Parallel Solvers
for Spline Collocation Equations

Christina C. Christara
Department of Computer Science
University of Toronto
Toronto, Ontario, Canada M5S 3G4
Tel.: (416) 978-7360, Fax : (416) 978-1931
e-mail: ccc@cs.toronto.edu

Abstract

A variety of solvers for the spline collocation equations arising from the discretisation of elliptic Partial Differential Equations (PDEs) are considered. The convergence properties of semi-iterative and Krylov subspace acceleration methods applied to the system of spline collocation equations are studied. The preconditioners tested include incomplete factorisation and SSOR for both the natural and multicolour orderings, domain decomposition based on Schur complement methods with nonoverlapping subdomains, or Schwarz methods with overlapping subdomains, and multigrid methods.

The parallelisation of some of the above iterative methods is studied and their advantages and disadvantages discussed. The communication requirements of the methods are discussed when the methods are implemented on distributed memory machines. Results which show that spline collocation methods are very competitive for the solution of PDEs are presented.

1. Introduction.

This paper considers the computation of linear elliptic Boundary Value Problems (BVPs) by spline collocation and a variety of linear solvers.

Smooth spline collocation methods offer an alternative to Galerkin Finite Element Methods (FEMs) as well as to Hermite spline collocation methods. The linear system arising from the discretisation of linear elliptic Boundary Value Problems (BVPs) in two or three dimensions by spline collocation is non-symmetric in general, but diagonally dominant under certain conditions. When either quadratic or cubic splines are used, the linear system is of lower order than the Hermite spline collocation system arising from the discretisation of the same BVP, and of smaller bandwidth and higher sparsity compared to either the respective spline Galerkin linear system or the Hermite spline collocation linear system. In addition, spline collocation methods, as all FEMs, do not require interpolation, in contrast to Finite

Difference Methods. Discretisation by collocation does not require any integration, unlike Galerkin FEMs.

Despite the advantages, spline collocation methods have not been extensively known and used, due to the fact that their standard formulation leads to non-optimal solution approximations [1]. In addition, the proofs of convergence of spline collocation methods require stronger assumptions on the smoothness of the solution and on the differential operators of the BVP than the respective Galerkin FEMs. Relatively recently, optimal formulations have been derived for certain degree splines and uniform grid discretisations [2], [3], [4], [5], [6], [7], [8], [9], [10].

In the following section, the formulation of optimal spline collocation methods is briefly presented. In the next three sections, the performance of certain solvers on the linear system of spline collocation equations is discussed. It is shown that methods, such as domain decomposition, with either overlapping or non-overlapping subdomains, and various iterative methods, with preconditioning techniques, such as multigrid or Incomplete Factorisation, perform well on the spline collocation equations. In Section 6, the parallel computation of some of the above methods is discussed. The type of parallelism studied is of coarse or medium grain. Results from the implementation of the methods on an iPSC/2 hypercube with up to 32 processors and on an Intel Paragon grid multiprocessor with up to 56 processors are presented. Section 7 summarises the conclusions from this study. The Appendix includes the tables, figures and test problems referred to throughout the paper.

2. Spline collocation methods.

Consider the second-order, two-dimensional, linear elliptic BVP described by a PDE

$$\begin{aligned} \mathbf{L}u \equiv a u_{xx} + b u_{xy} + c u_{yy} + d u_x + e u_y + f u = g \\ \text{in } \Omega \equiv (\omega_1, \omega_2) \times (\omega_3, \omega_4) \end{aligned} \tag{1.1}$$

subject to mixed boundary conditions

$$\mathbf{B}u \equiv \alpha u + \beta u_n = \gamma \text{ on } \partial\Omega \equiv \text{boundary of } \Omega \tag{1.2}$$

where u, a, b, c, d, e, f, g, α, β, γ are functions of x and y, and u_n denotes the normal derivative of u.

We assume the discretisation of the domain Ω into a rectangular grid. Spline collocation methods for discretising the BVP (1.1)-(1.2) assume, as other FEMs, the representation of the approximate solution as a linear combination of basis functions spanning an approximate space. In the case of splines, the approximate space for Problem (1.1)-(1.2) is the tensor product of one-dimensional splines, i.e., piecewise polynomials of degree k and continuity $k-1$ on the nodes of the grid. A set of points, called data points or collocation points, is chosen in $\Omega \cup \partial\Omega$. For k odd, the data points are usually the nodes of the grid, while for k even, the midpoints.

The standard formulation of spline collocation methods determines the approximation u_Δ to the solution u of (1.1)-(1.2) by forcing u_Δ to satisfy exactly the operator equation (1.1) on the data points in Ω and the boundary operator equation (1.2) on the data points on $\partial\Omega$. This formulation leads to non-optimal solution approximations, in the sense that the convergence order of the spline collocation approximation is lower than the order of the spline interpolant in the same approximation space.

The formulation of the optimal spline collocation methods is based on the construction of appropriate perturbations $\mathbb{P}_{\mathbb{L}}$ and $\mathbb{P}_{\mathbb{B}}$ of the operators $\mathbf{L}$ and $\mathbf{B}$ respectively. Two formulations were derived: a) the *one-step* or *extrapolated* methods, in which u_Δ is determined by forcing it to satisfy exactly a perturbed operator equation $(\mathbf{L}+\mathbb{P}_{\mathbb{L}})u_\Delta = g$ on the data points in Ω and a perturbed boundary operator equation $(\mathbf{B}+\mathbb{P}_{\mathbb{B}})u_\Delta = \gamma$ on the data points on $\partial\Omega$, and b) the *multiple-step* or *deferred-correction* methods, in which a low (second) order approximation is generated first, and in subsequent steps higher order approximations are generated by moving the perturbation operators to the right side of the PDE problem and applying them to the lower order approximations of the previous step(s). The number of such steps required depends on the order of the BVP and the degree of splines used. For the second-order BVP (1.1)-(1.2) and quadratic or cubic splines two steps suffice.

Both formulations are equivalent with respect to convergence properties, but the deferred-correction methods are more efficient with respect to time and memory requirements [11], [8], due to fact that they give rise to a smaller bandwidth linear system, with fewer nonzero entries per row than the extrapolated methods. In this paper, we restrict ourselves to the case of quadratic splines. The two-step Quadratic Spline Collocation (QSC) matrix for Problem (1.1)-(1.2) is block tridiagonal, with at most nine non-zero entries per row and diagonally dominant under certain conditions [12].

3. Schwarz Splitting Methods.

Schwarz Splitting Methods (SSMs) assume the partition of the domain Ω into (possibly) overlapping subdomains, so that each subdomain has a fictitious boundary, which belongs to the (interior of the) neighbour subdomains. A generic algorithm for SSMs is shown below.

Schwarz splitting algorithm
for each subdomain
    choose initial fictitious boundary conditions
endfor
repeat until satisfied
    for each subdomain
        solve subdomain (using current fictitious boundary conditions)
        update fictitious boundary conditions (using neighbours' solutions)
    endfor
    check stopping criterion
endrepeat

In the above generic Schwarz splitting algorithm, there are several parameters left to determine. The choice of fictitious boundary conditions, such as Dirichlet, Neumann, mixed or alternating Dirichlet-Neumann [13], plays an important role in the convergence of the SSM. The amount of overlapping is known to affect the convergence of the SSM. The larger the overlap the faster the convergence, but a large overlap has the disadvantage of much 'duplicate' computation. SSMs in conjunction with accelerations techniques [14] and preconditioning [15], [16] are very effective methods for the computation of PDEs. Generalised versions of the SSM allow the overlapping area to be very small or none, assuming that the fictitious boundary conditions have been chosen appropriately [13]. Additive [17] or multicoloured multiplicative versions of SSMs are very appropriate for the parallel computation of BVPs. Important advantages of the SSM over the Schur Complement Method discussed in the next section are the flexibility in the choice of discretisation method and solver in each subdomain, and the applicability to non-rectangular domains, which are the union of rectangular ones.

Standard acceleration techniques can be used in order to accelerate the above generic Schwarz splitting algorithm, in which case, the SSM is used as preconditioner. Other preconditioners, e.g. multigrid, can then be added.

Accelerated and unaccelerated SSMs applied to the two-step QSC equations were tested. The acceleration technique used in the accelerated methods is GMRES. It worths noting that Conjugate Gradient (CG) did not converge, due to the unsymmetry of the matrix. Two preconditioners were tested: the first is the plain SSM, and the second is SSM together with multigrid. For the latter case, the two preconditioned residual vectors, the one by SSM and the other by multigrid, are added. For the unaccelerated version, Dirichlet fictitious boundary conditions were used. For the accelerated versions, the residual is first split into pieces corresponding to the subdomains, by duplicating overlapping components and by setting the components corresponding to the fictitious boundary equal to zero, then the pieces are preconditioned locally, then merged back to a global preconditioned residual by summation of the overlapping components. The proof of convergence of SSM on the QSC equations for

rectangular domains or domains that are the union of rectangular domains in based on a discrete maximum principle [18].

In the following, a $d_n \times d_m$ domain decomposition is assumed, i.e., d_n subdomains in the x-dimension and d_m in the y-dimension. The total number of subdomains is denoted by D, i.e., $D = d_n \cdot d_m$. The number of overlapping grid lines along one dimension of a subdomain is denoted by *ovl*, while the grid size in one dimension by n. For brevity, we denote the unaccelerated SSM by 'Jacobi', the accelerated by GMRES SSM by 'GMRES', and the accelerated by GMRES and preconditioned with a coarse grid correction SSM by 'coarse'.

3.1. Performance of the SSM on the QSC equations.

The experiments show that the number of unaccelerated iterations required for convergence grows proportionally to the number of subdomains and the grid size in one dimension and is inversely proportional to the amount of overlapping. The number of accelerated by GMRES iterations required for convergence grows proportionally to the square root of d_n and n and inversely proportional to the square root of *ovl*, while the number of accelerated by GMRES and preconditioned by a coarse grid iterations remains almost constant as the parameters vary.

In the figures of this section, we study the convergence of the three versions of SSM as the parameters n, d_n, d_m and *ovl* vary. The results presented in the figures correspond to the first step of the QSC equations corresponding to Problem 1. Tables 3.1 and 3.2 include the convergence and timing results which were used to draw the figures of this section. Table 3.1 also includes results from the second step of QSC. The initial guess for the first step is the zero vector, while for the second step the already computed solution from the first step. Table 3.2 includes results from the Neumann Problem 2. The stopping criterion used for the accelerated methods is the relative Euclidean norm of the residual, while for the unaccelerated one the relative difference of the solutions of two consecutive iterations. The tolerance was set to 10^{-5}. The subdomain solves were carried out by direct band solvers, and the ordering of the points within a subdomain is such that the bandwidth is minimum. For the implementation of GMRES, the KSP package by W. Gropp and B. Smith was used [19]. All experiments were carried on a Sparcstation1 in double precision.

We first test the convergence of the SSMs as the overlapping increases. From Figure 3.1(a), we see that the unaccelerated SSM is much more sensitive to the number of overlapping lines than the accelerated SSMs, and it requires many more iterations, unless the number of overlapping lines increases to about 20% of the grid size in one dimension. We also see that the 'coarse' SSM is almost insensitive to the number of overlapping grid lines. In Figure 3.1(b), we plot similar data as in Figure 3.1(a), only for the accelerated SSMs, and for 4×4 domain decomposition.

We then test the convergence of the SSMs as the number of subdomains increases. From Figure 3.2(a), we see that the number of iterations grows sublinearly with the number of subdomains in the case of the 'GMRES' SSM. The 'coarse' SSM is almost insensitive to the number of subdomains. In Figure 3.2(b), we plot similar data as in Figure 3.2(a), only for the accelerated SSMs, assuming the number of subdomains in each dimension is equal to the square root of the total number of subdomains, i.e., a box-wise decomposition.

