

Numerical Experiments With a Domain Decomposition Method for Parabolic Problems on Parallel Computers

G erard Meurant*

Abstract.

In this paper, we describe numerical experiments on parallel computers obtained with a Domain Decomposition preconditioner [8] and the conjugate gradient method for solving linear systems arising from parabolic partial differential equations in two dimensional domains. The method relies on patching local overlapping solutions and leads to a completely parallel preconditioner for computers with a moderate number of processors matching the number of subdomains. A very useful variant is obtained when using approximate solutions for the subproblems. Numerical examples are described on a Sequent Symmetry S81 with 8 processors and a Cray Y-MP/832. They show that very good speed ups and performances can be obtained with this kind of method.

1. Introduction.

We are interested in solving linear systems arising from the implicit discretization of linear partial differential parabolic equations like,

$$\frac{\partial u}{\partial t} - \frac{\partial}{\partial x} \left(a(x, y) \frac{\partial u}{\partial x} \right) - \frac{\partial}{\partial y} \left(b(x, y) \frac{\partial u}{\partial y} \right) = f$$
$$\text{in } \Omega \subset R^2,$$

$$u|_{\partial\Omega} = 0, \quad u(x, 0) = u_0(x).$$

However, more general boundary conditions can be handled without problems. The model problem that is to be used is the heat equation :

$$\frac{\partial u}{\partial t} - \Delta u = f,$$

with Dirichlet boundary conditions, Ω being the unit square.

*CEA, Centre d'Etudes de Limeil-Valenton, 94195 Villeneuve St Georges cedex, France.

To time discretize this problem, for stability and efficiency an implicit Crank–Nicolson scheme is used that can be described in the following framework :

consider the general parabolic partial differential equation,

$$\frac{\partial u}{\partial t} + Lu = f,$$

with L being a second order self-adjoint linear elliptic operator. Space is discretized with a standard finite difference scheme with $n + 1$ points on each side of the square domain Ω , h being the mesh size,

$$h = \frac{1}{n + 1}.$$

Then we obtain an $n^2 \times n^2$ sparse matrix A from L ,

$$L \Rightarrow \frac{1}{h^2}A$$

where A is a block tridiagonal matrix (corresponding to the steady state problem $Lu = f$). With the standard five point scheme, A is a diagonally dominant M-matrix.

Time is discretized with a centered scheme : with $t \in [0, T]$, $k = \Delta t$ being the time step, m referring to the values of vectors at time mk , we have

$$\frac{u^{m+1} - u^m}{k} + \frac{1}{2h^2}(Au^{m+1} + Au^m) = \frac{1}{2}(f^{m+1} + f^m),$$

or

$$2\frac{h^2}{k}(u^{m+1} - u^m) + Au^{m+1} + Au^m = h^2(f^{m+1} + f^m).$$

At every time step we have to solve a linear system

$$\left(2\frac{h^2}{k}I + A\right)u^{m+1} = 2\frac{h^2}{k}u^m - Au^m + h^2(f^{m+1} + f^m).$$

The advantages of the implicit scheme are its second order accuracy (which means that the truncation error is $O(k^2) + O(h^2)$) and unconditional stability that allows larger time steps. In practice the problem will not be solved in the form written in the last equation, but is rewritten in the following way :

$$\text{let } \theta = 2\frac{h^2}{k}, \text{ and } A_t = \theta I + A,$$

$$A_t x = h^2(f^{m+1} + f^m) - 2Au^m,$$

then,

$$u^{m+1} = x + u^m.$$

Remark that for our problem, A_t is a symmetric strictly diagonally dominant M-matrix.

To solve this symmetric positive definite system, a very popular iterative method is the pre-conditioned Conjugate Gradient (PCG) :

let x^0 given and,

$$r^0 = b - A_i x^0,$$

for $k = 0, 1, \dots$ until convergence,

$$\begin{aligned} M z^k &= r^k, \\ \beta_k &= \frac{(r^k, z^k)}{(r^{k-1}, z^{k-1})}, \quad \beta_0 = 0, \\ p^k &= z^k + \beta_k p^{k-1}, \\ \alpha_k &= \frac{(r^k, z^k)}{(A_i p^k, p^k)}, \\ x^{k+1} &= x^k + \alpha_k p^k, \\ r^{k+1} &= r^k - \alpha_k A_i p^k. \end{aligned}$$

M being the preconditioning matrix or preconditioner.

