ELSEVIER

Applied Numerical Mathematics 32 (2000) 219–245

APPLIED NUMERICAL MATHEMATICS

www.elsevier.nl/locate/apnum

Interface relaxation methods for elliptic differential equations ☆

J.R. Rice *, P. Tsompanopoulou [1], E. Vavalis [1]

Purdue University, Computer Science Department, West Lafayette, IN 47907, USA

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.

Keywords: Interface relaxation; Domain decomposition; Multiphysics problems; Elliptic differential equations

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 [19] and survey the associated preconditioning strategies [4,7] 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

☆ Work supported in part by PENED grants 95-602 and 95-107, NSF grants CCR-9202536 and CDA-9123502, and AFOSR FM 49620-92-J-0069. An early version of this paper was presented at the Ninth International Domain Decomposition Conference.

* Corresponding author. E-mail: jrr@cs.purdue.edu

[1] Authors permanent address: University of Crete, Mathematics Department, 714 09 Heraklion, Greece and IACM, FORTH, 711 10 Heraklion, Greece.

0168-9274/00/$20.00
PII: S0168-9274(99)00016-1

methods and the existence of equivalence relations between them have already received a great deal of study [2,3,5].

Interface relaxation methods are taking us a step beyond non-overlapping domain decomposition [28]. 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 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 these methods is beyond the scope of this paper. 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 [6] of certain interface operators (i.e., Lagrange multipliers [11,12] or Steklov–Poincaré operators [26,27]). We will only briefly describe these techniques.

The rest of the paper 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. 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 [1,9,20,30].

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 [35].
- 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 [13].
- 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 [13].

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 [33] and for a general formulation and analysis of interface relaxation methods to [24]. 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 Fig. 1 where the generic relaxation formula $g_{i,j}$ (based on the 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 $b_{i,j}^{\text{New}}$ to the solution on the interface $\Gamma_{i,j}$ between them.

To formally describe this method we consider the differential problem

$$Du = f \quad \text{in } \Omega, \qquad Bu = c \quad \text{on } \partial\Omega, \tag{1}$$

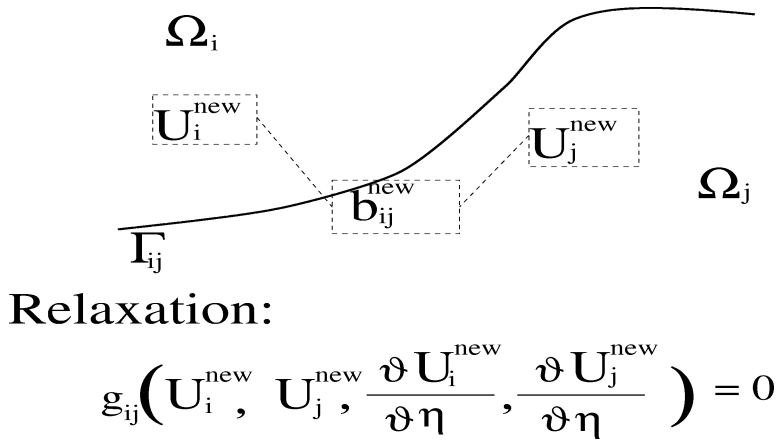

Fig. 1. The interface relaxation mechanism.

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

$$D_i u = f_i \quad \text{in } \Omega_i, \qquad G_i u = 0 \quad \text{on } \partial\Omega_i \backslash \partial\Omega, \qquad B_i u = c_i \quad \text{on } \partial\Omega_i \cap \partial\Omega, \tag{2}$$

where $i = 1, \ldots, p$. These differential problems are coupled through the interface conditions $G_i u = 0$ 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_i is associated with the interface relaxation method and different selections for the G_i'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 (1) at differential level and then discretize the resulting differential subproblems (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) instead of the global differential problem (1).
- They can utilize software parts technology by reusing existing "legacy" software parts for solving the individual subproblems (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, etc.).

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 and not a numerical analysis problem (see [24] for a discussion).

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 $u = \text{solve\_pde}(ui, \text{d}ui)$ 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 $\text{d}ui$. 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 $k = 0, 1, 2, \ldots$

$u^{(k+1/2)} = \text{solve\_pde}(ui)$ in each subdomain,

$$\text{d}ui = \beta \frac{\partial u_R^{(k+1/2)}}{\partial x} + (1-\beta) \frac{\partial u_L^{(k+1/2)}}{\partial x} \text{ on each interface,}$$

$u^{(k+1)} = \text{solve\_pde}(\text{d}ui)$ in each subdomain,

$$ui = \alpha u_R^{(k+1)} + (1-\alpha) u_L^{(k+1)} \text{ on each interface,}$$

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 [25]. In particular, in [34] a convergence analysis of the method is carried out at a differential level using Hilbert space techniques. In [36] 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 [31] to obtain sharp convergence results and to estimate optimum values for the relaxation parameters involved for simple model problems.

The geometric (**GEO**) *contraction based method*

GEO estimates the new solution for each subdomain by solving a Dirichlet problem and is classified as a *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 Fig. 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 $k = 0, 1, 2, \ldots$

$$ui^{(k+1)} = ui^{(k)} - \frac{w_L w_R}{w_L + w_R}\left(\frac{\partial u_L^{(k)}}{\partial x} - \frac{\partial u_R^{(k)}}{\partial x}\right) \text{ on each interface,}$$

$$u^{(k+1)} = \text{solve\_pde}(ui^{(k+1)}) \text{ in each subdomain.}$$

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

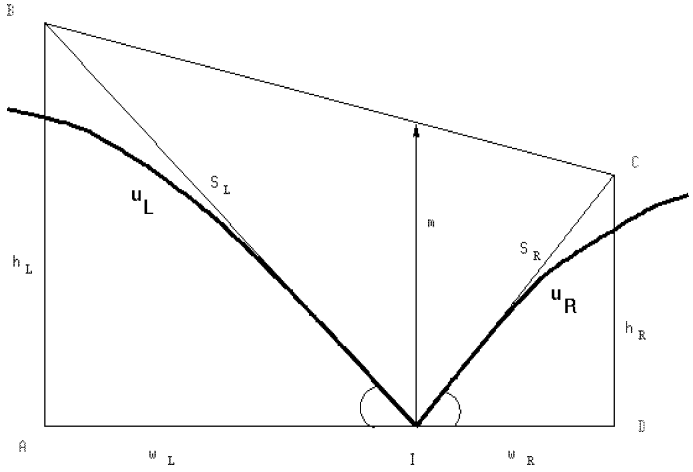

Fig. 2. Cross-section perpendicular to the interface where u_L and u_R have slopes S_L and S_R at the interface point I. Changing the values of u_L and u_R by m makes these slopes equal.

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 $\partial u_R^{(i)}/\partial x$, $\partial u_L^{(i)}/\partial x$.

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

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

Step 2. Apply linearization to solve the equations approximately:

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

Step 3. Approximate the unknown derivatives by differences:

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

Step 4. Solve for δ_R and δ_L from

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

so the

$$\delta_R = \delta_R = \delta = \frac{\left[\partial u_R^{(2)}/\partial x - \partial u_L^{(2)}/\partial x\right]}{A_L - A_R}.$$

The values of $\partial u_L/\partial x$ and $\partial u_R/\partial x$ 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 $k = 2, 3, 4\ldots$

Solve for corrections $\delta_L = \delta_R = \delta$ from the interface system as above:

$$\begin{aligned}
&\left(u_L^{(k)} + \delta_L\right) - \left(u_R^{(k)} + \delta_R\right) = 0,\\
&\frac{\partial u_L^{(k)}}{\partial x}\left(u_L^{(k)} + \delta_L\right) - \frac{\partial u_R^{(k)}}{\partial x}\left(u_R^{(k)} + \delta_R\right) = 0,\\
&ui^{(k+1)} = ui^{(k)} + \delta \text{ on each interface,}\\
&u^{(k+1)} = \text{solve_pde}(ui^{(k+1)}) \text{ in each subdomain.}
\end{aligned}$$

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 [23] and analyzed later in [10,18]. One solves the local PDE on the subdomains using Robin conditions on the interface lines by matching a convex combination of Dirichlet and Neumann data from the neighboring subdomains.