We next test the convergence of the SSMs as the grid size increases. From Figure 3.3(a), we see that while the number of iterations increases linearly with the grid size in the case of the unaccelerated SSM, it slowly increases in the case of the accelerated SSM and is almost insensitive in the case that a coarse grid correction is introduced. In Figure 3.3(b), we plot similar data as in Figure 3.3(a), only for the accelerated SSMs, and for a 4×4 domain decomposition.

On the other hand, as shown in Figure 3.4(a), the number of iterations remains relatively insensitive or constant with respect to the grid size, when the ratio of the grid size over the number of overlapping grid lines in one dimension remains constant. This is true for all three versions of the SSM tested.

The choice of box-wise or strip-wise decompositions does not affect the 'coarse' SSM, while the 'GMRES' SSM converges faster for box-wise decompositions. This is shown in Figure 3.4(b), where the total number of subdomains D remains constant and equal to 16, and the subdomains in the x- and y-dimensions take the values 1, 2, 4 and 16, 8, 4 respectively.

Finally, in Figure 3.5, we plot the time required to solve the QSC equations corresponding to Problems 1 and 2, versus the grid size. Problem 2, which has Neumann conditions in three out of four boundary lines, is distinguished by '+'. The domain decomposition and number of overlapping grid lines are chosen so that the minimum time is achieved for each grid size. The two accelerated versions of SSM are tested. The slope of each line in a log-log plot is also indicated. The SSM with the coarse grid correction, with a slope of 2.13, is asymptotically almost optimal, as expected.

The per iteration time of 'coarse' SSM is 15-20% larger than that of 'GMRES' SSM, the percentage reducing as the grid size increases. The 'coarse' SSM iteration includes the application of the extension and restriction operators, with optimal complexity $O(n^2)$, and a direct solution of a coarse problem of size 1/4 of the problem size in one dimension, i.e., 1/16 of the problem size in both dimensions. Despite the slightly larger per iteration time of the 'coarse' SSM versus the 'GMRES' SSM, the former clearly outperforms the latter in terms of both absolute total time and asymptotic performance, as expected.

From our experiments, it can be concluded that the optimal, with respect to total time, number of overlapping grid lines for the accelerated SSM with the coarse grid correction, is always one, while for the accelerated SSM without coarse grid correction it varies slightly with

the grid size, and for the unaccelerated SSM it is about 10% of the grid size in one dimension. The optimal domain decomposition is the strip-wise, due to the fact that the subdomains are solved directly. On the other hand, since the 'coarse' SSM is insensitive to the box- or strip-wise decomposition, there is not high motivation to try box-wise decompositions with iterative solution of subdomains. In Table 3.3, we present the best, with respect to total time, combination of number of overlapping grid lines and domain decomposition for several grid sizes for Problems 1 and 2.

4. Domain Decomposition − Schur Complement Methods.

We assume the subdivision, either strip-wise or box-wise, of the domain Ω into $d_n \times d_m$ non-overlapping subdomains. This subdivision leaves some data points and elements on the interfaces between subdomains, while the rest of the points and elements are either in the interior of the subdomains, or on the boundary of Ω. The width of the interfaces is the minimum that does not allow any interaction between interiors of different subdomains, that is, it is equal to the half-width of the discretisation stencil. Thus, for QSC the width of the interfaces is one. If the equations and unknowns of the linear system of equations arising from the discretisation of the BVP are ordered according to the subdivision of the domain, that is, first the interface equations/unknowns, then the interior to subdomains including the boundary of Ω equations/unknowns, the system takes the form of a 2×2 block system as in (4.1).

$$\begin{bmatrix} A_{00} & A_{01} \\ A_{10} & A_{11} \end{bmatrix} \cdot \begin{bmatrix} \mathrm{x}_0 \\ \mathrm{x}_1 \end{bmatrix} = \begin{bmatrix} \mathrm{b}_0 \\ \mathrm{b}_1 \end{bmatrix} \tag{4.1}$$

A_{11} is block diagonal, reflecting the absence of interaction between subdomains' equations/unknowns, while A_{00}, A_{01} and A_{10} are sparse. A_{00} represents the relation between interface equations/unknowns, and A_{01}, A_{10} represent the interaction between subdomains and interfaces.

Systems such as (4.1) can be solved by first solving the *capacitance* or *Schur complement* system

$$C\mathrm{x}_0 = w \tag{4.2}$$

for x_0, where $C = A_{00} - A_{01}A_{11}^{-1}A_{10}$ and $w = \mathrm{b}_0 - A_{01}A_{11}^{-1}\mathrm{b}_1$, and then solving

$$A_{11}\mathrm{x}_1 = \mathrm{b}_1 - A_{10}\mathrm{x}_0 \tag{4.3}$$

for x_1.

The method of first computing w, then solving the capacitance system (4.2), and finally solving (4.3) is referred to as a Schur Complement Method (SCM). The capacitance system is often solved by Preconditioned Conjugate Gradient (PCG) methods [20], [21], [22], [23], [24], [25], which do not require explicit knowledge of the entries of C, and thus no explicit computation of A_{11}.

For the two-step QSC equations corresponding to Problem (1.1)-(1.2) we choose as preconditioner for C the tridiagonal part T of A_{00}. This preconditioner takes substantial information from A_{00} [26], and if the ordering of the interface equations/unknowns is chosen carefully, it is block diagonal, each block corresponding to the east and north interfaces of one subdomain. More specifically, the ordering of the equations/unknowns of the east and north interfaces of a subdomain is done in a continuous way, e.g. bottom-up for the east interface and then right-to-left for the north interface. It worths noticing that, despite the unsymmetry of the capacitance system of the QSC equations, PCG with preconditioner T converged.

4.1. Performance of the SCM on the QSC equations.

Below, we present some results from the application of the SCM to the two-step QSC equations. The experiments show that the number of iterations required for convergence of PCG applied to the capacitance system of QSC equations with preconditioner T grows linearly with the square root of the order of C, i.e. the number of interface equations/unknowns n_0. This is proportional to the grid size n in one dimension and the number of subdomains in each dimension, i.e. $n_0 = O(n(d_n + d_m))$.

In Figures 4.1(a,b) and 4.2, we study the convergence of the SCM as the parameters d_n and n increase. It is shown that the number of iterations required for convergence grows linearly with the square root of d_n and n. The results for these three figures correspond to both steps of the QSC method for Problem 1. Asymptotically similar results were obtained for the non-self-adjoint Problem 3. The number of iterations for Problem 3 was about twice of that for Problem 1. The approximate space satisfies the boundary conditions by construction. The initial guess for x_0 that was used for step 1 of the QSC method was the zero vector, while in step 2 we use the already computed solution vector x_0 from step 1. The stopping criterion used is the relative Euclidean norm of the residual. The subdomain solves were carried out by direct band solvers, and the ordering of the points within a subdomain is such that the bandwidth is minimum. The tolerance was set to 5×10^{-7}. The computation was carried out on a Sparcstation1 in double precision.

Finally, in Figure 4.3, we plot the time required to solve by SCM the QSC equations corresponding to Problems 1 and 3, versus the grid size. Problem 3, which has a non-self-adjoint operator, is distinguished by '+'. The domain decomposition is chosen so that the minimum time is achieved for each grid size. Results from the first step of the QSC method are presented. The slope of each line in a log-log plot is also indicated.

From our experiments, it can be concluded that the optimal, with respect to total time, domain decomposition is the strip-wise with about as many subdomains in one dimension as one quarter of the grid lines in that dimension. This is true, despite the fact that box-wise decompositions require fewer iterations to converge than strip-wise ones with the same total number of subdomains. This is due to the fact that the subdomains are solved directly. In

Table 4.2, we present the best, with respect to total time, domain decomposition for several grid sizes for Problems 1 and 3.

5. Iterative methods.

Iterative methods have been successfully used for the efficient solution of large sparse linear systems arising from the discretisation of BVPs by several discretisation methods. In this section, we present some results from the application of several iterative methods to the system of QSC equations. The methods tested are:

Rich-CG: CG algorithm without preconditioning.

SSOR-CG: PCG algorithm with SSOR preconditioning.

SSOR-SI: SSOR algorithm with semi-iterative acceleration.

IFi-CG: PCG algorithm with Incomplete Factorisation (IF) preconditioning. The level i of the incomplete factorisation controls the fill-in allowed.

IFim-CG: Same as IFi-CG with row-sum equality modification.

IFi-OM: Truncated Orthomin algorithm with incomplete factorisation preconditioning. The level i is defined as above. The number of previous step directions varies.

IFim-OM: Same as IFi-OM with row-sum equality modification.

MGJ-BCGS:Bi-CGstab algorithm with a point Jacobi and a coarse grid preconditioner.

Two orderings were tested: natural and four-colour ordering. The WATSIT [27] package was used for all Incomplete Factorisation (IF) preconditioning methods, with relative residual tolerance 5×10^{-7} and tolerance 10^{-11} for the absolute difference between approximations in two consecutive iterations. The NSPCG package [28] was used for all SSOR acceleration methods, with relative residual tolerance 5×10^{-7}. (WATSIT was preferred over NSPCG for IF-based methods, because it was found to be faster. WATSIT does not include SSOR-based methods.) Method MGJ-BCGS uses simple local extension and restriction operators developed and analysed in [29]. The coarse problem size was 1/4 of the problem size in one dimension, i.e., 1/16 of the problem size in both dimensions. For the implementation of the BCGS acceleration the KSP package [19] was used. When using a coarse grid and a Jacobi preconditioner, BCGS was preferred to GMRES due to slightly faster convergence.

Both steps of the QSC method were tested, but the results presented in the figures correspond to the first step of the QSC equations. Tables 5.1 and 5.2 include the convergence and timing results which were used to draw the figures of this section. These tables also include results from the second step of QSC. The initial guess vector for the first step of the QSC method was the zero vector, while for the second step the initial guess vector was the already computed solution from the first step. In most cases, the second step requires about half the number of iterations of the first step. The computation was carried out on a

Sparcstation1 in double precision.

In [30] we prove that for certain PDE operators and boundary conditions the norm of the inverse matrix of QSC equations is bounded, thus its condition number grows linearly with the problem size. In addition, for PDE operators with constant coefficients and no first derivatives, the matrix of QSC equations is symmetric positive definite. Therefore, we expect the number of CG iterations to grow proportionally to n. In [29] we prove the convergence of point Jacobi integrated by a coarse grid preconditioning technique for a model one-dimensional BVP and show that the error decreases by a factor of at least 2 at every iteration.

5.1. Performance of iterative methods on the naturally ordered QSC equations.

In Figure 5.1, we plot the number of iterations required for the convergence of some iterative methods applied to the system of two-step QSC equations arising from the discretisation of Problem 1, versus the grid size in one dimension. A natural ordering of equations and unknowns is assumed. The slopes of the graphs for the IF-based methods are clearly less than 1, while for SSOR−CG is only slightly less than 1. Modification has a positive effect for the Laplace PDE operator. More specifically the slope for IF0-CG is about 0.85, while for IF0m-CG the slope is 0.5. It is worth noticing that the number of iterations for IF1m-CG is $O(\log^2 n)$ for the first step of the QSC equations and $O(\log n)$ for the second step. The number of iterations for MGJ-BCGS slightly reduces with n. The slope of the graphs for Rich−CG is 1, as expected. From experiments on other PDE problems, which are not listed here, we concluded that most iterative methods tested are quite sensitive to the PDE operator. Exceptions are methods with a coarse grid preconditioner, like MGJ-BCGS, and IF-based methods with modification. We also note that methods for symmetric systems require in general more than double the number of iterations when applied to unsymmetric systems, if they converge.

In Figure 5.2, we plot the time taken by the iterative methods to converge, for the same problem as in Figure 5.1, versus the grid size in one dimension. The slope of each line in a log-log plot is also indicated. Among the methods tested the fastest is MGJ-BCGS, which is super-optimal, next being IF1m-CG and IF2m-CG for the symmetric case, and IF1m-CG and IF1m-OM for the unsymmetric case. We should note, that the memory requirements of the IF-based methods increase with the level of fill-in, therefore level higher than 1 is not very beneficial.

