A SCHWARZ SPLITTING VARIANT OF CUBIC SPLINE COLLOCATION METHODS FOR ELLIPTIC PDEs

E.N. Houstis, J.R. Rice* and E.A. Vavalis**
Department of Computer Science
Purdue University
West Lafayette, IN 47907

ABSTRACT

We consider the formulation of the Schwarz alternating method for a new class of elliptic cubic spline collocation discretization schemes. The convergence of the method is studied using Jacobi and Gauss-Seidel iterative methods for implementing the interaction among subdomains. The Schwarz Cubic Spline Collocation (SCSC) method is formulated for hypercube architectures and implemented on the NCUBE (128 processors) machine. The performance and convergence of the hypercube SCSC algorithm is studied with respect to domain partition and subdomain overlapping area. The numerical results indicate that the partition and mapping of the SCSC on the NCUBE is almost optimal while the speedup obtained is similar to other domain decomposition techniques.

1. INTRODUCTION

In this paper we consider the parallelization of a new cubic spline collocation scheme [Hous 88a] for solving elliptic Partial Differential Equations (PDEs) following the basic idea of the Schwarz alternating method (SAM). Recently a large number of papers ([Bjor 86], [Ehrl 86], [Keye 87], [Meie 87], [Rodr 84,86], [Tang 87]), analyze the convergence and complexity of the method while conclude that SAM is a powerful alternative for solving elliptic PDEs in parallel architectures.

It appears that there is limited information about the performance of the method on real parallel machines. The objective of this study is to identify the dominant parameters of SAM and study its performance with respect to various parameter values. Specifically in section 2 we formulate SAM for the cubic spline collocation equations corresponding to a model problem. Further we study its convergence characteristics when the interaction among subdomains is implemented according to Jacobi and Gauss-Seidel iterative schemes. In section 3 we present a hypercube implementation of SCSC and analyze its complexity. Finally in section 4 we present the performance of the method measured by the execution time and speedups obtained as a function of domain partition and subdomain overlapping area. This results almost optimal mapping of the computation (SCSC) on the NCUBE machine and similar speedups as other domain decomposition methods [Keye 87].

2. SCHWARZ CUBIC SPLINE COLLOCATION.

2.1. The Cubic Spline Collocation Method.

In this section we give a brief description of the Cubic Spline Collocation method. We are interesting in approximating the solution $u(x,y)$ of the elliptic linear partial differential equation

This research supported in part by NSF grants CCR-8704826, 500-13981704, AFOSR grant 84-0385 and ESPRIT project 1588.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specfic permission.

© ACM 1988 0-89791-273-X/88/0007/1746 $1.50

$$Lu \equiv \alpha D_x^2 u + \beta D_x D_y u + \gamma D_y^2 u + \delta D_x u + \varepsilon D_y u + \zeta u = -f \quad (2.1)$$
$$\text{in } \Omega = [a,b] \times [c,d]$$

subject to homogeneous Dirichlet or Neumann boundary conditions

$$Bu \equiv 0 \text{ on } \partial\Omega \equiv \text{ boundary of } \Omega \quad (2.2)$$

where Bu is u, $D_x u$ or $D_y u$.

The Cubic Spline Collocation method introduced in [Hous 88a] involves the determination of the bicubic spline piecewise polynomial $u_\Delta(x,y)$, over a uniform partition $\Delta \equiv \{x_k = a + kh_x;\ k = 0\,N\} \times \{y_l = c + lh_y;\ l = 0 \text{ to } M\}$ of Ω, that satisfies exactly the boundary conditions and an operator equation $L'u_\Delta = -f$ at the interior grid points Ω_I of Δ, and additional spline end conditions. Here L' is a high order perturbation of L and can be described, at the interior notes Ω_I by stencil S.1. For later use we denote by Ω_B the boundary nodes of Δ except the corner points Ω_C.

$$\begin{array}{ccc} & \frac{\gamma}{12} D_y^2 S(x_i, y_{j+1}) & \\ \frac{\alpha}{12} D_x^2 S(x_{i-1}, y_j) & \begin{array}{c}\frac{10}{12}[\alpha D_x^2 S(x_i,y_j) + \gamma D_y^2 S(x_i,y_j)] \\ +\delta D_x S(x_i,y_j) + \epsilon D_y S(x_i,y_j) \\ +\beta D_x D_y S(x_i,y_j) + \zeta S(x_i,y_j)\end{array} & \frac{\alpha}{12} D_x^2 S(x_{i+1}, y_j) \\ & \frac{\gamma}{12} D_y^2 S(x_i, y_{j-1}) & \end{array}$$

Stencil S.1. Definition of L'S in terms of the cubic spline interpolant S and its derivatives at Ω_I grid points.

