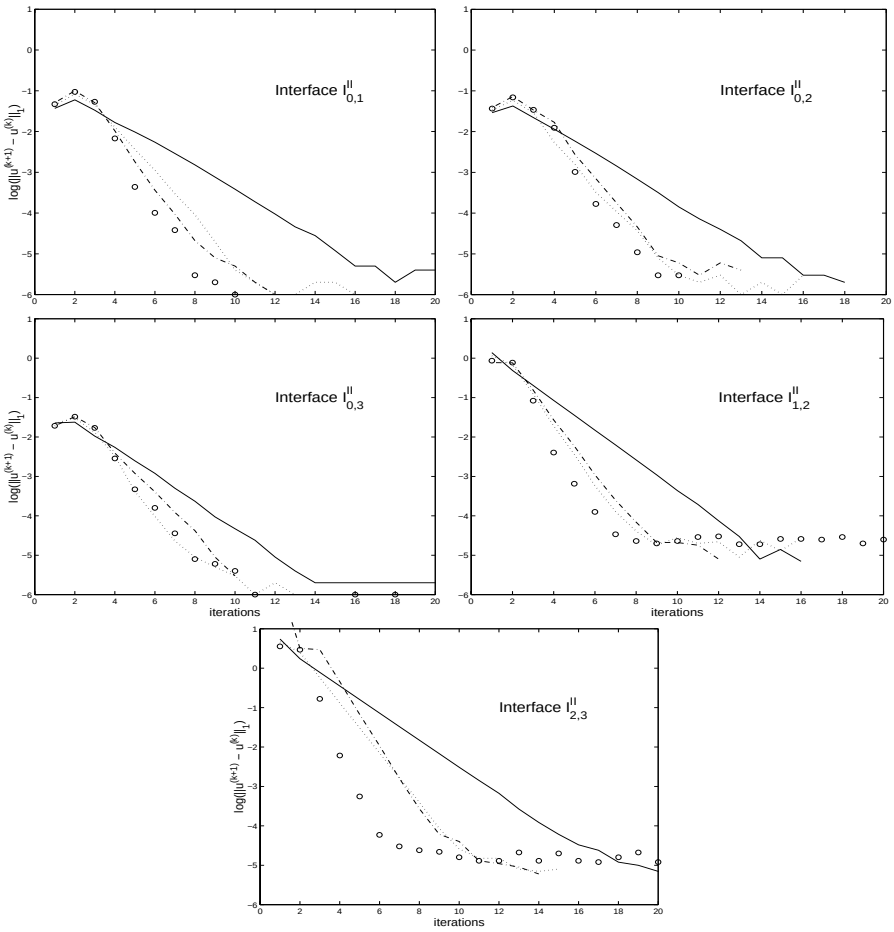


Figure 6.7: Reduction of the L_1 norm of the difference of two successive iterants on each interface using the **ROB** scheme for PDE2. Solid lines, dotted lines and circles denote data using $\lambda_i = 1$, $\lambda_i = 2$ and $\lambda_i = 4$ for $i = 0, \ldots, 4$ respectively. The dash-dotted lines denote data using the theoretically determined optimum values for λ_i's shown in Table 6.2.

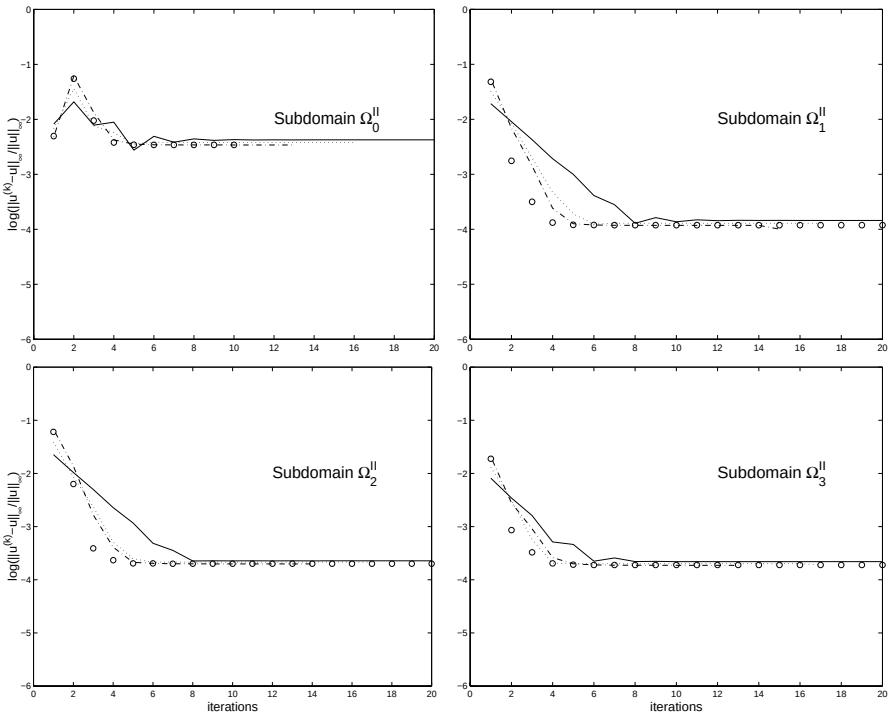


Figure 6.8: Reduction of the L_∞ norm of the relative errors in each subdomain using the **ROB** scheme for PDE2. Solid lines, dotted lines and circles denote data using $\lambda_i = 1$, $\lambda_i = 2$ and $\lambda_i = 4$ for $i = 0, \ldots, 3$ respectively. The dash-dotted lines denote data using the theoretically determined optimum values for λ_i's shown in Table 6.2.

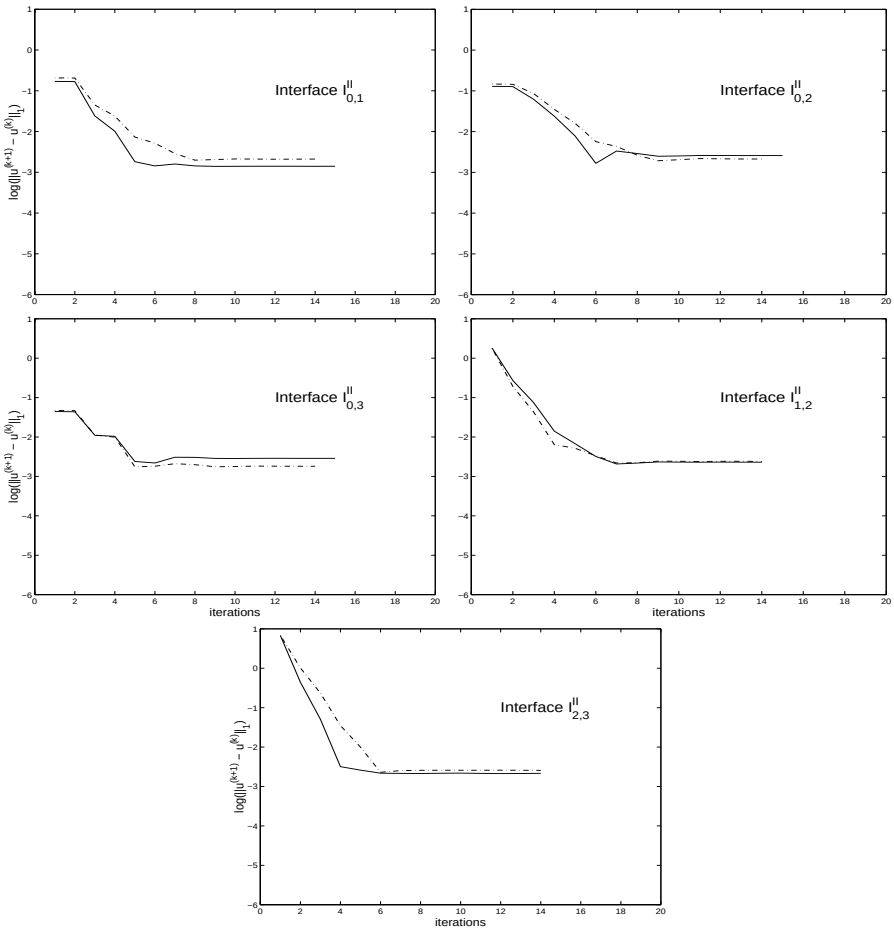


Figure 6.9: Reduction of the L_1 norm of the difference of two successive iterants on each interface using the **AVE** scheme for PDE2. The solid lines, denote data using $\alpha_i = 0.5$, for $i = 0, \ldots, 4$ and the dash-dotted lines denote data using the theoretically determined optimum values for the α_i's and β_i's as these are shown in Table 6.2.

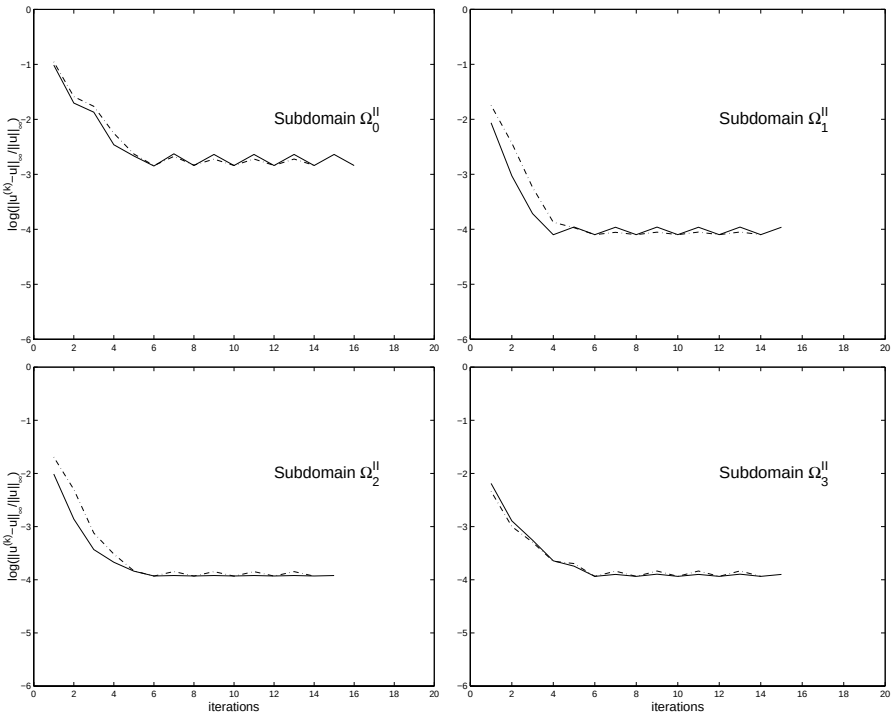


Figure 6.10: Reduction of the L_∞ norm of the relative errors in each subdomain using the **AVE** scheme for PDE2. The solid lines, denote data using $\alpha_i = 0.5$, for $i = 0, \ldots, 4$ and the dash-dotted lines denote data using the theoretically determined optimum values for the α_i's and β_i's as these are shown in Table 6.2.

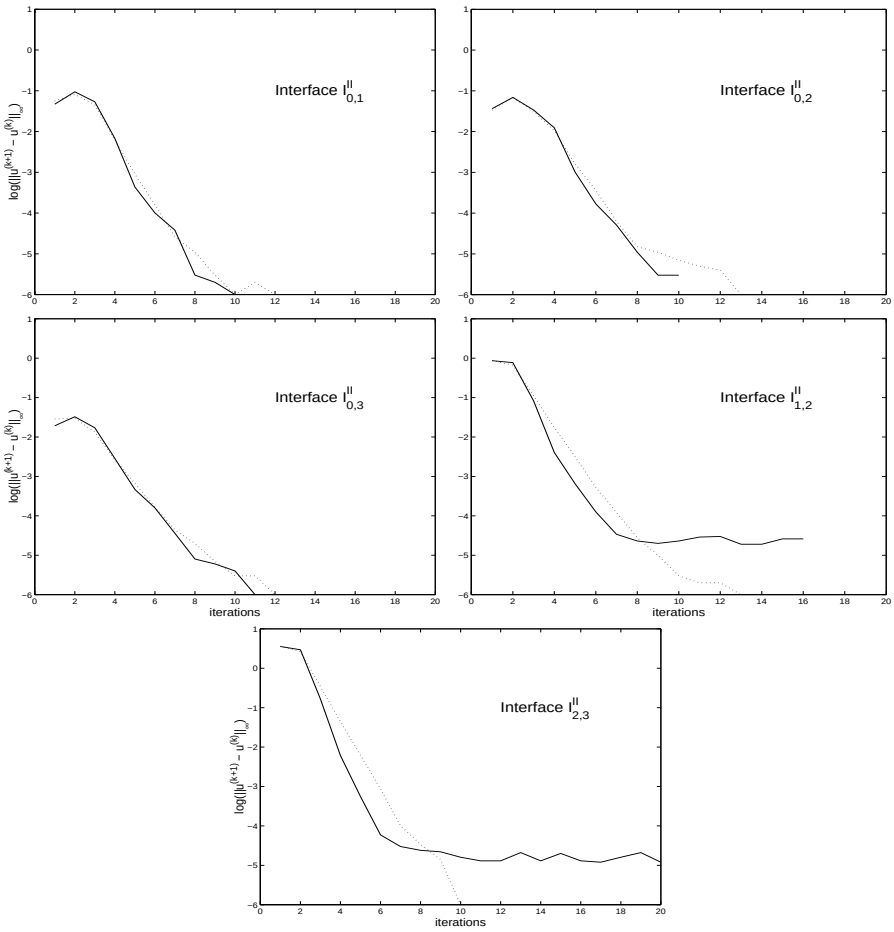


Figure 6.11: Reduction of the L_1 norm of the difference of two successive iterants on each interface using the **ROB** scheme and the Finite Element module on all subdomains for PDE2 with $\lambda_i = 4$ for $i = 0, \ldots, 4$. Solid lines denote data using as space discretization parameter $h = .10$, the dotted lines denote data using $h = .20$.

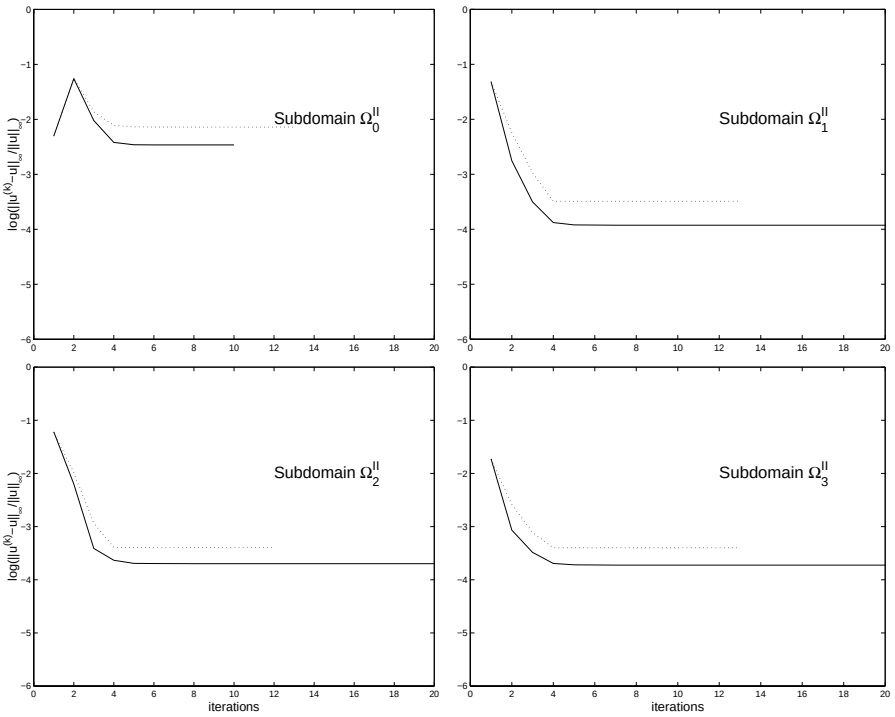


Figure 6.12: Reduction of the L_∞ norm of the relative errors in each subdomain using the **ROB** scheme and the Finite Element module on all subdomains for PDE2 with $\lambda_i = 4$ for $i = 0, \ldots, 4$. Solid lines denote data using as space discretization parameter $h = .10$, the dotted lines denote data using $h = .20$.