5.2. Performance of iterative methods on the four-colour ordered QSC equations.

In Figure 5.3, we plot the number of iterations required for the convergence of some iterative methods applied to the system of two-step QSC equations arising from the discretisation of Problem 1, versus the grid size in one dimension. A four-colour ordering of equations and unknowns is assumed. Among the three four-colour orderings of a 9-point-star stencil we chose the one that minimises the communication requirements [31], when the methods are

implemented on distributed machines. The convergence of the iterative solvers with the other four-colour orderings was not very different. The data corresponding to Rich−CG and MGJ−BCGS are the same as for the natural ordering, since these methods are insensitive to ordering, but we do plot the data for comparison.

In Figure 5.4, we plot the time taken by the iterative methods to converge, for the same problem as in Figure 5.3, versus the grid size in one dimension. The slope of each line in a log-log plot is also indicated. The ranking of the methods is similar as that from Figure 5.2, with the exception of IF0−CG which becomes as slow as the unpreconditioned Rich−CG.

It is known that the convergence rate of some iterative methods is lower with the multicolour ordering than with the natural ordering [32]. In the case of QSC equations, the effect of ordering on the convergence of iterative methods is more evident in the case of unsymmetric systems. Multicoloured SSOR-based methods require about 20-30% additional iterations, while multicoloured IF-based methods with 0 level of fill-in require about double the number of iterations. On the other hand, the convergence of IF-based methods with level 1 or 2 of fill-in is little affected in the case of symmetric systems, but their computational complexity is affected as the fill-in increases, due to the fact that the number of nonzero matrix entries generated through the process of fill-in is larger with the multicolour ordering than with the natural ordering. We should note that only level 0 fill-in IF-based methods are straightforward parallelisable with the multicolour ordering.

6. Parallel Computation.

Most of the methods described in the previous sections are very appropriate for parallel computation. Their formulation leads to a natural way of distributing data and computation to processors, either by following a geometrical decomposition of the domain Ω, or by adopting a splitting of the algorithmic constructs, e.g. matrix rows. We have tested the parallel performance of several of these methods on both shared and distributed memory machines. In this section, we briefly discuss the parallel formulation of the methods discussed and present some performance results.

6.1. Spline collocation methods.

Spline collocation methods, as well as any discretisation method for BVPs based on stencils, are inherently parallel. Each row of the spline collocation matrix and the respective right side vector component correspond to one collocation point. The computation of the entries of a row of the spline collocation matrix and of the respective right side vector component can be done independently of the computation of the entries of another row and of other right side vector components. Thus one (or more) row(s) of the matrix and the respective right side vector component(s) can be assigned to one processor. Usually, this assignment follows a geometrical decomposition of the domain, according to which the points/rows belonging to one

(or more) subdomain(s) are assigned to one processor. The subdomain layout, in turn, matches a layout of processors and it can be one-dimensional (strip-wise), or two-dimensional (box-wise).

Collocation methods have a relative advantage with respect to Galerkin FEMs, due to the fact that discretisation by collocation is purely local, with one only function evaluation required per data point and per function of the PDE operators $\mathbf{L}$ and $\mathbf{B}$, while Galerkin methods require some communication for the summation of values corresponding to nearby points, in order to compute the appropriate integrals. The two-step QSC method, though, also requires some communication in the second step, when computing the effect of the perturbation operators $\mathbb{P}_{\mathbb{L}}$ and $\mathbb{P}_{\mathbb{B}}$.

The communication required by the second step of the QSC method (as well as that required by Galerkin) is local, as long as the assignment of matrix rows to processors follows the layout of processors. In the case that more than one subdomains are assigned per processor, in order to preserve the locality of communication, the layout of the subdomains should be a refinement of the layout of processors.

6.2. Schwarz Splitting Methods.

Like all geometry decomposition methods, SSMs are very appropriate for parallel computation. The data and computation corresponding to one subdomain are relatively independent of the data and computation corresponding to another subdomain. Thus we can assign one (or more) subdomain(s) per processor.

The communication requirements arise when (a) vector components corresponding to points in the overlapping area need to be computed, (b) vector components corresponding to points on the fictitious boundary need to be computed, (c) a matrix-vector product needs to be computed, (d) an inner product needs to be computed, and (e) the stopping criterion test takes place. Other communication requirements might arise depending on the method used. For example, when GMRES is used, a small global least squares problem has to be solved.

Communication for (a), (b) and (c) is local and along each dimension of the layout of processors/subdomains. The amount of data to be communicated is proportional to the perimeter of the subdomains for all three cases (a), (b) and (c). It is also proportional to the amount of overlapping for (a) and to the width of the discretisation stencil for (c). Global communication is required for (d) and (e), but the amount of data to be communicated between any two processors is small and fixed (one floating point number or logical flag).

Below we discuss two cases where the unaccelerated SSM may be considered advantageous over the accelerated SSMs and the SCM with respect to parallel computation.

1. In the unaccelerated SSMs, the amount of data to be transferred depends neither on the discretisation stencil, nor on the amount of overlapping. Only one line of data needs to be

transferred from a processor to each neighbour, the data on the fictitious boundary (case (b) above). Cases (a) and (c) above do not apply. But, when a Schwarz preconditioner is used for an iterative/acceleration method applied to the original linear system of equations, this is not true. In this case, communication of type (a) takes place, since, according to the method we have implemented, summation takes place for the overlapping components of the local preconditioned residuals to produce a global preconditioned residual. On the other hand, the 'coarse' SSM has an optimal number of overlapping lines equal to 1, therefore, the amount of data to be transferred does not depend on the amount of overlapping. Moreover, communication of type (c) takes places, since any iterative/acceleration method requires a matrix-vector multiplication, and thus the amount of data to be transferred depends on the discretisation stencil. This is true for both the accelerated SSMs and SCM.

2. The computation of the SSM does not depend on an inner product or norm. The norm calculation is required only for the stopping criterion test, and thus it can be carried out occasionally and not necessarily at each iteration. (This is also true for iterative solvers like Jacobi and SOR but not for CG-based methods.)

Both "advantages" discussed above are minor and are offset by the faster convergence of accelerated methods.

6.3. Schur Complement Methods.

The assignment of data and computation to processors in the case of SCMs is similar to that of SSMs with minor differences, thus the communication requirements are of similar type. In SCMs, there is no overlapping area to be processed. In addition, when computing the matrix-vector product of the capacitance matrix C by a vector v, a certain order in the migration of data is dictated by the order according to which the intermediate products $A_{10}v$, $A_{11}^{-1}(A_{10}v)$, $A_{01}(A_{11}^{-1}(A_{10}v))$ are computed [26].

6.4. Iterative Methods.

We assume that some rows or columns of the spline collocation matrix are assigned per processor. This assignment may follow the assignment of points per processor as described in Section 6.1 for the generation of the discretisation matrix. Vectors, e.g. right side, residual, etc., are assigned to the processors by components in a similar way.

The typical computation performed in one iteration of an iterative method consists of several vector-vector operations, e.g. addition or inner product of vectors, one (or more) matrix-vector product(s), possibly a forward and/or back substitution and the stopping criterion test. Certain methods may require other types of computation as well. For example, a method with a coarse grid correction like MGJ-BCGS requires local restriction and extension operators to be applied to vectors, as well as a solution of a coarse grid problem.

The addition of vectors is straightforward parallelisable and requires no communication. The inner product of vectors is performed by first computing partial inner products in each processor, then summing-up the partial inner products using global communication, as in case (d) of Section 6.2. The stopping criterion test (i.e. a norm calculation) is performed similarly (case (e) of Section 6.2). The matrix-vector product requires local communication, as in case (c) of Section 6.2.

Forward and back substitutions have degree of parallelism bounded by the bandwidth of the matrix, that is, by the grid size in one dimension, and relatively large communication requirements. Therefore, they are not very appropriate for parallel computation. Forward and back substitutions arise when an SOR based method is used, or when performing SSOR or IF preconditioning. In such cases, a multicolour ordering of the equations and unknowns is suggested.

When such an ordering is adopted, the degree of parallelism grows linearly with the total size of the problem and the communication requirements remain local. In the case of four-colour ordered QSC equations and a two-dimensional layout of processors, during a forward substitution and a triangular matrix-vector multiplication four send/receive operations per processor suffice [31], if the computation of the colours is carried out in the order of the colours around the point, where the four processors' subdomains meet. This type of communication is similar to the communication required during a matrix-vector multiplication. In the case of IF- and SSOR-based methods two such communications are needed per iteration, while truncated methods for unsymmetric systems, like Orthomin, need this type of communication as many times as previous step directions used.

As discussed in Section 5, the multicolour ordering often affects the convergence of iterative methods in a negative way. Methods like IF−CG and SSOR−CG become equally slow or even slower than Rich−CG, which is insensitive to ordering and well-parallelisable with the natural ordering. Thus there is no benefit from using multicolouring.

The coarse grid preconditioning, on the other hand, has local communication requirements when the extension and restriction operators are applied. It also requires the solution of the coarse grid problem, hence it has limited degree of parallelism. In addition, some global communication or duplicate computation is required, unless an inexact solution takes place, in which case, if there are at least as many coarse points as processors, only local communication is required.

6.5. Performance on the Paragon and the iPSC/2.

In this section, we present speedup and efficiency results from some of the methods considered.

We consider the parallel computation of Rich-CG applied to the QSC equations for Problem 1 for various grid sizes and processors layouts on the Paragon. At each iteration communication takes place for one matrix-vector multiplication, the computation of two inner products and one stopping criterion test. For the global summation of partial inner products and for the stopping criterion test the built-in functions `gssum` and `gshigh` were used. Table 6.1 presents the time spent in communication during the matrix-vector multiplication on the Intel Paragon for various grid sizes. The time is slightly affected by the processors layout, more specifically, by the maximum number of neighbours per processor, as expected. For example, a 2×8 layout is advantageous over a 4×4 one, since the former requires fewer transfers per processor. The time increases very little with the grid size and with the number of points along the perimeter of a subdomain/processor. This is due to the fact that the per byte latency factor is very small. Table 6.2 shows the time spent during a global summation on the Paragon. Although the interconnection network of the Paragon is a two-dimensional grid, the time seems to increase almost logarithmically (excluding the increase from 2 to 4 processors) with the number of processors. As shown in Figures 6.1(a,b), the speedup on the Paragon grows almost linearly with the number of processors, and increases with the size of the problem for the same number of processors.

We next consider the parallel computation of the SCM, with the tridiagonal preconditioner described in Section 4 used for the capacitance system of the QSC equations for Problem 1 on the iPSC/2. Similar type of communication as for Rich−CG is required at each iteration. For the global summation of partial inner products and for the stopping criterion test we used our own software which implements a logarithmic with respect to the number of processors fan-in/out. Table 6.3 presents the time spent in communication during the matrix-vector multiplication on the iPSC/2 for various grid sizes. Table 6.4 shows the time spent during a global summation on the iPSC/2. Communication on the iPSC/2 seems to be about 2-3 times slower than on the Paragon. (Floating-point computation on the iPSC/2 seems to be about 35 times slower than on the Paragon.) In Figure 6.2, we plot the speedup obtained on the iPSC/2 for generating the discretisation matrix and the right side vector and the speedup for the solution procedure by the SCM method with PCG solution of the capacitance matrix. The grid size remains fixed, the domain decomposition fixed to 32×1 subdomains and the number of processors varies from 1 to 32, thus assigning 32 to 1 subdomains per processor. The efficiencies observed are close to optimal.

Due to the fact that the ratio of computation over communication cost on the iPSC/2 is larger than on the Paragon, we expect to have higher efficiencies on the iPSC/2 for the same problem size.

7. Comparison and conclusions.

We have considered QSC methods for the solution of elliptic BVPs in two dimensions, a variety of solvers for QSC equations and their parallel computation.