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

Here ρ is a relaxation parameter. The convergence of this method was analyzed in [23] 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 [17] an ADI-based modification for accelerating the convergence of the **ROB** scheme is proposed and analyzed. A modification of **ROB** that extends its applicability and frees it from the cross-point trouble is formulated and analyzed in [8]. Another variation that uses the tangential derivatives in addition to the normal derivative for smoothing is given in [32] 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 [14] (see also the references therein). It alternates Dirichlet and Neumann interface conditions in space and can be described by

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

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 [14] 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 [15].

The Shooting (**SHO**) *method*

This method is proposed in [21] 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 $D(ui^{(k)}) \equiv \partial u_L^{(k)}/\partial x - \partial u_R^{(k)}/\partial x = 0$ on the interfaces using a fixed point (Picard) iteration scheme to obtain new values.

for $k = 0, 1, 2, \ldots$

$\alpha^{(k+1)} = \dfrac{\alpha^{(k)}|D(ui^{(k+1)})|}{|D(ui^{(k)}) - D(ui^{(k+1)})|}$ 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 [22] 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 $k = 0, 1, 2, \ldots$

$u^{(k+1/2)} = \text{solve\_pde}(ui)$ in each subdomain,

$\mathrm{d}ui = \dfrac{1}{2}\left(\dfrac{\partial u_R^{(k+1/2)}}{\partial x} + \dfrac{\partial u_L^{(k+1/2)}}{\partial x}\right)$ on each interface,

$u^{(k+1)} = \text{solve\_pde0}(\mathrm{d}ui)$ $(Lu = 0)$ in each subdomain,

$ui = ui - \frac{1}{2}\rho\left(u_R^{(k+1)} + u_L^{(k+1)}\right)$ on each interface.

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

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 [26] and [27] 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–Poincare operator on the complementary region. An idea similar to the above approach is used in [11] and [12] 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 multiplier 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 square fit to approximately satisfy the over determined 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.

4. Numerical experiments

An extensive and systematic performance evaluation study of all the interface relaxation schemes presented above is under way for general 2-dimensional decompositions using the SciAgents [13] system. In this section we present and discuss numerical performance data mainly for 1-dimensional problems. These problems might be too simple to be of practical importance, but these experiments can be very illuminating for understanding the nature of the interface relaxation method. 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 $u'' - \gamma u = f$ in $[0, 1]$, where f is selected such that $u(x) = e^{x+4}x(x-1)(x-0.7)$ 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. [2] Central differences are used to discretize the differential equation. The interval $[0, 1]$ 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 schemae in a straight forward and naive way. In particular, we set $\alpha = \beta = \frac{1}{2}$ in **AVE**, $w_L = w_R =$ half the length of the associated subdomain in **GEO**, $\rho = \frac{1}{2}$ in **ROB** and **SPO**, and $\theta = \frac{1}{2}$ in **SCO**. We also set $\gamma = 20$ for all data except Fig. 8. We select this, not very common, value of γ in order to increase the experimental data (see the discussion of Figs. 3 and 4) that can be fitted into the plots (in particular, in Figs. 3 and 8) so a clear qualitative comparison picture can be easily drawn.

We have verified that the convergence rate of all methods is independent of the local grid size. A very representative set of data is given in Table 1 where we present the convergence factor

$$\phi_k = \sqrt[k]{\frac{\|Du^{(k)} - f\|_\infty}{\|Du^{(0)} - f\|_\infty}} \quad \text{for } k = 2, 4, 6, 8, 10 \tag{3}$$

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 Fig. 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 Fig. 3). Nevertheless, **AVE** is the fastest for 2 and 4 subdomains. **NEW**and **SHO** behave in

[2] `http: //www.cs.purdue.edu/homes/mav/projects/dom_dec_code`

Table 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]

		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	0.11	4.89E−2	0.33	7.53E−3	0.48	4.56E−3	0.58	4.3E−3	0.64
	$N = 160$	6.24E−1	0.11	4.57E−2	0.33	4.23E−3	0.48	1.25E−3	0.58	1.0E−3	0.64
GEO	$N = 80$	2.74E−0	0.08	8.42E−1	0.32	2.99E−1	0.48	1.06E−1	0.58	3.7E−2	0.64
	$N = 160$	2.74E−0	0.08	8.40E−1	0.32	2.99E−1	0.48	1.06E−1	0.58	3.7E−2	0.64
NEW	$N = 80$	6.10E−0	0.03	7.20E−1	0.33	1.57E−1	0.48	1.97E−1	0.58	1.7E−1	0.64
	$N = 160$	6.10E−0	0.02	7.22E−1	0.33	1.58E−1	0.48	1.08E−1	0.58	7.8E−2	0.64
ROB	$N = 80$	5.43E−0	0.15	3.21E−0	0.36	1.96E−0	0.50	1.23E−0	0.59	7.9E−1	0.64
	$N = 160$	5.45E−0	0.15	3.24E−0	0.36	2.00E−0	0.50	1.27E−0	0.59	8.1E−1	0.64
SCO	$N = 80$	2.09E−0	0.13	4.83E−1	0.34	8.22E−2	0.48	2.04E−2	0.58	1.1E−2	0.64
	$N = 160$	2.09E−0	0.13	4.90E−1	0.34	8.41E−2	0.48	1.49E−2	0.58	3.4E−3	0.64
SHO	$N = 80$	5.40E−0	0.04	3.82E−1	0.33	1.12E−1	0.48	6.96E−2	0.58	4.5E−3	0.64
	$N = 160$	5.40E−0	0.04	3.83E−1	0.33	1.13E−1	0.48	6.79E−2	0.58	3.9E−3	0.64
SPO	$N = 80$	2.31E−0	0.09	3.74E−1	0.33	6.59E−2	0.48	1.25E−2	0.58	2.3E−3	0.64
	$N = 160$	2.31E−0	0.09	3.73E−1	0.33	6.57E−2	0.48	1.26E−2	0.58	2.6E−3	0.64

a similar and rather erratic way. We also plot in Fig. 5 the convergence factor ϕ_k (formula (3)) 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 $\gamma = 20$ 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. 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 approximates zero. This is clearly reflected in Fig. 4, where we present the convergence rate data, as in Fig. 3 but now for $\gamma = 1$. Besides the break down of **AVE** and **SPO** for more than two subdomains, it is also seen a general decrease, relatively to the $\gamma = 20$ case, on the rates of convergence.

We believe that the specific convergence pattern might give important information about the convergence characteristics. To explore this, Fig. 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 Fig. 6 correspond to their Dirichlet sweeps. In Fig. 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.