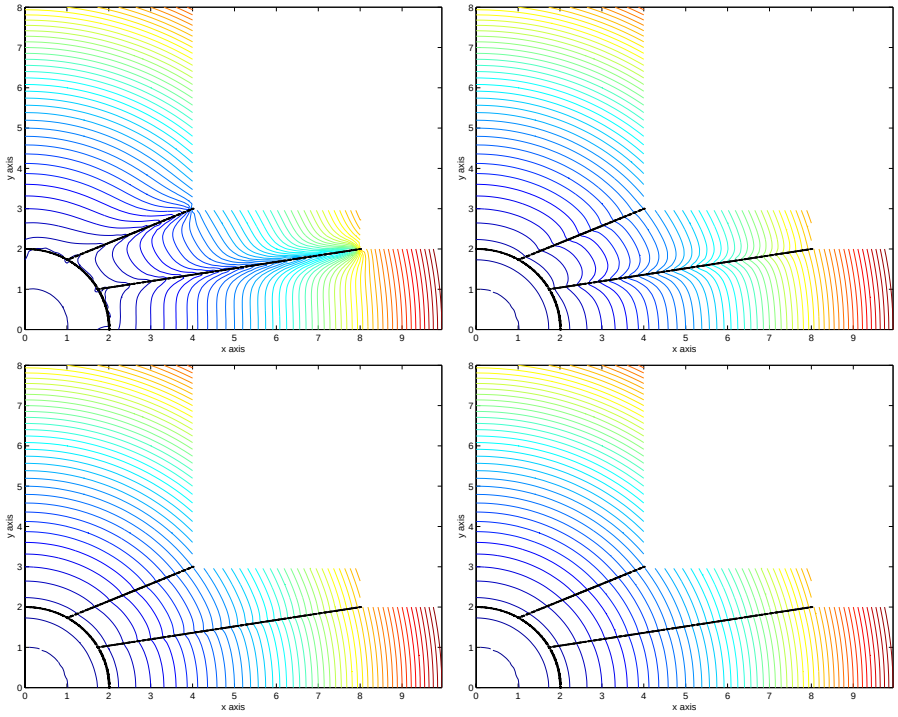


Figure 6.13: Contour plots of the first 4 iterants $(u^k, k = 1, 2, 3, 4)$ of PDE2 computed by the **ROB** scheme using the theoretically determined optimum relaxation parameters.

successive iterants of the **ROB** scheme for the different values of the λ_i's. The configuration used to obtain these data was the same as the one for PDE2. The "optimum" values determined from the theoretical analysis were $\lambda_0 = 0.640536$, $\lambda_1 = 0.6510$, $\lambda_2 = 0.66155$, $\lambda_3 = 0.921287$ and $\lambda_4 = 1.08033$. As is clearly seen these values lead to the fastest convergence. In addition, by comparing the slopes of the lines in Figures 6.7 and 6.14, we see that the rate of convergence for the two problems PDE2 and PDE3 that differ both on the size of the subdomains and the PDE operators applied on each one of them, is approximately the same. Finally, we also see the effectiveness of the scheme, in the sense that it offers full accuracy in less than 20 iterations.

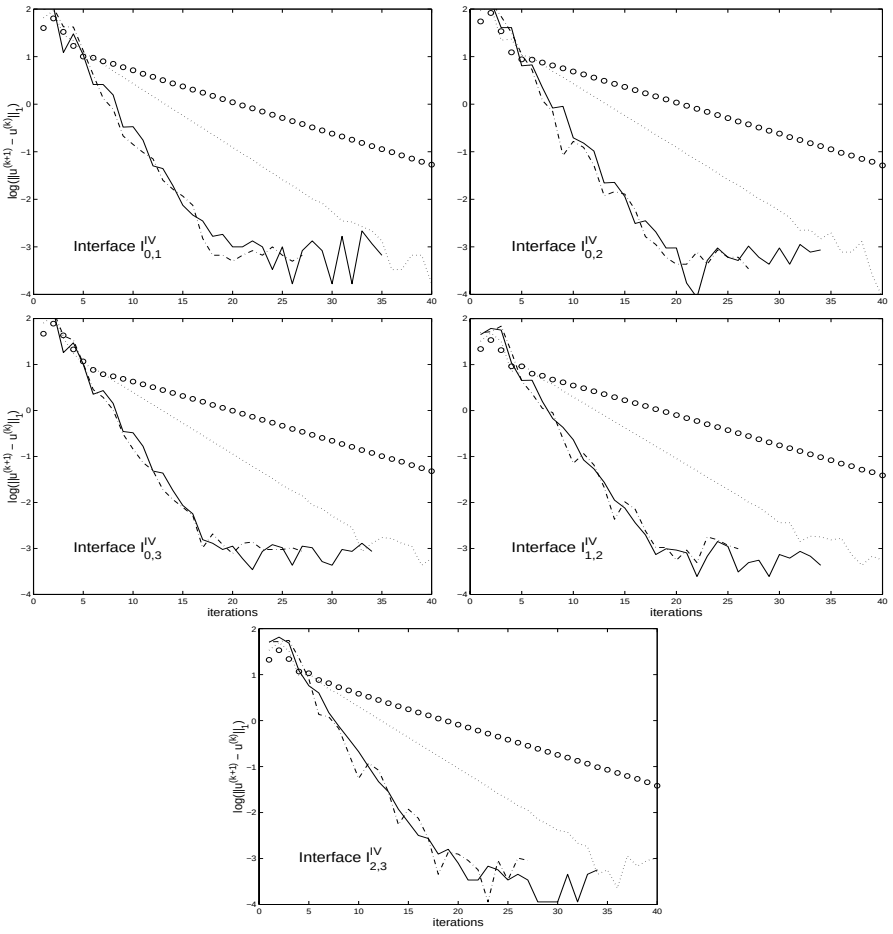


Figure 6.14: Reduction of the L_1 norm of the difference of two successive iterants on each interface using the **ROB** scheme for PDE3. Solid lines, dotted lines and circles denote data using $\lambda_i = 1$, $\lambda_i = 2$ and $\lambda_i = 4$ for $i = 0, \ldots, 4$ respectively. The dash-dotted lines denote data using the theoretically determined optimum values for λ_i's shown in Table 6.2.

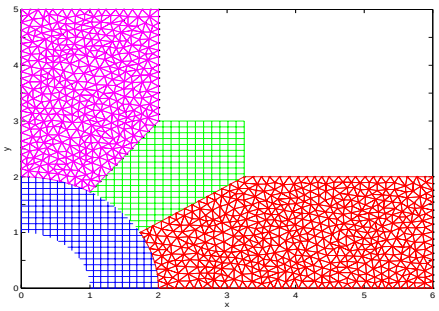


Figure 6.15: Space discretization for the PDE4 using cartesian grids on subdomains Ω_0^{IV} and Ω_2^{IV} and triangular elements on subdomains Ω_1^{IV} and Ω_3^{IV} with discretization respectively with $h = .1$ in both cases

6.5 PDE4: A Non-Linear PDE with General Decomposition

For the experiments in this section we have discretized subdomains Ω_0^{IV} and Ω_2^{IV} using rectangular elements, and subdomains Ω_1^{IV} and Ω_3^{IV} using triangular elements as it is shown in Figure 6.15. In all four subdomain discretizations we have used $h = 0.1$. We have linearized the PDE operators in 6.2.4 using the Newton's method and the resulting linearized PDEs were discretized using the Ellpack's 5-point-star module for subdomains Ω_0^{IV} and Ω_2^{IV} and the finite element module for subdomains Ω_1^{IV} and Ω_3^{IV}. This way subdomains 0,1,2 and 3 resulted in 400, 570, 400, and 433 linear equations (and unknowns) respectively. These linear systems were solved using the Gauss Elimination module for banded matrices.

In Figure 6.16 we plot in three dimensions the computed by the **ROB** scheme solution after iterations 1, 2, 3 and 25 and in Figure 6.17 the convergence history in terms of the difference of successive iterants with 4 different set of values of the relaxation parameters. The ones represented by the dash-dotted lines are for the "optimum" values determined by using the "approximate" cartesian partition (defined by Ω^{III}) and the associated procedure used for PDE2 and described in section 6.4. These values were $\lambda_0 = 0.5338$, $\lambda_1 = 0.5977$, $\lambda_2 = 0.6615$, $\lambda_3 = 1.0357$ and $\lambda_4 = 1.0474$. It is seen in Figure 6.17 that these values lead to divergence. This is explained from the fact that the differential operators in the subdomains are far away from the Helmholtz model problem we have analyzed theoretically. Other than that the scheme seems to converge fast enough, offering two correct decimal digits in about 5 iterations.

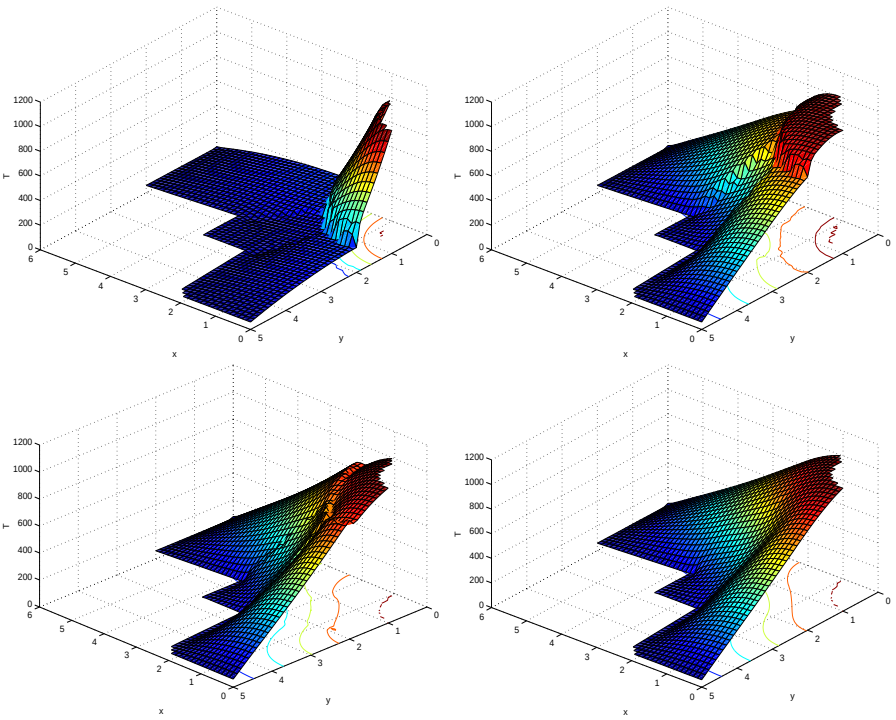


Figure 6.16: Three-dimensional plots of the 1st, 2nd, 3rd and 25th iterants $(u^k, k = 1, 2, 3, 25)$ of PDE4 computed by the **ROB** scheme using $\lambda_i = 4$ for $i = 0, \ldots, 4$.

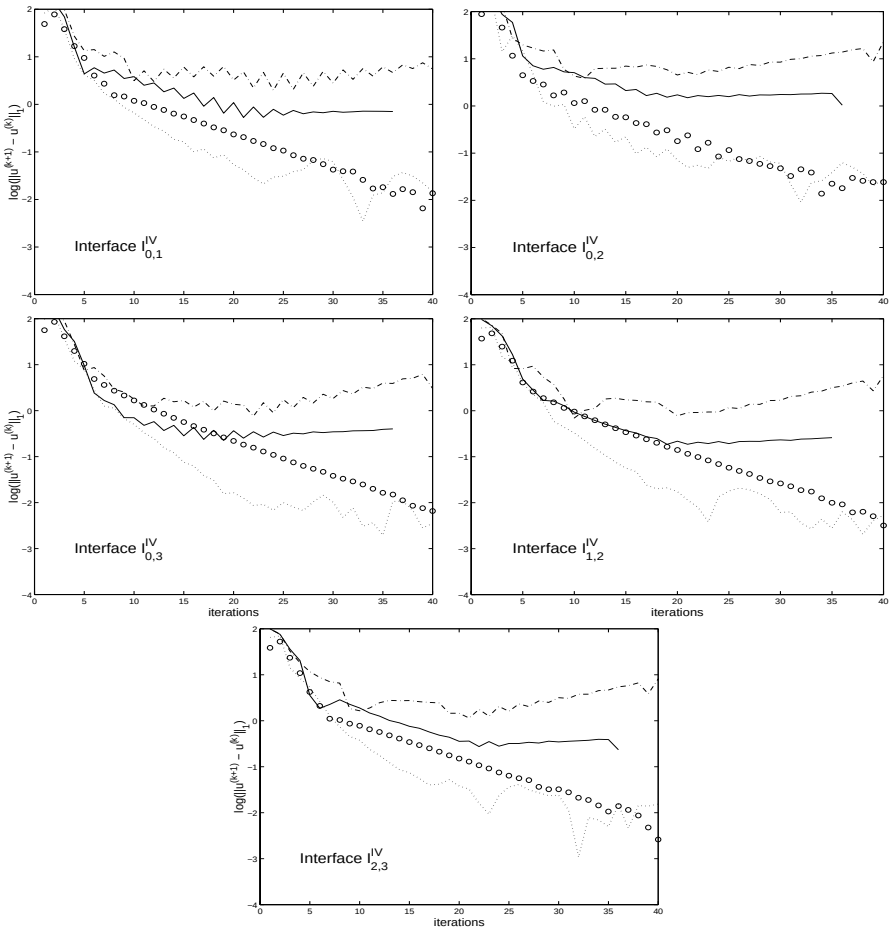


Figure 6.17: Reduction of the L_1 norm of the difference of two successive iterants on each interface using the **ROB** scheme for PDE4. Solid lines, dotted lines and circles denote data using $\lambda_i = 1$, $\lambda_i = 2$ and $\lambda_i = 4$ for $i = 0, \ldots, 3$ respectively. The dash-dotted lines denote data using the theoretically determined optimum values for λ_i's.

Chapter 7

Conclusions and Future Plans

Abstract

The general conclusions drawn from this Thesis are presented. A list of further research directions are also presented and several on-going research efforts are briefly described.

7.1 Conclusions

We propose a methodology for building a powerful simulation engine for composite PDE problems of elliptic type by properly combining existing models and software components. Our approach enjoys the following properties:

Problem simplification. It dramatically simplifies the complexity of the physical problem by (1) considering subproblems that involve simpler local physical rules acting on simpler geometries, and (2) providing a convenient abstraction of the modeling and solution processes while simultaneously providing a modeling practice that yields a closer representation of the physical world.

Reduction in software development time. It drastically reduces the time to develop a simulation engine by permitting the heavy reuse of legacy scientific software. Note that the basic (software) build-in blocks of a system are typically already existing problem solving environments.

Parallelism and scalability. It allows heterogeneous distributed resources to be harnessed by using a naturally parallel and highly scalable approach. The network of collaborative solvers is easily mapped onto a wide variety of distributed high performance computing architectures. Challenging parallelization issues like data partitioning, assignment and load balancing are easier handled on the level of physics rather than on the level of computational abstractions.