QSC methods are an effective alternative for the solution of elliptic BVPs, as they have optimal order of convergence, and can be combined with a variety of solution techniques, including domain decomposition with overlapping or non-overlapping subdomains, various iterative techniques and preconditioners, like IF, SSOR and multigrid. Most of our results on the convergence of the methods tested agree with the results of other researchers on linear systems arising from the discretisation of BVPs by variational type finite element methods or finite difference methods.

Considering the serial time complexity of the solvers tested, the multigrid method is unquestionably the fastest solver. The SSM method with coarse grid preconditioning, is asymptotically the second fastest solver, but in absolute times both the SCMs and the IF−CG methods are very fast solvers. We point out that the SCM was accelerated by CG, while SSM by GMRES, which, with the restart of 20 used, requires on the average several matrix-vector products per iteration. This fact explains the large per iteration time of the accelerated SSMs compared to the SCMs, and consequently the good performance of the SCMs in absolute terms. We were unable to explain the discrepancy of the fact that, although both the SCM and the SSM were applied to non-symmetric matrices the former did converge with CG while the latter did not.

Considering the sensitivity of the solvers to the properties of the linear system, SSMs and MGJ-BCGS are less sensitive than SCMs and other iterative methods. We have carried out several numerical experiments with the PDE Problems 1, 2 and 3, as well as with other operators with nonconstant coefficients (not listed). Most methods tested are applicable to symmetric and midly nonsymmetric linear systems, arising from the discretisation of elliptic operators in which the coefficient of the unknown function u is of different sign than the coefficient of the second order derivatives u_{xx} and u_{yy}. A few iterative methods (e.g. Rich-CG, IF0-CG) were applicable only to very restricted operators.

Considering the parallel computation of the solvers, domain decomposition methods possess high degree of parallelism, increasing linearly with the size of the problem in all its dimensions. CG and GMRES accelerations are both well parallelisable, but the least squares solution of GMRES has limited parallelism. So does the coarse grid correction method. IF and SSOR based methods also have limited parallelism, unless combined with multicolour ordering which hinders convergence and restricts the range of applicability. The SCMs offer a reasonable compromise between fast serial solvers and highly parallelisable ones.

All methods considered have local and global communication requirements. Global communication time needs to be taken into serious consideration since it increases with the number

of processors. Global communication requirements arise when an inner product has to be computed (Krylov subspace methods) and when a stopping criterion test has to be performed (all methods). In some cases, inner products may be grouped, in order to reduce communication costs. The stopping criterion test may be performed occasionally only, thus reducing communication cost. Certain methods have additional global communication requirements, e.g. GMRES (the solution of a small least squares problem), and multigrid (the solution of a coarse problem). Local communication time can be kept constant, if the number of processors increases proportionally to the problem size. The type of local communication varies little among the methods tested.

In Figure 7.1, we compare a few of the methods considered with respect to their computational complexity, by plotting the time required for the completion of their execution, versus the grid size in one dimension. Whenever there is a choice of domain decomposition, we chose the one which has the lowest (serial) computational complexity. We also point out which methods are applicable to a midly nonsymmetric problem, and which are highly parallelisable. We include the time taken by the standard banded Gauss elimination algorithm for comparison.

Appendix

The Appendix includes all tables, figures and BVPs referred to through this manuscript.

Table 3.1. Number of iterations and time in seconds required for the convergence of three versions of SSM applied to the two-step QSC equations corresponding to Problem 1, for several grid sizes, domain decompositions and overlapping grid lines, and for the two steps of the QSC method.

grid size	no. of subdom.	domain decomp.		no. of overlap. grid lines	equat.	Jacobi no. of iter. step	Jacobi solution time step	Jacobi per iter. time	GMRES no. of iter. step		GMRES solution time step	GMRES per iter. time	coarse no. of iter. step		coarse solution time step	coarse per iter. time
n	D	d_n	d_m	ovl	%	1	1		1	2	1		1	2	1	
65	2	2	1	1	1.05				24	16	45.54	1.022	13	8	37.43	1.262
				2	1.08	60	69.83	0.337	16	11	38.77	1.036	14	9	40.46	1.304
				4	1.14	29	54.50	0.387	11	8	37.31	1.125	14	9	44.54	1.400
				6	1.20	19	49.16	0.403	10	7	39.49	1.212	12	8	45.16	1.483
				8	1.26	15	50.93	0.453	9	7	42.57	1.330	12	8	49.67	1.591
	4	4	1	1	1.14				31	21	38.09	0.748	14	9	28.97	1.006
				2	1.23	87	76.51	0.147	20	14	33.71	0.843	19	12	37.88	1.106
				4	1.42	42	58.78	0.172	15	10	35.44	0.974	16	10	40.82	1.249
				6	1.60	29	61.40	0.220								
	4	2	2	1	1.08				30	19	52.30	1.028	14	9	39.52	1.286
				2	1.16	89	97.46	0.190	21	16	47.23	1.137	16	10	44.99	1.355
				4	1.30	42	73.18	0.227	18	13	50.29	1.253	18	12	55.10	1.518
				6	1.44	28	67.57	0.247	15	11	53.00	1.391	16	10	58.79	1.663
				8	1.59	21	70.79	0.292	14	10	59.98	1.586	16	11	67.50	1.856
	8	4	2	1	1.16				50	31	53.68	0.770	16	10	31.60	1.026
				2	1.33	116	102.41	0.080	34	22	47.27	0.869	21	12	42.58	1.185
				4	1.61	56	78.46	0.099	22	15	47.62	1.111	21	12	52.45	1.396
				6	1.92	38	83.22	0.132								
	16	4	4	1					50	36	55.69	0.799	17	11	33.82	1.060
				2	1.51	144	141.61	0.050	36	23	53.42	0.944	22	14	47.26	1.265
				4	2.00	69	111.86	0.069	22	16	56.44	1.300	20	13	58.46	1.532
				6	2.56	47	124.60	0.094								
97	2	2	1	1	1.03				28	20	150.05	3.000	14	8	116.27	3.553
				2	1.05	87	259.35	0.989	19	14	127.57	3.090	14	9	119.80	3.628
				4	1.09	41	182.15	1.082	13	10	117.90	3.257	15	10	133.88	3.842
				6	1.13	28	160.10	1.117	11	8	118.45	3.438	13	9	133.12	4.015
				8	1.18	21	157.23	1.217	10	8	125.28	3.655	13	9	143.65	4.235
				9	1.20	19	158.25	1.261	10	7	129.32	3.745	13	8	147.90	4.322
	4	4	1	1	1.08				46	28	135.68	2.130	14	9	75.10	2.673
				2	1.15	124	249.64	0.373	23	17	95.38	2.340	21	13	104.35	2.982
				4	1.28	60	171.39	0.414	18	12	95.75	2.560	19	12	109.25	3.158
				6	1.40	41	166.57	0.499	14	10	100.15	2.935	17	12	119.03	3.530
				8	1.53	32	171.60	0.581	13	9	115.22	3.338	16	11	132.88	3.925
	4	2	2	1	1.06				40	25	189.23	3.060	15	9	120.70	3.605
				2	1.11	128	359.57	0.524	27	19	157.40	3.203	17	11	134.85	3.755
				4	1.19	61	247.01	0.598	20	15	150.55	3.500	20	14	163.17	4.082
				6	1.29	40	212.27	0.645	18	13	159.05	3.803	17	11	164.90	4.370
				8	1.38	31	213.06	0.727	16	12	169.60	4.173	18	13	188.02	4.733
	8	4	2	1	1.11				50	42	146.50	2.162	16	10	82.85	2.715
				2	1.21	166	328.80	0.199	48	28	156.82	2.365	23	14	113.00	3.015
				4	1.40	80	225.36	0.233	27	19	133.07	2.873	24	14	134.17	3.383
				6	1.59	54	219.41	0.293	22	16	141.88	3.385	21	13	150.68	4.000
				8	1.79	41	226.88	0.353	18	13	150.52	3.840	20	14	170.55	4.438
	16	4	4	1	1.18				50	50	149.45	2.202	17	10	86.38	2.763
				2	1.33	204	435.61	0.114	50	33	171.35	2.503	25	16	123.57	3.105
				4	1.63	99	308.08	0.141	29	20	149.20	3.055	23	15	146.67	3.745
				6	1.97	67	310.13	0.189	22	17	167.28	3.965	21	14	177.20	4.610

Table 3.2. Number of iterations and time in seconds required for the convergence of two versions of SSM applied to the first step of the two-step QSC equations corresponding to Problems 1 and 2, for the indicated grid size, and for several domain decompositions and overlapping grid lines.

					Problem 1						Problem 2	
					GMRES			coarse			GMRES	coarse
grid size	no. of subdom.	domain decomp.		no. of overlap. grid lines	no. of iter. step	solution time step	per iter. time	no. of iter. step	solution time step	per iter. time	no. of iter. step	no. of iter. step
n	D	d_n	d_m	ovl	1	1		1	1		1	1
129	2	2	1	1	35	417.41	7.129	14	281.73	8.147	74	15
				3	17	312.35	7.564	15	313.02	8.623	22	16
				5	14	302.30	7.732	15	325.42	8.797	17	17
				6	13	302.10	7.866	14	325.02	8.944	16	16
				7	12	305.18	8.080	14	336.34	9.159	15	16
	4	4	1	1	58	368.90	5.045	14	160.06	5.990	127	15
				3	22	217.75	5.802	17	203.55	6.673	45	19
				4	20	214.73	5.916	20	236.24	6.995	38	25
				5	18	218.89	6.231	16	223.11	7.276	28	17
		2	2	1	53	550.67	7.196	15	292.05	8.187	87	16
				3	29	413.79	7.759	17	337.95	8.764	61	19
				5	21	387.06	8.563	20	393.69	9.326	41	25
				6	20	388.00	8.536	17	379.62	9.547	38	20
	8	8	1	1	85	375.20	3.799	15	123.28	4.710	258	17
				3	29	202.14	4.589	18	170.65	5.641	71	20
		4	2	1	85	492.92	4.911	16	169.17	5.849	247	17
				3	45	347.80	5.682	20	226.39	6.707	74	21
				5	29	299.22	6.425	21	275.91	7.759	58	24
				6	25	299.15	6.914	24	319.46	8.047	49	30
				7	23	305.79	7.430	20	301.43	8.335	44	24
	16	16	1	1	114	429.47	3.343	17	121.34	4.315	440	19
		8	2	1	99	430.34	3.812	17	134.13	4.777	372	18
				3	59	350.88	4.737	22	202.00	5.929	190	26
				5	38	318.32	5.896	22	253.93	7.209	83	25
		4	4	1	91	524.61	4.924	17	176.95	5.909	247	17
				3	47	377.08	5.923	20	237.98	6.968	119	23
				5	32	349.81	6.975	21	306.32	8.522	70	22
				6	26	350.88	7.846	23	355.30	9.058	59	29
				7	24	367.70	8.623	20	351.62	9.547	50	24

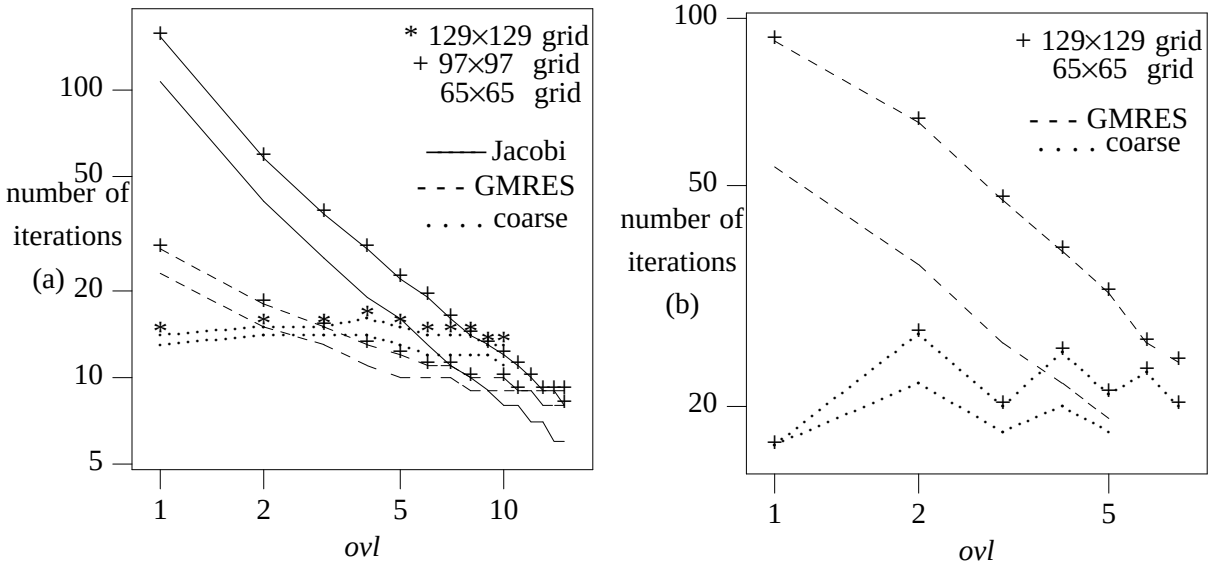

Figure 3.1. Log-log scale plot of the number of iterations required for convergence of three versions of the SSM on the QSC equations corresponding to Problem 1 for the indicated grid sizes, versus the number of overlapping grid lines *ovl* between neighbouring subdomains. A 2×1 domain decomposition is chosen for (a), while a 4×4 domain decomposition is chosen for (b).