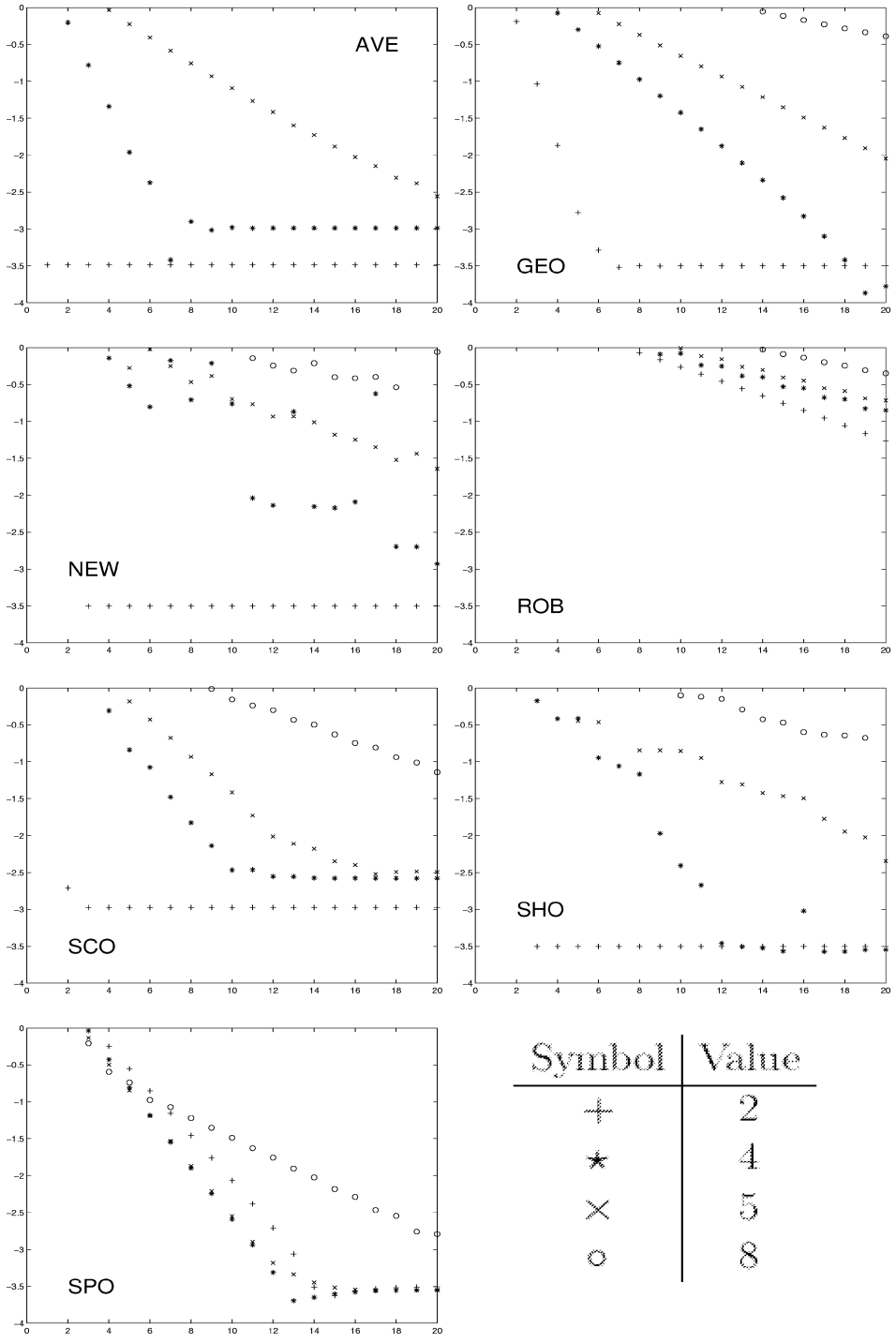

Fig. 3. Convergence plots for 2 (+), 4 (∗), 5 (×) and 8 (○) subdomains, for $\gamma = 20$. On the x-axis we have the iteration number and on the y-axis the logarithm of the max-norm of the error.

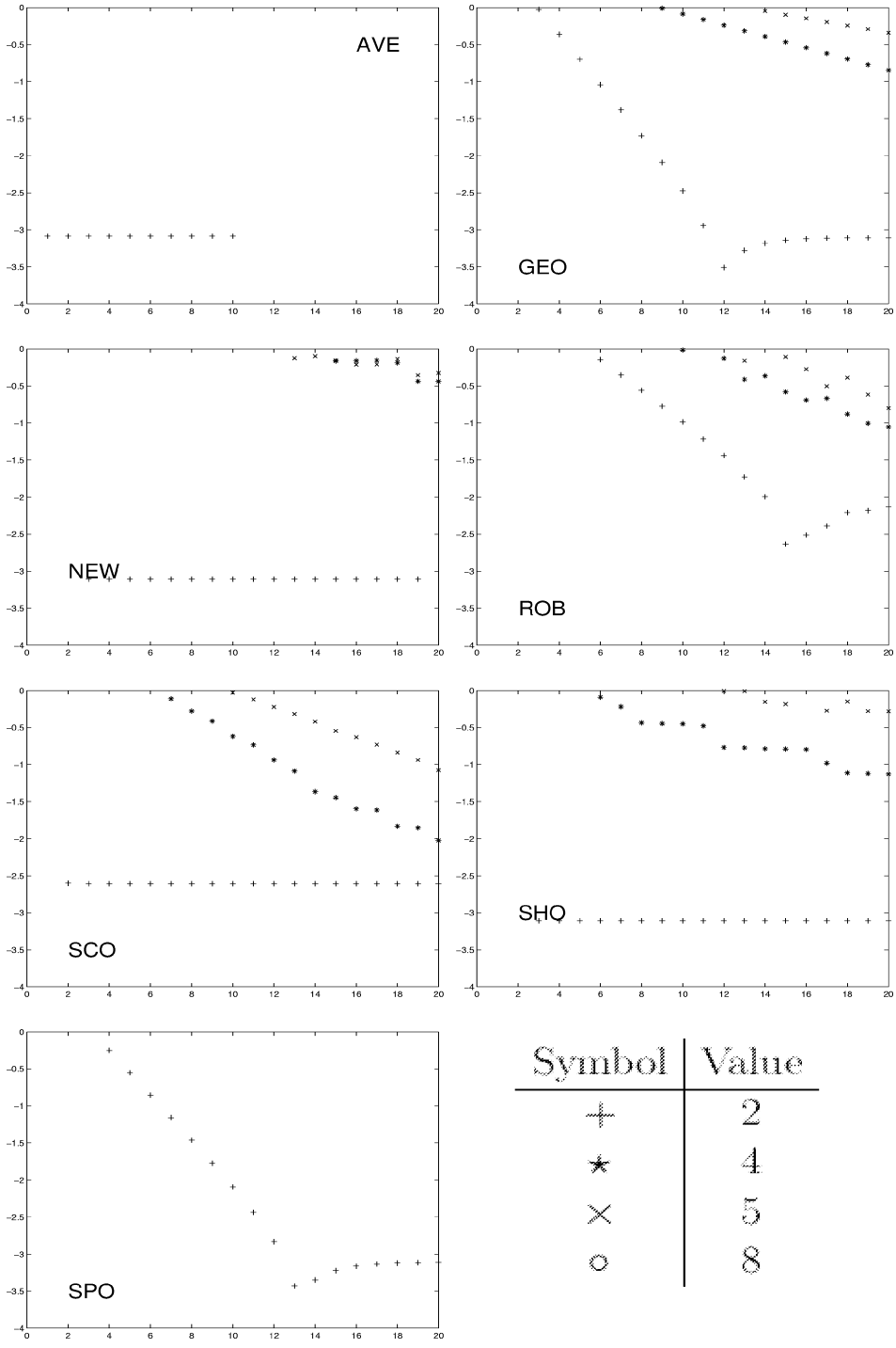

Fig. 4. Convergence plots for 2 (+), 4 (∗), 5 (×) and 8 (○) subdomains, for $\gamma = 1$. On the x-axis we have the iteration number and on the y-axis the logarithm of the max-norm of the error.