Cooperability and adaptivity. Due to natural problem partitioning, collaboration of groups of modelers is naturally promoted and also simplifiable at each stage of the modeling process. Each modeler can select the particular component of the physical artifact that fits his expertise, build his own local model and adapt any of the local parameters involved, in a dynamic manner, at any stage of the computation. The only information he needs to give his neighbors, for collaboration, is the information on relaxation at the interfaces.

Numerical efficiency. It increases the efficiency of the overall numerical scheme by allowing one to use the most appropriate numerical method for each particular subproblem.

This Thesis focuses on the Interface Relaxation methods that consist the core of the system. Their methodology appears to be an attractive alternative to the traditional domain decomposition approach in particular when modeling multiphysics/mul- tidomain problems are considered. Nevertheless, their theoretical analysis appears to be challenging and already radically different analysis techniques have appeared in the literature ranging from numerical linear algebra to finite element and furthermore to continuous Banach spaces.

For the theoretical analysis in this Thesis we have restricted ourselves to the Helmholtz equation in one- and two-dimensions. Specifically, we have proved the convergence of three

Interface Relaxation methods at continuous level and one at discrete level. We have considered 1-dimensional domain partitions in an arbitrary number of subdomains of different sizes. In all cases we have shown that the rate of convergence is independent of all the parameters of the local PDE solvers. Specifically, the convergence does not depend on the local space discretization step h, or the local PDE discretization scheme. This is true even for the **GEO** method, a variation of which [45] has been shown to have convergence depended on h. In contrary, in our formulation and analysis of **GEO** , even at discrete - finite difference - level, we have proved its independence of h. For all the three methods analyzed we were able to determine regions of convergence. For the **ROB** and **AVE** methods we have obtained "optimum" values for the relaxation parameters in the sense that they minimize the spectral radius of the iteration matrix. All theoretical results have been confirmed with extensive numerical data. Additional experiments show that these results hold for more general problems and show certain characteristics of the methods. Specifically, we have observed that:

- The **AVE** method seems to converge faster in most cases. Nevertheless, its Neumann step reduces its robustness.
- The **ROB** method is the most robust and converges for all of the problems we have tried. This is in agreement with the fact that this method converges for any positive value of its parameters.
- The **GEO** appears to be more sensitive to its parameters than the other two in the sense that small variations to the values of these parameters lead either to large increase of the number of iterations needed for convergence or even to divergence.

Most of the methods converge fast enough so they do not require preconditioning. In fact the "global" precondition takes, in the Interface Relaxation era, a new notion as being raised to solely the PDE level discarding, thus, the particular matrices and other data structures. This way, preconditioners are easy to be computed, in a natural straightforward way, by means of "easy" approximations of the local PDE subproblems and might further accelerate the convergence.

It is important to make clear that the proposed collaborating method is not to replace existing numerical methods and/or software but to provide the general framework, the necessary theoretical results and the practical tools to build on top of them by properly integrating them into a common distributed solving environment and to effectively orchestrate them in a simple and effective way.

The task of designing, analyzing, implementing and evaluating Collaborative PDE solvers is obviously not accomplished. Several interesting, both mathematically and computationally, problems are still open. The general directions that we envision the future research will focus are listed in Section 7.2. In Section 7.3 we briefly describe specific research task that we hope to accomplish in the near future.

7.2 General Future Research Directions

We believe that research for collaboration in the PDE solving procedure will attract an increasing number of researchers in the future. In particular we expect results soon in the following research areas.

Theoretical analysis at PDE level: As mentioned previously in this Thesis Interface Relaxation is an iteration defined at the continuum (mathematical) level; its convergence properties are mainly a question of mathematical analysis, not of numerical analysis. Nevertheless, a generic analysis similar to more than one century old Schwarz splitting result for overlapping domain partitioning is still lacking. The only study so far that has its flavor is the one by P.-L. Lions [40] associated with the **ROB** scheme which definitely needs to be extended for other relaxers. At any rate, recent advances in the theory of PDEs will be required. As an example we state the new technique presented in [66] which we believe will prove itself a powerful theoretical tool for the analysis of Interface Relaxation for general linear and nonlinear PDE problems. This kind of analysis is based on a new line of reasoning that will provide new intuition about the dynamics of Interface Relaxation.

More space dimensions: In principle, all of the Interface Relaxation methods considered can be extended to more than two dimensions. Nevertheless, the convergence analysis of all of them is expected to be more complicated and the implementation of some of them will be much more difficult than of others. For example one should expect that the implementation of the **AVE** method will be easily extensible to three dimensions while it seems to us that additional work is needed for the proper implementation of the **SHO** method. Finally, for all the methods the interpolation problems involved will be much more challenging since it will require generic two-dimensional interpolants of particular characteristics in accuracy and in preservation of shape.

Jump conditions on the interfaces: Many physical problems, (e.g. alloy solidification [14]) involve interfaces where jump conditions of various kinds are imposed. It seems that both the general framework and our implementation of the relaxation procedures can

be extended for such problems easily. It also seems that the analysis (at least for model problem cases) is also easily extendable. Recent studies for linear [15] and nonlinear problems [73] in this direction are very encouraging.

Interpolation and un-matching grids: To implement the mediators (see Sections 5.2 and 5.3) in the case where the grid or meshes of two neighboring subdomains do not match on their interface one needs to specify how the intrepolant will be constructed so it accomplishes the accuracy requirements and on which "closest" (see Section 5.3) points will be defined on. This task is more crucial for three-dimensional PDE problems. We have evidence that the recent theoretical results and software developed by Ron and C. DeBoor [16] will provide a strong background to solve this problem that definitely requires investigation.

General PDE problems and advanced relaxers: As the composite PDE problem gets more complicated and difficult the need of more advanced relaxers is increasing. In fact, recently many mathematicians from different research disciplines try to properly couple PDE problems of different type. Some of such advanced relaxers are listed in Section 2.3. An excellent and up-to-date source of other such relaxers for general PDE problems could be the recent book [49] regardless of the fact that it formulates and analyzes these relaxers from the domain decomposition viewpoint.

For the implementation viewpoint we expect more work on:

Integration of legacy PDE solvers: So far the SciAgents system contains as lega- cy solvers for the subdomain problems only the ELLPACK modules. Nevertheless, it has been designed as an open architecture and we believe that it clearly defines an interface that allows the easy incorporation of other external legacy solvers too.

Systematic performance evaluation of relaxers: Using Interface Relaxation o- ne faces several crucial questions (e.g., which relaxation scheme to select for a given problem? Where to place the interfaces?, etc). These questions remain for most of the practical problems unanswered. Therefore a systematic performance evaluation study that uses a knowledge based system like the one in [72] is needed to help us extract practical knowledge.

Further exploitation the Agent technology: SciAgents could benefit more from agent computing. Specifically, one might decide to allow selected agents to travel through the network. This does not seem to be feasible for the `PDESolver` agents (due to their large state) but it might be the appropriate choice for the `PDEMediator` agents.

Furthermore software agents might add capabilities for automatic adaptation and speculative computing to SciAgents.

7.3 On-going Research Efforts

In this final section we present those problems that we have clearly identified, we have made some preliminary investigations through this thesis and we believe that are both important and tractable.

Estimating Interface Relaxation Parameters in the General Case

As clearly seen in the previous Chapters most of Interface Relaxation schemes involve relaxation parameters that can significantly improve the convergence properties of the various Interface Relaxation methods if their values are properly chosen. These parameters depend on many components (PDE operator, PDE domain, splitting, ...) of the original problem. This, unfortunately makes the selection of "optimum" values for the relaxation parameters a hard and challenging problem.

Several papers have been devoted in theoretically obtaining optimum values for the interface relaxation parameters [75, 77]. Nevertheless the analysis is done for model problems and subdomain splittings and although they provide important information fail to assist a non-expert user to select values for those parameters for problems with moderate complexion. When this complication is increased even experts can not effectively select parameters.

The main objective of one of our future studies will be to provide an adaptive heuristic mechanism for automatically selecting "good" relaxation parameters for general differential equations and domain decompositions. This proposed mechanism uses Automatic Differentiation (AD) and utilizes the existing analytically estimated values [56].

Specifically, as it has been already observed [58, 24] the value of the relaxation parameters plays a crucial role in the convergence of the various interface relaxation methods. Fine tuning of these parameters can greatly improve the rate of convergence while a bad choice might lead to unacceptable slow convergence or even divergence. They depend on both the eigenvalues of the differential operator and the geometric and decomposition parameters of the domain. We next present a framework to adaptively adjust the values of the relaxation parameters α_i and β_i of the averaging interface relaxation method (**AVE**) towards its optimal value. The methodology of the proposed scheme is general enough to treat complicated (possibly non-linear) differential operator and arbitrary domain decompositions in one, two or three dimensions, has already been applied with success to the successive over-relaxation

(SOR) iterative method [29] and can be readily applied to virtually any interface relaxation methods.

Consider the particular interface $\Gamma_{i,j} \in R^d, d = 0, 1, 2$ between subdomains Ω_i and Ω_j. The interface relaxation equations on $\Gamma_{i,j}$ are given by

$$\mathsf{u}'(x) = \beta_{i,j} \left(\frac{\partial u_i^{(2k)}(x)}{\partial \nu} - \frac{\partial u_j^{(2k)}(x)}{\partial \nu} \right) + \frac{\partial u_j^{(2k)}(x)}{\partial \nu} \ , \quad x \in \Gamma_{i,j} \tag{7.3.1}$$

where $\frac{\partial}{\partial \nu}$ represents the outward normal derivative to $\Gamma_{i,j}$ and

$$\mathsf{u}(x) = \alpha_{i,j} \left(u_i^{(2k+1)}(x) - u_j^{(2k+1)}(x) \right) + u_j^{(2k+1)}(x) \ , \quad x \in \Gamma_{i,j}. \tag{7.3.2}$$

We are interested in investigating how different values of the relaxation parameters affect the quality of the iterants $\mathsf{u}(x)$ and $\mathsf{u}'(x)$. We can achieve that by computing the derivative of a certain quantity, that reflects the quality of the iterants, with respect to these relaxation parameters and use it to update their values. We can consider two such quantities: the L_2 norm of the residual $r = \| D\mathsf{u}(x) - f(x) \|$ on the interface $\Gamma_{i,j}$ or the size of the Gerschgorin disc associated with that particular interface as considered in the Chapters 2-4. We should mark that the first choice might not work for interfaces where there are discontinuities and the second one assumes that the iteration matrix M [57] has been obtained.

We first concentrate on the first option. Assuming that we are able to compute the sensitivity of the residual with respect to $\alpha_{i,j}$ as its partial derivative $r_\alpha \equiv \frac{\partial r}{\partial \alpha_{i,j}}$ after performing the Neumann (7.3.1) relaxation then we can use a secant method to update the value of the relaxation parameter in the next iteration. Specifically, we obtain the following simple adaptive formula

$$\alpha_{i,j}^{(k+1)} = \alpha_{i,j}^{(k)} - r_\alpha^{(k)} \frac{\alpha_{i,j}^{(k)} - \alpha_{i,j}^{(k-1)}}{r_\alpha^{(k)} - r_\alpha^{(k-1)}}, \tag{7.3.3}$$

while the equivalent formula associated with the Dirichlet step (7.3.2)

$$\beta_{i,j}^{(k+1)} = \beta_{i,j}^{(k)} - r_\beta^{(k)} \frac{\beta_{i,j}^{(k)} - \beta_{i,j}^{(k-1)}}{r_\beta^{(k)} - r_\beta^{(k-1)}}. \tag{7.3.4}$$

can be derived similarly. Obviously there is no way to analytically calculate r_α since there is no closed form for r, as a function of the $\alpha_{i,j}$'s and $\beta_{i,j}$'s, available. Nevertheless such function is given in a form of an algorithm while relaxing each interface. Therefore a possible solution is to use *automatic differentiation* [51] to obtain "good" values for the relaxation parameters. In the automatic differentiation framework one applies a process

that generates code which uses lookup tables and mechanically applies the chain rule to compute the derivative of a function given in a form of a computer program. Among the various software infrastructures that exploit the automatic differentiation idea we have select the ADIFOR package [2] for our implementation.

To move to our second option we consider the Gerschgorin disk associated with the ith interface and try to simultaneously minimize its area and move its center as closely to the origin as possible. This can be done by minimizing the sum of the absolute value of the elements in the ith row of the iteration matrix M with respect to α and β. Thus we can use automatic differentiation to calculate the Jacobian matrix and solve the associated equations for the optimum values of the relaxation parameters.

Reaction Diffusion Equations

Work on domain decomposition type methods so far has focused on elliptic problems. However, time dependent PDEs have the following attractions:

- They avoid most of the problems concerning the convergence of the iterated relaxation methods.
- All real world problems are time dependent anyway.

Primitive interface relaxation schema for time dependent PDEs have been proposed several decades ago (e.g. [52]). They are known as *time marching* methods and their analysis is very similar to ODEs (implicit methods allow one to use elliptic solvers).

To the best of our knowledge, the Interface Relaxation idea has not yet been applied to time depended PDE problems. This is in contrast to the well established Schwartz methods which have been widely used. In particular recently a Wavefront Relaxation procedure [25] has been analyzed and very effectively applied for reaction diffusion problems.

Our interface relaxation framework, considered for the elliptic case in this thesis, can be naturally extended for composite reaction diffusion equations through such a Wavefront Relaxation procedure too. For this we start with a domain defined as the union of a set of subdomains, i.e. $\Omega = (\bigcup_{i=1}^{p} \Omega_i) \times [0,T)$, and consider the composite reaction diffusion equation on the above domain Ω,

$$\frac{\partial u}{\partial t} = L_i u \quad on \quad \Omega_i \times [0,T), \quad for \quad i = 1,\ldots,p \tag{7.3.5}$$

$$u = u_a \quad on \quad \Omega_i,\ i = 1,\ldots,p \ at \quad t = 0$$

$$Bu = u_b \quad on \quad \partial\Omega \tag{7.3.6}$$

$$\tilde{I}u_- = \tilde{I}u_+ \quad on \quad \Gamma_{i,j} \ for \ i,j = 1,\ldots,p \ with \ i \neq j,$$

where B is the boundary operator representing Dirichlet, Neumann or mixed boundary conditions and where u_a and u_b are given functions representing the initial and boundary values of the problem respectively. $\Gamma_{i,j} \equiv \partial\Omega_i \bigcap \partial\Omega_j$ denotes the interface between subdomains Ω_i and Ω_j, u_- and u_+ represent the solution on the left and on the right of the interface respectively, $\tilde{I}$ is a given interface condition operator and L_i is a space differential operator which for simplicity we set to

$$L_i u(x,t) = \sum_{i=1}^{s} \left\{ c_i^2(x,t) \frac{\partial^2 u(x,t)}{\partial x_i^2} \right\} + f_i(u) \tag{7.3.7}$$

where $x = (x_1, \ldots, x_s)$, $x_i \in \mathbb{R}$ for $i = 1, \ldots, s$.

