High Performance Computing of Elliptic Partial Differential Equations with Spline Collocation

Christina C. Christara
Department of Computer Science
University of Toronto
Toronto, Ontario, Canada M5S 1A4
Tel.: (416) 978-7360, Fax : (416) 978-1931
e-mail: ccc@cs.toronto.edu

Abstract

We consider a variety of solvers for the spline collocation equations arising from the discretisation of elliptic Partial Differential Equations. We study the convergence properties of semi-iterative and conjugate gradient acceleration methods applied to the system of spline collocation equations. 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.

We study the parallelisation of some of the above iterative methods and discuss their advantages and disadvantages. 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 BVPs 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]. 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, we briefly describe the formulation of optimal spline collocation methods. In the next three sections, we present the performance of certain solvers on the linear system of spline collocation equations. We show 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, we describe the parallel computation of some of the above methods. 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 14 processors are presented. Section 5 summarises the conclusions from this study. The Appendix includes the test problems used.

2. Spline collocation methods.

Consider the linear elliptic Boundary Value Problem (BVP) described by a Partial Differential Equation (PDE)

$$\mathbf{L}u \equiv au_{xx} + bu_{xy} + cu_{yy} + du_x + eu_y + fu = g \quad \text{in } \Omega \equiv (\omega_1,\omega_2)\times(\omega_3,\omega_4) \tag{1.1}$$

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 Finite Element Methods, 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 deter-

mines the approximation u_Δ to the solution u of (1.1)-(1.2) by forcing it 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 as well. 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 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 this case, the SSM is used as preconditioner. Other preconditioners, e.g. multigrid, can then be added.

3.1. Performance of the SSM.

We present some results from the application of the SSM to the two-step QSC equations. We test accelerated and unaccelerated SSMs. 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. 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].

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'. 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}. All experiments were carried on a Sparcstation1 in double precision. The subdomain solves were carried out by direct band solvers. The results presented here correspond to the first step of the QSC equations. Although we do not include the results from the second step, we can briefly say that the number of iterations required for convergence for the second step of QSC is about half or two thirds of the number of iterations for the first step, assuming that the initial guess for the first step is the zero vector, while for the second step the already computed solution from the first step. In the following, we assume a $d_n \times d_m$ domain

decomposition, i.e., d_n subdomains in the x-direction and d_m in the y-direction. We also denote the total number of subdomains by D, i.e., $D = d_n \cdot d_m$.

We first test the convergence of the SSMs as the overlapping increases. From Figure 3.1, 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 accelerated SSM with the coarse grid correction is almost insensitive to the number of overlapping grid lines.

In Figure 3.2, we plot similar data as in Figure 3.1, only for the accelerated SSMs, and for 4×4 domain decomposition. The insensitivity of the SSM with the coarse grid correction to the number of overlapping grid lines is evident in this plot too.

We then test the convergence of the SSMs as the number of subdomains increases. From Figure 3.3, we can see that the number of iterations grows linearly with the number of subdomains in the case of the unaccelerated method, but less than linearly in the accelerated methods. The method with the coarse grid correction is almost insensitive to the number of subdomains.

In Figure 3.4, we plot similar data as in Figure 3.3, only for the accelerated SSMs, assuming the number of subdomains in each direction is equal to the square root of the total number of subdomains, i.e., a box-wise decomposition. The insensitivity of the SSM with the coarse grid correction to the number of subdomains is evident in this plot too.

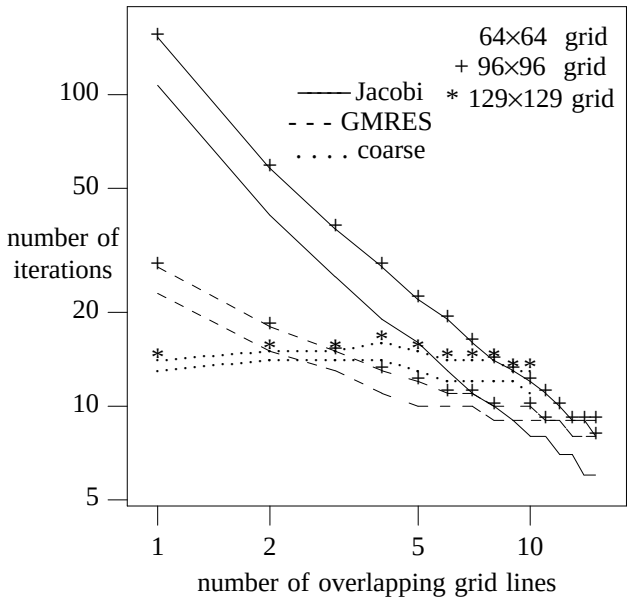

Figure 3.1. Log-log scale plot of the number of iterations required for convergence of three versions of the SSM on the QSC equations for Problem A.1 and the indicated grid sizes, versus the number of overlapping grid lines between neighbouring subdomains. A 2×1 domain decomposition is chosen.