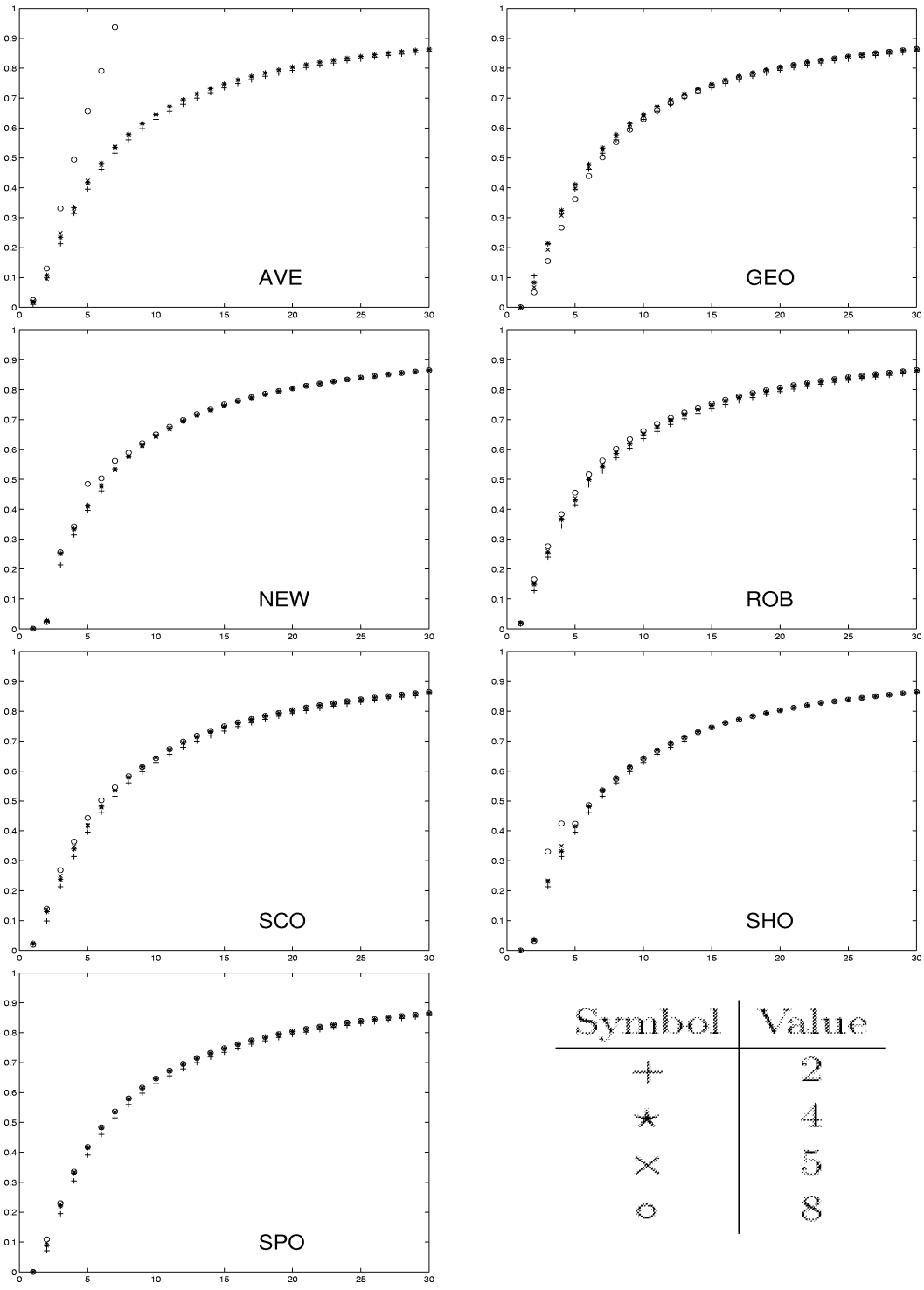

Fig. 5. The convergence factor ϕ_k versus iterations for all methods, for 2 (+), 4 (∗), 5 (×) and 8 (○) subdomains.

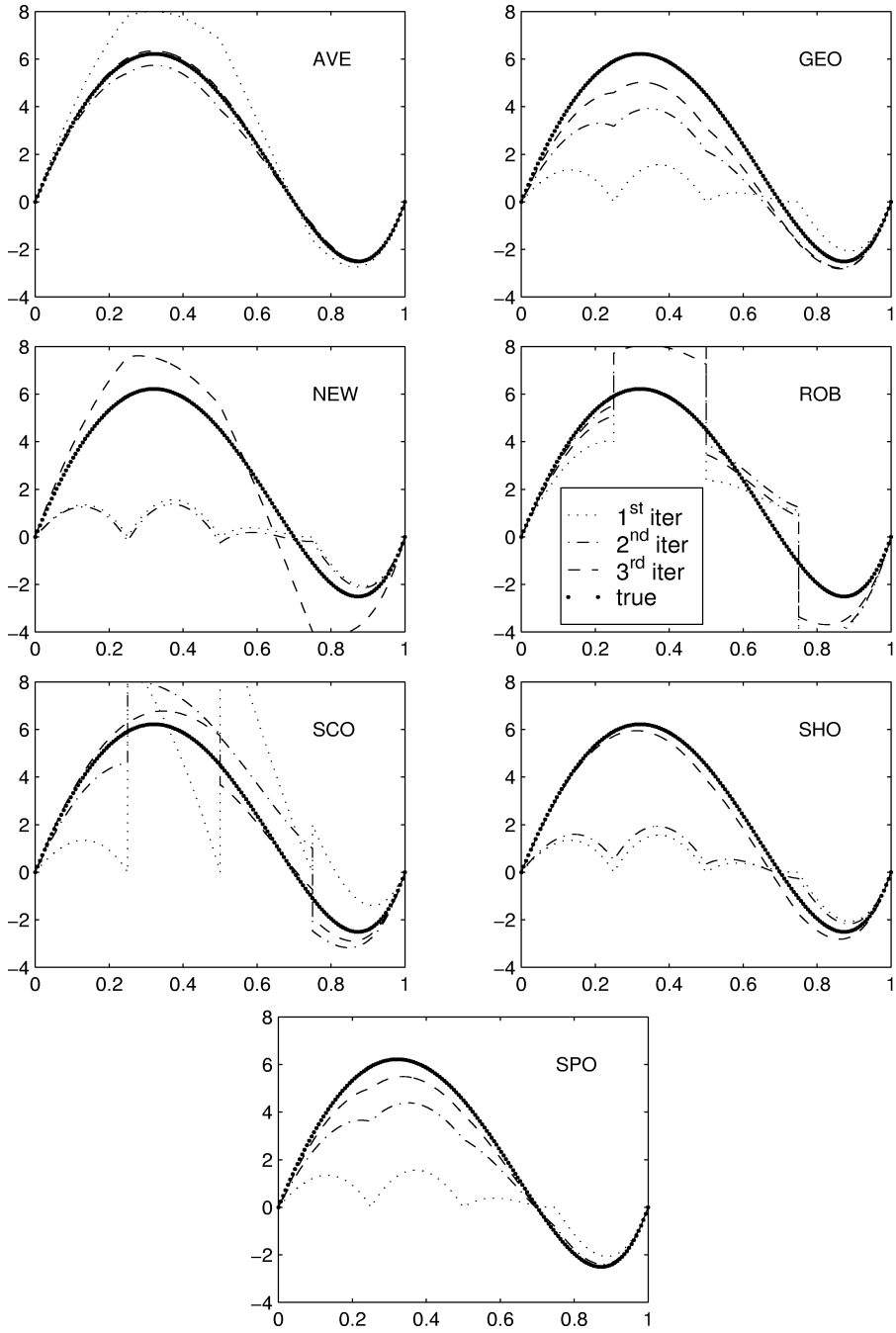

Fig. 6. Convergence history of the seven relaxers in a 4 subdomain decomposition and $\gamma = 20$. The true solution (solid line) is plotted along with the first (dotted line), second (dot-dashed line), and third (dashed line) computed solutions.

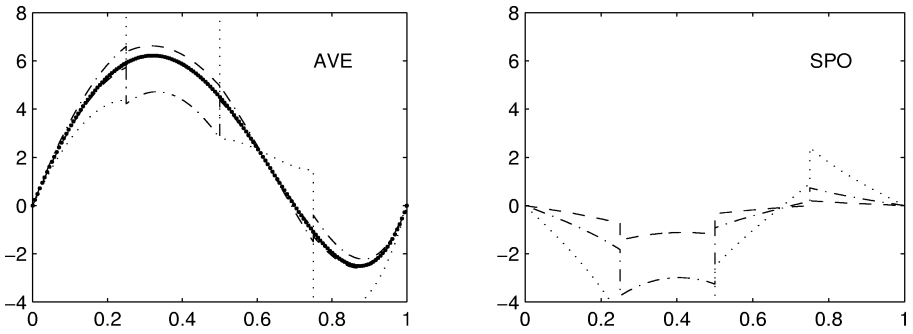

Fig. 7. Convergence history of the two-step relaxers, **AVE** and **SPO**, during the Neumann sweep.