Within each iteration, an additional linear system must be solved,

$$M z = r.$$

In this paper, the target computer architecture will be a parallel machine with a few (of the order of 10) processors and a shared memory. Hence, it is of interest to devise a preconditioner as parallel as possible, that is with the smallest possible number of synchronization points. However, M must also be chosen to have a good rate of convergence for PCG, cf. [4]. Many different solutions to this problem have been proposed during the last years, (for references see [1], [3]). Here, we will study parallel preconditioners M that do require minimal synchronisation between the different processors. Section 2 describes the domain decomposition method introduced in [8] and the possible extensions. Section 3 gives numerical results on shared memory parallel computers like the Sequent Symmetry S81 and the Cray Y-MP/832.

2. The Domain Decomposition method.

In this Section, the domain decomposition preconditioner M is defined. We use a method that was proposed in a different setting as a direct method by Y. Kuznetsov [6] and defined in [8]. The domain Ω is divided into non-overlapping subdomains Ω_i , $i = 1, \dots, l$. For the sake of the numerical experiments on the unit square, the Ω_i s will be squares (also called boxes), see figure 1, and it is supposed that their boundaries are parallel to the rows and columns in the mesh but not mesh lines.

Each Ω_i is extended to a domain $\widehat{\Omega}_i$ that overlaps the neighbours of Ω_i (restricted of course to Ω), see figure 2 where Ω_i corresponds to the gray area and $\widehat{\Omega}_i$ to the same plus the stripped area.

To solve $M z = r$ the following steps are performed : for each subdomain $\widehat{\Omega}_i$, $i = 1, \dots, l$, let \widehat{A}_i be the $n_i \times n_i$ matrix arising from the discretization of the problem on $\widehat{\Omega}_i$ with homogeneous

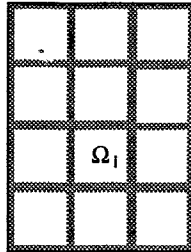


Figure 1
Boxes

Dirichlet boundary conditions. Let \hat{r}_i be the vector of length n_i that will be the right hand side on $\hat{\Omega}_i$ whose entries are equal to those of r for components corresponding to mesh points in Ω_i and 0 elsewhere. Then let \hat{z}_i be defined by solving

$$\hat{A}_i \hat{z}_i = \hat{r}_i,$$

we extend \hat{z}_i to a vector z_i on the whole Ω with 0 components corresponding to mesh points outside $\hat{\Omega}_i$. The global solution z is simply defined as

$$z = \sum_{i=1}^l \hat{z}_i.$$

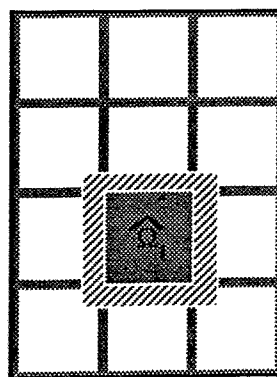


Figure 2
Subdomain $\hat{\Omega}_i$

The main problem is to know where to put the artificial boundaries. For the method used by Y. Kuznetsov [6], it was clear that if the problem has to be solved with a precision ϵ and if q is the

maximum number of subdomains that overlap at a given mesh point, then each subproblem must be solved with a precision of ϵ/q . Therefore, it is enough to ask for the solution to be smaller than the maximum of the right hand side $\times \epsilon/q$.

There is nothing so simple here, because we are solving for the difference of the solution at two successive time steps and, as the Conjugate Gradient method is used, the linear system under consideration has the residual as the right hand side. It is not precisely known what is the influence of the precision on the preconditioner solve over the rate of convergence of CG.

However, it is clear that the action of the preconditioner M can be viewed as a solve with A_i as the matrix with a certain precision ϵ . Then, if the following criteria :

$$\frac{\|r^k\|}{\|r^0\|} \leq 10^{-p}$$

is used to stop CG iterations, and as only a moderate precision is needed for the solve (usually $p = 2$ or $p = 3$), one possibility is to compute the overlapping asking for the terms of the inverse to be smaller than 10^{-p} , see [8] for formulas giving an estimate of the extent of overlapping.