The convergence and the applicability of the method are discussed in [Hous 88]. It has been shown that the method can be integrated successfully with iterative methods and runs more efficiently than collocation methods based on Hermite cubics and Galerkin methods based on cubic splines. Three implementations of the method exist in ELLPACK [Rice 85] for various forms of the equation (2.1) and mixed Dirichlet or Neumann boundary conditions on parts of the boundary. In this section we consider the *interior* formulation of the Cubic Spline Collocation method for the Laplace equation on the unit square. A detailed analysis of the Schwarz Cubic spline Collocation for general elliptic operators and for the *one* and *two step* formulations is given in [Hous 88b].

In order to determine the corresponding discrete equations we adopt the following representation of u_Δ

$$u_\Delta(x,y) = \sum_{i=0}^{N} \sum_{j=0}^{M} U_{i,j}\, B_i(x) B_j(y) \quad (2.3)$$

where the B_i' s are the basis functions for the one-dimensional cubic splines space that satisfy exactly the boundary conditions. We determine u_Δ such that it satisfies

(i) the **interior** collocation equations in Ω_I :

$$\left[L'u_\Delta - (-f) \right]_{(x_i,y_j)} = 0$$

where L' is defined by Stencil S.1 with $\beta \equiv \delta \equiv \varepsilon \equiv 0$ and $\alpha \equiv \gamma \equiv 1$,

(ii) the **boundary** collocation equations in Ω_B :

$$D_x^2 u_\Delta = -f_{i,j} \text{ or } D_y^2 u_\Delta = -f_{i,j},$$

(iii) and the **corner** collocation equations in Ω_C :

$$D_x^2 D_y^2 u_\Delta = -D_x^2 f_{i,j}.$$

The boundary and corner collocation equations can be explicitly solved to obtain the boundary unknowns $U_{i,j}$, $i = 0,N$ and $j = 0,1,\ldots,M$ and $U_{i,j}$, $i = 1,2,\ldots,N-1$ and $j = 0,M$. After the elimination of the above unknowns the interior equations can be written in a matrix form as

$$A\,\mathbf{U} = \mathbf{b} \quad (2.4)$$

where A is a quintdiagonal block matrix of the following form:

$$\begin{bmatrix} D_1 & E_1 & F_1 & & & & & & & \\ C_2 & D_2 & E_2 & F_2 & & & & & & \\ \cdot & \cdot & \cdot & \cdot & \cdot & & & & & \\ & B_k & C_k & D_k & E_k & F_k & & & & \\ & & B_{k+1} & C_{k+1} & D_{k+1} & E_{k+1} & F_{k+1} & & & \\ & & & & \cdot & \cdot & \cdot & \cdot & \cdot & \\ & & & & B_{l-1} & C_{l-1} & D_{l-1} & E_{l-1} & F_{l-1} & \\ & & & & & B_l & C_l & D_l & E_l & F_l \\ & & & & & & \cdot & \cdot & \cdot & \cdot \\ & & & & & & B_{N-2} & C_{N-2} & D_{N-2} & E_{N-2} \\ & & & & & & & B_{N-1} & C_{N-1} & D_{N-1} \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ \cdot \\ U_k \\ U_{k+1} \\ \cdot \\ U_{l-1} \\ U_l \\ \cdot \\ U_{N-2} \\ U_{N-1} \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \cdot \\ b_k \\ b_{k+1} \\ \cdot \\ b_{l-1} \\ b_l \\ \cdot \\ b_{N-2} \\ b_{N-1} \end{bmatrix}$$

Notice that all blocks in A are $(M-1)\times(M-1)$ matrices where $C_{N-1} = E_1 = quindiag(1,16,14,16,1)$ with $C_{N-1}^{1,1} = C_{N-1}^{M-1,M-1} = 13$ $D_{N-1} = D_1 = quindiag(4,13,-148,13,4)$ with $D_{N-1}^{1,1} = D_{N-1}^{M-1,M-1} = -152$ and the rest of the block matrices B_i, C_i, D_i, E_i and F_i are $B_i = F_i = tridiag(1,4,1)$, $C_i = E_i = quindiag(1,16,14,16,1)$ and

$D_i = quindiag(4,14,-144,14,4)$. Assuming that the equations or collocation points have been ordered with respect to vertical mesh lines from left to right, the solution vector consists of the components $\mathbf{U}_i = \left[U_{i,1}, U_{i,2}, \ldots, U_{i,M-2}, U_{i,M-1} \right]^T$. The right hand side vector involves the values of the function f and the effect of the elimination of the boundary conditions.

2.2. The Schwarz Cubic Spline Collocation Method.

For simplicity of the presentation we decompose the domain Ω into two overlapping subdomains Ω_1 and Ω_2 as shown in Figure 2.1. Also without loss of generality we assume that the *new* boundary lines *AB* and *CD* are the l^{th} and the k^{th} grid lines in the x direction respectively.