In the case that a unique solution u to the above problem exists (this obviously depends on certain continuity, smoothness and boundness conditions) we are interested in deriving a procedure that, by solving the individual PDEs defined by each one of the reaction diffusion equations in 7.3.5 subject to some interface conditions $\tilde{I}$, generates a sequence of local solutions with each one of them converging to the restriction of the global solution u on the associated subdomain. Such a procedure can be easily defined by the following iterative scheme whose $(k+1)st$ iteration is defined for $i = 1, \ldots, p$ by the following set of reaction diffusion problems

$$\begin{aligned}
\frac{\partial u_i^{(k+1)}}{\partial t} &= L_i u_i^{(k+1)} \quad \text{on} \quad \Omega_i \times [0,T)\,, \\
u_i^{(k+1)} &= u_a \quad \text{on} \quad \Omega_i, \quad \text{at} \quad t = 0, \\
B u_i^{(k+1)} &= u_b \quad \text{on} \quad \partial\Omega \backslash \Gamma_{i,j} \quad \text{for } j = 1, \ldots, p \text{ with } i \neq j, \\
\tilde{I} u_i^{(k+1)} &= \tilde{I} u_j^{(k)} \quad \text{on} \quad \Gamma_{i,j} \quad \text{for } j = 1, \ldots, p \text{ with } i \neq j,
\end{aligned} \tag{7.3.8}$$

The convergence analysis and the implementation of the above scheme seems to be feasible and is under way. Note that this scheme allows, under certain consistency and stability conditions [6], different time steps on different subdomains in a natural and easy way.

The Biharmonic Equations

An interesting extension of the Interface Relaxation idea that we are currently working on, is its application to fourth order PDE problems. To the best of our knowledge the only Interface Relaxation scheme for fourth order elliptic problems is one based on **SCO**. The formulation, the preliminary analysis of the one-dimensional case and the implementation of this scheme was presented in [26]. As an alternative, one might consider to relax the

interfaces using an extension of **ROB** which seems to be one of the most promising schemes for the second order problems. Let us consider the Biharmonic *clamped rod* model problem define by

$$\begin{aligned} \frac{d^4u}{dx} &= f \quad for \quad x \in \Omega \equiv (-a,b), \\ u(-a) = u(b) = \left.\frac{du}{dx}\right|_{x=-a} &= \left.\frac{du}{dx}\right|_{x=b} = 0. \end{aligned} \tag{7.3.9}$$

Assume now that Ω is split into $\Omega_1 \equiv (-a,0)$ and $\Omega_2 \equiv (0,b)$ and define the equivalent to (7.3.9) problem as follows

$$\begin{aligned} \frac{d^4v}{dx} &= f \quad \text{for} \quad x \in \Omega_1, \qquad \frac{d^4w}{dx} = f \quad \text{for} \quad x \in \Omega_2, \\ v(-a) = \left.\frac{dv}{dx}\right|_{x=-a} &= 0, \qquad w(b) = \left.\frac{dw}{dx}\right|_{x=b} = 0, \end{aligned} \tag{7.3.10}$$

$$\begin{aligned} v(0) = w(0), \qquad & \left.\frac{dv}{dx}\right|_{x=0} = \left.\frac{dw}{dx}\right|_{x=0}, \\ \left.\frac{d^2v}{dx^2}\right|_{x=0} = \left.\frac{d^2w}{dx^2}\right|_{x=0}, \qquad & \left.\frac{d^3v}{dx^3}\right|_{x=0} = \left.\frac{d^3w}{dx^3}\right|_{x=0}. \end{aligned} \tag{7.3.11}$$

From the above one might derive several **ROB** based Interface Relaxation schemes by properly combining the last four conditions (7.3.11) on the interface ($x = 0$) into two conditions. One possibility is to group the values of the functions and the its first derivative in one condition and group the second and third derivatives in the second condition. This leads to the following Interface Relaxation method for the *clamped rod* model problem (7.3.9).

$$\frac{d^4v^{(k+1)}}{dx} = f \quad for \quad x \in \Omega_1, \tag{7.3.12}$$

$$\begin{aligned} v^{(k+1)}(-a) &= \left.\frac{dv^{(k+1)}}{dx}\right|_{x=-a} = 0, \\ \left.\frac{dv^{(k+1)}}{dx}\right|_{x=0} + \lambda v^{(k+1)}(0) &= \left.\frac{dw^{(k)}}{dx}\right|_{x=0} + \lambda w^{(k)}(0), \\ \left.\frac{d^3v^{(k+1)}}{dx^3}\right|_{x=0} + \lambda' \left.\frac{d^2v^{(k+1)}}{dx^2}\right|_{x=0} &= \left.\frac{d^3w^{(k)}}{dx^3}\right|_{x=0} + \lambda' \left.\frac{d^2w^{(k)}}{dx^2}\right|_{x=0}, \end{aligned} \tag{7.3.13}$$

$$\frac{d^4 w^{(k+1)}}{dx} = f \ \ for \ \ x \in \Omega_2, \tag{7.3.14}$$

$$w^{(k+1)}(b) = \left.\frac{dw^{(k+1)}}{dx}\right|_{x=b} = 0,$$

$$-\left.\frac{dw^{(k+1)}}{dx}\right|_{x=0} + \lambda w^{(k+1)}(0) = -\left.\frac{dv^{(k)}}{dx}\right|_{x=0} + \lambda v^{(k)}(0), \tag{7.3.15}$$

$$-\left.\frac{d^3 w^{(k+1)}}{dx^3}\right|_{x=0} + \lambda' \left.\frac{d^2 w^{(k+1)}}{dx^2}\right|_{x=0} = -\left.\frac{d^3 v^{(k)}}{dx^3}\right|_{x=0} + \lambda' \left.\frac{d^2 v^{(k)}}{dx^2}\right|_{x=0},$$

where λ and λ' are the relaxation parameters. A preliminary convergence analysis of the above scheme for one-dimensional model problems has been already carried out but it remains to be extended for more general problems and decompositions. It also remains to be determined if it is preferable to use any of the other possible **ROB** schemes.

Developing Initial Guesses

All the Interface Relaxation schemes presented require an initial guess of the solution on the interfaces to start with. One might decide to start the iterations with a zero initial guess or use a naïve way of extending the given boundary values. This latter approach has been used for all the experiments presented in Chapters 2, 3 and 4. Nevertheless, we expect that for many problems with discontinuities or for three–dimensional problems a more reasonable initial guess will be needed. Such a guess can be obtained by various approximation methods that extend the boundary conditions into the interior of the domain using either a blending technique [54], a Monte Carlo method or a wavelet approach [30]. In any case, better initial guesses provide faster solutions and more robust computations.

Applications

The Collaborative PDE solving framework has been proposed to solve important and complex multi-physics, multi-component practical problems. In particular there are currently two on-going projects that heavily use the proposed framework and parts of the software components of our implementation which was presented in Chapter 5. These are the GasTurbin[1] project [41], and the project on Collaborative Air Pollution Models [59, 70, 71]. Preliminary software system designs, theoretical analysis and experimental studies for both the above two projects are so far very encouraging.

[1] `http://www.cs.purdue.edu/research/cse/gasturbn/`

Appendix A

The Algorithms for Seven Relaxation Methods.

Algorithm 1: The Dirichlet/Neumann Averaging (**AVE**) Method

$$g_i^i = \beta_i \left.\frac{du_i^{(2k)}}{dx}\right|_{x=x_i} + (1-\beta_i) \left.\frac{du_{i+1}^{(2k)}}{dx}\right|_{x=x_i}, \quad i = 1, \ldots, p-1.$$

$$\begin{array}{l|l|l}
Lu_1^{(2k+1)} = f \text{ in } \Omega_1 & Lu_i^{(2k+1)} = f \text{ in } \Omega_i & Lu_p^{(2k+1)} = f \text{ in } \Omega_p \\
u_1^{(2k+1)}\Big|_{x=x_0} = 0 & \left.\frac{du_i^{(2k+1)}}{dx}\right|_{x=x_{i-1}} = g_{i-1}^{i-1}, \ i = 2, \ldots, p-1 & \left.\frac{du_p^{(2k+1)}}{dx}\right|_{x=x_{p-1}} = g_{p-1}^{p-1} \\
\left.\frac{du_1^{(2k+1)}}{dx}\right|_{x=x_1} = g_1^1 & \left.\frac{du_i^{(2k+1)}}{dx}\right|_{x=x_i} = g_i^i & u_p^{(2k+1)}\Big|_{x=x_p} = 0
\end{array}$$

$$h_i^i = \alpha_i \left. u_i^{(2k+1)}\right|_{x=x_i} + (1-\alpha_i) \left. u_{i+1}^{(2k+1)}\right|_{x=x_i}, \quad i = 1, \ldots, p-1.$$

$$\begin{array}{l|l|l}
Lu_1^{(2k+2)} = f \text{ in } \Omega_1 & Lu_i^{(2k+2)} = f \text{ in } \Omega_i & Lu_p^{(2k+2)} = f \text{ in } \Omega_p \\
u_1^{(2k+2)}\Big|_{x=x_0} = 0 & u_i^{(2k+2)}\Big|_{x=x_{i-1}} = h_{i-1}^{i-1}, \ i = 2, \ldots, p-1 & u_p^{(2k+2)}\Big|_{x=x_{p-1}} = h_{p-1}^{p-1} \\
u_1^{(2k+2)}\Big|_{x=x_1} = h_1^1 & u_i^{(2k+2)}\Big|_{x=x_i} = h_i^i & u_p^{(2k+2)}\Big|_{x=x_p} = 0
\end{array}$$

Algorithm 2: The Geometric (**GEO**) Contraction Based Method

$$g_i^i = \frac{u_i^{(k)} + u_{i+1}^{(k)}}{2} + \frac{w_i^i w_i^{i+1}}{w_i^i + w_i^{i+1}} \left(-\frac{du_i^{(k)}}{dx} + \frac{du_{i+1}^{(k)}}{dx} \right)\Bigg|_{x=x_i}, \quad i = 1, \ldots, p-1.$$

$$\begin{array}{l|l|l}
Lu_1^{(k+1)} = f \quad \text{in} \quad \Omega_1 & Lu_i^{(k+1)} = f \quad \text{in} \quad \Omega_i & Lu_p^{(k+1)} = f \quad \text{in} \quad \Omega_p \\
u_1^{(k+1)}\Big|_{x=x_0} = 0 & u_i^{(k+1)}\Big|_{x=x_{i-1}} = g_{i-1}^{i-1} \quad , \quad i = 2, \ldots, p-1 & u_p^{(k+1)}\Big|_{x=x_{p-1}} = g_{p-1}^{p-1} \\
u_1^{(k+1)}\Big|_{x=x_1} = g_1^1 & u_i^{(k+1)}\Big|_{x=x_i} = g_i^i & u_p^{(k+1)}\Big|_{x=x_p} = 0
\end{array}$$

Algorithm 3: The Newton's (**NEW**) Method

$$\left.\begin{aligned}\alpha_i^i &= \frac{\left.\frac{du_i^{(k)}}{dx}\right|_{x=x_i} - \left.\frac{du_i^{(k-1)}}{dx}\right|_{x=x_i}}{\left.u_i^{(k)}\right|_{x=x_i} - \left.u_i^{(k-1)}\right|_{x=x_i}} \\ \alpha_i^{i+1} &= \frac{\left.\frac{du_{i+1}^{(k)}}{dx}\right|_{x=x_i} - \left.\frac{du_{i+1}^{(k-1)}}{dx}\right|_{x=x_i}}{\left.u_{i+1}^{(k)}\right|_{x=x_i} - \left.u_{i+1}^{(k-1)}\right|_{x=x_i}}\end{aligned}\right\} i = 1, \ldots, p-1$$

$$\left.\begin{aligned}h_i^i &= \left.u_i^{(k)}\right|_{x=x_i} + \frac{\left.\alpha_i^{i+1}u_{i+1}^{(k)}\right|_{x=x_i} - \left.\alpha_i^{i+1}u_i^{(k)}\right|_{x=x_i} - \left.\frac{du_{i+1}^{(k)}}{dx}\right|_{x=x_i} + \left.\frac{du_i^{(k)}}{dx}\right|_{x=x_i}}{\alpha_i^{i+1} - \alpha_i^i} \\ h_i^{i+1} &= \left.u_{i+1}^{(k)}\right|_{x=x_i} + \frac{\left.\alpha_i^{i}u_{i+1}^{(k)}\right|_{x=x_i} - \left.\alpha_i^{i}u_i^{(k)}\right|_{x=x_i} - \left.\frac{du_{i+1}^{(k)}}{dx}\right|_{x=x_i} + \left.\frac{du_i^{(k)}}{dx}\right|_{x=x_i}}{\alpha_i^{i+1} - \alpha_i^i}\end{aligned}\right\} i = 1, \ldots, p-1.$$

$$\left.\begin{aligned}&Lu_1^{(k+1)} = f \text{ in } \Omega_1 \\ &\left.u_1^{(k+1)}\right|_{x=x_0} = 0 \\ &\left.u_1^{(k+1)}\right|_{x=x_1} = h_1^1\end{aligned}\right| \left.\begin{aligned}&Lu_i^{(k+1)} = f \text{ in } \Omega_i \\ &\left.u_i^{(k+1)}\right|_{x=x_{i-1}} = h_{i-1}^i \quad , \quad i = 2, \ldots, p-1 \\ &\left.u_i^{(k+1)}\right|_{x=x_i} = h_i^i\end{aligned}\right| \left.\begin{aligned}&Lu_p^{(k+1)} = f \text{ in } \Omega_p \\ &\left.u_p^{(k+1)}\right|_{x=x_{p-1}} = h_{p-1}^p \\ &\left.u_p^{(k+1)}\right|_{x=x_p} = 0\end{aligned}\right.$$

Algorithm 4: The Robin Relaxation (**ROB**) Method

$$\left.\begin{aligned}g_i^i &= \left.\frac{du_{i+1}^{(k)}}{dx}\right|_{x=x_i} + \lambda_i \left.u_{i+1}^{(k)}\right|_{x=x_i} \\ g_i^{i+1} &= -\left.\frac{du_i^{(k)}}{dx}\right|_{x=x_i} + \lambda_i \left.u_i^{(k)}\right|_{x=x_i}\end{aligned}\right\} i = 1, \ldots, p-1.$$

$$\left.\begin{aligned}&Lu_1^{(k+1)} = f \quad \text{in} \quad \Omega_1 \\ &\left.u_1^{(k+1)}\right|_{x=x_0} = 0 \\ &\left.\frac{du_1^{(k+1)}}{dx}\right|_{x=x_1} + \lambda_1 \left.u_1^{(k+1)}\right|_{x=x_1} = g_1^1\end{aligned}\right| \left.\begin{aligned}&Lu_p^{(k+1)} = f \quad \text{in} \quad \Omega_p \\ &-\left.\frac{du_p^{(k+1)}}{dx}\right|_{x=x_{p-1}} + \lambda_{p-1} \left.u_p^{(k+1)}\right|_{x=x_{p-1}} = g_{p-1}^p \\ &\left.u_p^{(k+1)}\right|_{x=x_p} = 0\end{aligned}\right.$$

$$\left.\begin{aligned}&Lu_i^{(k+1)} = f \quad \text{in} \quad \Omega_i \\ &-\left.\frac{du_i^{(k+1)}}{dx}\right|_{x=x_{i-1}} + \lambda_{i-1} \left.u_i^{(k+1)}\right|_{x=x_{i-1}} = g_{i-1}^i \\ &\left.\frac{du_i^{(k+1)}}{dx}\right|_{x=x_i} + \lambda_i \left.u_i^{(k+1)}\right|_{x=x_i} = g_i^i\end{aligned}\right\} i = 2, \ldots, p-1.$$