As we will see below in the numerical experiments, the problem with the method we have just described, is the following. The extent of overlapping one needs to obtain a given number of iterations is fixed (it is not even a function of the mesh size, it is only given by the properties of the underlying operator), so if we want to increase the number of subdomains, the area of each Ω_i decreases but the area of $\widehat{\Omega}_i$ must remain the same. As the problems are solved on $\widehat{\Omega}_i$, the work per iteration increases as we have more subdomains, the problem being that the cost of the Linpack solves is proportional to $n_i\sqrt{n_i}$ for boxes. It can also be seen that the number of iterations increases slightly when the number of subdomains is larger. These two reasons imply that, unfortunately, this method is not worth using with more than four subdomains.

To try to solve this problem, an obvious idea is to replace the "exact" Linpack solves by an approximate solve that is less costly. The method tested in this study is to use an incomplete Choleski decomposition for the subproblems. Let $\widehat{L}_i\widehat{L}_i^T$ be a modified incomplete Choleski decomposition of \widehat{A}_i (cf. [2], [5]), \widehat{L}_i being a lower triangular matrix with the same sparsity pattern as the lower triangular part of \widehat{A}_i (the so-called MIC(0) method), then \widehat{z}_i is defined as the solution of

$$\widehat{L}_i\widehat{L}_i^T\widehat{z}_i = \widehat{r}_i.$$

We use the same choices of overlapping as in the "exact" method. In this method that we will refer as MIC, the cost of one solve is only $9n_i$.

If this method would have been used with stripes instead of boxes, it would have resemble a method of Radicati & Robert [9]. However, in their method, the right hand side is not chosen in the same way.

4. Numerical experiments.

As an example, the two dimensional model problem,

$$\frac{\partial u}{\partial t} - \Delta u = f,$$

is solved with Dirichlet boundary conditions, Ω being the unit square.

The right hand side f is computed such that the exact solution u is

$$u(x, y, t) = \sin(\pi t)xy(1-x)(1-y)\exp(xy).$$

We integrate in time over the interval $[0, T]$ with $T = 0.5$. As the scheme is second order accurate, we choose to have the time step to be of the order of the mesh size. For the experiment, we took $h = 1/34$ and $k = 1/60$; with these values, $\theta = 0.104$ and the maximum error in the max norm over $[0, T]$ is $0.11 \cdot 10^{-4}$. The mesh nodes are ordered with the “natural” or rowwise ordering (bottom to top, left to right). At each time step, we ask for a reduction of the residual l_2 norm by a factor of 10^{-2} . We solve the problem with square boxes as shown in figure 1.

First, we present results on a Sequent Symmetry S81. The machine we used has 10 processors (Intel 80386 + Weitek 1167 floating point accelerators) and a 32 Mbytes shared memory. Each processor has a 64 Kbytes cache; they are connected via a bus whose peak transfer speed is 80 Mbytes/second. In the following experiments, we use at most only 8 processors leaving 2 processors available to the operating system and a dedicated machine. The parallelism in the algorithm was described using Sequent specific directives. The iterations of simple PCG loops were shared in parallel by the different processors. The subdomains solves were parallelized adding a directive in front of the loop over the subdomains. Each subdomain was completely handled by one processor.

The only point that has to be carefully handled is the phase of the algorithm where the summation of the contributions of the different overlapping subregions is done as, then, the processors must access some shared variables. The values on domains Ω_i s (grey area in figure 3) can be summed in parallel without problem as these regions are disconnected. Some care must be given to the summation over $\widehat{\Omega}_i/\Omega_i$. One way to do this is to separate the “corners” (vertical stripes on figure 3) and the rest of this region (oblique stripes), as shown in figure 3. Then, it can be seen that, as long as there is no overlapping between two domains that are not adjacent to each other, the summation can be done in parallel for all the “corners” and then in parallel for the rest of the region as they are all disconnected.

Figure 4 gives the total number (over $[0, T]$) of CG iterations as a function of the overlapping. Here, the overlapping means the number of mesh lines in the overlapping region in one direction. As the problem is isotropic, we take the same overlapping in both directions. Eight is the “optimal” overlapping with respect to the number of iterations as we are doing only two iterations per time step.