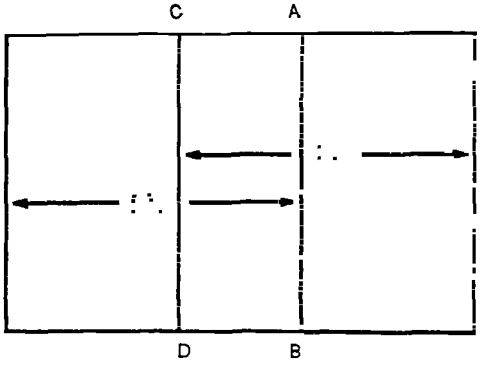

Figure 2.1. Two vertical overlapping subdomains.

According to Schwarz splitting scheme the original problem (2.1), (2.2) can be formulated as two coupled problems,

$$Lu_1 = -f \quad in \quad \Omega_1 \qquad Bu_1 = g_1 \quad on \quad \partial\Omega_1, \qquad g_1 = \begin{cases} u_2 & on\ line\ AB \\ 0 & on\ the\ other\ boundary\ lines\ of\ \partial\Omega_1 \end{cases} \tag{2.5}$$

and

$$Lu_2 = -f \quad in \quad \Omega_2 \qquad Bu_2 = g_2 \quad on \quad \partial\Omega_2, \qquad g_2 = \begin{cases} u_1 & on\ line\ CD \\ 0 & on\ the\ other\ boundary\ lines\ of\ \partial\Omega_2 \end{cases} \tag{2.6}$$

The Schwarz Alternating Algorithm starts by giving u_2 an initial guess on the boundary line *AB*. Then (2.5) is solved and its solution is used to determine g_2 on *CD*. Finally (2.5) is solved and its solution is used to determine g_1. The entire process is repeated until appropriate stopping criterion is satisfied.

The Schwarz Cubic Spline Collocation (SCSC) method consists of discretizing the subproblems (2.5) and (2.6) using the Cubic Spline Collocation method described in section 2.1 while the interaction between the subproblems is computed by an iterative method. For both the convergence analysis and the implementation of the method, it is more convenient to formulate the SCSC method at the discretization level.

The above described Schwarz Alternating Algorithm can be viewed (see [Meie 86], [Rodr 84], [Tang 87]) as a 2x2 Block Gauss-Seidel iteration of the following matrix equation

$$\tilde{A}\tilde{\mathbf{U}} = \begin{bmatrix} A_1 & P \\ K & A_2 \end{bmatrix} \begin{bmatrix} \mathbf{U}^1 \\ \mathbf{U}^2 \end{bmatrix} = \begin{bmatrix} \mathbf{b}^1 \\ \mathbf{b}^2 \end{bmatrix}, \tag{2.7}$$

where

$$\mathbf{U}^1 = \left[\mathbf{U}_1, \mathbf{U}_2, \ldots, \mathbf{U}_l\right]^T, \quad \mathbf{U}^2 = \left[\mathbf{U}_k, \mathbf{U}_{k+1}, \ldots, \mathbf{U}_{N-1}\right]^T,$$
$$\mathbf{b}^1 = \left[\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_l\right]^T, \quad \mathbf{b}^2 = \left[\mathbf{b}_k, \mathbf{b}_{k+1}, \ldots, \mathbf{b}_{N-1}\right]^T,$$

$$A_1 \equiv \begin{bmatrix} D_1 & E_1 & F_1 & & & \\ C_2 & D_2 & E_2 & F_2 & & \\ . & . & . & . & . & \\ & B_{l-1} & C_{l-1} & D_{l-1} & E_{l-1} & \\ & & B_l & C_l & D_l & \end{bmatrix}, \quad A_2 \equiv \begin{bmatrix} D_k & E_k & F_k & & & \\ C_{k+1} & D_{k+1} & E_{k+1} & F_{k+1} & & \\ . & . & . & . & . & \\ & B_{N-2} & C_{N-2} & D_{N-2} & E_{N-2} & \\ & & B_{N-1} & C_{N-1} & D_{N-1} & \end{bmatrix},$$

$$P \equiv \begin{bmatrix} & & & & \\ & & 0 & & \\ 0 & & & & 0 \\ & & & & \\ & \bar{F}_{l-1} & \bar{0} & & \\ & E_l & F_l & & \end{bmatrix} \quad \text{and} \quad K \equiv \begin{bmatrix} & B_k & C_k & & \\ & 0 & B_{k+1} & & \\ 0 & - & - & & \\ & & & & 0 \\ & & 0 & & \\ & & & & \end{bmatrix}.$$

The general case of decomposing the domain in both horizontal and vertical directions is presented in detail in [Hous 88b].