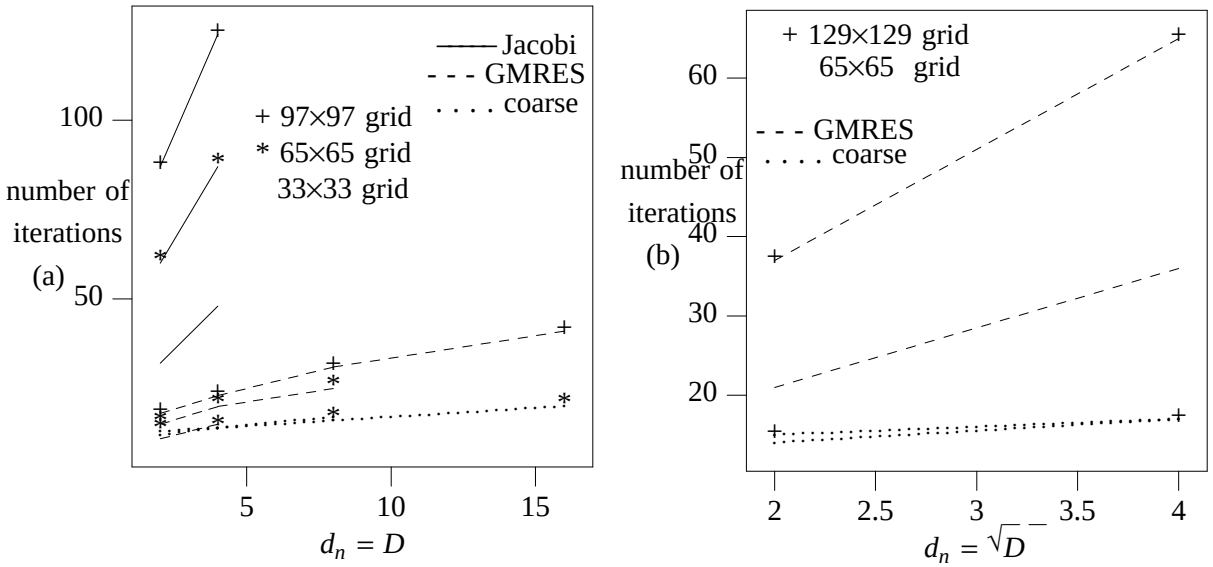

Figure 3.2. Plot of the number of iterations required for convergence of three versions of the SSM on the QSC equations corresponding to Problem 1 for the indicated grid sizes, versus the number of subdomains d_n in one dimension assuming a strip-wise decomposition for (a), and a square box-wise decomposition for (b). The number of overlapping grid lines is kept constant along each of the plotted lines; *ovl* = 1 for 'coarse', *ovl* = 2 for 'GMRES' and for 'Jacobi'.

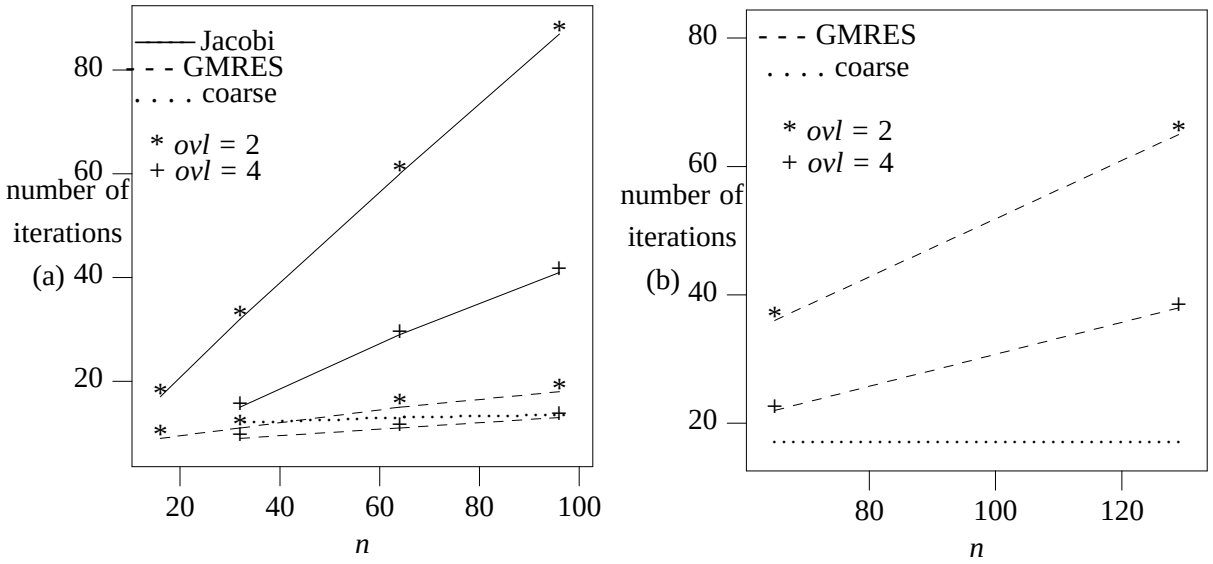

Figure 3.3. Plot of the number of iterations required for convergence of three versions of the SSM on the QSC equations for Problem 1 and for the indicated number of overlapping grid lines *ovl*, versus the grid size *n* in one dimension. A 2×1 domain decomposition is chosen for (a), while a 4×4 domain decomposition is chosen for (b).

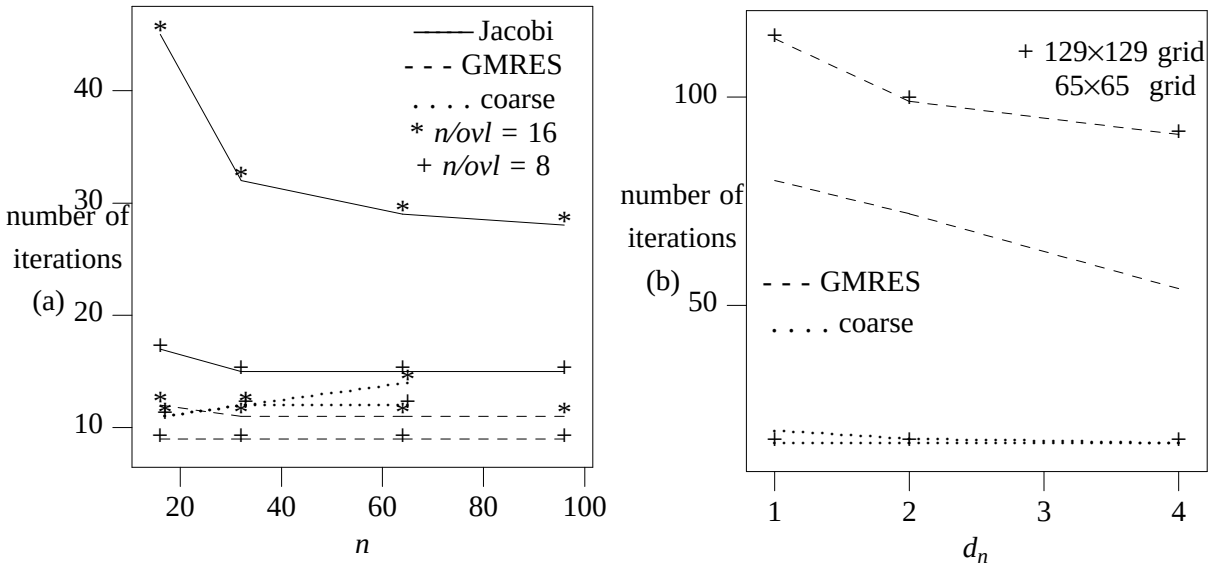

Figure 3.4. Plot of the number of iterations required for convergence of three versions of the SSM on the QSC equations for Problem 1, when (a) *n/ovl* remains constant versus the grid size *n* in one dimension (a 2×1 domain decomposition is chosen), and (b) the total number of subdomains remains constant, $D = 16$, versus the number of subdomains d_n in one dimension (the overlapping *ovl* is chosen to be 1).

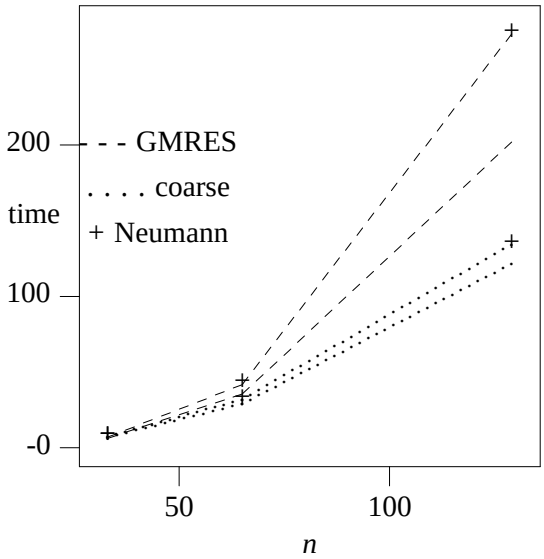

Figure 3.5. Plot of the time that two versions of SSM require to solve the QSC equations corresponding to Problems 1 and 2 versus the grid size n in one dimension. The optimal combination of domain decomposition and overlapping grid lines is chosen. The slopes in a log-log plot of the lines from top to bottom are 2.68, 2.54, 2.16 and 2.13.

Table 3.3. The optimal, with respect to total time, combination of number of overlapping grid lines and domain decomposition for three versions of SSM applied to the two-step QSC equations corresponding to Problems 1 and 2, for several grid sizes. Entries on the second row for the same grid and method indicate almost equivalent timing results for different domain decompositions.