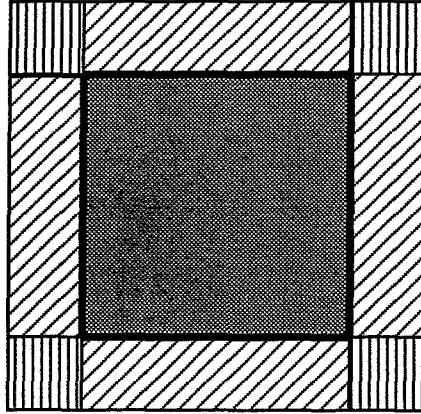


Figure 3
Partition of the subdomains

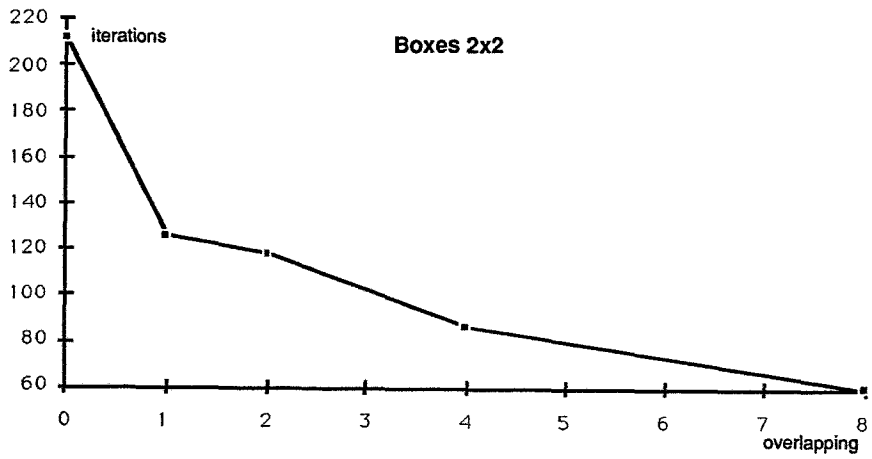


Figure 4
Number of iterations with 4 subdomains

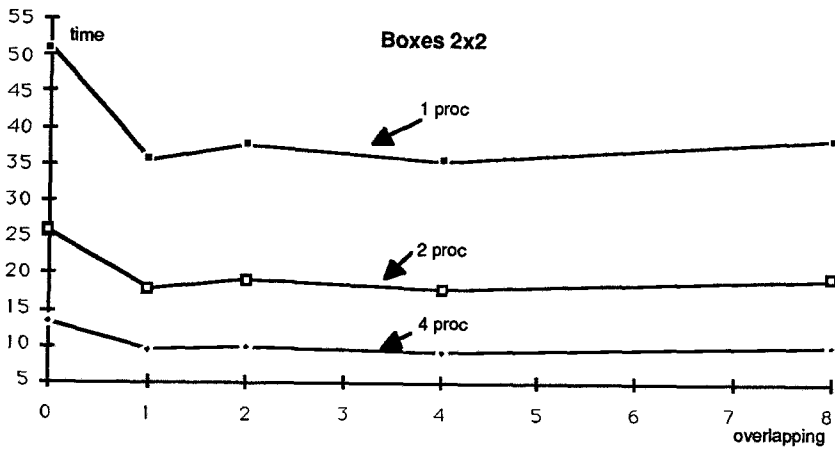


Figure 5
Sequent elapsed times with 4 subdomains

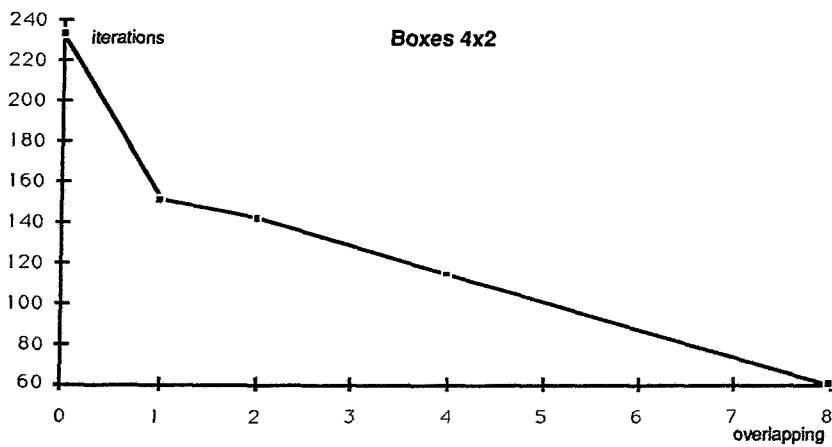


Figure 6
Number of iterations with 8 subdomains

Figure 5 shows that, unfortunately, if an overlapping of 8 is the best for the number of iterations, it is not the same for the computing (elapsed) time with 1, 2 and 4 processors. With this respect an overlapping of 1 is better and even though the gain on using no overlapping at all is really modest. This is because of the large cost of the solves on each subdomain. It can be surprising that the results in Figure 5 are not exactly the same as those of [8]. However, first of

all the coding was change in order to have more parallelism in the “summing” phase and secondly with the mesh size we chose here the size of all the subdomains are the same which was not the case in [8].

In [8] a comparison was done of the present algorithm with a diagonal preconditioner. Here, we found 523 iterations and an elapsed time of 21.6 seconds on one processor and 3.5 seconds on 8 processors. This is of the same order as what we found with our algorithm but as it was shown on [8], there are more difficult problems where the diagonal preconditioner is not competitive at all.