2.3. Iterative methods for Subproblem Interaction.

Recently, it has been noticed that the choice of Gauss-Seidel as the iterative scheme for handling the interaction between the subproblems might not be the best. Its slow convergence forced many researchers, [Bjor 86], [Ehrl 86], [Keye 87], [Meie 86], [Tang 87], to apply, successfully, SOR and preconditioned conjugate gradient acceleration techniques. Theoretical results on the convergence rates and the choice of the optimum acceleration parameters are also available, but the computational cost to estimate these parameters is not

fully analyzed. Also the Jacobi method has been considered in [Rodr 84], [Tang 87]. Its advantage is its inherent parallelism.

In this paper we will consider only the Jacobi and Gauss-Seidel iterative schemes. Their convergence properties are studied in [Hous 88b] and summarized here in the following theorem.

Theorem 2.1 *The Jacobi and Gauss-Seidel Schwarz Cubic Spline Collocation Methods converge from any initial approximation. Furthermore the spectral radius of the Gauss-Seidel iteration matrix is half the spectral radius of the Jacobi iteration matrix.*

The SCSC method is dominated by a number of parameters, ranging from non overlapping strips to overlapping subdomains with arbitrary area. In this paper we consider only partitions of strips or boxes of equal area. This restriction greatly simplifies the formulation and analysis of the method and it is not an unreasonable constraint for smooth elliptic problems on rectangular domains.

Before implementing SCSC in parallel, it is essential to examine the dominance of parameters like the number of subdomains, the number of iterations, the size of the overlapping area and the type of domain partition on the convergence and complexity of SCSC on sequential machines. The SCSC algorithm can be described as

STEP 0 discretize

STEP 1 for id = 1,2,... number_of_subdomains do:
|_factor matrix(id)

STEP 2 for iter = 1,2,... until convergence do:
| for id = 1, number_of_subdomains do:
| | back_solve matrix(id)
|__|__update rhs(id)

At Step 0 we apply the one step interior Cubic Spline Collocation to form the coefficient matrix and the right hand side on the whole domain. At Step 1 we use band Gauss elimination to factor the coefficient matrices of all subdomains. At Step 2 we back solve these matrices, update the right hand sides and iterate until convergence.

It is clear that the number of iterations is not such a dominant performance factor since most of the time is spend at Step 0 to factor the matrices. This suggests that Jacobi's slow convergence might not be a serious disadvantage. Specifically, the application of Cubic Spline Collocation on an $n \times m$ interior grid points leads to a coefficient of size $nm \times nm$ and bandwidth $w = 2m + 1$. Assuming that the ordering of the equations is done from bottom to top, left to right, the number of operations to factor this matrix is $nmw(w+1)$ while the cost to back solve it is $nm(2w+1)$. It is important to notice that when we decompose the domain in horizontal strips we decrease m (the number of grid lines in y direction), and the bandwidth w while n (the number of grid lines in x direction) is kept constant. The opposite is true in case of vertical strip decomposition. In Figure 2.2 we give the total (steps 0, 1 and 2) VAX 8600 time in seconds to solve the self adjoint equation $D_x(e^{xy}D_x u) + D_y(e^{-xy}D_y u) - u/(1+x+y) = f(x,y)$ with Dirichlet boundary conditions on the unit square as a function of the number of non overlapping vertical or horizontal strips for several grids. The convergence tolerance for each grid size is selected in such a way that it guarantees the fourth order convergence. The data indicate that the non overlapping domain decomposition effects the complexity or performance of the sequential SCSC.

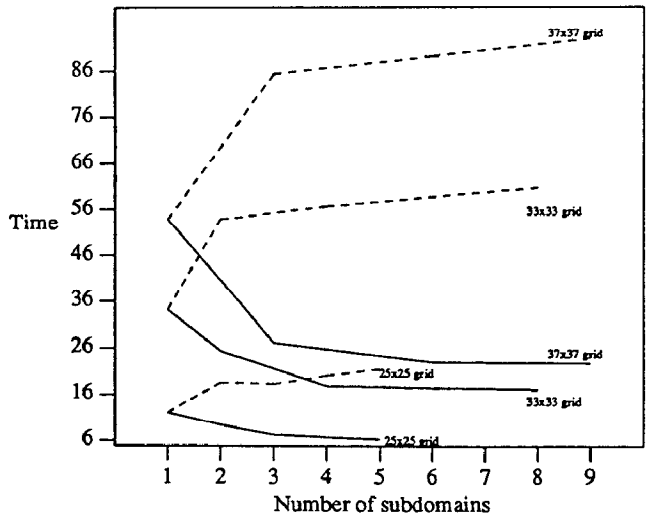

Figure 2.2. Timing in seconds of the sequential SCSC on VAX 8600 in double precision. The solid lines correspond to a domain decomposition in horizontal non overlapping strips while the dashed lines in vertical non overlapping strips.