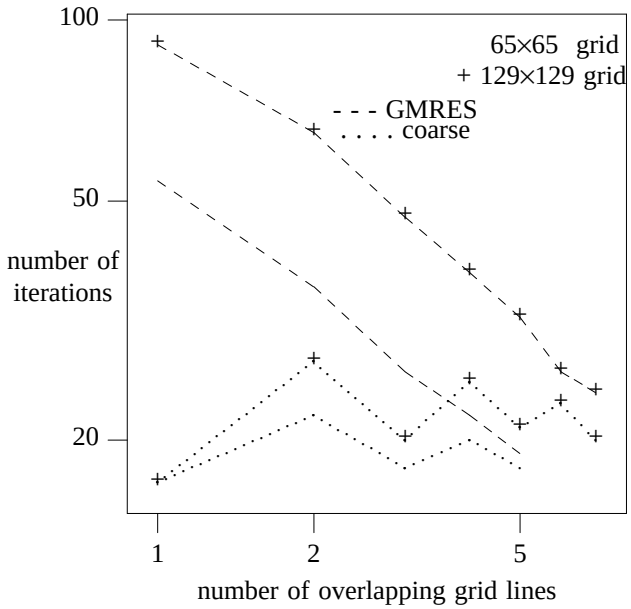

Figure 3.2. Log-log scale plot of the number of iterations required for convergence of two versions of the SSM on the QSC equations for Problem A.1 and the indicated grid sizes, versus the number of overlapping grid lines between neighbouring subdomains. A 4×4 domain decomposition is chosen.

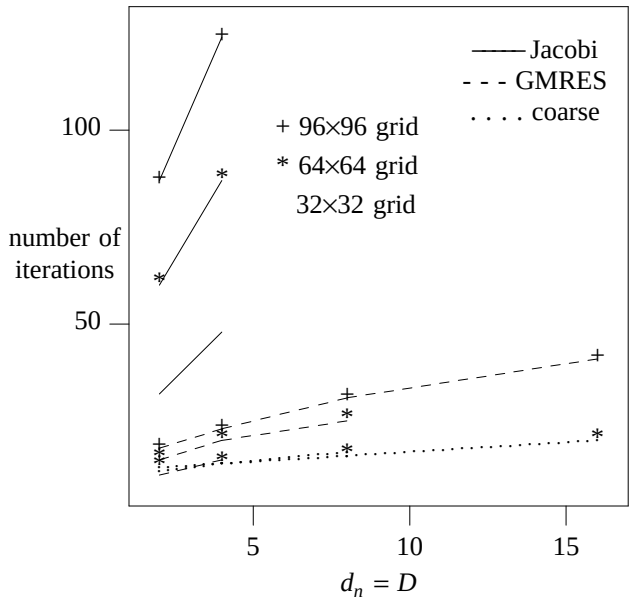

Figure 3.3. Plot of the number of iterations required for convergence of three versions of the SSM on the QSC equations for Problem A.1 and the indicated grid sizes, versus the number of subdomains d_n in one direction assuming a strip-wise decomposition. The number of overlapping grid lines is kept constant along each of the plotted lines; 1 for 'coarse', 2 for 'GMRES' and for 'Jacobi'.

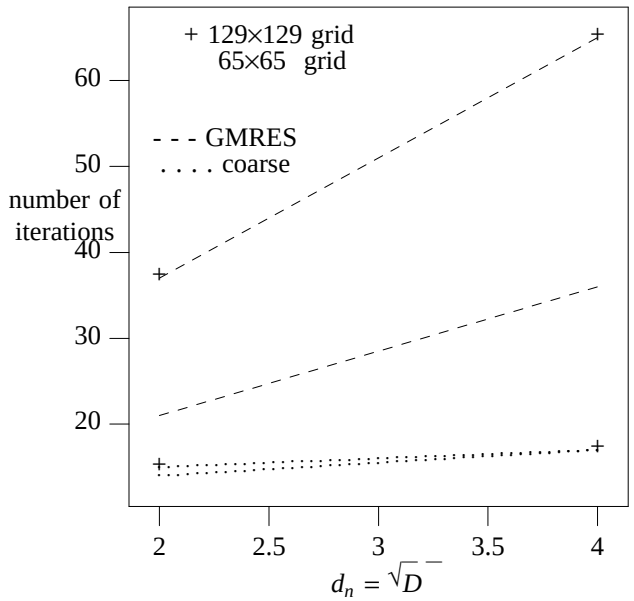

Figure 3.4. Plot of the number of iterations required for convergence of two versions of the SSM on the QSC equations for Problem A.1 and the indicated grid sizes, versus the number of subdomains d_n in one direction assuming a square box-wise decomposition. The number of overlapping grid lines is kept constant along each of the plotted lines; 1 for 'coarse' and 2 for 'GMRES'.