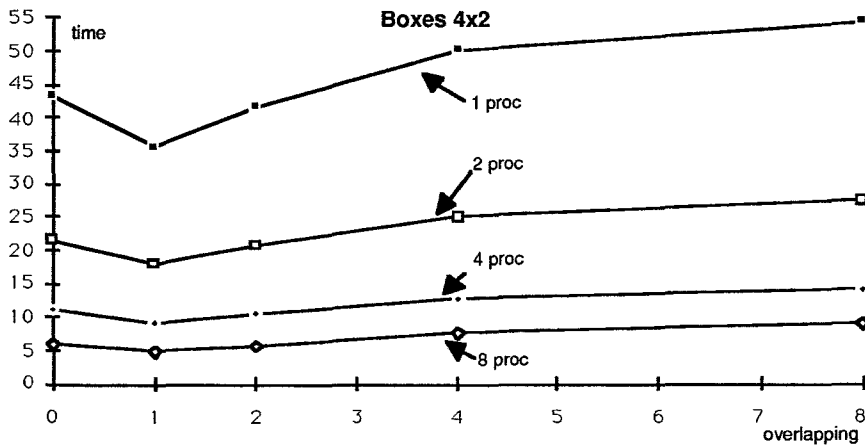


Figure 7

Sequent elapsed times with 8 subdomains

Figures 6 and 7 show that the problem is the same (or even worse) with more subdomains and an exact solve. Here, we have 4 divisions in the x direction, and 2 in the y direction matching the maximum number of processors. In this case, the speed up with 8 processors and an overlapping of 1 is 7.5 which shows that the machine is correctly used and that the algorithm is well parallelized.

Figure 8 gives the number of iterations as a function of $1/h$ for a fixed time step of $1/60$ and a fixed extent of overlapping, that is one mesh line for $h = 1/18$, two for $h = 1/34$ and four for $h = 1/66$. It is seen that the number of iterations is asymptotically constant, which means that this method can be useful for very large problems.

Now, we replace the exact Linpack solves for the subdomains by the approximate MIC solves. Figure 9 shows the number of iterations for MIC. The “optimal” overlapping for the number of iterations is 8 as then the number of iterations is 118 to be compared with 116 when MIC is used on the whole domain.