Algorithm 5: The Schur complement (**SCO**) Method

For a $c \in \{1, \ldots, p\}$

$$\left.\begin{array}{l} h_{i-1}^{i} = \left\{ \begin{array}{ll} 0 & , \ i = 1 \\ \theta_{i-1} \ u_{i-1}^{(k+1)}\Big|_{x=x_{i-1}} + (1-\theta_{i-1}) \ u_{i}^{(k)}\Big|_{x=x_{i-1}} & , \ i = 2, \ldots, c-1 \end{array} \right. \\ g_i^i = \left. \frac{du_{i+1}^{(k)}}{dx} \right|_{x=x_i} \\ Lu_i^{(k+1)} = f \quad in \quad \Omega_i \quad , \quad u_i^{(k+1)}\Big|_{x=x_{i-1}} = h_{i-1}^{i} \quad , \quad \left. \frac{du_i^{(k+1)}}{dx} \right|_{x=x_i} = g_i^i \end{array} \right\} i = 1, \ldots, c-1$$

$$h_{c-1}^{c} = \theta_{c-1} \ u_{c-1}^{(k+1)}\Big|_{x=x_{c-1}} + (1-\theta_{c-1}) \ u_{c}^{(k)}\Big|_{x=x_{c-1}}$$

$$h_{c}^{c} = \theta_{c} \ u_{c+1}^{(k)}\Big|_{x=x_{c}} + (1-\theta_{c}) \ u_{c}^{(k)}\Big|_{x=x_{c}}$$

$$Lu_c^{(k+1)} = f \quad in \quad \Omega_c \quad , \quad u_c^{(k+1)}\Big|_{x=x_{c-1}} = h_{c-1}^{i} \quad , \quad u_c^{(k+1)}\Big|_{x=x_c} = h_c^c$$

$$\left.\begin{array}{l} h_i^i = \left\{ \begin{array}{ll} \theta_i \ u_{i+1}^{(k)}\Big|_{x=x_i} + (1-\theta_i) \ u_i^{(k)}\Big|_{x=x_i} & , \ i = c+1, \ldots, p-1 \\ 0 & , \ i = p \end{array} \right. \\ g_{i-1}^{i} = \left. \frac{du_{i-1}^{(k+1)}}{dx} \right|_{x=x_{i-1}} \\ Lu_i^{(k+1)} = f \quad in \quad \Omega_i \quad , \quad \left. \frac{du_i^{(k+1)}}{dx} \right|_{x=x_{i-1}} = g_{i-1}^{i} \quad , \quad u_i^{(k+1)}\Big|_{x=x_i} = h_i^i \end{array} \right\} i = c+1, \ldots, p$$

Algorithm 6: The Shooting (**SHO**) Method

$$\left.\begin{aligned} D_i^{(k-1)} &= \left.\frac{du_i^{(k-2)}}{dx} - \frac{du_{i+1}^{(k-2)}}{dx}\right|_{x=x_i} \\ D_i^{(k)} &= \left.\frac{du_i^{(k-1)}}{dx} - \frac{du_{i+1}^{(k-1)}}{dx}\right|_{x=x_i} \\ \alpha_i^{(k)} &= \frac{\alpha_i^{(k-1)}|D_i^{(k)}|}{|D_i^{(k-1)} - D_i^{(k)}|} \\ g_i^i &= \left. u_i^{(k)}\right|_{x=x_i} - \alpha_i^{(k)} D_i^{(k)} \end{aligned}\right\} i = 1, \ldots, p-1.$$

$$\left|\begin{aligned} Lu_1^{(k+1)} &= f \quad \text{in} \quad \Omega_1 \\ \left. u_1^{(k+1)}\right|_{x=x_0} &= 0 \\ \left. u_1^{(k+1)}\right|_{x=x_1} &= g_1^1 \end{aligned}\right| \left|\begin{aligned} Lu_i^{(k+1)} &= f \quad \text{in} \quad \Omega_i \\ \left. u_i^{(k+1)}\right|_{x=x_{i-1}} &= g_{i-1}^{i-1} \quad , \quad i = 2, \ldots, p-1 \\ \left. u_i^{(k+1)}\right|_{x=x_i} &= g_i^i \end{aligned}\right| \left|\begin{aligned} Lu_p^{(k+1)} &= f \quad \text{in} \quad \Omega_p \\ \left. u_p^{(k+1)}\right|_{x=x_{p-1}} &= g_{p-1}^{p-1} \\ \left. u_p^{(k+1)}\right|_{x=x_p} &= 0 \end{aligned}\right|$$

Algorithm 7: The Steklov–Poincaré operator (**SPO**) Method

$$\left|\begin{aligned} Lu_1^{(k+1)} &= f \text{ in } \Omega_1 \\ \left. u_1^{(k+1)}\right|_{x=x_0} &= 0 \\ \left. u_1^{(k+1)}\right|_{x=x_1} &= h_1^1 \end{aligned}\right| \left|\begin{aligned} Lu_i^{(k+1)} &= f \text{ in } \Omega_i \\ \left. u_i^{(k+1)}\right|_{x=x_{i-1}} &= h_{i-1}^i \quad , \quad i = 2, \ldots, p-1 \\ \left. u_i^{(k+1)}\right|_{x=x_i} &= h_i^i \end{aligned}\right| \left|\begin{aligned} Lu_p^{(k+1)} &= f \text{ in } \Omega_p \\ \left. u_p^{(k+1)}\right|_{x=x_{p-1}} &= h_{p-1}^p \\ \left. u_p^{(k+1)}\right|_{x=x_p} &= 0 \end{aligned}\right|$$

$$g_i^{i+1} = g_i^i = \frac{1}{2}\left(\left.\frac{du_i^{(k+1)}}{dx}\right|_{x=x_i} - \left.\frac{du_{i+1}^{(k+1)}}{dx}\right|_{x=x_i} \right) \quad , \quad i = 1, \ldots, p-1.$$

$$\left|\begin{aligned} L\phi_1^{(k+1)} &= 0 \text{ in } \Omega_1 \\ \left. \phi_1^{(k+1)}\right|_{x=x_0} &= 0 \\ \left. \frac{d\phi_1^{(k+1)}}{dx}\right|_{x=x_1} &= g_1^1 \end{aligned}\right| \left|\begin{aligned} L\phi_i^{(k+1)} &= 0 \text{ in } \Omega_i \\ \left. \frac{d\phi_i^{(k+1)}}{dx}\right|_{x=x_{i-1}} &= g_{i-1}^i \quad , \quad i = 2, \ldots, p-1 \\ \left. \frac{d\phi_i^{(k+1)}}{dx}\right|_{x=x_i} &= g_i^i \end{aligned}\right| \left|\begin{aligned} L\phi_p^{(k+1)} &= 0 \text{ in } \Omega_p \\ \left. \frac{d\phi_p^{(k+1)}}{dx}\right|_{x=x_{p-1}} &= g_{p-1}^p \\ \left. \phi_p^{(k+1)}\right|_{x=x_p} &= 0 \end{aligned}\right|$$

$$h_i^{i+1} = h_i^i = u_i^i - \frac{\rho_i}{2}\left(\left.\phi_i^{(k+1)}\right|_{x=x_i} + \left.\phi_{i+1}^{(k+1)}\right|_{x=x_i} \right) \quad , \quad i = 1, \ldots, p-1.$$

Appendix B

Implementation of a Collaborative PDE Problem Solving Environment

B.1 SciAgents GUI(SAtool)

The implementation in the computer of the mathematical methods for solving PDEs is a task that requires a lot of time and effort even simple model problems. Specially, when someone has to write/debug code that handles the descritization of the PDE domain and operator. This task becomes more difficult, almost impossible, when multi-component problems are to be considered. Thus existing software(e.g., libraries, executables etc.) should be used to reduce the complexity of the implementation of such *large programs*. Usually these *programs* require a lot of data in such details that their input files are long and most of the times unreadable even by their frequent users. Graphical User Interfaces are applied to facilitate the use of these programs. Their main task is to provide the user easier ways to set the input data correctly, execute the right program calling the proper functions and finally to interpret the output data.

SciAgents is a program that solves two-dimensional multi-PDE problems. Because of the complexity of composite PDE problems, it takes almost the same time to (1) define the input data of the problem, (2) specify the computing power that will be used to solve the problem in parallel and (3) to actually compute the solution. It takes a lot of time and effort to operate **SciAgents** *manually*. **SciAgents** is based on **PELLPACK**, uses three interface relaxation methods to handle the interfaces, and is operationable on a local network of workstations.

Some of the questions that come up when dealing with such problems are: *How is the convergence to the solution of the PDE affected by the particular relaxation method? How to select optimum/good parameters of each particular method? How to determine initial guesses on the interfaces?* etc. These are very important issues and, since there are no theoretical results, we need to experiment a lot to empirically understand them first. In addition to this, we can use **SciAgents** to verify the results that come from a theoretical analysis of a relaxation method for model problems.

From the above, it is clear that the necessity to built a GUI for **SciAgents**, was of great importance. Let us name this GUI **SAtool**. The **SAtool** design philosophy is similar to **Pelltool**(the GUI of PELLPACK), and in many cases it uses some of the individual tools either as they appear in **Pelltool** or extended/modified. **SAtool** is presented in the form of the a brief user's guide and its implementation is not presented here.

Introductory Window

The introductory window (Figure B.1) pops up when the user types **SAtool**. This command panel allows the user to start defining a new multi-PDE problem, load an existing one, or `Quit`. The `About SA` button provides information for the SciAgents, while the `Help` provides

Figure B.1: The initial window that appears typing **SAtool**

information about **SAtool** using hypertext files. Clicking on the `Load` or `New File` buttons starts the SAsession (Figure B.2).

B.2 SciAgents Session

The SAsession window (Figure B.2) organizes the effort that one needs to go through in order to specify the multi-PDE problem for the SA infrastructure. The `Quit`, `Save`, `Save as`, `Print` are essentially the same as in **Pelltool** (with only few extensions added) and are already fully functional (except from `Print`). The `Reset` button clears up the interconnectivity of the SciAgents network of machines. Also starts up the httpd daemon and the SciAgents-server of default or user defined machines. The `Help` button provide information about the **The SAsession window** and the functionalities of each button. The `Clear` button clears the editor, on the right part of the window and the whole definition of an existing problem.

The session editor shown on the right is an emacs-type editor and can be used to manually generate the SciAgents input file that is currently needed. Using this file requires the user to declare the number of solvers, i.e. subdomains of the general domain of the PDE and the number of the mediators which is the number of the interfaces. Then for each of the mediators one provides the following information: its *id number*, the *number of solvers* that it has to collaborate and exchange data the *id number* of that interface (related with the specific mediator) for each subdomain, and the particular *relaxation scheme* each mediator uses. The tolerance needed for the convergence criteria is given next, and then the coordinates of the starting and the ending point of the interface are given with the value of the solutions on those taken from either boundary conditions (for interface points on the original boundary) or initial guess otherwise. Then machine names are declared. First of all is the machine for displaying the various outputs, then the one for the global control. Next are the machines where solvers will run (each one might run on a different machine), and, finally, are the machines where the mediators will run. A first version of the input file is generated by the **Boundary, Interface and BC Editor** (see below), and contains

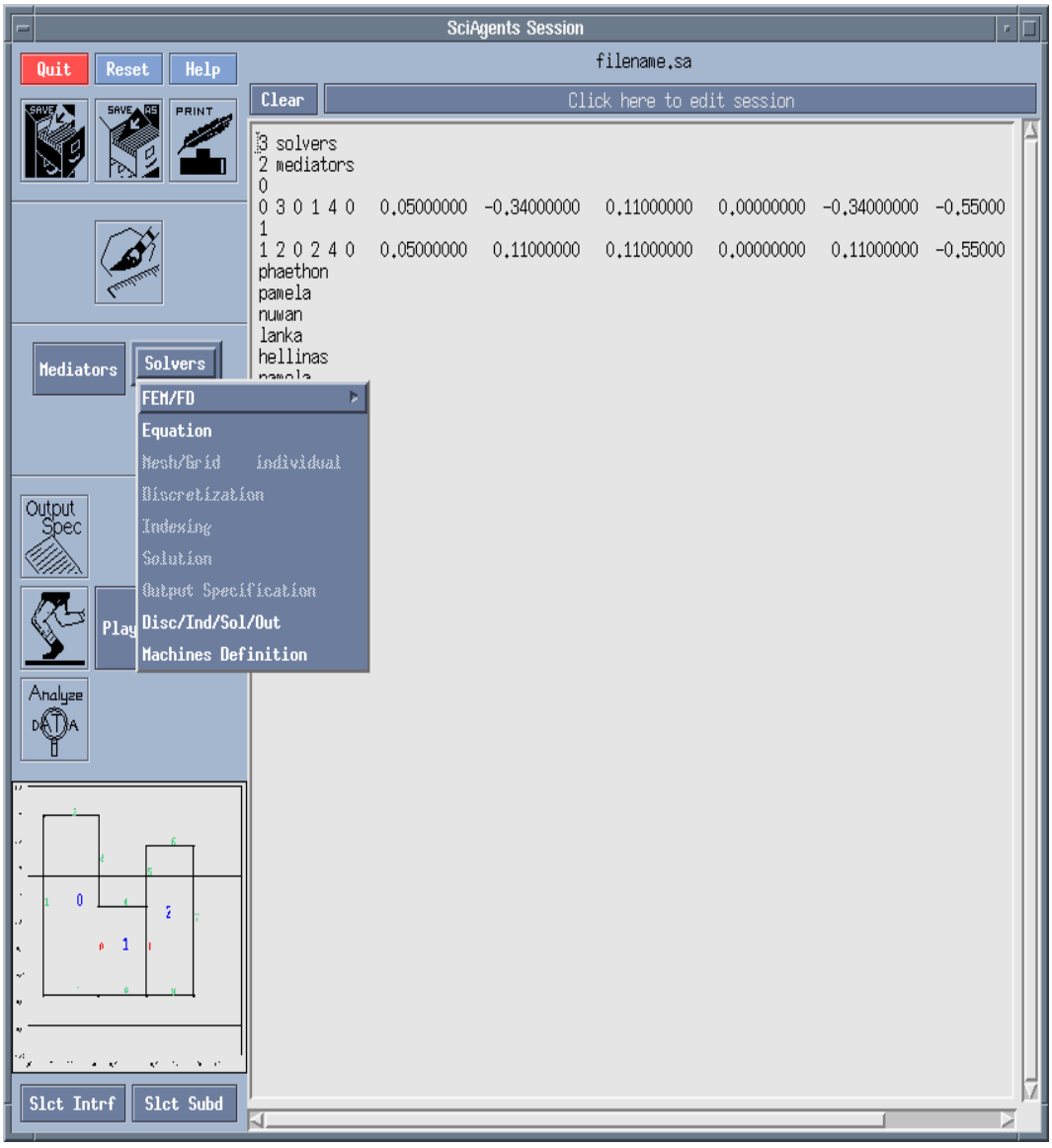


Figure B.2: SAsession window

information on how interfaces and solvers are matched together. The rest information can be added either manually by the user or using the corresponding buttons.

Clicking the geometry button, (Figure B.3), raises **Boundary, Interface and BC Editor**. This tool (described below) assists the user in drawing the domain of the composite problem.

The canvas on the lower left is used to display a sketch of the composite articraft as shown in Figure B.5. This might be used for a successive view of the global domain which makes the selection of one or a set of subdomains or interfaces much easier. The selection mechanism is provided by the buttons `Slct Subd` and `Slct Intrf` and applies to any of the `Mediator`, `Solvers`, `Output`, `PlayBack` and Analyze data buttons whose actions are described below. Right now clicking on the canvas one can view an enlarged image of the global domain.

B.3 The Boundary and Interface Specification Editor.

This editor (Figure B.4), is an interactive, graphical editor and it is raised by clicking on the icon shown in Figure B.3, from the **SciAgents Session** window. It is used to define the

Figure B.3: Icon that invokes the Boundary, Interface and BC Editor

outer boundary of the domain of the PDE problem, the interfaces (which may come from the physics of the problem or not) and to specify the conditions that apply to the outer boundary and to the interfaces. In addition it is used to define the subdomains that the PDE problem decomposition. It also produces textual descriptions, one for of each subdomain, and stores them in separate files, that can be used later when **Pelltool** is used to solve each subproblem. Finally, the editor creates an image (Figure B.5) of the decomposed and saved domain and passes it to the **SciAgents Session** where it is displayed in the canvas area at the lowest left.