As it is obviously expected, and already seen in Figs. 3 and 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 γ $(= 10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 20, 30)$ in the operator and observe the convergence for a 4 subdomain uniform partition of Ω. Our data are presented in Fig. 8 which uses the same axes as in Fig. 3. As γ 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 $\gamma \leqslant 1$ while the data for the rest methods are split in two groups, one for $\gamma = 10, 20$ and 30 and one for $\gamma = 0.1, 0.01$ and 0.001 with the case of $\gamma = 1$ in the later 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 0.2 to 0.4, 0.6 and 0.8. In Fig. 9, where we consider the four different 2 subdomain partitions, we clearly see that none of the schemes is very is sensitive to this change. Again the axes are as in Fig. 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 straight forward way for the experiments described above. In Fig. 10 we systematically vary the values of these parameters and present convergence plots for the 2 subdomain case with the interface point at 0.8. 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 Fig. 10 are: **AVE** $(\alpha = \beta = 0.3)$, **GEO** $(w_L = w_R = 0.6)$, **ROB** $(\rho = 0.9)$, **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 for **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 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., $\|u^{(k+1)} - u^{(k)}\|_\infty < 10^{-5}$) as a function of the starting subdomain q. Specifically, for Table 2 we use a uniform decomposition of 8 subdomains and we start the domain decomposition scheme from

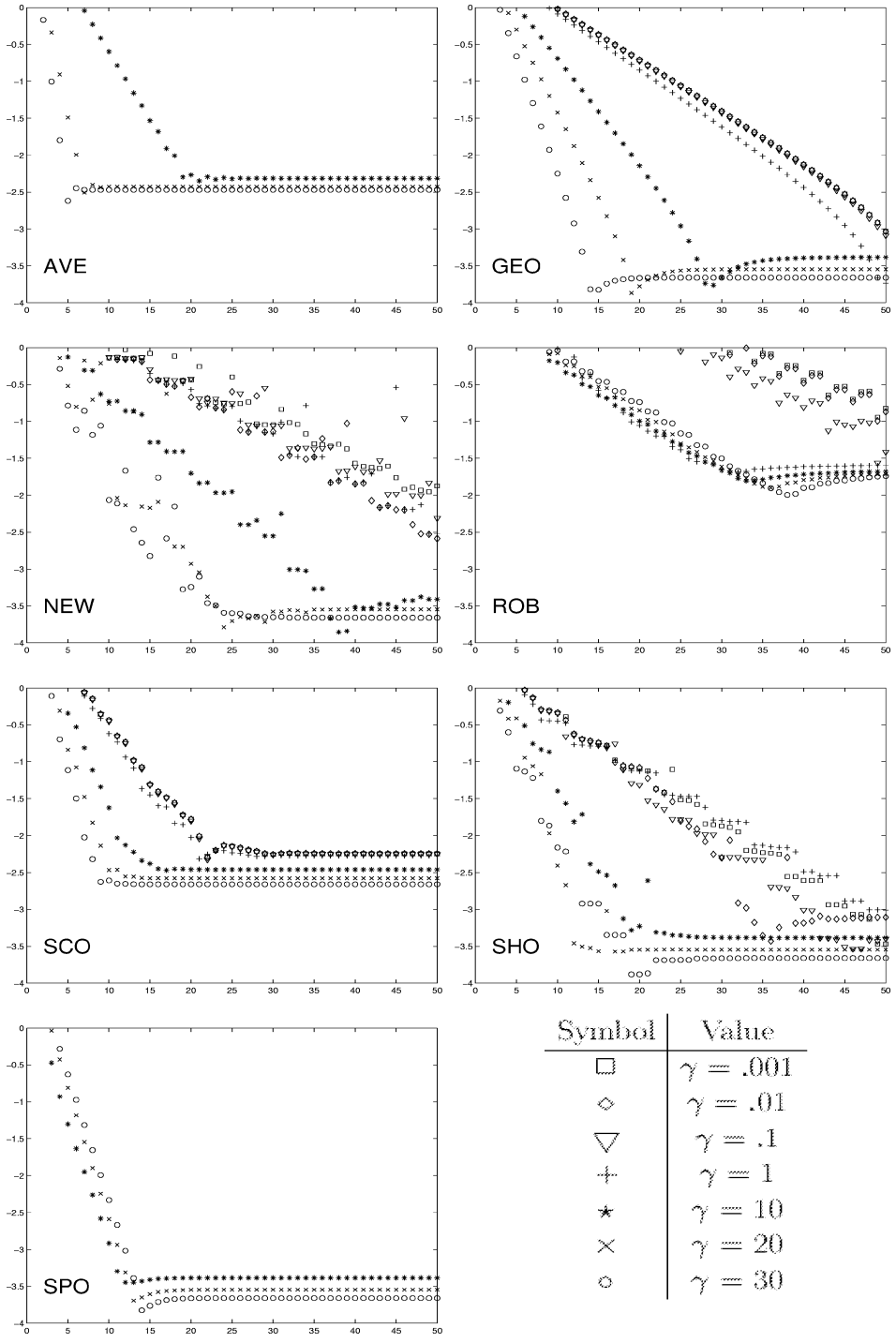

Fig. 8. The effect of the coefficient γ on the convergence rate of the relaxation schemes with a four subdomain partition of $[0, 1]$. The legends and the value of γ are given in the lower right.

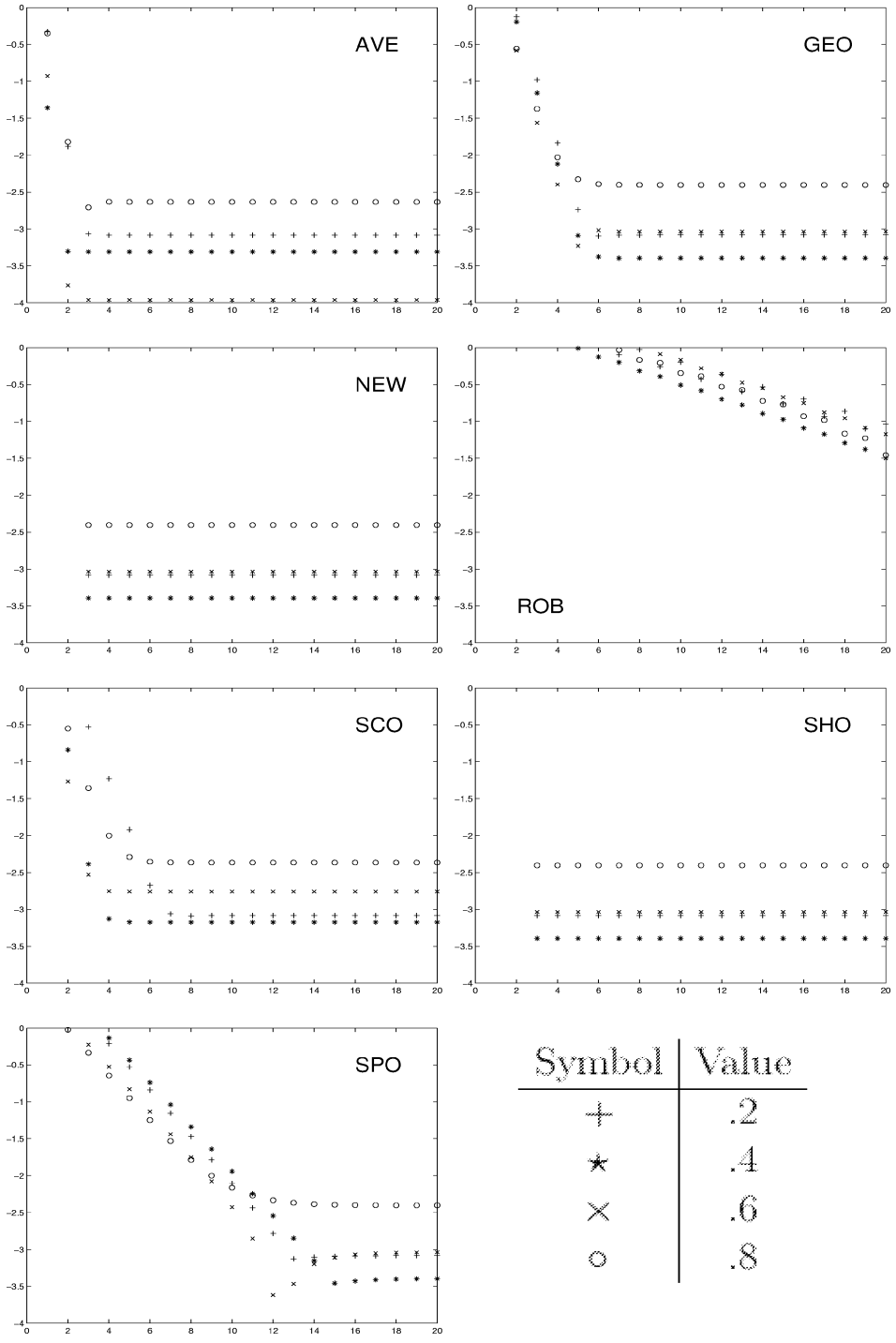

Fig. 9. The convergence of the relaxation schemes for non-uniform 2 subdomain decompositions. The interface point ip is placed at 0.2 (+), 0.4 (∗), 0.6 (×) and 0.8 (○).