Figure 10 compares the number of iterations with the exact solve and the approximate MIC

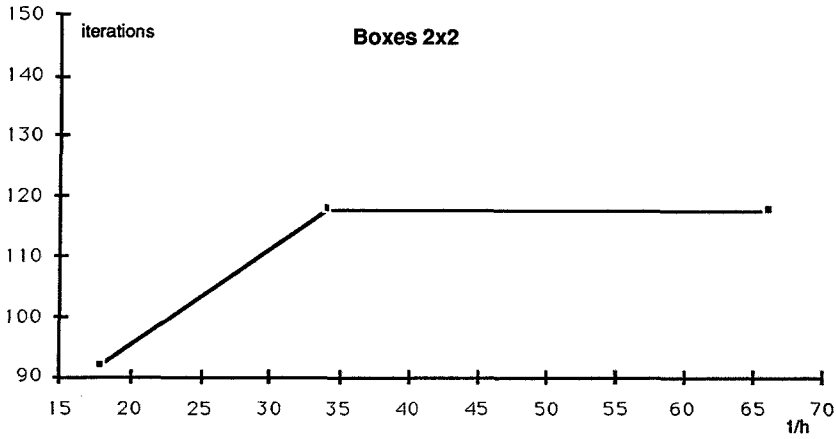


Figure 8
 Number of iterations as a function of $1/h$

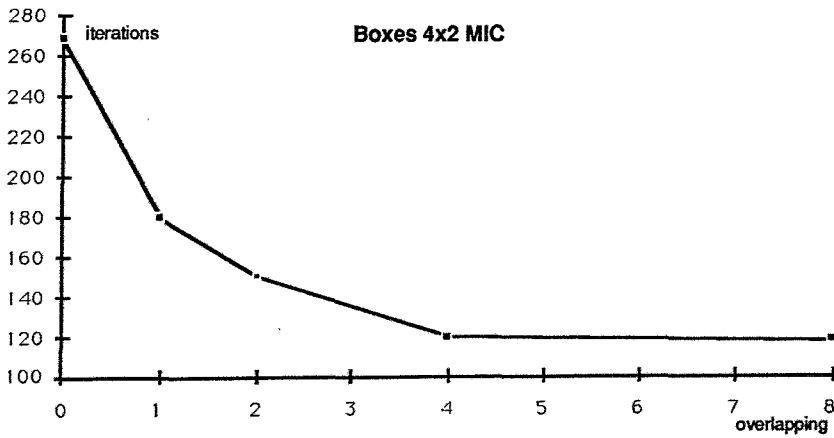


Figure 9
 Number of iterations with 8 subdomains and MIC

one as a function of the overlapping. It can be seen that with an overlapping of 2 or 4 the numbers of iterations are very close for the exact and approximate solvers and therefore the MIC solve must be much cheaper.

Figure 11 shows the elapsed times as a function of the overlapping for different numbers of processors. The optimal overlapping is 4 whatever the number of processors is. This is shown more