	Problem 1						Problem 2			
	coarse		GMRES		Jacobi		coarse		GMRES	
n	$d_n \times d_m$	ovl	$d_n \times d_m$	ovl	$d_n \times d_m$	ovl	$d_n \times d_m$	ovl	$d_n \times d_m$	ovl
33	2×1	1	2×1	2	2×1	4	4×1	1	2×1	2
65	8×1	1	4×1	2	2×1	6	8×1	1	2×1	4
									4×1	4
97	8×1	1	4×1	3	2×1	10				
129	16×1	1	8×1	3			8×1	1	4×1	6
							16×1	1		

Table 4.1. Number of iterations and time in seconds required for the SCM with PCG solution of the Schur complement system (4.2) of the two-step QSC equations corresponding to Problems 1 and 3, for several grid sizes and domain decompositions, and for the two steps of the QSC method.

grid size	no. of subdom.	domain decomp.		rows in C	Problem 1 number of iterations		solution time	per iter. time	Problem 3 number of iterations		solution time
n	D	d_n	d_m	n_0	nit		$t(n,d_n,d_m)$		nit		$t(n,d_n,d_m)$
					step		step		step		step
					1	2	1		1	2	1
64	2	2	1	63	19	13	16.66	0.270	40	16	27.08
	4	2	2	125	20	14	16.83	0.143	47	24	29.96
		4	1	189	26	16	10.19	0.091	53	25	18.14
	8	4	2	249	31	20	11.51	0.056	65	30	21.31
		8	1	441	33	20	7.72	0.049	79	34	16.71
	16	4	4	369	30	20	10.95	0.042	62	33	20.16
		8	2	497	37	25	8.66	0.044	89	40	18.99
		16	1	945	44	26	8.37	0.040	102	47	18.46
	32	8	4	609	41	25	9.58	0.036	84	41	18.21
		16	2	993	50	33	9.60	0.046	112	54	20.42
	64	8	8	833	40	27	9.77	0.046	90	43	20.18
		16	4	1089	52	32	10.97	0.052	115	55	22.57
128	2	2	1	127	27	15	204.20	1.995	55	14	309.65
	4	2	2	253	28	17	206.21	0.989	63	24	334.43
		4	1	381	36	18	102.88	0.557	72	25	176.96
	8	4	2	505	42	23	114.30	0.288	82	31	195.28
		8	1	889	46	23	67.38	0.205	100	33	131.35
	16	4	4	753	42	25	117.60	0.192	87	32	202.20
		8	2	1009	49	27	71.87	0.117	114	40	147.92
		16	1	1905	56	32	50.37	0.099	142	45	121.19
	32	8	4	1249	58	30	79.33	0.083	116	41	149.21
		16	2	2017	68	35	60.15	0.085	155	53	131.35
		32	1	3937	71	42	53.52	0.095	192	61	138.38
	64	8	8	1729	55	31	75.37	0.073	123	42	155.31
		16	4	2241	72	37	63.94	0.077	160	55	137.27
		32	2	4033	88	47	66.35	0.089	217	71	158.70
	128	16	8	2689	70	39	63.97	0.074	162	52	142.10
		32	4	4225	69	51	54.39	0.090	220	69	166.09

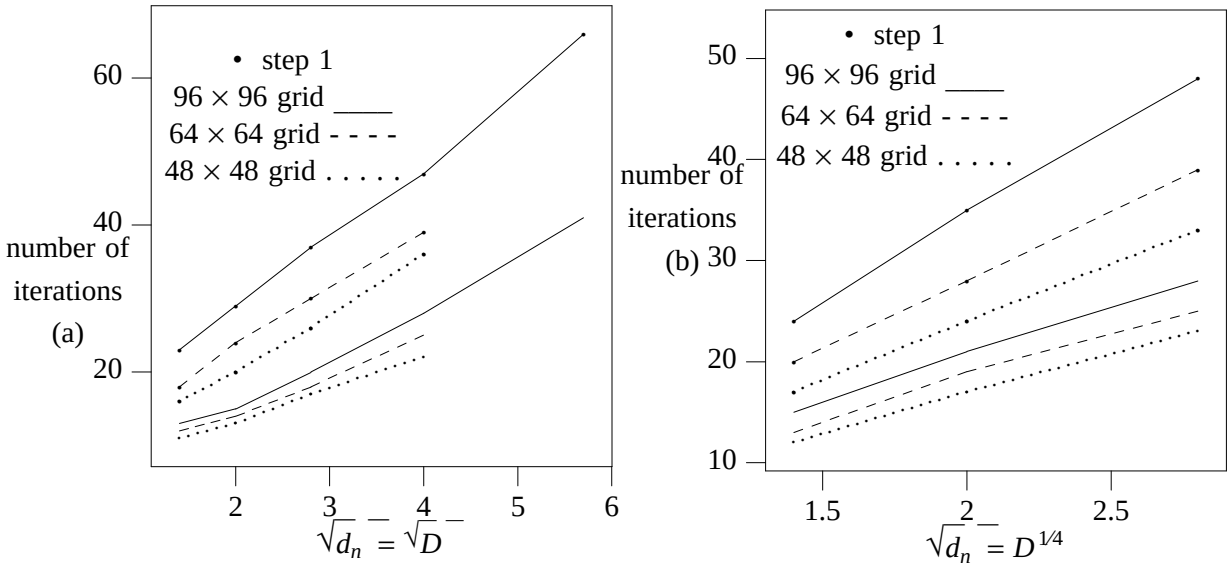

Figure 4.1. Plot of the number of iterations required for convergence of PCG applied to the Schur complement system of the two-step QSC equations corresponding to Problem 1, versus the square root of the number of subdomains d_n in one dimension, for various grid sizes and for the two steps of the QSC method. In (a) $d_m = 1$ and $d_n = D$, while in (b) $d_n = d_m = \sqrt{D}$.

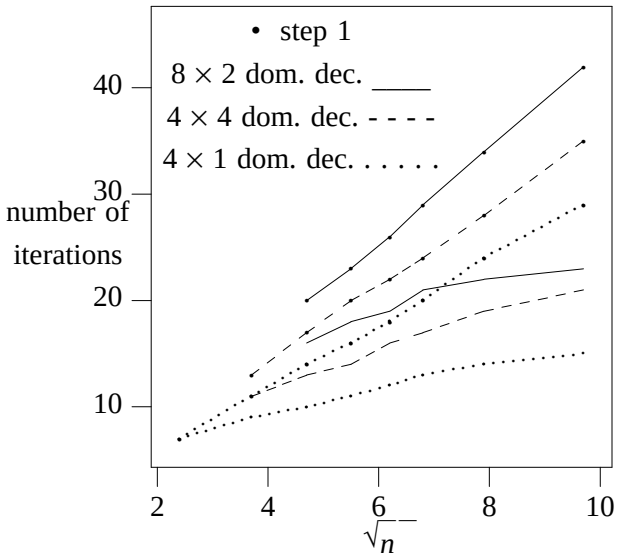

Figure 4.2. Plot of the number of iterations required for convergence of PCG applied to the Schur complement system of the two-step QSC equations corresponding to Problem 1, versus the square root of the grid size n in one dimension, for various domain decompositions and for the two steps of the QSC method.

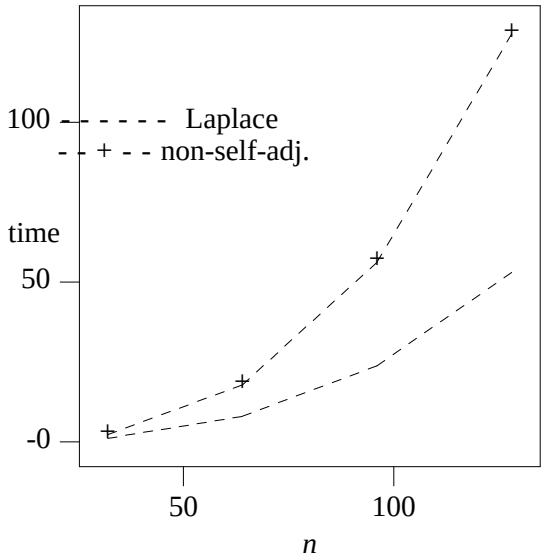

Figure 4.3. Plot of the time that SCM requires to solve the QSC equations corresponding to Problems 1 and 3 versus the grid size n in one dimension. The optimal domain decomposition is chosen. The slopes in a log-log plot of the lines from top to bottom are 2.91 and 2.78.

Table 4.2. The optimal, with respect to total time, domain decomposition for SCM with PCG solution of the capacitance matrix of the two-step QSC equations corresponding to Problems 1 and 3, for several grid sizes. Entries on the second row for the same grid and problem indicate almost equivalent timing results for different domain decompositions.

n	Problem 1 $d_n \times d_m$	Problem 3 $d_n \times d_m$
32	8×1	4×1
64	16×1	16×1
96	16×1	16×1
128	16×1	16×1
	32×1	32×1

Table 5.1. Number of iterations and time in seconds required for the convergence of various iterative methods applied to the two-step QSC equations corresponding to Problems 1 and 3 and natural ordering for several grid sizes and for the two steps of the QSC method. A '−' appears in the case of divergent iterations.

method	grid size	no. of prev. dir.	Problem 1				Problem 3			
			no. of iterations		solut. time	per iter. time	no. of iterations		solut. time	per iter. time †
	n		step 1	step 2	step 1		step 1	step 2	step 1	
Rich-CG	65		133	109	25.78	0.18	−			
	97		200	160	86.72	0.39	−			
	129		269	210	206.02	0.70	−			
SSOR-SI	65		172	77	69.25	0.40	166	80	64.70	0.39
	97		274	105	245.90	0.90	247	108	187.02	0.75
	129		300	127	511.97	1.71	300	130	484.43	1.61
SSOR-CG	65		58	40	23.75	0.41	120	66	46.20	0.38
	97		86	57	78.81	0.92	193	88	146.35	0.75
	129		115	73	202.40	1.76	262	106	425.45	1.62
IF0-CG	65		37	25	14.87	0.37	−			
	97		54	32	45.19	0.78	−			
	129		70	39	111.18	1.51	−			
IF1-CG	65		26	17	12.48	0.41	37	22	17.69	0.43
	97		36	22	35.53	0.87	67	31	65.78	0.92
	129		47	27	86.13	1.66	109	40	199.85	1.76
IF2-CG	65		19	14	10.87	0.43	24	15	13.59	0.46
	97		28	17	32.28	0.94	43	22	47.88	0.98
	129		36	21	76.91	1.79				
IF0m-CG	65		24	16	10.21	0.37	24	15	10.82	0.40
	97		30	18	26.81	0.79	31	18	28.99	0.85
	129		35	21	59.53	1.53	37	20	65.33	1.62
IF1m-CG	65		19	13	9.69	0.41	17	11	9.26	0.44
	97		23	15	24.61	0.88	22	13	24.66	0.93
	129		28	16	55.95	1.69	27	15	56.41	1.78
IF2m-CG	65		16	10	9.77	0.43	14	9	9.19	0.46
	97		20	13	24.98	0.94	18	10	23.93	1.00
	129		23	13	53.99	1.78				
IF1-OM	65	2					37	22	17.70	0.43
	97	2					67	31	65.72	0.92
	129	2					109	40	199.83	1.76
	65	5					29	19	16.08	0.50
	97	5					48	25	54.53	1.05
	129	5					89	33	172.79	1.85
IF0m-OM	65	2					24	15	10.89	0.40
	97	2					31	18	28.91	0.84
	129	2					37	20	65.36	1.62
	65	5					21	14	10.82	0.46
	97	5					27	16	28.88	0.97
	129	5					33	18	67.36	1.88
IF1m-OM	65	2					17	11	9.32	0.44
	97	2					22	13	24.69	0.93
	129	2					27	15	56.43	1.78
	65	5					15	11	9.23	0.49
	97	5					20	12	25.18	1.05
	129	5					26	14	56.71	1.86
MGJ-BCGS	65		13	6	10.71	0.82	14	9	11.43	0.82
	97		11	7	20.79	1.89	13	10	24.37	1.87
	129		10	8	34.16	3.41	12	10	40.77	3.39

† The per iteration time of SSOR-based methods turns out to be smaller for the unsymmetric case than for the symmetric one. This discrepancy is due to the different storage scheme used in the two cases and possibly due to the implementation of the matrix-vector multiplication in NSPCG.