One can also figure out that the increase of the number of the sub-

domains does not increase dramatically the total cpu time while increases significantly the parallelism.

3. A HYPERCUBE REALIZATION OF SCSC METHOD.

To assure load balancing we restrict ourselves to the homogeneous case, where all subdomains are of the same size. In addition, we assume that the number of processors we use is equal to the number of subdomains. We start by considering the particularly simple decomposition given in Figure 3.1(a) and partition the computation of SCSC by simply assigning one processor to each subdomain.

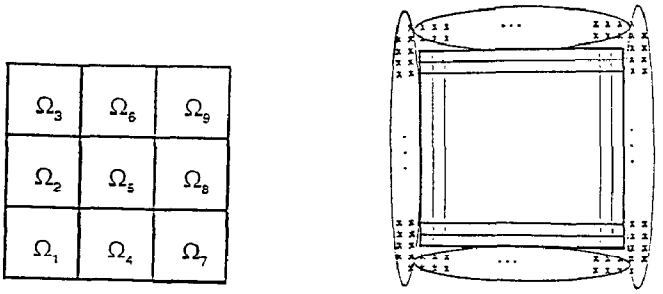

Figure 3.1. A simple 3×3 decomposition (a) and a subdomain with its data dependencies (b).

It is clear that neighbor processors share some degrees of freedom (the time symbol x's in Figure 4.1(b)) which are associated with the boundary of neighbor subdomains. Thus each processor needs to send the values of these unknowns to neighbor processors which must update the corresponding right hand sides accordingly. If we assume that in each subdomain we use an $n_i \times m_i$ grid then within one iteration, the amount of data to be communicated is $A_{comm} = 4(n_i + m_i + 3)$ while the total amount of computation (to update the right hand side and back solve) is $A_{comp} = 12(n_i + m_i) + \sqrt{n_i m_i}\ 2(m_i + 1)$. It is important to notice that the ratio $R_{eff} = \frac{A_{comm}}{A_{comp}}$ decreases as we increase the local grid $n_i \times m_i$ while is independent of the number of subdomains.

The first step of the hypercube implementation of SCSC (HSCSC) is executed on the host processor and consists of discretizing problem (2.1), (2.2) over domain Ω using **Cubic Spline Collocation**, by forming the global collocation coefficient matrix. Furthermore, the host processor sends the local coefficient matrices corresponding to various subdomains to the associated processor nodes, together with an initial guess of the corresponding degrees of freedom. The second step takes place at each processor node and consists of the factorization of each local coefficient matrix.

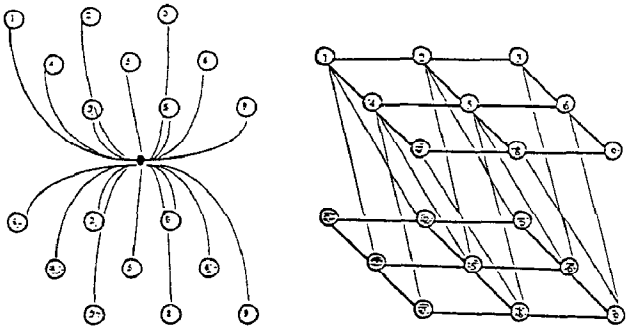

Figure 3.2. Communication schemes for (a) Jacobi and (b) Gauss-Seidel of the first two iterations for the 3×3 decomposition given in Figure 3.1(a).

If we use the Gauss-Seidel iteration scheme to implement the interaction among subdomains then each node processor gets the values of the degrees of freedom left to its left boundary from its nearest neighbor in the west direction as shown in Figure 3.1(b) and the corresponding degrees of freedom from the nearest neighbor in the south direction before updating the right hand side and back solving. Next, it sends the appropriate data to the west, north, south and east neighbor and waits to get data from the north and west before starting the next iteration. In Jacobi, at the beginning, all processors become fully utilized by working on each subdomain. When they finish they all send and receive data from neighbors and start working on the next iteration. In Figure 3.2 we give the data dependence graph of the proposed parallel Jacobi and Gauss-Seidel iterations.

The above iterative procedures can be described in pseudo C code as follows :

HOST PROGRAM

```
call init;
call generate_collocation_matrix;

for ( node = 1; node < number_of_subdomains; node++ ) {
        call send_parameters(node);
        call send_local_matrix(node);
        call get_local_solution(node);
}

call form_solution;
```

NODE PROGRAM (GAUSS-SEIDEL)

```
call get_parameters;
call get_local_matrix;
call factor_matrix;

for ( iter = 1; iter < max_iterations; iter++ ) {
        call get_solution(south,west);
        call modify_rhs
        call backsolve;
        call send_solution(north,east,south,west);
        call get_solution(north,east);
}

call send_local_solution(host);
```