The editor itself consists of a command panel and a drawing window below it. If the **SciAgents Session** editor contains the specification of a previously defined problem (which the whole domain and the subdomains already defined) then the drawing window is used to display and modify the domain of a previously defined problem. (This facility is under construction). If no domain is defined, then the drawing window contains only an empty

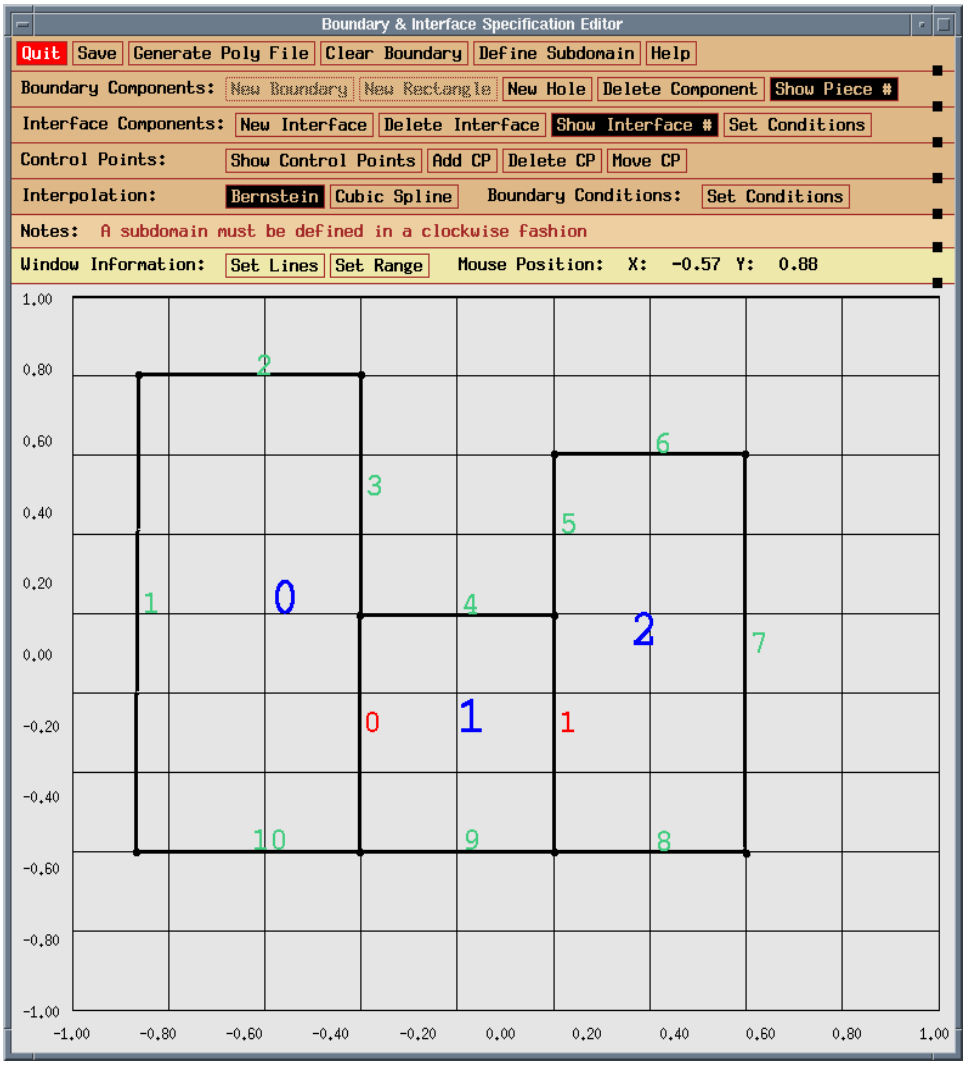


Figure B.4: Domain, Interface and BC Editor

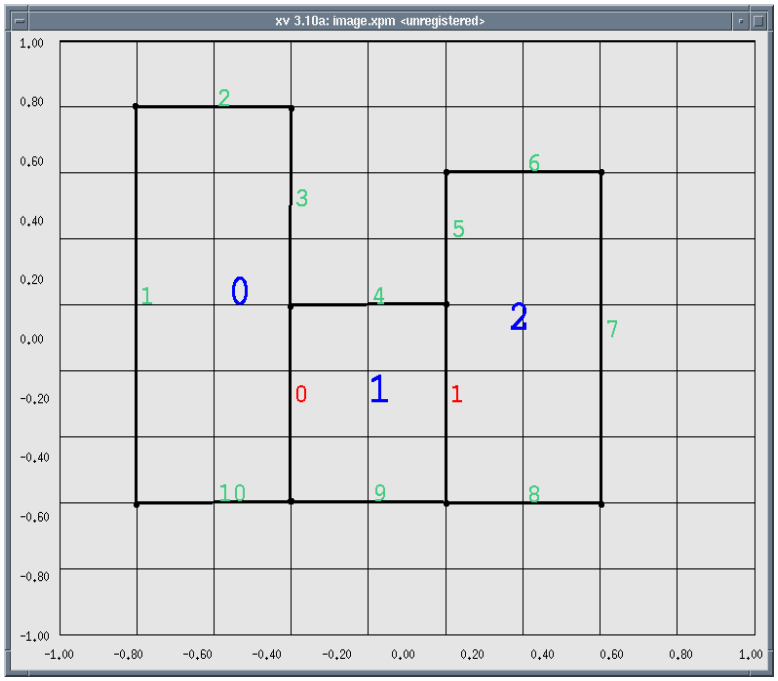


Figure B.5: Drawing Area image that is saved in the canvas of SciAgents Session window.

grid. In the command panel there are buttons that allow the user to define and manipulate the domain and the subdomains of the PDE problem and to set conditions on each piece of both the boundary and the interfaces. The buttons on the command panel can be activated by clicking on them with the left mouse button, while the middle button has no affect and the right one provides help messages for each operation. In the drawing window there are two working modes. Some of the operations, operate in the edit mode and the rest operate in the command mode. Operations like `Add CP` operates on the control points that can be selected by clicking on them with the left mouse button. A boundary component is selected by clicking on one of its pieces or on one of its control points with any of the mouse buttons. If an operation switches the mode into the edit mode, clicking on the right mouse button switches to the command mode.

Descriptions for some of the buttons:

Quit: Quits the editor. If the domain created is not saved, it asks the user to save it or not.

Save: Saves the textual representation of each subdomain displayed in the editor. Each subdomain is stored in a file named 'subdomainN.e', where N is the order number of the

subdomain. If the subdomains were already defined and/or stored in files, they are overwritten. It also writes the contents of the drawing area to a file, named input.sa (the input to **SA** file).

Clear Boundary: Clears the drawing window, by deleting both the outer boundary and the interfaces if there are any.

Define Subdomain: Defines the subdomains of the problem. It uses the already drawn pieces of the outer boundary and/or the interface pieces. In the simplest case where there is only one subdomain (i.e., the PDE domain) this is defined automatically. A subdomain is defined by selecting sequentially boundary and/or interface pieces in a *clockwise* order. In the edit mode of `Define Subdomain` button the left mouse button determines the first piece of the subdomain, the right mouse button determines the last piece and the middle mouse button is used to specify the rest pieces of the subdomain. When the last piece is selected, the definition of the subdomain is completed and the current mode is switched into command mode. This button is used each time a new subdomain is defined.

Help: Provides information about the buttons of the editor and their functionality using hypertext files.

New Boundary: Defines the outer boundary component. The control points must be defined sequentially and in clockwise order. In the edit mode of 'New Boundary' the left mouse button must be used to define the first control point of the boundary component. In addition it defines control points of the current boundary piece. The middle mouse button complete the current boundary piece by defining its endpoint. The shape of the new piece is completely defined by its control points and the selected interpolation scheme. The right mouse button completes the boundary component and exits the edit mode. It completes the last piece of the boundary by closing it but does not define a control point. After that a default boundary condition $U = true(x,y)$ is assigned to every edge of the boundary.

New Rectangle: Provides an easy way to define a rectangular outer domain, whose range is set using the `Set Range`.

Delete Component: Deletes the selected boundary component. Now it deletes the outer boundary and the interfaces if there are any.

Show Piece #: Displays the piece numbers of the outer boundary.

New Interface: Defines a new interface piece. It must begin and end on beginning/ending points of existing pieces (either boundary pieces or previously defined interfaces pieces). An interface component may consist of many pieces. The edit mode of `New Interface` is very similar to the edit mode of `New Boundary`. The left mouse button must be used to specify among all the existing control points, the beginning point of the current interface. Also it can be used to define control points of the current piece of the interface. The middle mouse

button completes the current interface piece by defining its endpoint (i.e., define one more control point in the interior of the outer domain). The shape of the new piece is completely defined by its control points and the selected interpolation scheme. The right mouse button completes the interface component and exits the edit mode. It completes the last piece of the current interface by selecting an existing control point. Note that the first and the last points of the interface component do not define more control points since they are already defined. After completing the current component, a default interface condition U = *rinterface(x,y)* is assigned to every edge of the interface. This operation must be selected each time a new interface is defined.

Delete Interface: Deletes the selected interface component. All the interface components that are built based on points of the selected interface, are also deleted.

Show Interface #: Displays the piece numbers of the interface components. The number is globally determined over all interfaces.

Set Conditions: Specifies the interface conditions (that come from the physics of the PDE problem) associated with each piece of the selected interface component. To specify that a boundary piece is an interface component use the `Set Conditions` button 2 rows below this one.

Show Control Point: Displays the control points of the boundary/interface pieces in the drawing window. This is helpful when adding, deleting, moving control points.

Add CP: Adds control point to the drawn pieces. To add a new control point after an existing control point, select the existing one (i.e., click on it with the left/middle mouse button) and without releasing the button, drag the cursor to the new point's location. Release the mouse button. The affected pieces are redrawn. If the selected point is a beginning/ending point and it is selected by the left mouse button, then the new point is added to the left boundary/interface piece of the selected point and is the new beginning/ending point. If the selection is made by the middle mouse button, then the new point is added to the right piece of the selected point and the beginning/ending point remains at the same location. Many control points can be added to various pieces before exiting the edit mode. To complete the operation and switch to command mode, click on the right mouse button. Currently the selected point can not be a beginning/ending point since interface pieces might need to be moved.

Delete CP: Deletes control points of the drawn pieces. To delete a control point click on it with the left/middle mouse button. If the selected control point is a beginning/ending point and if the point selection is done with the left mouse button, then the the new begin/end point is the previous (but not begin point) of the left piece. If the selection is done with the middle mouse button then the new begin/end point is the next (but not endpoint) of

the right piece. To complete the operation and switch to command mode, click on the right mouse button. Currently this operation should not be used to control points *near* begin/end points.

Move CP: Moves control points of drawn pieces. To move a control point select it (using left/middle mouse button) and, without releasing the button, drag the cursor to the point's new location. To complete the operation and switch to command mode, click on the right mouse button. Currently the selected point can not be a beginning/ending point since interface pieces might need to be moved.

Bernstein: Uses Bernstein polynomial interpolation of the control points to define curved pieces.

Cubic Spline: Uses the cubic Spline interpolation of the control points to define curved pieces.

Set Conditions: Specifies the boundary and interface conditions associated with each piece of the selected boundary/interface component. When this operation is selected, the Boundary/Interface Conditions Specification Editor displays the current number and condition of each piece of the selected component. The pieces are numbered in the order they are created. To see this numbering select the `Show Piece/Interface #` before selecting `Set Conditions`. To modify the condition for one of the pieces, erase the current one and enter the new expression in its place. To modify all the conditions, select clear to delete the current conditions and enter the new. To save the new conditions, select `Continue` and select `Cancel` to quit the Boundary/Interface Conditions Specification Editor without saving the new conditions.

Set Lines: Sets the number of the grid lines in the X and Y directions. The lines of the grid are simply a guide for placing control points at correct locations and not part of the discretization of the domain. To change the number of the lines, place the cursor in the rectangle that contains the current numbers, erase them and enter the number of lines. Select `Continue` to save and quit, click on `Cancel` to quit without saving.

Set Range: Sets the range of the drawing area for the X and Y directions. The four numbers, separated by spaces, should be erased before the new values are entered. The first and the second numbers specify the minimum and maximum in X direction and the third and the forth numbers specify the minimum and maximum in the Y direction. Select `Continue` to save and quit and click on `Cancel` to quit without saving.

Mouse Position: Displays the coordinates of the mouse cursor. Functional only when the cursor is in the drawing area.

To define a domain and decompose it to several subdomains, follow the following procedure:

- *Clear* the drawing window, if an old domain definition exists.
- Specify the range of the drawing area, using `Set Range`.
- Specify the number of grid lines in each direction, using `Set Lines`.
- Select `Show Control Points` operation, if any of the outer boundary or the interfaces are curved pieces.
- Define the outer boundary of the PDE problem using the `New Boundary` or the `New Rectangular` operations.
- Define the interfaces using `New Interface`.
- Define the conditions for each component using the `Set Condition` operation.
- Use `Define Subdomain` to specify the subdomains of the decomposed PDE problem.
- Save the subdomains in their files and post an image of the drawing area in the canvas on the lower left part of SciAgents Session tool.

We might want to investigate the possibility to provide a second way to create the main domain of the problem. One could create the subdomains separately and then move them, by clicking with the left button of the mouse inside them, to the desired position. They would have to match and create the main domain.

B.4 The Mediator Editor.

This editor (Figure B.6) allows the user to specify (1) the machine that will handle the selected (by `Slct Intrf` button) mediators, (2) the relaxation method that will be used, and (3) the value of the parameter(s) of the selected method. The user can edit the inputs of all dialog boxes and can either select one of the existing relaxation schemes or select `CUSTOM` to pop-up an editor for specifying a new scheme. Two relaxations methods are already ready to be specified through the editor. The `AVE` and the three variations of `Default` defined by T. Drashansky in his Ph.D. The `Help` button again provide information on the utilities of this Editor, using hypertext files.

B.5 The Solver Editor.