Table 5.2. Number of iterations and time in seconds required for the convergence various iterative methods applied to the two-step QSC equations corresponding to Problems 1 and 3 and a four-colour ordering for several grid sizes and for the two steps of the QSC method. A '−' appears in the case of divergent iterations.

method	grid size	no. of prev. dir.	Problem 1				Problem 3			
			no. of iterations		solut. time	per iter. time	no. of iterations		solut. time	per iter. time
	n		step 1	step 2	step 1		step 1	step 2	step 1	
SSOR-SI	65		188	93	102.13	0.54	228	99	146.46	0.64
	97		283	127	351.86	1.24	300	131	376.65	1.25
	129		300	157	724.07	2.41				
SSOR-CG	65		71	50	38.61	0.54	172	88	109.10	0.63
	97		106	71	132.38	1.25	265	120	328.08	1.24
	129		141	91	346.20	2.45	300	186	839.21	2.79
IF0-CG	65		63	42	25.10	0.38	−			
	97		94	52	78.20	0.80	−			
	129		113	67	194.84	1.67	−			
IF1-CG	65		26	17	14.16	0.45	88	43	43.96	0.47
	97		37	23	41.28	0.96	163	61	170.15	1.01
	129		48	27	106.04	1.97	265	76	565.76	2.09
IF2-CG	65		24	16	15.41	0.48	61	31	34.40	0.50
	97		34	21	43.81	1.03	115	46	132.80	1.08
	129		44	25	111.48	2.10	187	40	428.87	2.19
IF0m-CG	−									
IF1m-CG	65		21	14	12.21	0.45	20	16	12.13	0.48
	97		26	16	31.61	0.98	29	19	35.47	1.02
	129		29	17	70.18	2.00	39	34	92.96	2.08
IF2m-CG	65		20	14	13.88	0.48	19	14	13.77	0.51
	97		25	16	36.58	1.07	27	19	38.83	1.09
	129		29	17	81.62	2.11	34	23	94.69	2.20
IF1-OM	65	2					88	43	43.99	0.47
	97	2					163	61	170.09	1.01
	129	2					265	76	561.62	2.08
	65	5					60	31	34.54	0.54
	97	5					97	37	115.94	1.14
	129	5					126	46	294.41	2.25
IF0m-OM	−									
IF1m-OM	65	2					20	16	12.13	0.48
	97	2					29	19	35.47	1.02
	129	2					39	34	93.02	2.08
	65	5					19	14	12.79	0.54
	97	5					25	17	34.53	1.15
	129	5					29	21	77.20	2.26

Figure 5.1

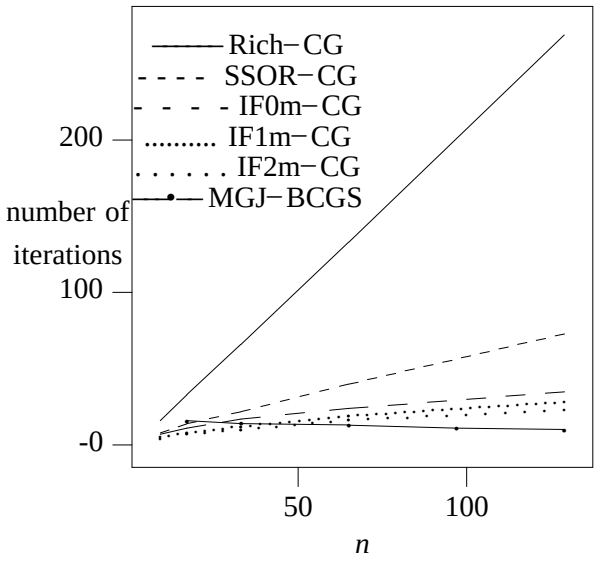

Figure 5.2

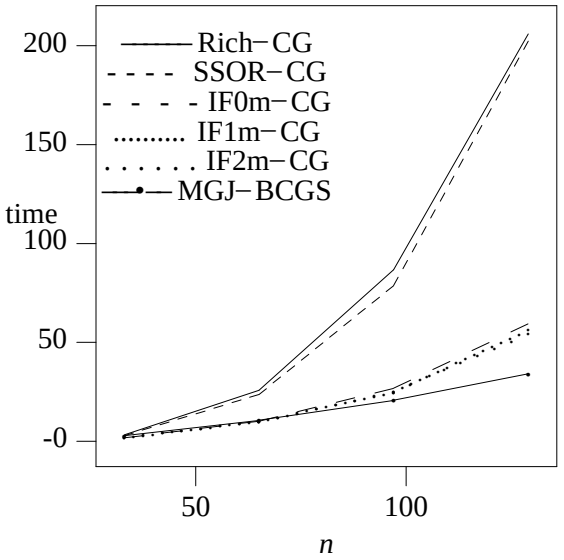

Figure 5.1. Plot of the number of iterations required for convergence of several iterative methods applied to the naturally ordered QSC equations corresponding to Problem 1, versus the grid size n in one dimension.

Figure 5.2. Plot of the time taken by several iterative methods to solve the naturally ordered QSC equations corresponding to Problem 1, versus the grid size n in one dimension. The slopes in a log-log plot are: Rich−CG 3.05; SSOR−CG 3.09; IF0m−CG 2.56; IF1m−CG 2.57; IF2m−CG 2.52; MGJ−BCGS 1.79;

Figure 5.3

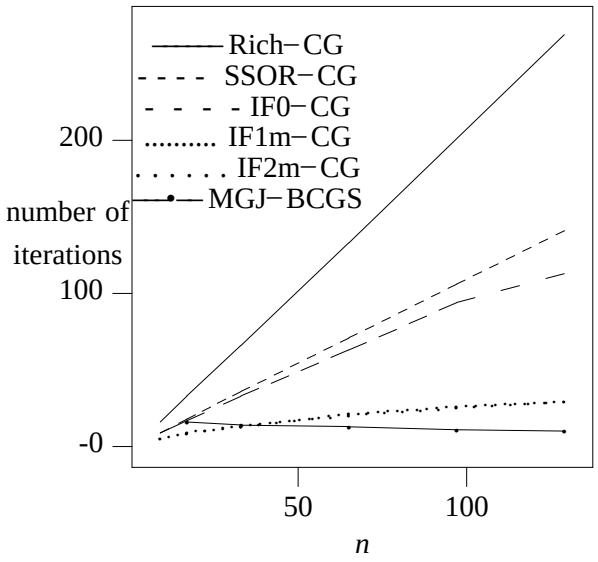

Figure 5.4

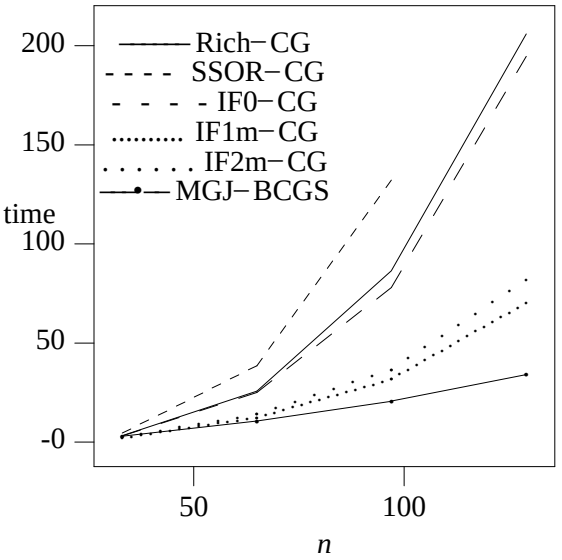

Figure 5.3. Plot of the number of iterations required for convergence of several iterative methods applied to the four-colour ordered QSC equations corresponding to Problem 1, versus the grid size n in one dimension.

Figure 5.4. Plot of the time taken by several iterative methods to solve the four-colour ordered QSC equations corresponding to Problem 1, versus the grid size n in one dimension. The slopes in a log-log plot are: Rich−CG 3.05; SSOR−CG 3.16; IF0m−CG 3.01; IF1m−CG 2.50; IF2m−CG 2.53; MGJ−BCGS 1.79;

Table 6.1. Communication time in milliseconds on P processors for the local exchange of data required to compute a matrix-vector multiplication, when the matrix arises from a 9-point-star stencil, with natural ordering, for various grid sizes n and processor layouts on the Intel Paragon.

P	2	4		8		16		32	14	28	56
n layout	1 × 2	2 × 2	1 × 4	2 × 4	1 × 8	4 × 4	2 × 8	4 × 8	1 × 14	2 × 14	4 × 14
113	0.49	0.77	0.41	0.75	0.54	1.02	0.79	1.01	0.53	0.79	1.02
169	0.53	0.79	0.49	0.78	0.59	1.04	0.84	1.07	0.59	0.83	1.06
225	0.59	0.83	0.56	0.81	0.64	1.04	0.85	1.06	0.67	0.84	1.05
449	0.68	0.84	0.99	1.03	1.15	1.05	1.01	1.12	1.08	0.97	1.12

Table 6.2. Communication time in milliseconds for a global summation of P floating-point numbers distributed in the P processors on the Intel Paragon.

P	2	4	8	14	16	28	32	56
	0.14	0.47	0.72	0.92	1.00	1.21	1.26	1.46

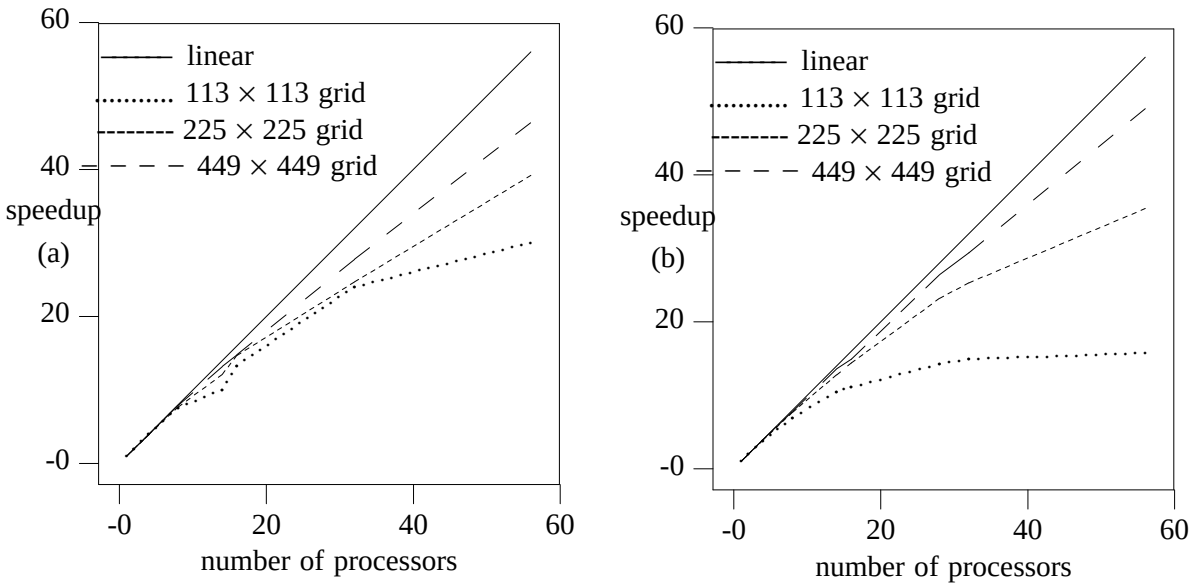

Figure 6.1. Plot of speedup obtained on the Intel Paragon for (a) generating the QSC equations for Problem 1 and (b) solving them (a 9-point-star matrix in natural ordering) by Rich−CG (an iterative method which requires a matrix-vector multiplication and two inner products per iteration), versus the number of processors. The efficiency varies for (a) the discretisation phase from 96% to 83% for the 449×449 grid (87% for 32 processors) and from 98% to 70% for the 225×225 grid (77% for 32 processors), and for (b) the solution phase from 100% to 87% for the 449×449 grid (92% for 32 processors) and from 100% to 63% for the 225×225 grid (79% for 32 processors).

Table 6.3. Communication time in milliseconds on *P* processors for the local exchange of data required to compute the product of the QSC capacitance matrix by a vector, for various grid sizes *n*, domain decompositions and processor layouts on the Intel iPSC/2.