We next test the convergence of the SSMs as the grid size increases. From Figure 3.5, 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.

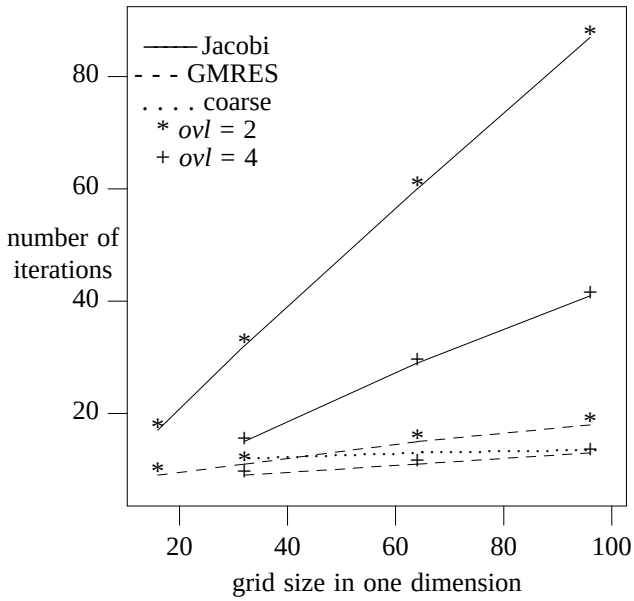

Figure 3.5. Plot of the number of iterations required for convergence of three versions of the SSM on the QSC equations for Problem A.1 and for the indicated number of overlapping grid lines *ovl*, versus the grid size in one direction. A 2×1 domain decomposition is chosen.

In Figure 3.6, we plot similar data as in Figure 3.5, only for the accelerated SSMs, and for a 4×4 domain decomposition. The insensitivity of the SSM with the coarse grid correction to the grid size is evident in this plot too.

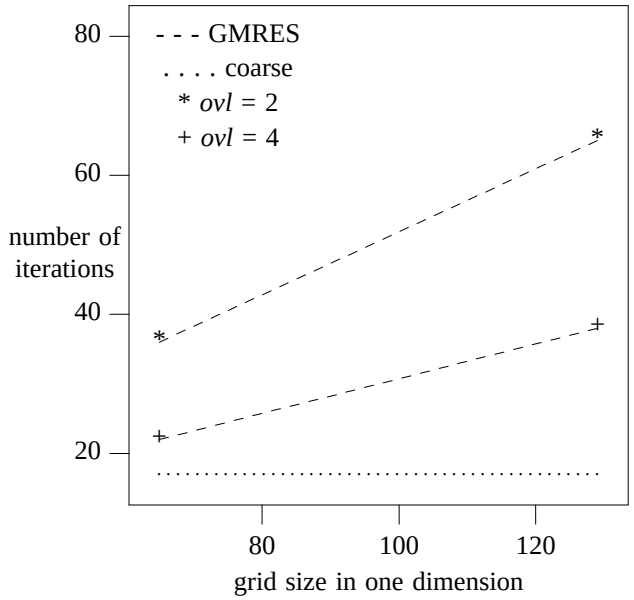

Figure 3.6. Plot of the number of iterations required for convergence of two versions of the SSM on the QSC equations for Problem A.1 and for the indicated number of overlapping grid lines *ovl*, versus the grid size in one direction. A 4×4 domain decomposition is chosen.

On the other hand, as shown in Figure 3.7, 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 direction 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 SSM with the coarse grid correction, while the accelerated SSM without the coarse grid correction converges faster for box-wise decompositions. This is shown in Figure 3.8, where the total number of subdomains *D* remains constant and equal to 16, and the subdomains in the *x* and *y*-directions take the values 1, 2, 4 and 16, 8, 4 respectively.

Finally, in Figure 3.9, we plot the time required to solve the QSC equations corresponding to Problems A.1 and A.2, versus the grid size. Problem A.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.

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. In Tables 3.1 and 3.2, 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 A.1 and A.2 respectively.