For each subdomain we need to specify the PDE equation, the type of discretization (grid/mesh), the PDE discretization scheme and the method to solve the linear algebraic

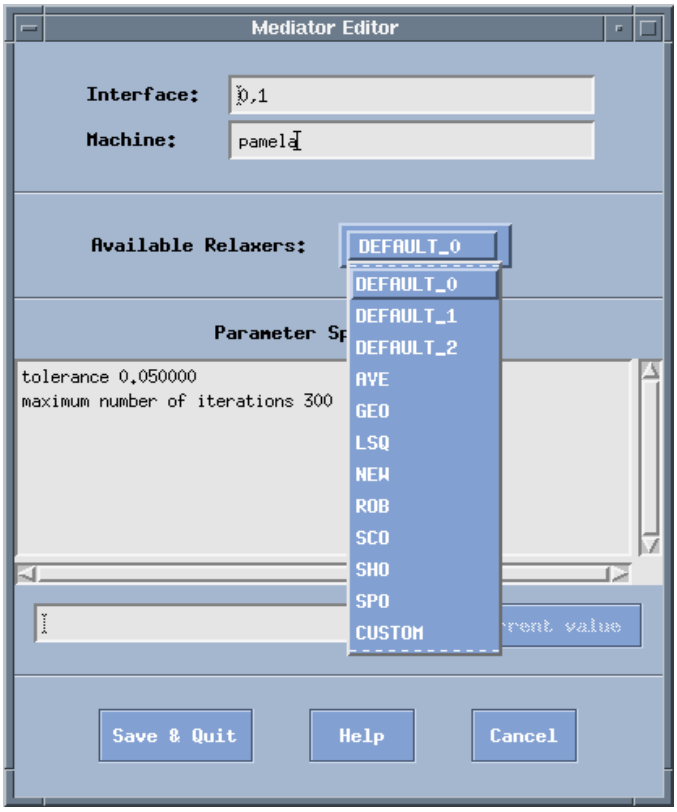


Figure B.6: Mediator Editor

system. This specification is made by clicking on the `Solvers` button in the SAsession window, after the *id number* of the solver(s) has been selected through the `Slct Subd` button. Clicking on the `Solvers` raises a small pull down menu and the user selects how to proceed. First of all has to select the type of discretization, either *FEM* or *FDM*, and then click on `Equation` to define the PDE equation. This will be done through the SciAgent enhanced version of **Pelltool** that will be started for the selected solvers. Having defined the equation and the boundary for a subdomain, the corresponding .e file will contain default specifications for discretization, indexing, solution and output segments. If the user wish to modify one or more of these specifications, she/he has to select each subdomain separately (using the `Slct Subd` button) and the click on the `Disc/Index/Sol/Out` button on the `Solvers` pull down menu. The **Pelltool** session will come up with the selected subdomain's .e file so one can proceed with the required modifications. To save the modified .e file the user has to click onth `Save` button of the Pelltool's Session. To define the mesh discretization scheme, the user has to work each .e file individually since the mesh/grid depends strongly

Figure B.7: Machines Specification Editor

on the geometry of each subdomain. Therefore first click on `Slct Subd` button to select a subdomain and then click on `Disc/Index/Sol/Out` button. The **Pelltool** will come up with the corresponding .e file and now the user can go on as usual. At this state some output files must be defined for the boundary points and the values of the function on these points, and this can be done automatically be clicking on the `Agent` button on the command panel of the Pelltool's Session. The machines can be specified when on clicks on the `Machines Definition` button. An editor comes up like in Figure B.7 and the user has to fill in the boxes with the machine names chosen from the Local Access Network.

B.6 The Run Tool.

The global execution of **SciAgents** starts when all definitions and specifications are completed, and the user clicks on the `Run` icon. This raises a window similar to Pellpack's ExecuteTool. When the actual execution starts an Execution Trace window pops-up which selectively displays the execution procedure graphically (Figure B.8). Appropriate colors are used to indicate what the agents are actually doing (computing or waiting for data). This window can also provide the values of the convergence criteria and iteration numbers, list the messages being sent and allow the user to pause or stop the execution and modify certain parameters.

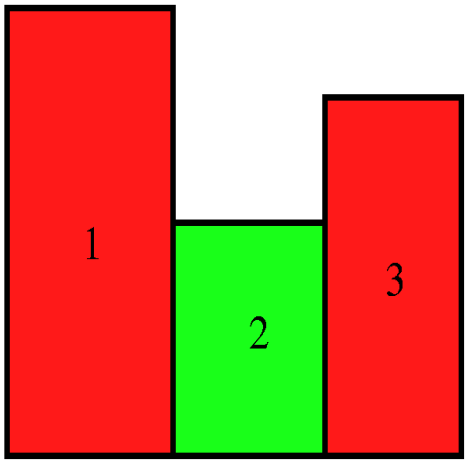


Figure B.8: Supervising Execution Graph

B.7 The PlayBK Tool.

The **PlayBk** button raises the Execution and Communication Info window. It contains a graph (similar to the one described in the Run tool above) of the main domain with its subdomains and two time diagrams, one for the mediators and the other for the solvers. The graph, illustrated in Figure B.8, shows if a solver is working or waiting to receive new data for its interfaces. Colors are used to provide this information to the user, for example, a subdomain is colored green when its solver is working and red when its solver is waiting. Other time diagrams like the ones in Figure B.9 might be added. There is a great possibility to use **ParaGraph** (a graphical display system for visualizing the behavior and the performance of parallel programs on message-passing multicomputer architectures) to process and visualize the execution trace information.

B.8 The Output Display.

The `Output Spec` and the `Analyze DATA` buttons raise Pellpack's Output Specification Editor and Analyze data editors with which one can specify views the various data after the computation is complete. Right now the user can plot an image of the function (or the first derivatives) on the global domain, running the script (written in Perl) `preplot` (the whole path for that script is `/p/pses/sciagents/src/front_end/scri- pts/preplot`) to process and manipulate the `.out` and `.mesh` files of each solver and the files that contains the boundary points on the interfaces. This script also copies from the script directory 3

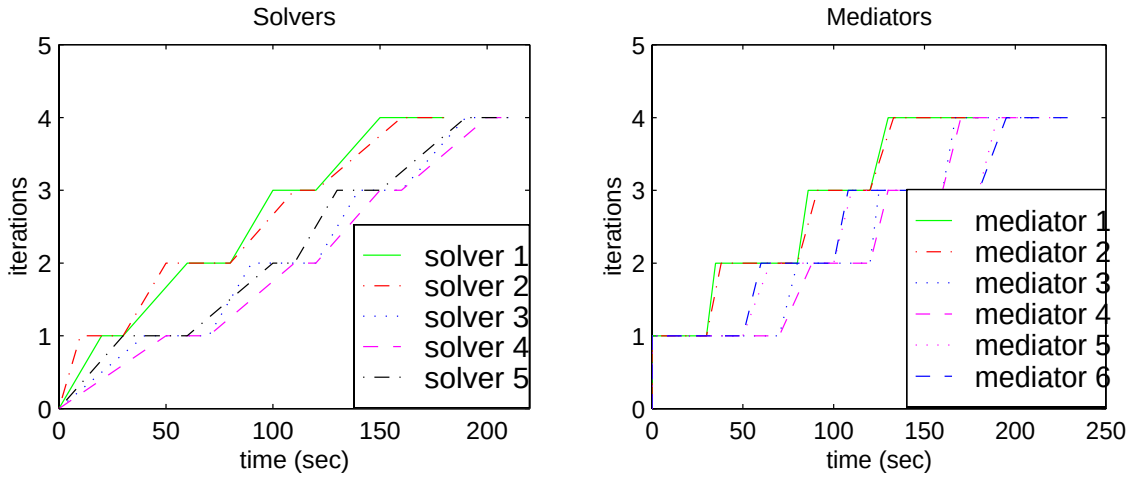


Figure B.9: Execution Trace Diagrams

MATLAB files (`plotu.m plotux.m plotuy.m`) that can be used to make the plot of either the function or its derivatives.

B.9 The Selection Buttons.

The two buttons `Slct Subd` and `Slct Intrf` at the bottom of the **SAsession Window** have similar functionality. If one clicks on `Slct Subd` a dialog box comes up, like in Figure B.10, and the user needs to define the number of the subdomains that will apply other functionalities of the GUI. For the case of subdomains, these functionalities can be the *Solver Editor*, the *Output Specification Editor*, the *Analyze Data Editor* and the *PlayBack tool* The numbers of the selected interfaces is specified in the dialog box raised by clicking on the `Slct Intrf` button, where the user then can apply the *Mediator Editor* and the *PlayBack tool*.

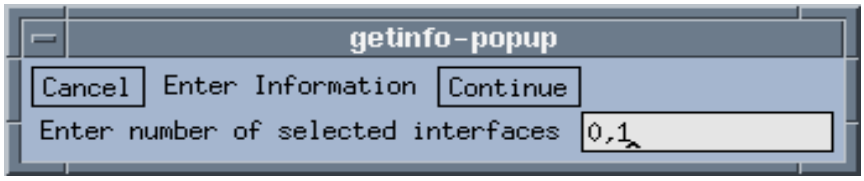


Figure B.10: Select Interface/Subdomain Dialog Box

Appendix C

Analysis for the AVE method

C.1 Minimize the max-norm of M^D

The max-norm of M^D is given by

$$||M^D||_\infty = \max_{1\le i\le p-1} f_i,$$

where

$$f_1(\alpha_1) = \left|\frac{\alpha_1 m_1}{n_1\gamma_1} - \frac{(1-\alpha_1)n_2}{m_2\gamma_2}\right| + \frac{2(1-\alpha_1)}{m_2\gamma_2},$$

$$f_i(\alpha_i) = \frac{2\alpha_i}{m_i\gamma_i} + \left|\frac{\alpha_i n_i}{m_i\gamma_i} - \frac{(1-\alpha_i)n_{i+1}}{m_{i+1}\gamma_{i+1}}\right| + \frac{2(1-\alpha_i)}{m_{i+1}\gamma_{i+1}}, \quad i = 2,\ldots,p-2, \tag{C.1.1}$$

$$f_{p-1}(\alpha_{p-1}) = \frac{2\alpha_{p-1}}{m_{p-1}\gamma_{p-1}} + \left|\frac{\alpha_{p-1}n_{p-1}}{m_{p-1}\gamma_{p-1}} - \frac{(1-\alpha_{p-1})m_p}{n_p\gamma_p}\right|.$$

Considering the fact that $\gamma_i = \gamma, i = 1,\ldots,p$, as in Theorem 3.3.4, the formulas (C.1.1) can be simplified as

$$f_1(\alpha_1) = \left|\frac{\alpha_1 m_1}{n_1\gamma} - \frac{(1-\alpha_1)n_2}{m_2\gamma}\right| + \frac{2(1-\alpha_1)}{m_2\gamma},$$

$$f_i(\alpha_i) = \frac{2\alpha_i}{m_i\gamma} + \left|\frac{\alpha_i n_i}{m_i\gamma} - \frac{(1-\alpha_i)n_{i+1}}{m_{i+1}\gamma}\right| + \frac{2(1-\alpha_i)}{m_{i+1}\gamma}, \quad i = 2,\ldots,p-2, \tag{C.1.2}$$

$$f_{p-1}(\alpha_{p-1}) = \frac{2\alpha_{p-1}}{m_{p-1}\gamma} + \left|\frac{\alpha_{p-1}n_{p-1}}{m_{p-1}\gamma} - \frac{(1-\alpha_{p-1})m_p}{n_p\gamma}\right|.$$

We present the analysis for the general case (i.e., for $f_i, i = 2,\ldots,p-2$). The minimum for f_1 and f_{p-1} is obtained in the same way.

The function f_i obtains its minimum value (see Figure C.1) at α_i^* or α_i^{**}, where α_i^* is the intersection point of the lines $\frac{2\alpha_i}{m_i\gamma}$ and $\frac{2(1-\alpha_i)}{m_{i+1}\gamma}$ and is equal to $\alpha_i^* = \frac{m_i}{m_i+m_{i+1}}$, while α_i^{**} is the root of the quantity in the absolute value, and is equal to $\alpha_i^{**} = \frac{m_i n_{i+1}}{m_{i+1}n_i+m_i n_{i+1}}$. Substituting α_i^* and α_i^{**} in f_i we get the following equalities

$$f_i(\alpha_i^*) = \frac{4+|n_i-n_{i+1}|}{\gamma(m_i+m_{i+1})}, \tag{C.1.3}$$

and

$$f_i(\alpha_i^{**}) = \frac{2(n_i+n_{i+1})}{\gamma(m_{i+1}n_i+m_i n_{i+1})}. \tag{C.1.4}$$

Next, we compare the values in (C.1.3) and (C.1.4) and prove that $f_i(\alpha_i^{**})$ is the minimum. To do so, we show that the quantity $\gamma(f_i(\alpha_i^{**}) - f_i(\alpha_i^*))$ is negative under the assumption $n_i > 2, i = 1,\ldots,p-1$. It is easy to see that

$$f_i^* \equiv \gamma(f_i(\alpha_i^{**}) - f_i(\alpha_i^*)) = \frac{2(m_{i+1}-m_i)(n_{i+1}-n_i) - |n_{i+1}-n_i|(m_{i+1}n_i+m_i n_{i+1})}{(m_i+m_{i+1})(m_{i+1}n_i+m_i n_{i+1})}.$$

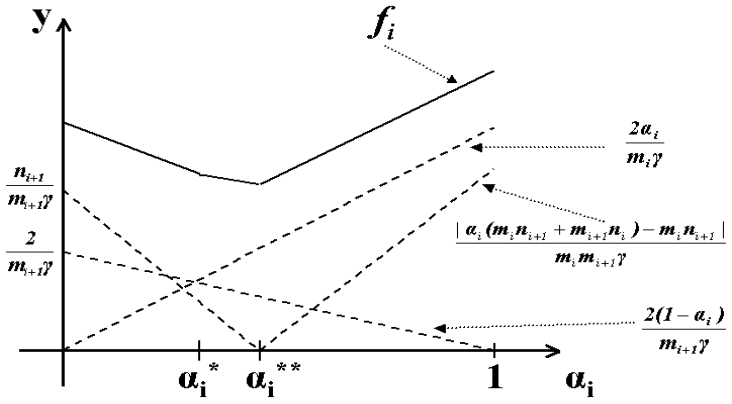


Figure C.1: The components of the sum of the absolute values of the elements of the i^{th} row of matrix M^D.

We derive 2 cases with respect to the difference in the absolute value and we have that

$$f_i^* = \begin{cases} (n_{i+1}-n_i)\left(m_{i+1}(n_i+2)+m_i(n_{i+1}-2)\right), & n_i > n_{i+1}, \\ -(n_{i+1}-n_i)\left(m_{i+1}(n_i-2)+m_i(n_{i+1}+2)\right), & n_i \le n_{i+1}. \end{cases}$$

In the first branch, where $n_i > n_{i+1}$, the quantity in the second parenthesis is positive under the assumption that $n_i > 2$ and therefore f_i^* is negative, while in the second branch the first parenthesis is positive and the second one is negative assuming that $n_{i+1} > 2$.

Hence, under the constrain $n_i > 2, i = 1, \ldots, p-1$ (the only constrain of Theorem 3.3.4), we prove that $f_i(\alpha_i^{**}) < f_i(\alpha_i^*)$, which makes α_i^{**} to be the minimum of the function f_i.

C.2 Minimize the max-norm of M^N

The max-norm of M^N is given by

$$||M^N||_\infty = \max_{1\le i\le p-1} g_i,$$

where

$$g_1(\beta_1) = \left|\tfrac{\beta_1 n_1\gamma_1}{m_1} - \tfrac{(1-\beta_1)n_2\gamma_2}{m_2}\right| + \tfrac{2(1-\beta_1)\gamma_2}{m_2},$$

$$g_i(\beta_i) = \tfrac{2\beta_i\gamma_i}{m_i} + \left|\tfrac{\beta_i n_i\gamma_i}{m_i} - \tfrac{(1-\beta_i)n_{i+1}\gamma_{i+1}}{m_{i+1}}\right| + \tfrac{2(1-\beta_i)\gamma_{i+1}}{m_{i+1}}, \quad i = 2, \ldots, p-2, \tag{C.2.1}$$

$$g_{p-1}(\beta_{p-1}) = \tfrac{2\beta_{p-1}\gamma_{p-1}}{m_{p-1}} + \left|\tfrac{\beta_{p-1}n_{p-1}\gamma_{p-1}}{m_{p-1}} - \tfrac{(1-\beta_{p-1})n_p\gamma_p}{m_p}\right|.$$

Working in the same way as in previous section, we prove that the $\beta_i, i = 1, \ldots, p-1$ as defined in (3.3.27) and (3.3.28) are the optimum values, in the sense that they minimize the max-norm of matrix M^N.