NODE PROGRAM (JACOBI)

```
call get_parameters;
call get_local_matrix;
call factor_matrix;

for ( iter = 1; iter < max_iterations; iter++ ) {
        call modify_rhs
        call backsolve;
        call send_solution(north,east,south,west);
        call get_solution(north,east,south,west);
}

call send_local_solution(host);
```

4. NUMERICAL EXPERIMENTS ON THE NCUBE/7 (128 processors).

In this section we present the numerical performance of HSCSC in double precision on an NCUBE/7 hypercube machine with 128 processors. In all experiments, we have applied HSCSC on the self adjoint elliptic equation

$$D_x(e^{xy}D_x u) + D_y(e^{-xy}D_y u) - u/(1+x+y) = f(x,y) \qquad (4.1)$$

with Dirichlet boundary conditions on the unit square. The right hand side f corresponds to the exact solution $u = 0.75e^{xy}\sin(\pi x)\sin(\pi y)$. In these experiments, the local to each processor, linear system of equations is solved by Gauss elimination using the LINPACK routines for banded matrices DGBFA and DGBSL. The convergence tolerance for each grid size is selected in such a way that guarantees the fourth order convergence of the Cubic Spline Collocation. All timings are given in seconds.

					GAUSS-SEIDEL		JACOBI	
cube	global grid	domains	local grid	error	iterations	time	iterations	time
0	9×9	1	9×9	1.4E-4	1	.21		
1		1×2	9×5		5	.20	11	.16
1		2×1	5×9		7	.33	17	.31
2		2×2	5×5		10	.21	23	.14
0	13×13	1	13×13	3.3E-5	1	.77		
1		1×2	13×7		8	.69	16	.53
1		2×1	7×13		12	1.41	22	1.02
2		2×2	7×7		15	.60	31	.45
3		2×4	7×4		18	.35	37	.25
3		4×2	4×7		21	.45	42	.33
4		4×4	4×4		25	.27	49	.21
0	17×17	1	17×17	1.1E-5	1	20.6*		
1		1×2	17×9		11	10.5	21	7.9
1		2×1	9×17		16	23.2	31	17.4
2		2×2	9×9		22	9.4	43	6.9
3		2×4	9×5		28	4.9	55	3.8
3		4×2	5×9		31	7.0	63	5.1
4		4×4	5×5		37	3.6	76	3.0
0	21×21	1	21×21	4.8E-6	1	26.4*		
2		2×2	11×11		29	20.7	57	15.8
3		2×4	11×6		36	9.7	71	7.5
3		4×2	6×11		41	15.5	81	11.2
4		4×4	6×6		50	7.2	99	5.6
0	25×25	1	25×25	2.3E-6	1	51.0*		
2		2×2	13×13		37	40.3	74	32.1
3		2×4	13×7		46	18.0	91	14.2
3		4×2	7×13		52	30.1	105	22.3
4		4×4	7×7		62	12.8	125	10.5
5		4×8	7×4		82	7.8	164	6.8
5		8×4	4×7		97	10.5	187	8.5
6		8×8	4×4		112	6.7	232	5.8
0	33×33	1	33×33	7.3E-7	1	144.5*		
3		2×4	17×9		66	48.5	131	39.1
3		4×2	9×17		76	89.3	151	66.6
4		4×4	9×9		91	34.0	182	28.3
5		4×8	9×5		122	18.7	245	16.9
5		8×4	5×9		142	27.2	286	22.7
6		8×8	5×5		173	14.4	347	13.3
0	41×41	1	41×41	3.0E-7	1	329.6*		
4		4×4	11×11		121	75.6	236	61.5
5		4×8	11×6		162	37.4	314	32.7
5		8×4	6×11		189	59.5	375	49.1
6		8×8	6×6		229	28.0	458	25.6
0	65×65	1	65×65	4.2E-8	1	1949.5*		
5		4×8	17×9		300	195.5	591	172.9
5		8×4	9×17		350	346.8	685	290.7
6		8×8	9×9		435	142.8	832	125.0
7		8×16	9×5		600	83.3	1175	79.0
7		16×8	5×9		700	117.3	1360	107.1

Table 4.1. Timing and number of iterations of HSCSC for non overlapping subdomains. (* predicted value)

In Table 4.1, we summarize the performance of HSCSC method on the NCUBE/7. Various configurations are used and in the first column we give the order of the corresponding subcube. The second column gives the grid used to discretize the original domain and the fourth column gives the grid used for each subdomain. The third column gives the number of non-overlapping subdomains in x and y directions that fits to the current NCUBE configuration. The error achieved is reported in the fifth column. It is easy to see the

optimal fourth order convergence of the Cubic Spline Collocation method. Columns six and eight give the number of iterations needed by Jacobi and Gauss-Seidel schemes to achieve the discretization error. It is important to notice that Gauss-Seidel needs precisely half the number of iterations that Jacobi needs. That was expected since from Theorem 2.1 we know that the spectral radius of the Gauss-Seidel is half the spectral radius of the Jacobi. It can also be seen that the direction of splitting is a dominating factor for the parallel performance, but not as strong as in the sequential case. Finally in column seven and nine we list the total cpu time (the time to read data from the host processor is not included). Based on the above data the Jacobi scheme is at least 10 percent faster than the Gauss-Seidel scheme.