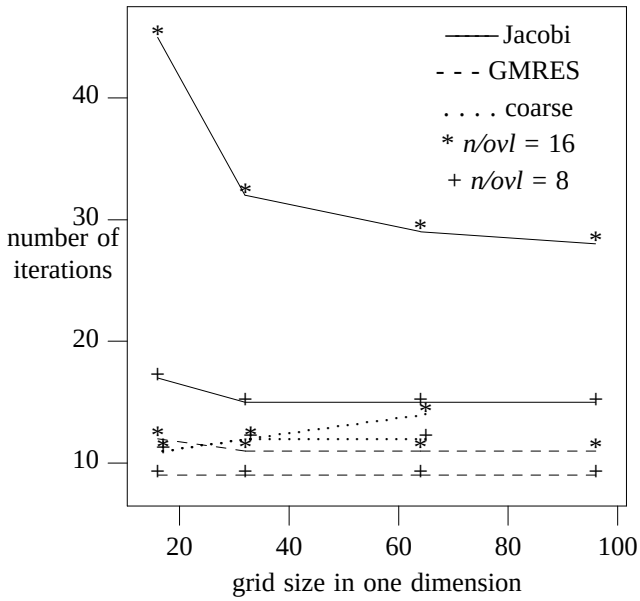

Figure 3.7. Plot of the number of iterations required for convergence of three versions of the SSM on the QSC equations for Problem A.1 when *n/ovl* remains constant versus the grid size *n* in one direction. A 2×1 domain decomposition is chosen.

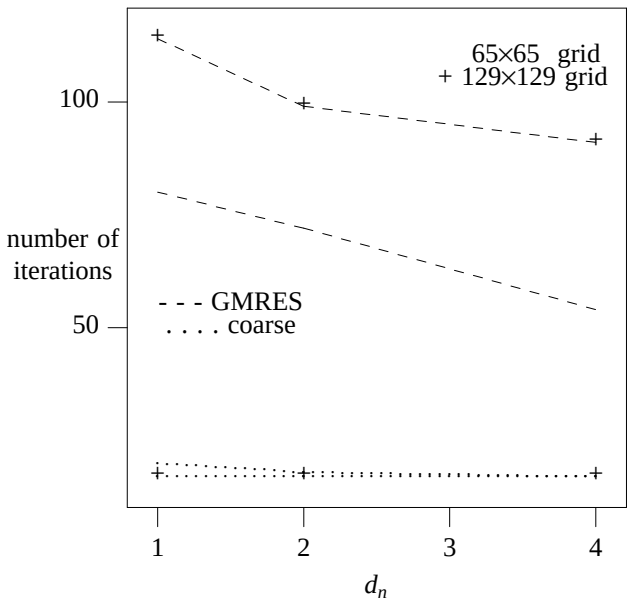

Figure 3.8. Plot of the number of iterations required for convergence of three versions of the SSM on the QSC equations for Problem A.1 when the total number of subdomains remains constant versus the number of subdomains in one direction.

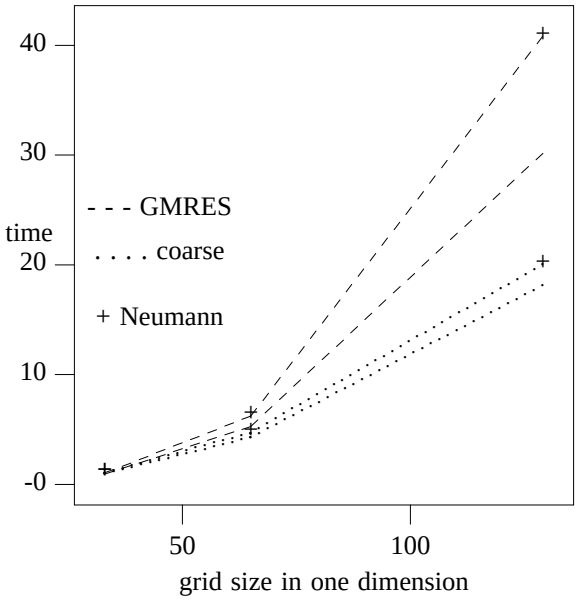

Figure 3.9. Plot of the time that two versions of SSM require to solve the QSC equations corresponding to Problems A.1 and A.2 versus the grid size in one dimension. The optimal combination of domain decomposition and overlapping grid lines is chosen. The slopes in a log-log plot of the lines from top to bottom are 2.68, 2.54, 2.16 and 2.13.

	coarse		GMRES		Jacobi	
grid	dom. dec.	overl	dom. dec.	overl	dom. dec.	overl
33×33	1×2	1	1×2	2	1×2	4
65×65	1×8	1	1×4	2	1×2	6
96×96					1×2	10
129×129	1×16	1	1×8	3		

Table 3.1. The optimal, with respect to total time, combination of number of overlapping grid lines and domain decomposition for several grid sizes for Problem A.1.

	coarse		GMRES	
grid	dom. dec.	overl	dom. dec.	overl
33×33	1×4	1	1×2	2
65×65	1×8	1	1×2	4
			1×4	4
129×129	1×8	1	1×4	6
	1×16	1		

Table 3.2. The optimal, with respect to total time, combination of number of overlapping grid lines and domain decomposition for several grid sizes for Problem A.2. Entries on the second row for the same grid and method indicate equivalent timing results for different domain decompositions.