VITA

VITA

Panagiota Tsompanopoulou was born in Heraklion, Crete, Greece. In 1988 she enrolled at the Department of Mathematics of University of Crete, and received the Bachelor Degree in Mathematics in 1991. She joined the Institute of Applied and Computational Mathematics, FORTH, in June 1991 as Research Assistant and worked there until July 1996. In August 1992 she enrolled at the graduate program of Department of Mathematics of Univ. of Crete, and received the M.Sc. in Applied Mathematics in February 1995. She worked at Purdue University, Indiana, USA, from August 1996 to December 1997, as a Visiting Instructor in the Department of Mathematics, and from January 1998 to June 2000, as a Research Associate in the Computer Science Department. In January 1998 she got admission by the Graduate School of Purdue University where she received a M.Sc. in Computer Sciences in May 1999. She received the degree of Doctor of Philosophy in August 2000 from University of Crete, Greece.

Bibliography

[1] A. Bamberger, R. Glowinski, and Q.H. Tran. A domain decomposition method for the acoustic wave equation with discontinuous coefficients and grid change. *SIAM J. Numer. Anal.*, 34(2):603–639, 1997.

[2] C. Bischof, A. Carle, P. Khademi, and Mauer A. ADIFOR 2.0: Automatic differentiation of Fortran 77 programs. *IEEE Computational Science & Engineering*, 3:18–32, 1996.

[3] P. E. Bjorstad and O. Widlund. To overlap or not overlap: A note on a domain decomposition method for elliptic problems. *SIAM J. Sci. Stat. Comput.*, 10(2):1053–1061, 1989.

[4] L. Bölöni and D.C. Marinescu. An object-oriented framework for building collaborative network agents. *Intelligent Systems and Interfaces*, 2000.

[5] L. Bölöni, D.C. Marinescu, J.R. Rice, P. Tsompanopoulou, and E.A. Vavalis. Agent based networks for scientific simulation and modelling. *Concurancy: Practice and Experience.*

[6] L. Cao and J. Zhu. Accurate local time stepping for solving partial differential equations. Technical Report TRCS99-11, Department of Mathematics and Statistics, Mississippi State University, 1999.

[7] J.G. Caputo, N. Flytzanis, Y., and E. Vavalis. Two-dimensional effects in josephson junctions: I static properties. *Physical Review*, E54:2092–2021, 1996.

[8] T. F. Chan, T. Y. Hou, and P. L. Lions. Geometry related convergence results for domain decomposition algorithms. *SIAM J. Numer. Anal.*, 28(2):378–391, 1991.

[9] T. F. Chan and D. C. Resasco. A survey of preconditioners for domain decomposition. Technical Report /DCS/RR-414, Yale University, 1985.

[10] Tony Chan, Thomas Hou, and P. L. Lions. Geometry related convergence results for domain decomposition algorithms. *SIAM J. Numer. Anal.*, 28:378–391, 1991.

[11] Tony F. Chan and Danny Goovaerts. On the relationship between overlapping and nonoverlapping domain decomposition methods. *SIAM J. Matrix Anal. Appl.*, 13:663–670, 1992.

[12] Tony F. Chan and Thomas Y. Hou. Eigendecomposition of domain decomposition interface operators for constant coefficient elliptic problems. *SIAM J. Sci. Stat. Comput.*, 12:1471–1479, 1991.

[13] Tony F. Chan and Tarek P. Mathew. Domain decomposition algorithms. In *Acta Numerica 1994*, pages 61–143. Cambridge University Press, 1994.

[14] J. Crank. *Free and moving boundary problems.* Oxford University Press, Oxford, 1984.

[15] Yang D. A nonoverlapping Scharz method for elliptic interface problems with strongly discontinuous coefficients. *Numerical Algorithms*, (to appear) 2000.

[16] Carl de Boor and Amos Ron. The least solution for the polynomial interpolation problems. *Math. Zeit*, 210:705–727, 1992.

[17] Q. Deng. An analysis for a nonoverlapping domain decomposition iterative procedure. *SIAM J. Sci. Comput.*, 18:1517–1525, 1997.

[18] B. Despres. Domain decomposition method and the Helmholtz problem. In L. Halpern G. Cohen and P. Joly, editors, *Mathematical and Numerical Aspects of Wave Propagation Phenomena*, pages 44–52. SIAM, 1991.

[19] B. Despres. *Methodes de decomposition de domaines pour les problemes de propagation d'ondes en regime harmonique.* PhD thesis, Universite Paris IX Dauphine, UER Mathematiques de la Decision, 1991.

[20] M. R. Dorr. Domain decomposition via Lagrange multipliers. Technical Report 98532, Lawrence Livermore National Laboratory, Livermore. CA, 1988.

[21] M. R. Dorr. On the discretization of interdomain coupling in elliptic boundary-value problems. In Roland Glowinski, Gene H. Golub, Gérard A. Meurant, and Jacques Périaux, editors, *First International Symposium on Domain Decomposition Methods for Partial Differential Equations*, pages 17–37, SIAM, Philadelphia, PA, 1988.

[22] T.T. Drashansky. *An agent–based approach to building multidisciplinary problem solving enviroments.* PhD thesis, Purdue University, Computer Science Department, December 1996.

[23] S. Franklin and A. Graesser. Is it an agent, or just a program? In *Proceedings of the Third International Workshop on Agent Theories, Architectures and Languages.* Springer Verlag, 1996.

[24] D. Funaro, A. Quarteroni, and P. Zanolli. An iterative procedure with interface relaxation for domain decomposition methods. *SIAM J. Numer. Anal.*, 25(6):1213–1236, 1988.

[25] M.J. Gander. A wavefront relaxation algorithm with overlapping splitting for reaction diffusion equations. *Numer. Linear Algebra Appl.*, 6:125–145, 1999.

[26] W. Heinrichs. Domain decomposition for fourth–order problems. *SIAM J. Numer. Anal.*, 30(2):435–453, 1993.

[27] P. Henrici. *Applied and Computational Complex Analysis.* John Wiley, New York, NY, 1974.

[28] E.N. Houstis, J. R. Rice, S. Weerawarana, A. Catlin, M. Gaitatzes, P. Papachiou, and K. Wang. Pellpack: A problem solving environment for pde based applications on multicomputer platforms. *ACM Trans. Math. Software*, 24:30–73, 1998.

[29] P. Hovland and M. Heath. Adaptive SOR: A case study in automatic defferentiation of algorithm parameters. Technical Report ANL/MCS-P672-0697, Argonne National Laboratory, 1997.

[30] S. Jaffard. Wavelet methods for fast resolution of elliptic problems. *SIAM J. Numer. Anal.*, 29:965–986, 1992.

[31] J. Douglas Jr. and C-S Huang. An acceletarated domain decomposition procedure based on Robin transmission conditions. Technical report, Purdue University, Department of Mathematics, 1996.

[32] J. Douglas Jr., P. J. Paes Leme, J. E. Roberts, and J. Wang. A parallel iterative procedure applicable to the approximate solution of the second order partial differential equations by mixed finite element methods. *Numer. Math.*, 65:95–108, 1993.

[33] D. Keyes and W. Gropp. A comparison of domain decomposition techniques for elliptic partial differential equations and their parallel implementation. *SIAM J. Sci. Stat. Comput.*, 8:s166–s202, 1987.

[34] S. Kim. A parallelizable iterative procecedure for the Helmholtz problem. *Appl. Numer. Math.*, 17:411–425, 1995.

[35] S.-B. Kim, A. Hadjidimos, E.N. Houstis, and J.R. Rice. Multi-parameterized Schwarz splittings. *Math. Comput. Simulation*, 42:47–76, 1996.

[36] C. H. Lai. On domain decomposition and shooting methods for two-point boundary value problem. In David E. Keyes and Jinchao Xu, editors, *Seventh International Conference of Domain Decomposition Methods in Scientific and Engineering Computing*, volume 180 of *Contemporary Mathematics*, pages 257–264. AMS, 1994.

[37] P. Le-Tallec, Y. De-Roecl, and M. Vidrascu. Domain decomposition methods for large linearly elliptic 3-dimensional problems. *J. Comput. Appl. Math.*, 34(1):93–117, 1991.

[38] T.B. Lim, L.N. Sankar, N. Hariharan, and N.N. Reddy. A technique for the prediction of propeller induced acoustic loads on aircraft structures. In *AIAA-93-4340*, pages 240–265, 1993.

[39] P. L. Lions. On the Schwarz alternating method III: A variant for nonoverlapping subdomains. In R. Glowinski, G.H. Golub, G.A. Meurant, and J. Periaux, editors, *Domain Decomposition Methods for Partial Differential Equations*, pages 202–223. SIAM, 1990.

[40] Pierre Louis Lions. On the Schwarz alternating method. II. In Tony Chan, Roland Glowinski, Jacques Périaux, and Olof Widlund, editors, *Domain Decomposition Methods*, pages 47–70, Philadelphia, PA, 1989. SIAM.

[41] S. Markus, E. Houstis, A. Catlin, J.R. Rice, P. Tsompanopoulou, E. Vavalis, D. Gottfried, and K. Su. An agent-based netcentric framework for multidisciplinary problem solving. Technical Report CS-TR-00-003, Purdue University, 2000.

[42] H.S. McFaddin. *An Object-based Problem Solving Environment for Collaborating PDE Solvers and Editors.* PhD thesis, Computer Science Department, Purdue University, 1992.

[43] H.S. McFaddin and J.R. Rice. Collaborating pde solvers. *Applied Numerical Mathematics*, 10:279–295, 1992.

[44] K. Miller. Numerical analogs of the Schwarz alteranting procedure. *Numer. Math.*, 7:91–103, 1965.

[45] M. Mu. Solving composite problems with interface relaxation. *SIAM J. Sci. Comput.*, 20:1394–1416, 1999.

[46] M. Mu and J.R. Rice. Modeling with collaborating PDE solvers - theory and practice. *Computing Systems in Engineering*, 6:87–95, 1995.

[47] R. Natarajan. Domain decomposition using spectral expansions of Steklov–Poincaré operators. *SIAM J. Sci. Comput.*, 16(2):470–485, 1995.

[48] R. Natarajan. Domain decomposition using spectral expansions of Steklov–Poincaré operators II: A matrix formulation. *SIAM J. Sci. Comput.*, 18(4):1187–1199, 1997.

[49] A. Quarteroni and A. Valli. *Domain decomposition methods for partial differential equations*. Oxford University Press, Oxford, 2000.

[50] Alfio Quarteroni, Yuri A. Kuznetsov, Jacques Périaux, and Olof B. Widlund, editors. *Domain Decomposition Methods in Science and Engineering: The Sixth International Conference on Domain Decomposition*, volume 157 of *Contemporary Mathematics*. AMS, 1994. Held in Como, Italy, June 15–19,1992.

[51] L.B. Rall. *Automatic differentiation: Techniques and applications*, volume 120 of *Lecture Notes in Computer Science*. Springer-Verlag, New York, 1981.

[52] J.R. Rice. Split runge-kutta method for simultaneous equations. *Journal of Research*, 64B:151–170, 1960.

[53] J.R. Rice. An Agent–based architecture for solving partial differential equations. *SIAM News*, 31, 1998.

[54] J.R. Rice and R.F. Boisvert. *Solving Elliptic Problems Using ELLPACK*. Springer-Verlag, New York, NY, 1985.

[55] J.R. Rice, P. Tsombanopoulou, E. Vavalis, and D. Yang. Domain decomposition methods for underwater acoustics problems. Technical report, Computer Science Department, Purdue University, W. Lafayette, IN, in preparation.

[56] J.R. Rice, P. Tsompanopoulou, and E. Vavalis. Automated estimation of relaxation parameters for interface relaxation. Technical Report CSD-TR-98-018, Purdue University, Computer Science Department, 1998.

[57] J.R. Rice, P. Tsompanopoulou, and E. Vavalis. Fine tuning interface relaxation methods for elliptic differential equations. Technical Report CSD-TR-98-017, Purdue University, Computer Science Department, 1998.

[58] J.R. Rice, P. Tsompanopoulou, and E.A. Vavalis. Review and performance interface relaxation methods for elliptic pdes. Technical Report CSD-TR-97-004, Purdue University, W. Lafayette. IN, 1996.

[59] J.R. Rice and E. Vavalis. Collaborative agents for modeling air pollution. *Systems Analysis Modeling Simulation*, 32:93 – 101, 1998.

[60] J.R. Rice, E. Vavalis, and D. Yang. Analysis of a non–overlapping domain decomposition method for elliptic PDEs. *J. Comput. Appl. Math.*, 87:11–19, 1997.

[61] H. A. Schwarz. *Gesammelte Mathematische Abhandlungen*, volume 2, pages 133–143. Springer Verlag, Berlin, Germany / Heidelberg, Germany / London, UK / etc., 1890. First published in Vierteljahrsschrift der Naturforschenden Gesellschaft in Zürich, volume 15, 1870, pp. 272–286.

[62] H.A. Schwarz. Uber einige abbildungsaufgaben. *J. Reine Angew. Math.*, 70:105–120, 1869.

[63] M.J. Smith. Computational considerations of an eluer/navier-stokes aeroelastic method for a hovering rotor. In *AIAA-95-182*, pages 193–215, 1995.

[64] R. V. Southwell. Stress-calculation in frameworks by the method of "systematic relaxation of constraints". *Proc. Roy. Soc. Edinburgh Sect. A*, 151:57–91, 1935. parts I and II.

[65] R. V. Southwell. Stress-calculation in frameworks by the method of "systematic relaxation of constraints". *Proc. Roy. Soc. Edinburgh Sect. A*, 153:41–76, 1935. part III.

[66] A. Tersenov. On quasilinear non–uniformly parabolic equations in general form. *J. of Differential Equations*, 142(2):263–276, 1998.

[67] P. Tsompanopoulou, L. Bölöni, D.C. Marinescu, and J.R.Rice. The design of software agents for a network of pde solvers. In *Proc. Workshop on Agent Technologies for High Performance Computing, at Agents 99*, pages 57–68, 1999.

[68] H.T.M. van der Maarel and A. Platschorre. Optimization of flexible computing in domain decomposition for a system of PDEs. In *Proc. of the Ninth Intl. Conf. on Domain Decomposition Methods*. 1998.

[69] R.S. Varga. *Matrix Iterative Analysis*. Prentice-Hall, Inc., New Jersey, 1962 (2^{nd} ed. revised and expanded, Springer; Berlin, Heidelberg, NewYork, 2000.).

[70] E. Vavalis. A collaborative framework for air pollution simulations. *NATO-ASI series*, pages 349–358, 1999.

[71] E. Vavalis. Agent and runtime support for collaborative air pollution models. *Systems Analysis Modeling Simulation*, (to appear)2000.

[72] V.S. Verykios. *Knowledge Discovery in Scientific Databases*. PhD thesis, Purdue University, 1999.

[73] A. Wiegmann and K.P. Bube. The immersed interface method for nonlinear differential equations with discontinuous coefficients and singular sources. *SIAM J. Numer. Anal.*, 35:177–200, 1998.

[74] J. Xu and J. Zou. Non–overlapping domain decomposition methods. Technical report, Mathematics Department, Pennsylvania State University, University Park, PA, 1996.

[75] D. Yang. A parallel iterative nonovelapping domain decomposition procedure for elliptic problems. *IMA J. Numer. Anal.*, 16:75–91, 1996.

[76] D. Yang. A non-overlapping domain decomposition method for elliptic interface prroblems. Technical Report IMA # 1472, University of Minnesota, 1997.

[77] D. Yang. A parallel domain decomposition algorithm for elliptic problems. *J. Comp. Math.*, 16:141–151, 1998.

[78] D. Yang. A parallel grid modification and domain decomposition algorithm for local phenomena capturing and load balanching. *J. Scientific Computing*, 12:99–117, 1998.