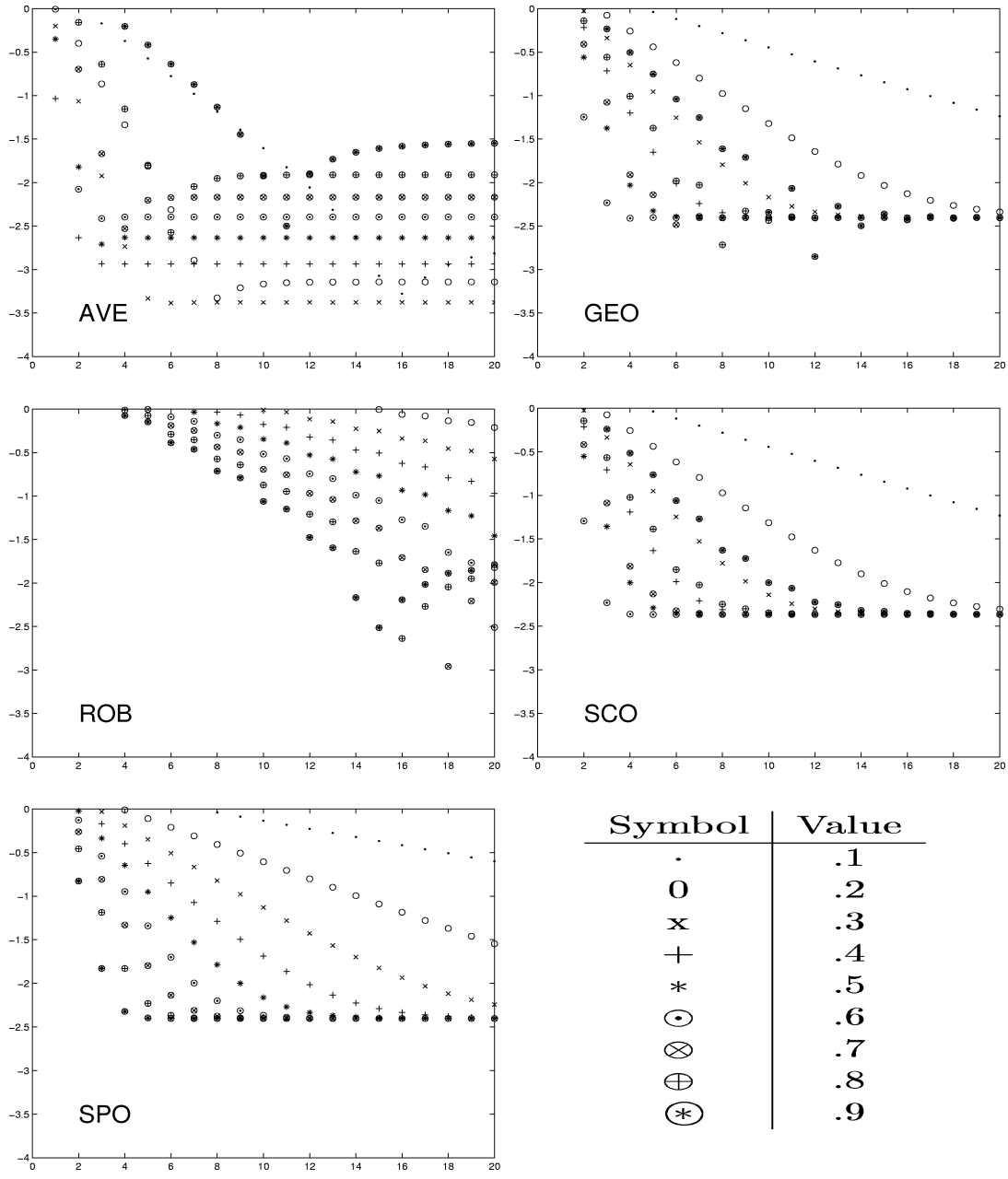

Fig. 10. The effect of the parameter selection on the convergence behavior of five relaxation methods for $\gamma = 1$. The legends and parameter values used are given in the lower right, the two parameters of **AVE** and **GEO** are both set equal to the value shown.

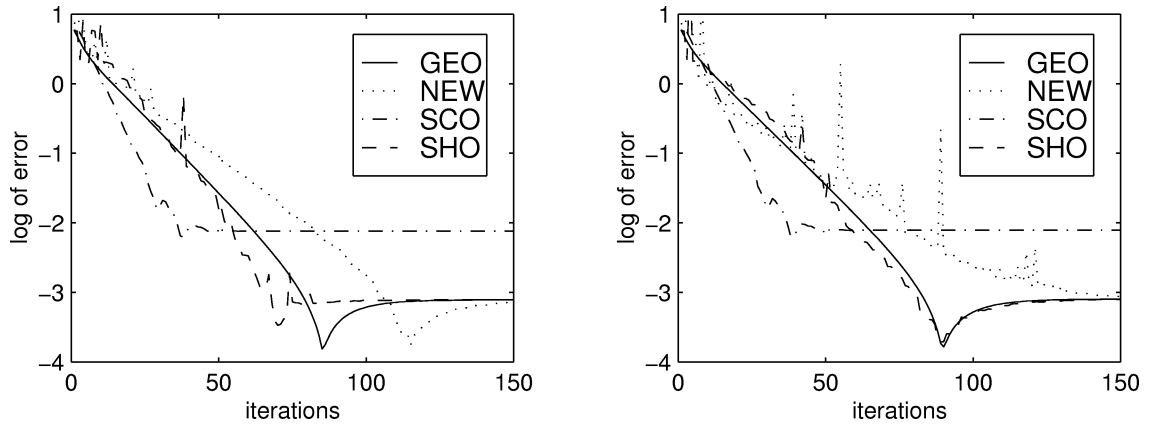

Fig. 11. The history of convergence of four relaxation methods for $u'' + \sin(2\pi x)u = f$ (on the left) and $u'' + \cos(2\pi x)u = f$ (on the right) in the case of 5 sub-domains of equal size.

Table 2
Number of iterations k to achieve $\|u^{(k+1)} - u^{(k)}\|_\infty < 10^{-5}$ for various starting subdomains in the **SCO** method

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

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 $-u'' + \cos(2\pi x)u = f$, that do not satisfy the ellipticity condition and appear (in a 2-dimensional form) in practical applications. Only four methods (**GEO**, **NEW**, **SCO**, **SHO**) converge for these indefinite problems. Fig. 11 shows the convergence behavior of these methods for both problems. We see that the convergence rate is comparable to that seen in Fig. 3. The rest either diverge or oscillate. Such behavior has been already noticed for some of the methods [10]. 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 [24] 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 2-dimensional problems using Ellpack [29] assuming "skyline" domains (a string of rectangles of different heights and widths).[3] This leads to 1-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 1-dimensional experiments presented above. The detailed presentation of our performance is beyond the scope of this article. Nevertheless, in the right plot of Fig. 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 $u(x) = \mathrm{e}^{y(x+4)}x(x-1)(x-0.7)y(y-0.5)$. We assume

[3] This Ellpack code is also available from our web page.