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 chosen so that there is no interaction between interiors of different subdomains. 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).

$$\begin{bmatrix} A_{00} & A_{01} \\ A_{10} & A_{11} \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \end{bmatrix} \tag{4.1}$$

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.

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

$$C x_0 = w \tag{4.2}$$

for x_0, where $C = A_{00} - A_{01}A_{11}^{-1}A_{10}$ and $w = b_0 - A_{01}A_{11}^{-1}b_1$, and then solving

$$A_{11}x_1 = b_1 - A_{10}x_0 \tag{4.3}$$

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 [19], [20], [21], [22], [23], [24], 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} [25], 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.

4.1. Performance of the SCM.

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 con-

vergence 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. $n_0 = O(n(d_n + d_m))$. It worths noticing that, despite the unsymmetry of the capacitance system of the QSC equations, PCG with preconditioner T converged.

In Figures 4.1, 4.2 and 4.3, 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 A.1. Asymptotically similar results were obtained for the non-self-adjoint problem A.3. The number of iterations for Problem A.3 was about twice of that for Problem A.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 desired precision ε was set to 5×10^{-7}. The computation was carried out on a Sparcstation1 in double precision.

Finally, in Figure 4.4, we plot the time required to solve by SCM the QSC equations corresponding to Problems A.1 and A.3, versus the grid size. Problem A.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.

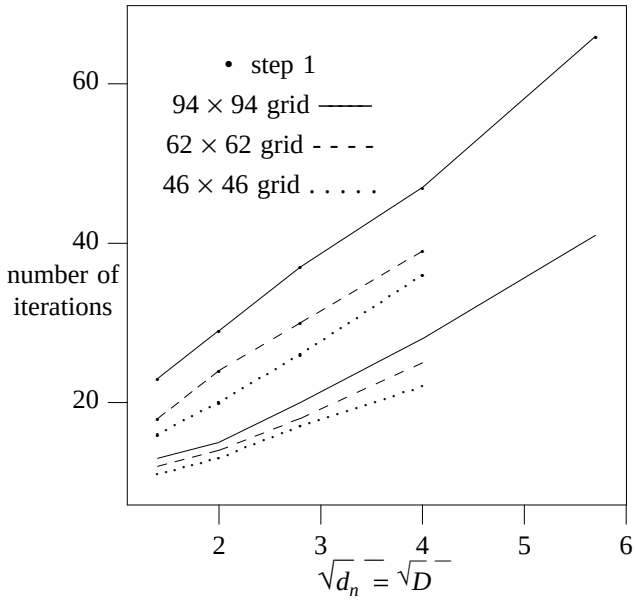

Figure 4.1. Plot of the number of iterations required for convergence of the SCM on the first and second step QSC equations for Problem A.1 and the indicated grid sizes, versus the square root of the number of subdomains d_n in one direction, assuming a strip-wise decomposition.

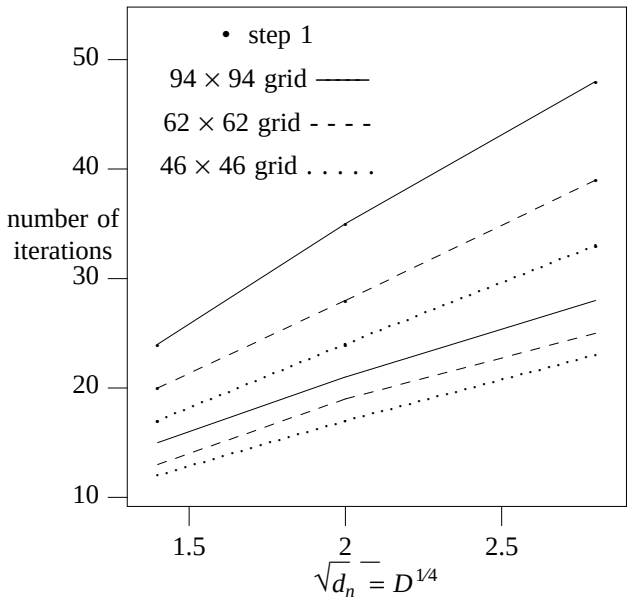

Figure 4.2. Plot of the number of iterations required for convergence of the SCM on the first and second step QSC equations for Problem A.1 and the indicated grid sizes, versus the square root of the number of subdomains d_n in one direction, assuming a square box-wise decomposition.