P	2	4	8	16		32		
n layout	2×1	4×1	8×1	8×2	16×1	8×4	16×2	32×1
48	1.28	3.20	3.20	2.56	3.20	2.88	2.56	
64	1.60	3.20	3.20	4.16	3.20	2.88	4.16	3.52
96	1.60	3.20	3.20	4.16	3.84	2.88	4.16	3.52

Table 6.4. Communication time in milliseconds for a global summation of *P* floating-point numbers distributed in the *P* processors on the Intel iPSC/2.

P	2	4	8	16	32
	1	1.5	2.1	2.7	3.3

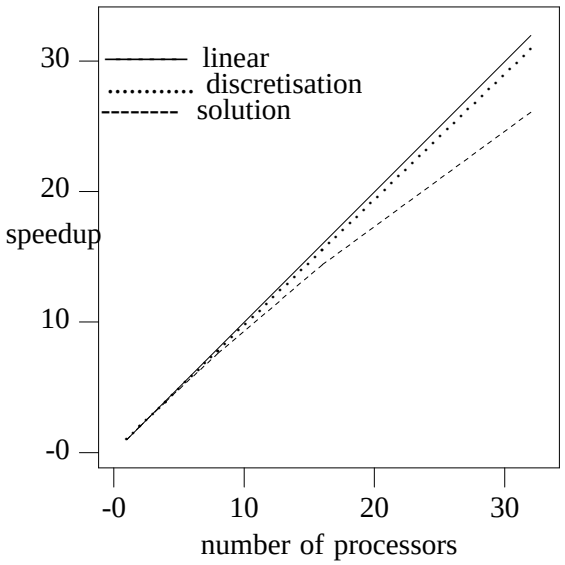

Figure 6.2. Plot of speedup obtained on the Intel iPSC/2 for generating the QSC equations for Problem 1 and solving them by the SCM with PCG solution of the capacitance matrix, versus the number of processors. The grid size is fixed to 94×94 and the domain decomposition to 32×1 subdomains. The efficiency varies from 99% to 96% for the discretisation phase and from 98% to 81% for the solution phase.

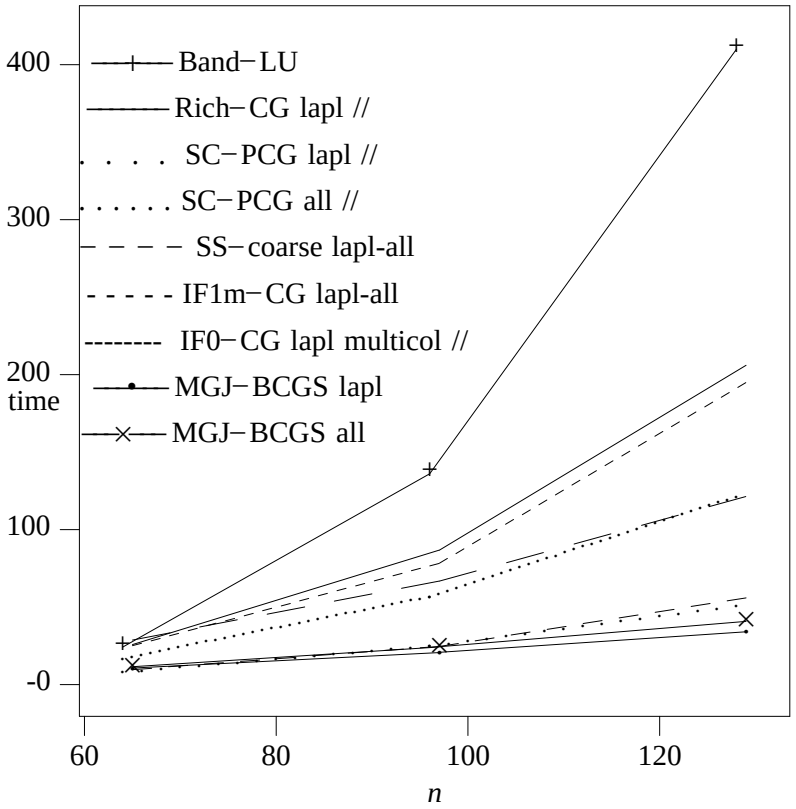

Figure 7.1. Plot of the time taken by several methods applied to the system of QSC equations corresponding to Problems 1 (‘lapl’) and 3 (‘all’) versus the grid size n in one dimension and for the first step of the QSC method. A ‘//’ denotes degree of parallelism increasing linearly with the total size of the problem. The slopes in a log-log plot are: Band−LU 4.09; Rich−CG 3.05; SC−PCG lapl 2.71; SC−PCG all 2.87; SS−coarse 2.11; IF1m−CG lapl-all 2.57; IF0−CG lapl multicol 2.98; MGJ−BCGS lapl 1.79; MGJ−BCGS all 1.86

Problem 1.

$$u_{xx} + u_{yy} = g \quad \text{in} \quad (0,1)\times(0,1)$$
$$u = 0 \quad \text{on} \quad x = 0,\ x = 1,\ y = 0,\ y = 1$$

Problem 2.

$$u_{xx} + u_{yy} = g \quad \text{in} \quad (0,1)\times(0,1)$$
$$u = 0 \quad \text{on} \quad x = 1$$
$$u_n = 0 \quad \text{on} \quad x = 0,\ y = 0,\ y = 1$$

Problem 3.

$$u_{xx} + u_{xy} + u_{yy} + u_x + u_y + u = g \quad \text{in} \quad (0,1)\times(0,1)$$
$$u = 0 \quad \text{on} \quad x = 0,\ x = 1,\ y = 0,\ y = 1$$

In all problems, the function g is chosen so that the exact solution to the problems is $u = x^{\frac{9}{2}}(x-1)^2 y^{\frac{9}{2}}(y-1)^2$.

References

[1] Ahlberg, J. H., and Ito, T., A collocation method for two-point boundary value problems, Math. Comp., 29 (1975), pp. 129–131, 761–776.

[2] Fyfe, D. J., The use of cubic splines in the solution of two-point boundary value problems, Comput. J., 17 (1968), pp. 188–192.

[3] Archer, D. A., Some collocation methods for differential equations, Rice University, Houston, TX, U.S.A., Ph.D. thesis, 1973.

[4] Daniel, J. W. and Swartz, B. K., Extrapolated collocation for two-point boundary value problems using cubic splines, J. Inst. Maths Applics, 16 (1975), pp. 161–174.

[5] Papamichael, N., and Soares, M. J., A class of cubic and quintic spline modified collocation methods for the solution of two-point boundary value problems, Brunel University, Tech. Rep. (1987), 41 pages.

[6] Irodotou-Ellina, M., Optimal spline collocation methods for high degree two-point boundary value problems, Aristotle University of Thessaloniki, Greece, Ph.D. thesis, 1987.

[7] Irodotou-Ellina, M. and Houstis, E. N., An $O(h^6)$ quintic spline collocation method for fourth order two-point boundary value problems, BIT, 28 (1988), pp. 288–301.

[8] Christara, C. C., Spline collocation methods, software and architectures for linear elliptic boundary value problems, Ph.D. thesis, Purdue University, IN, U.S.A., 1988.

[9] Houstis, E. N., Vavalis, E. A. and Rice, J. R., Convergence of an $O(h^4)$ cubic spline collocation method for elliptic partial differential equations, SIAM J. Numer. Anal., 25, 1 (1988), pp. 54−74.

[10] Houstis, E. N., Christara, C. C. and Rice, J. R., Quadratic spline collocation methods for two-point boundary value problems, Internat. J. Numer. Methods Engrg., 26 (1988), pp. 935−952.

[11] Houstis, E. N., Rice, J. R., Christara, C. C. and Vavalis, E. A., Performance of scientific software, Mathematical Aspects of Scientific Software, (J. R. Rice, ed.), Springer Verlag, 1988, pp. 123−156.

[12] Christara, C. C., Quadratic spline collocation methods for elliptic partial differential equations, BIT, 34:1 (1994), pp. 33−61.

[13] Lions, P.-L., On the Schwarz Alternating Method III: A Variant for Nonoverlapping Subdomains, in Domain Decomposition Methods for Partial Differential Equations, T. Chan, R. Glowinski, J. Periaux and O. Widlund eds., SIAM, 1990, pp. 202–223.

[14] Ehrlich, L. W., The numerical Schwarz alternating procedure and SOR, SIAM J. Sci. Statist. Comput., 7, 3 (1986), pp. 989–993.

[15] Rodrigue, G. and Simon, J., A generalisation of the numerical Schwarz algorithm, in Computing Methods in Applied Science and Engineering VI, R. Glowinski and J.-L. Lions eds., Elsevier Science Publ., 1984, pp. 273–283.

[16] Dihn, Q. V., Glowinski, R., and Periaux, J., Solving elliptic problems by domain decomposition methods with applications, in Elliptic Problem Solvers II, Academic Press, 1989, pp. 395–426.

[17] Dryja, M., An additive Schwarz algorithm for two- and three- dimensional finite element elliptic problems, in Domain Decomposition Methods, T. Chan, R. Glowinski, J. Periaux and O. Widlund eds., SIAM, 1989, pp. 168–172.

[18] Christara, C. C. and Smith, B., Convergence of Quadratic Spline Collocation on the Union of Rectangular Subdomains, in preparation.

[19] Gropp, W. D. and Smith, B., Users Manual for KSP: data-structure-neutral codes implementing Krylov space methods, Argonne National Laboratory, Tech. Rep. ANL-93/30 (1993).

[20] Bjorstad, P. E. and Widlund, O. B., Iterative methods for the solution of elliptic problems on regions partitioned into substructures, SIAM J. Numer. Anal., 23, 6 (1986), pp. 1097–1120.

[21] Bramble, J. H., Pasciak, J. E. and Schatz, A. H., The construction of preconditioners for elliptic problems by substructuring I, Math. Comp., 47 (1986), pp. 103–134.

[22] Dryja, M., A finite element − capacitance method for elliptic problems on regions partitioned into subregions, Numer. Math., 44 (1984), pp. 153–168.

[23] Dryja, M. and Proskurowski, W., Iterative methods in subspaces for solving elliptic problems using domain decomposition, University of Southern California, Tech. Rep. CRI-86-10 (1986).

[24] Keyes, D. E. and Gropp, W. D., A comparison of domain decomposition techniques for elliptic partial differential equations and their parallel implementation, SIAM J. Sci. Statist. Comput., 8 (1987), pp. s166–s202.

[25] Mandel, J., On block diagonal and Schur complement preconditioning, Numer. Math., 58 (1990), pp. 79–93.

[26] Christara, C. C., Schur complement preconditioned conjugate gradient methods for spline collocation equations, Proceedings of the 1990 International Conference on Supercomputing (ICS90), June 1990, Amsterdam, the Netherlands, pp. 108−120.

[27] Kightley, J. R. and Forsyth, P. A. (rev. E. F. D'Azevedo), MAT1 Iterative Sparse Matrix Solver: User's Guide, University of Waterloo, 1989.

[28] Oppe, T. C., Joubert, W. D. and Kincaid, D. R., NSPCG User's Guide, Version 1.0, Center for Numerical Analysis, the University of Texas at Austin, CNA-216, April 1988.

[29] Christara, C. C. and Smith, B., Multigrid and multilevel methods for quadratic spline collocation, to appear in BIT.

[30] Christara, C. C., Conjugate gradient methods for spline collocation equations, Proceedings of the fifth Distributed Memory Computing Conference (DMCC5), April 1990, Charleston, SC, U.S.A., pp. 550−558.

[31] Christara, C. C., Parallel Computation of Partial Differential Equations on Distributed Memory Machines, Advances in Computer Methods for Partial Differential Equations - VII R. Vichnevetsky, D. Knight, G. Richter eds. (Proceedings of the 7th IMACS International Conference on Computer Methods for Partial Differential Equations, Rutgers University, New Brunswick, June 22-24, 1992), pp. 142−149.

[32] Duff, I. S. and Meurant, G. A., The effect of ordering on preconditioned conjugate gradients, BIT, 29 (1989), pp. 635–657.

Acknowledgement

The author wishes to thank SINTEF Industrial Mathematics, Trondheim, Norway, for allowing the use of their Intel Paragon.