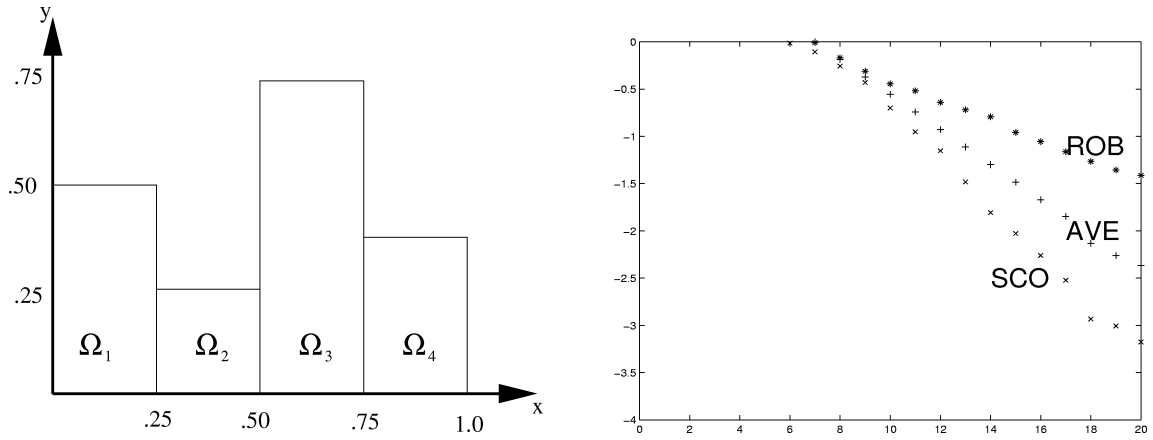

Fig. 12. The history of convergence of **AVE** (+ symbols), **ROB** (∗) and **SCO** (×) methods for $\Delta u - 10u = f$ (on the right) assuming the PDE domain and its partition given on the left.

Dirichlet boundary conditions. The PDE domain and its 1-dimensional partition into 4 subdomains $\Omega \equiv \bigcap_{i=1}^{4} \Omega_i$ is depicted in left plot of Fig. 12. The 5-point star Ellpack discretization module was used. The similarity of the convergence behavior of the three methods in 1-dimension (Fig. 8) and in 2-dimensions (Fig. 12) is easily observed.

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 Fig. 13. This figure also summarizes the mathematical and computational properties of the methods. It is seen that the speed of 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 Fig. 13), the particular method (finite element, finite difference, etc.) used to discretize the PDE operators (column "PDE discretization") and the geometric characteristics (rectangular or not, holes, wide angles, etc.) of the domains (column "PDE domain"). The theory and the experimental data available to explain all these cases and phenomena is very limited. In Fig. 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.

We start our iterations with a zero initial guess. Nevertheless, we expect that for many problems with discontinuities or for 2-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

	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**		

Fig. 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.

interior of the domain using either a blending technique [29] or a wavelet approach [16]. In any case, better initial guesses provide faster solutions and more robust computations.

The principal conclusions of this study 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.

Appendix

In this appendix we give the detailed algorithms for the seven relaxation methods in the 1-dimensional case.

Algorithm 1. The Dirichlet/Neumann averaging (**AVE**) method

$$g_i^i = \beta_i \frac{\mathrm{d}u_i^{(2k)}}{\mathrm{d}x}\bigg|_{x=x_i} + (1-\beta_i)\frac{\mathrm{d}u_{i+1}^{(2k)}}{\mathrm{d}x}\bigg|_{x=x_i}, \quad i = 1,\ldots,p-1,$$

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

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

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

Algorithm 2. The Geometric (**GEO**) contraction based method

$$g_i^i = u_i^{(k)} - \frac{w_i^i w_i^{i+1}}{w_i^i + w_i^{i+1}} \left(\frac{\mathrm{d}u_i^{(k)}}{\mathrm{d}x} - \frac{\mathrm{d}u_{i+1}^{(k)}}{\mathrm{d}x} \right)\Big|_{x=x_i}, \quad i = 1, \ldots, p-1,$$

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

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

Algorithm 3. The Newton's (**NEW**) method

$$\left.\begin{array}{l}
\alpha_i^i = \dfrac{\frac{\mathrm{d}u_i^{(k)}}{\mathrm{d}x}\big|_{x=x_i} - \frac{\mathrm{d}u_i^{(k-1)}}{\mathrm{d}x}\big|_{x=x_i}}{u_i^{(k)}|_{x=x_i} - u_i^{(k-1)}|_{x=x_i}} \\
\alpha_i^{i+1} = \dfrac{\frac{\mathrm{d}u_{i+1}^{(k)}}{\mathrm{d}x}\big|_{x=x_i} - \frac{\mathrm{d}u_{i+1}^{(k-1)}}{\mathrm{d}x}\big|_{x=x_i}}{u_{i+1}^{(k)}|_{x=x_i} - u_{i+1}^{(k-1)}|_{x=x_i}}
\end{array}\right\} \quad i = 1, \ldots, p-1,$$

$$\left.\begin{array}{l}
h_i^i = u_i^{(k)}\big|_{x=x_i} + \dfrac{\alpha_i^{i+1} u_{i+1}^{(k)}|_{x=x_i} - \alpha_i^{i+1} u_i^{(k)}|_{x=x_i} - \frac{\mathrm{d}u_{i+1}^{(k)}}{\mathrm{d}x}\big|_{x=x_i} + \frac{\mathrm{d}u_i^{(k)}}{\mathrm{d}x}\big|_{x=x_i}}{\alpha_i^{i+1} - \alpha_i^i} \\
h_i^{i+1} = u_{i+1}^{(k)}\big|_{x=x_i} + \dfrac{\alpha_i^i u_{i+1}^{(k)}|_{x=x_i} - \alpha_i^i u_i^{(k)}|_{x=x_i} - \frac{\mathrm{d}u_{i+1}^{(k)}}{\mathrm{d}x}\big|_{x=x_i} + \frac{\mathrm{d}u_i^{(k)}}{\mathrm{d}x}\big|_{x=x_i}}{\alpha_i^{i+1} - \alpha_i^i}
\end{array}\right\} \quad i = 1, \ldots, p-1,$$

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

Algorithm 4. The Robin relaxation (**ROB**) method

$$\left.\begin{array}{l}
g_i^i = \dfrac{\mathrm{d}u_{i+1}^{(k)}}{\mathrm{d}x}\Big|_{x=x_i} + \lambda_i u_{i+1}^{(k)}\big|_{x=x_i} \\
g_i^{i+1} = -\dfrac{\mathrm{d}u_i^{(k)}}{\mathrm{d}x}\Big|_{x=x_i} + \lambda_i\, \mathrm{d}u_i^{(k)}\big|_{x=x_i}
\end{array}\right\} \quad i = 1, \ldots, p-1,$$

$$\begin{array}{l|l}
Lu_1^{(k+1)} = f \text{ in } \Omega_1 & Lu_p^{(k+1)} = f \text{ in } \Omega_p \\
u_1^{(k+1)}\big|_{x=x_0} = 0 & -\dfrac{\mathrm{d}u_p^{(k+1)}}{\mathrm{d}x}\Big|_{x=x_{p-1}} + \lambda_{p-1} u_p^{(k+1)}\big|_{x=x_{p-1}} = g_{p-1}^p, \\
\dfrac{\mathrm{d}u_1^{(k+1)}}{\mathrm{d}x}\Big|_{x=x_1} + \lambda_1 u_1^{(k+1)}\big|_{x=x_1} = g_1^1 & u_p^{(k+1)}\big|_{x=x_p} = 0
\end{array}$$