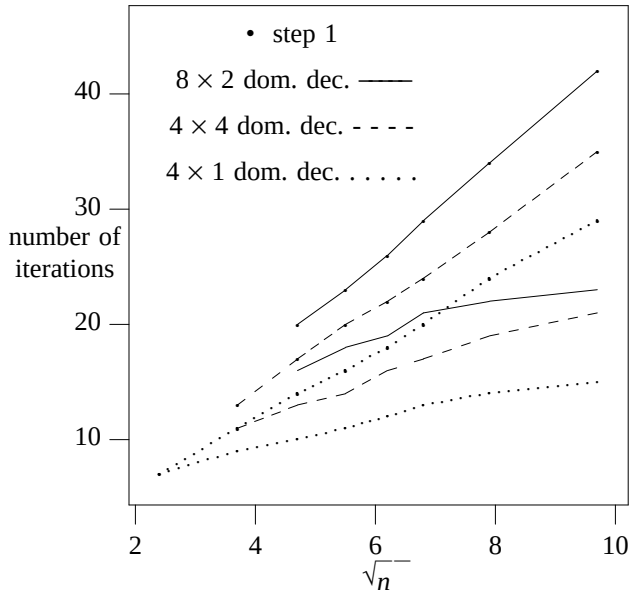

Figure 4.3. Plot of the number of iterations required for convergence of the SCM on the first and second step QSC equations for Problem A.1 and the indicated domain decompositions, versus the square root of the grid size in one dimension.

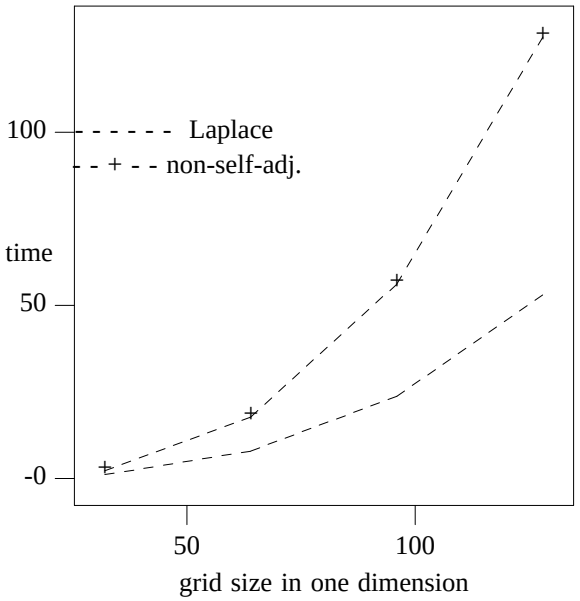

Figure 4.4. Plot of the time that SCM requires to solve the QSC equations corresponding to Problems A.1 and A.3 versus the grid size in one dimension. The optimal domain decomposition is chosen. The slopes in a log-log plot of the lines from top to bottom are 2.91 and 2.78.

grid	Problem A.1 dom. dec.	Problem A.3 dom. dec.
32×32	1×8	1×4
64×64	1×16	1×16
96×96	1×16	1×16
128×128	1×16	1×16
	1×32	1×32

Table 4.1. The optimal, with respect to total time, domain decomposition for several grid sizes for Problems A.1 and A.3. Entries on the second row for the same grid and problem indicate equivalent timing results for different domain decompositions.

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 direction as one quarter of the grid lines in that direction. 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 4.1, we present the best, with respect to total time, domain decomposition for several grid sizes for Problems A.1 and A.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 spline collocation 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.

In Figure 5.1, we plot the number of iterations required for the convergence of some of the above iterative methods applied to the system of two-step QSC equations arising from the discretisation of Problem A.1, versus the grid size n in one dimension. The slopes of the graphs for the IF-based methods are strictly less than 1, while for all other methods they are 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. From experiments on other PDE problems, we concluded that most iterative methods tested are quite sensitive to the PDE operator. Methods for symmetric systems require in general more than double the number of iterations when applied to unsymmetric systems, if they converge. Exceptions are methods with a coarse grid preconditioner, like MGJ-BCGS, and IF-based methods with modification.

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 superoptimal, 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.