Table 4.2 is similar to Table 4.1 but involves overlapping subdomains. In column five, we list the size of the overlapping area. More specifically we give the number of horizontal and vertical grid lines that are common to two subdomains. According to these data the increase of overlapping area results in rapid decrease of the number of iteration and the execution time for both Jacobi and Gauss-Seidel. It should be noticed that even for overlapping subdomains the Jacobi scheme is faster and that the Gauss-Seidel requires precicely half the number of Jacobi iterations.

In Table 4.3 we present the relative speedups achieved for the data given in Table 4.1, as the ratio of the execution times with two different Ncube/7 configurations on the same size problem. Specifically in the second and third columns we give the number of node processors used and in the last two columns the associated time ratio for Gauss-Seidel and Jacobi iterations.

Comparing these data against the relative speedups of domain decomposition methods for the Laplace equation presented in [Keye 87] we can claim that the Schwarz splitting can be implemented at least as effectively as the domain decomposition methods for the spline collocation methods.

					GAUSS-SEIDEL		JACOBI	
cube	global grid	subdomains	local grid	overlap	iterations	time	iterations	time
2	9×9	2×2	5×5	1×1	10	.21	23	.14
			6×6	2×2	5	1.29	10	.73
			7×7	4×4	3	1.87	8	2.21
2	13×13	2×2	7×7	1×1	15	.60	31	.45
			8×8	2×2	8	3.74	16	2.21
			9×9	4×4	5	4.69	10	2.49
			10×10	6×6	4	6.41	7	2.89
			11×11	8×8	3	8.50	5	3.76
4		4×4	4×4	1×1	25	.27	49	.21
			7×7	12×12	9	4.93	19	2.00
2	17×17	2×2	9×9	1×1	22	9.4	43	6.9
			10×10	2×2	11	8.8	22	5.6
			11×11	4×4	8	10.9	15	5.8
			12×12	6×6	6	13.1	11	6.4
			13×13	8×8	4	16.1	9	7.7
4		4×4	5×5	1×1	37	3.6	76	3.0
			8×8	12×12	14	8.5	28	3.8
2	21×21	2×2	11×11	1×1	29	20.7	57	15.8
			12×12	2×2	15	18.4	29	12.0
			13×13	4×4	10	20.4	20	12.2
4		4×4	6×6	1×1	50	7.2	99	5.6
			9×9	12×12	18	13.8	38	6.8
4	25×25	4×4	7×7	1×1	62	12.8	125	10.5
			10×10	12×12	22	20.7	46	10.4
			13×13	24×24	14	42.5	29	24.6
4	33×33	4×4	9×9	1×1	91	34.0	182	28.3
			12×12	12×12	33	44.2	68	24.6

Table 4.2. Timing and number of iterations of HSCSC algorithm for overlapping subdomains.

			T_q / T_p	
global grid	p	q	Gauss-Seidel	Jacobi
13×13	2	8	2.00	2.12
17×17	2	8	2.14	2.08
	4	16	2.61	2.30
21×21	4	16	2.85	2.82
25×25	4	16	3.14	3.05
	8	32	2.86	2.62
	16	64	1.91	1.81
33×33	8	32	2.59	2.31
	16	64	2.36	2.13
41×41	16	64	2.70	2.40
65×65	32	128	2.35	2.18

Table 4.3. Relative speedups of HSCSC with Jacobi and Gauss-Seidel iteration schemes for the data given in Table 4.1.

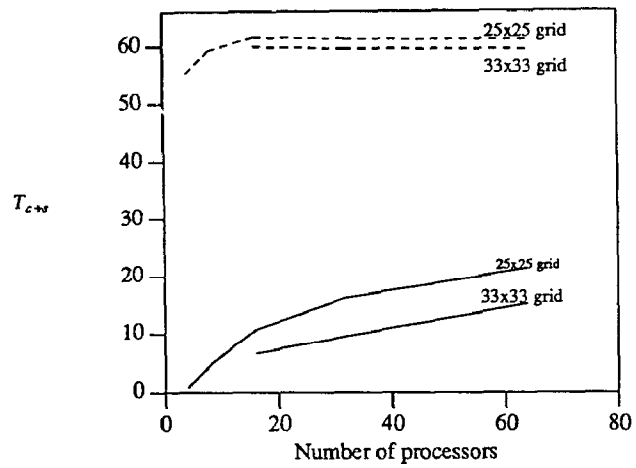

Figure 4.3. Percentage of the cpu time spent for communication and synchronization as a function of the number of processors. Solid lines represent the Jacobi and dashed lines the Gauss-Seidel iterations.