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

Algorithm 5. The Schur complement (**SCO**) method
For $c \in \{1, \ldots, p\}$,

$$\left.\begin{array}{l}
h_{i-1}^i = \begin{cases} 0, & i = 1 \\ \theta_{i-1} u_{i-1}^{(k+1)}\big|_{x=x_{i-1}} + (1-\theta_{i-1}) u_i^{(k)}\big|_{x=x_{i-1}}, & i = 2, \ldots, c-1 \end{cases} \\
g_i^i = \dfrac{\mathrm{d}u_{i+1}^{(k)}}{\mathrm{d}x}\Big|_{x=x_i} \\
Lu_i^{(k+1)} = f \text{ in } \Omega_i, \quad u_i^{(k+1)}\big|_{x=x_{i-1}} = h_{i-1}^i, \quad \dfrac{\mathrm{d}u_i^{(k+1)}}{\mathrm{d}x}\Big|_{x=x_i} = g_i^i
\end{array}\right\} \quad i = 1, \ldots, c-1,$$

$$\begin{array}{l}
h_{c-1}^c = \theta_{c-1} u_{c-1}^{(k+1)}\big|_{x=x_{c-1}} + (1-\theta_{c-1}) u_c^{(k)}\big|_{x=x_{c-1}}, \\
h_c^c = \theta_c u_{c+1}^{(k)}\big|_{x=x_c} + (1-\theta_c) u_c^{(k)}\big|_{x=x_c}, \\
Lu_c^{(k+1)} = f \text{ in } \Omega_c, \quad u_c^{(k+1)}\big|_{x=x_{c-1}} = h_{c-1}^i, \quad u_c^{(k+1)}\big|_{x=x_c} = h_c^c,
\end{array}$$

$$\left.\begin{aligned} h_i^i &= \begin{cases} \theta_i u_{i+1}^{(k)}\big|_{x=x_i} + (1-\theta_i)u_i^{(k)}\big|_{x=x_i}, & i = c+1,\dots,p-1 \\ 0, & i = p \end{cases} \\ g_{i-1}^i &= \left.\frac{\mathrm{d}u_{i-1}^{(k+1)}}{\mathrm{d}x}\right|_{x=x_{i-1}} \\ Lu_i^{(k+1)} &= f \text{ in } \Omega_i, \quad \left.\frac{\mathrm{d}u_i^{(k+1)}}{\mathrm{d}x}\right|_{x=x_{i-1}} = g_{i-1}^i, \quad u_i^{(k+1)}\big|_{x=x_i} = h_i^i \end{aligned}\right\} \quad i = c+1,\dots,p.$$

Algorithm 6. The Shooting (**SHO**) method

$$\left.\begin{aligned} g_i^i &= \left.\frac{\mathrm{d}u_{i+1}^{(k)}}{\mathrm{d}x}\right|_{x=x_i} + \lambda_i u_{i+1}^{(k)}\big|_{x=x_i} \\ g_i^{i+1} &= \left.\frac{\mathrm{d}u_i^{(k)}}{\mathrm{d}x}\right|_{x=x_i} + \lambda_i\, \mathrm{d}u_i^{(k)}\big|_{x=x_i} \end{aligned}\right\} \quad i = 1,\dots,p-1,$$

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

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

Algorithm 7. The Steklov–Poincaré operator (**SPO**) method

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

$$g_i^{i+1} = g_i^i = \frac{1}{2}\left(\left.\frac{\mathrm{d}u_i^{(k+1)}}{\mathrm{d}x}\right|_{x=x_i} - \left.\frac{\mathrm{d}u_{i+1}^{(k+1)}}{\mathrm{d}x}\right|_{x=x_i} \right), \quad i = 1,\dots,p-1,$$

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

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

References

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

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

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

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

[5] T.F. Chan, D. Goovaerts, On the relationship between overlapping and nonoverlapping domain decomposition methods, SIAM J. Matrix Anal. Appl. 13 (1992) 663–670.

[6] T.F. Chan, T.Y. Hou, Eigendecomposition of domain decomposition interface operators for constant coefficient elliptic problems, SIAM J. Sci. Statist. Comput. 12 (1991) 1471–1479.

[7] T.F. Chan, T.P. Mathew, Domain decomposition algorithms, in: Acta Numerica 1994, Cambridge University Press, Cambridge, 1994, pp. 61–143.

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

[9] B. Despres, Domain decomposition method and the Helmholtz problem, in: L. Halpern, G. Cohen and P. Joly (Eds.), Mathematical and Numerical Aspects of Wave Propagation Phenomena, SIAM, Philadelphia, PA, 1991, pp. 44–52.

[10] B. Despres, Methodes de decomposition de domaines pour les problemes de propagation d'ondes en regime harmonique, Ph.D. Thesis, Universite Paris IX Dauphine, UER Mathematiques de la Decision, 1991.

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

[12] M.R. Dorr, On the discretization of interdomain coupling in elliptic boundary-value problems, in: R. Glowinski, G.H. Golub, G.A. Meurant and J. Périaux (Eds.), First International Symposium on Domain Decomposition Methods for Partial Differential Equations, SIAM, Philadelphia, PA, 1988, pp. 17–37.

[13] T.T. Drashansky, An agent-based approach to building multidisciplinary problem solving environments, Ph.D. Thesis, Purdue University, Computer Science Department, December 1996.

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

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

[16] S. Jaffard, Wavelet methods for fast resolution of elliptic problems, SIAM J. Numer. Anal. 29 (1992) 965–986.

[17] J. Douglas Jr., C.-S. Huang, An accelerated domain decomposition procedure based on Robin transmission conditions, Technical Report, Department of Mathematics, Purdue University, 1996.

[18] J. Douglas Jr., P.J. Paes Leme, J.E. Roberts, 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 (1993) 95–108.
[19] D. Keyes, W. Gropp, A comparison of domain decomposition techniques for elliptic partial differential equations and their parallel implementation, SIAM J. Sci. Statist. Comput. 8 (1987) s166–s202.
[20] S. Kim, A parallelizable iterative procedure for the Helmholtz problem, Appl. Numer. Math. 17 (1995) 411–425.
[21] C.H. Lai, On domain decomposition and shooting methods for two-point boundary value problem, in: D.E. Keyes and J. Xu (Eds.), Seventh International Conference of Domain Decomposition Methods in Scientific and Engineering Computing, Contemporary Mathematics, AMS, Vol. 180, 1994, pp. 257–264.
[22] P. Le-Tallec, Y. De-Roecl, M. Vidrascu, Domain decomposition methods for large linearly elliptic 3-dimensional problems, J. Comput. Appl. Math. 34 (1) (1991) 93–117.
[23] 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 (Eds.), Domain Decomposition Methods for Partial Differential Equations, SIAM, Philadelphia, PA, 1990, pp. 202–223.
[24] M. Mu, Solving composite problems with interface relaxation, SIAM J. Sci. Comput. (1999, to appear).
[25] M. Mu, J.R. Rice, Modeling with collaborating PDE solvers—theory and practice, Comput. Systems Engrg. 6 (1995) 87–95.
[26] R. Natarajan, Domain decomposition using spectral expansions of Steklov–Poincaré operators, SIAM J. Sci. Comput. 16 (2) (1995) 470–485.
[27] R. Natarajan, Domain decomposition using spectral expansions of Steklov–Poincaré operators II: A matrix formulation, SIAM J. Sci. Comput. 18 (1997).
[28] J.R. Rice, An Agent-based architecture for solving partial differential equations, SIAM News 31 (1998).
[29] J.R. Rice, R.F. Boisvert, Solving Elliptic Problems Using ELLPACK, Springer, New York, 1985.
[30] J.R. Rice, P. Tsompanopoulou, E. Vavalis, D. Yang, Domain decomposition methods for underwater acoustics problems, Technical Report, Computer Science Department, Purdue University, W. Lafayette, IN, in preparation.
[31] J.R. Rice, E. Vavalis, D. Yang, Analysis of a non-overlapping domain decomposition method for elliptic PDEs, J. Comput. Appl. Math. 87 (1998) 11–19.
[32] H.T.M. van der Maarel, A. Platschorre, Optimization of flexible computing in domain decomposition for a system of PDEs, in: Proc. of the Ninth Int. Conf. on Domain Decomposition Methods, 1999, to appear.
[33] J. Xu, J. Zou, Non-overlapping domain decomposition methods, Technical Report, Mathematics Department, Pennsylvania State University, University Park, PA, 1996.
[34] D. Yang, A parallel iterative nonoverlapping domain decomposition procedure for elliptic problems, IMA J. Numer. Anal. 16 (1996) 75–91.
[35] D. Yang, A non-overlapping domain decomposition method for elliptic interface problems, Technical Report IMA # 1472, University of Minnesota, 1997.
[36] D. Yang, A parallel domain decomposition algorithm for elliptic problems, J. Comput. Math. 16 (1998) 141–151.