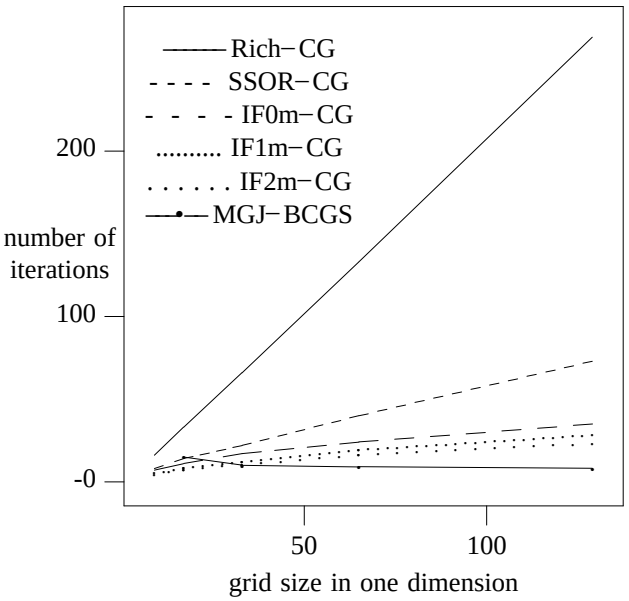

Figure 5.1. Plot of the number of iterations required for convergence of several iterative methods applied to the QSC equations for Problem A.1, versus the grid size in one dimension.

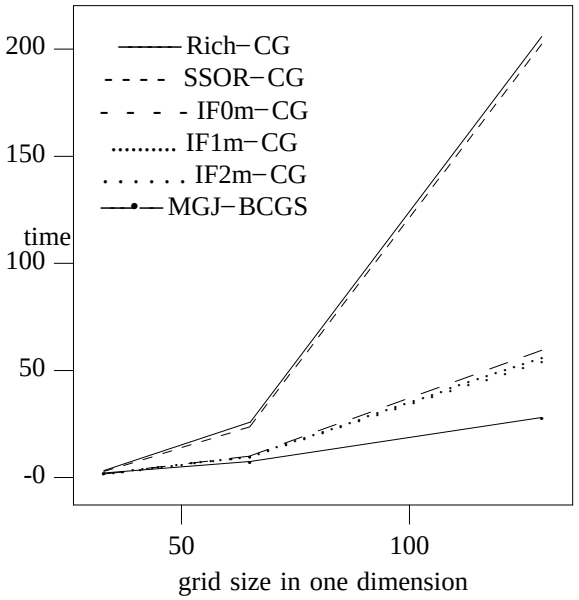

Figure 5.2. Plot of the time that several iterative methods require to solve the QSC equations corresponding to Problem A.1, versus the grid size in one dimension. The slopes in a log-log plot are: Rich−CG 3.05; SSOR−CG 3.1; IF0m−CG 2.57; IF1m−CG 2.6; IF2m−CG 2.54; MGJ−BCGS 1.88;

In all experiments, natural ordering of equations and unknowns is assumed. The WATSIT [26] 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 [27] 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 [28]. We tested both steps of the QSC method, but the results presented here are from the first step. In most cases, the second step requires about half the number of iterations of the first step. 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. The computation was carried out on a Sparcstation1 in double precision.

In [29] 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 [28] 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.

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. 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 belonging to one (or more) subdomain(s) are assigned to one processor. Thus, the layout of processors follows the subdomain layout, which 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 requires some local communication for the summation of values corresponding to nearby points, in order to compute the appropriate integrals. The two-step QSC method, though, requires some local communication in the second step, when computing the effect of the perturbation operators $\mathbb{P}_{\mathbb{L}}$ and $\mathbb{P}_{\mathbb{B}}$.

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) a matrix-vector product needs to be computed, c) an inner product needs to be computed, and d) 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) and b) is local and along each dimension of the layout of processors/subdomains. The amount of data to be communicated might depend on the amount of overlapping for a) and on the width of the discretisation stencil for b). Global communication is required for c) and d), but the amount of data to be communicated between any two processors is small and fixed (one floating point number or logical flag).

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 [25].

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 Subsection 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, like 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. 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. The amount of data to be communicated between any two processors is fixed (one floating point number or logical flag). The matrix-vector product requires local communication. The amount of data to be communicated depends on the width of the discretisation stencil. Forward and back substitutions have degree of parallelism bounded by the bandwidth of the matrix 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. This often affects the convergence of iterative methods in a negative way [30], [31]. The coarse grid preconditioning, on the other hand, has local communication requirements when the extension and restriction operators are applied, but requires the solution of the coarse grid problem, hence some global communication or duplicate computation.