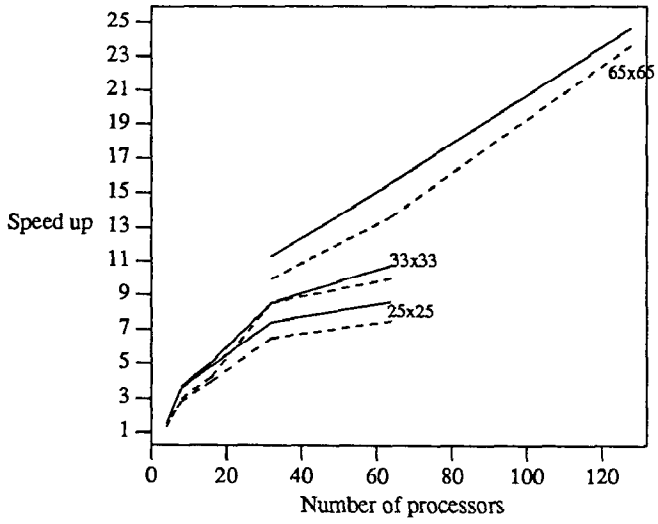

Figure 4.4. Speedups of HSCSC with Jacobi (solid lines) and Gauss-Seidel (dashed lines) iteration schemes for the data given in Table 4.1. (Speedup has been obtained with respect to sequential SCSC.)

Recently, ([Luba 84], [Reed 84]) has been realized that in parallel iterative methods one can avoid the synchronization penalty using a chaotic iterative scheme. In earlier work [Hous 87] we developed an asynchronous iterative method for the solution of a class of elliptic PDEs on bus architecture machines with success. With this in mind we have tried to remove synchronization from the SCSC algorithms described in section 4.1. The new iterative scheme converges unreasonable slow and the cpu time increases significantly.

REFERENCES

[Bjor 86] P. E. Bjorstad and O. B. Widlund, *Iterative methods for the solution of elliptic problems on regions partitioned into substructures*, SIAM J. Numer. Anal., Vol 23, No. 6, 1097–1119

[Ehrl 86] L. W. Ehrlich, *The numerical Schwarz alternating procedure and SOR*, SIAM J. Sci. Stat. Comput., Vol. 7, No. 3, 989–993.

[Hous 87] E. N. Houstis, J. R. Rice and E. A. Vavalis, *Parallelization of a new class of cubic spline collocation methods*, in: *Advances in Computer Methods for Partial Differential Equations*, Vol. VI, (R. Vichnevetsky, R. S. Stepleman, editor), 167–174.

[Hous 88a]E. N. Houstis, J. R. Rice and E. A. Vavalis, *Convergence of an $O(h^4)$ Cubic Spline-Collocation Method for Elliptic Partial Differential Equations*, SIAM J. Num. Anal., Vol 25, No 1, 54-73.

[Hous 88b]E. N. Houstis, J. R. Rice, E. A. Vavalis and A. Hadjidimos, *Convergence of Schwarz Decomposition for a Class of Cubic Spline Collocation Methods*. (In preparation).

[Keye 87] D. E. Keyes and W. D Gropp, *A comparison of domain decomposition techniques for elliptic partial differential equations and their parallel implementation*, SIAM J. Sci. Stat., Vol 8, No. 2, s166–s202

[Luba 84] B. Lubachevsky and D. Mitra, *Chaotic parallel computations of fixed points of nonnegative matrices of unit spectral radius*, IEEE, 109–115.

[Meie 86] U. Meier, *Two parallel SOR variants of the Schwarz alternating procedure*, Parallel Computing, 3, 205–215.

[Reed 84] D. A. Reed and M. L. Patrick, *A model of asynchronous iterative algorithms for solving large, sparse, linear systems*, IEEE, 402–409.

[Rice 85] J. R. Rice and R. F. Boisvert, *Solving Elliptic Problems Using Ellpack*, Springer-Verlag, New York, (1985).

[Rodr 84] G. Rodrigue and J. Simon, *Jacobi splittings and the method of overlapping domains for solving elliptic PDEs*, in: *Advances in Computer Methods for Partial Differential Equations*, Vol. V, (R. Vichnevetsky, R. S. Stepleman, editor), 383–386.

[Rodr 86] G. Rodrigue, *Some ideas for decomposing the domain of elliptic partial differential equations in the Schwarz process*. Comm. Appl. Num. Meth., Vol. 2, 245–249.

[Tang 87] W.P. Tang, *Schwarz splitting, a model of parallel computations*, Ph.D. Thesis, Stanford University.