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

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 A.1 for various grid sizes on the Paragon. Each iteration includes 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. The layout of processors is one-dimensional. As shown in Figure 6.1, the speedup on the Paragon grows almost linearly with the number of processors, and increases with the size of the problem.

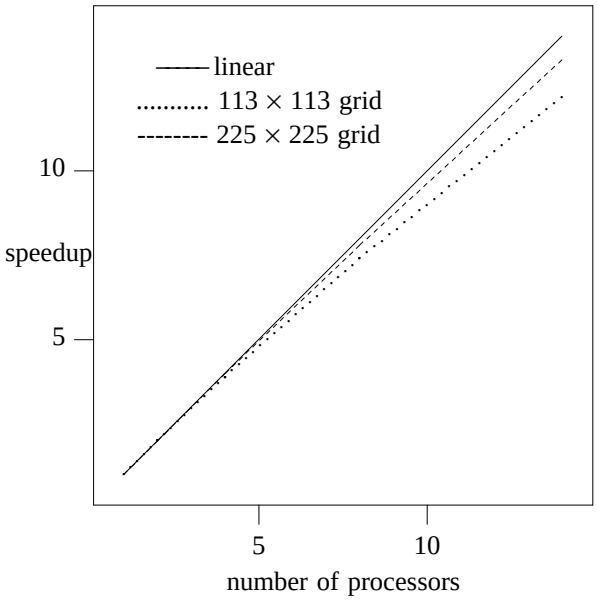

Figure 6.1. Plot of the speedup of Rich-CG applied to the QSC equations for Problem A.1 for various grid sizes on the Paragon, versus the number of processors. The corresponding efficiency for the lines plotted is 87% − 99% for the 113×113 grid and 95% − 100% for the 225×225 grid.

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 A.1 on the iPSC/2. Each iteration includes 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 we used our own software which implements a logarithmic with respect to the number of processors fan-in/out. We measure the speedup for generating the discretisation matrix and the right side vector and the speedup for the solution procedure.

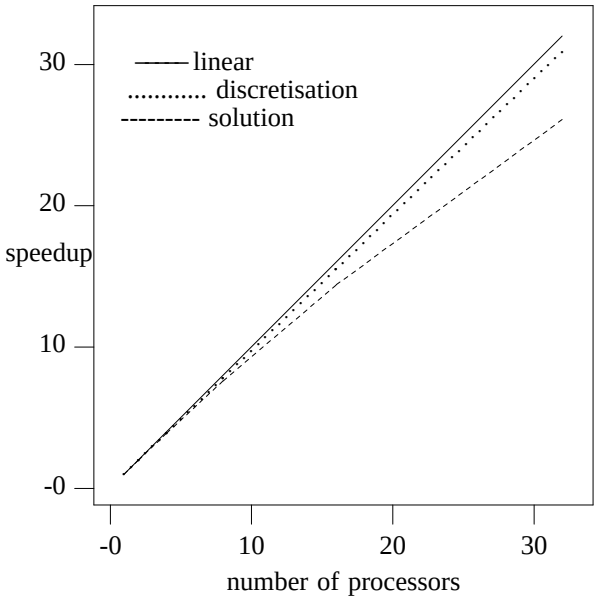

Figure 6.2. Plot of the speedup of QSC discretisation and of SCM applied to the QSC equations for Problem A.1 on the iPSC/2, versus the number of processors. The domain decomposition is fixed to 32×1 and the grid size to 94×94. The corresponding efficiency for the lines plotted is 96% − 99% for the discretisation and 81% − 98% for the solution.

7. 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. Among the solvers tested, the multigrid method is the fastest, but the coarse grid solution has limited parallelism. So does the SSM method with coarse grid preconditioning, which is the second fastest solver. The SSM method with GMRES acceleration and without coarse preconditioning is the next fastest. GMRES has limited parallelism, though, due to solving a least squares problem. Other competitive solvers include IF preconditioning, but the natural ordering methods also have limited parallelism, while the multicoloured versions do not converge as fast as the natural ordering versions. The SCMs offer a reasonable compromise between fast serial solvers and highly parallelisable ones.

Appendix

We include the description of the BVPs used for testing the methods considered.

Problem A.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 A.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 A.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] Bjorstand, 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.

[20] 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.

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

[22] 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).

[23] 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.

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

[25] 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.

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

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

[27] 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.

[28] Christara, C. C. and Smith, B., Multilevel methods for quadratic spline collocation, in preparation.

[29] 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.

[30] Christara, C. C., Multicolour orderings and iterative methods for elliptic equations, Proceedings of the Supercomputing Symposium '91 (SS '91), June 1991, Fredericton, NB, Canada, pp. 221−230.

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