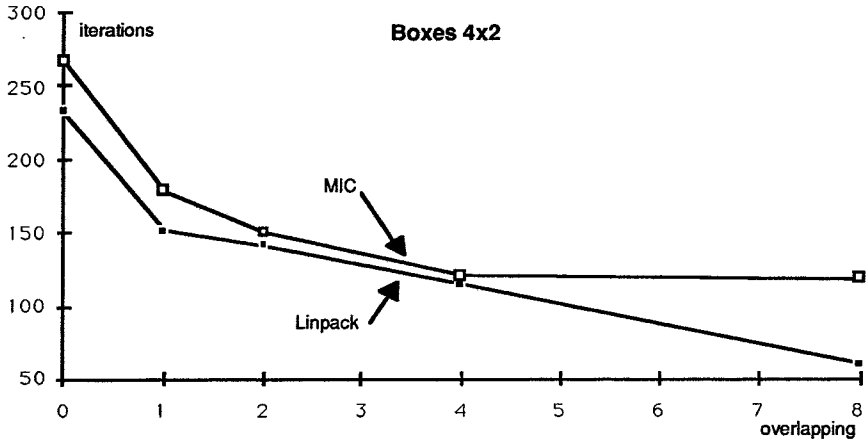


Figure 10

Comparison of numbers of iterations with exact and approximate solvers

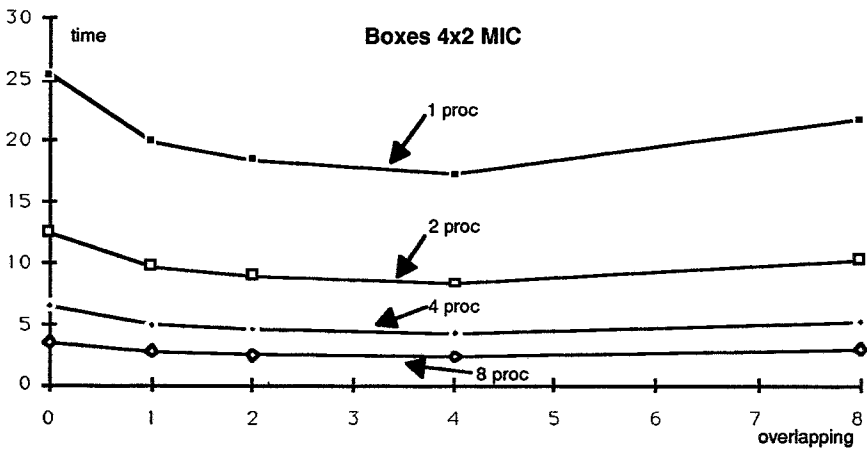


Figure 11

Sequential elapsed times with MIC

precisely on Figure 12 for 8 processors.

Figure 13 gives the number of iterations as a function of the overlapping for different numbers of subdomains. We see that if the number of mesh lines that overlap is constant for a fixed mesh size h , the number of iterations does not depend too much on the number of subdomains.

In Figure 14, the number of iterations is almost linear as a function of $1/h$ for a fixed extent

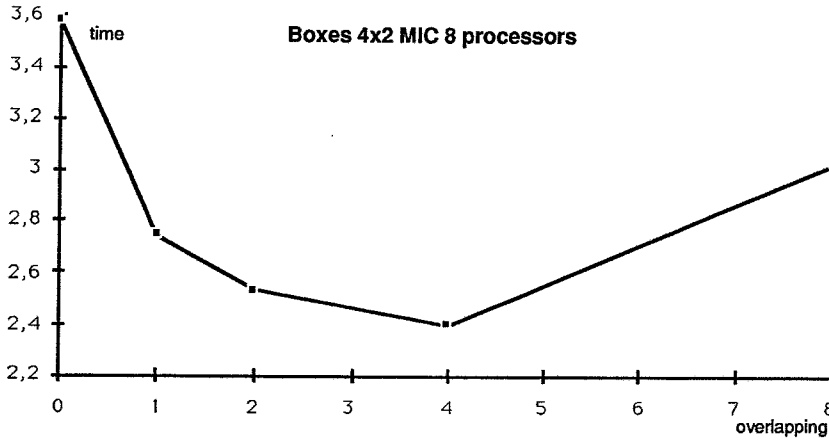


Figure 12
Sequent elapsed times on 8 processors with MIC

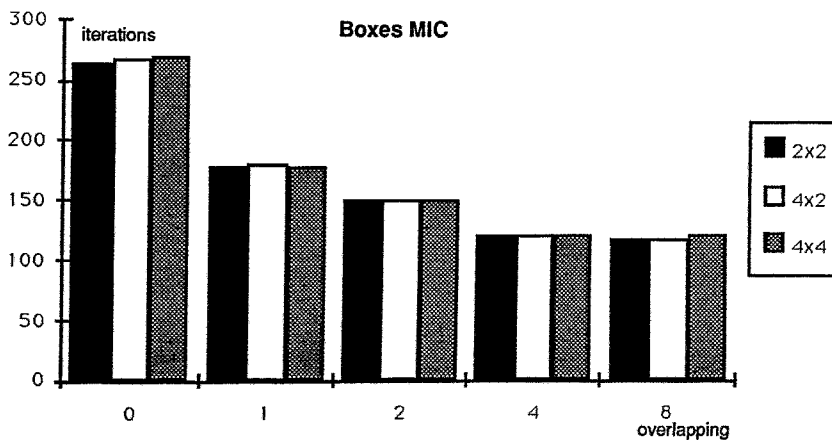


Figure 13
Number of iterations with MIC as a function of the number of subdomains

of overlapping (1, 2, 4 mesh lines).

Now, we briefly present some results on the CRAY Y-MP/832 using 8 processors. To parallelize the code we use the "Autotasking" feature that does some automatic parallelization. However, the parallelism over the subdomain solves cannot be find by the automatic parallelizer. Therefore, we have to add a few directives as with the Sequent. Figure 15 shows the elapsed times for 1 and

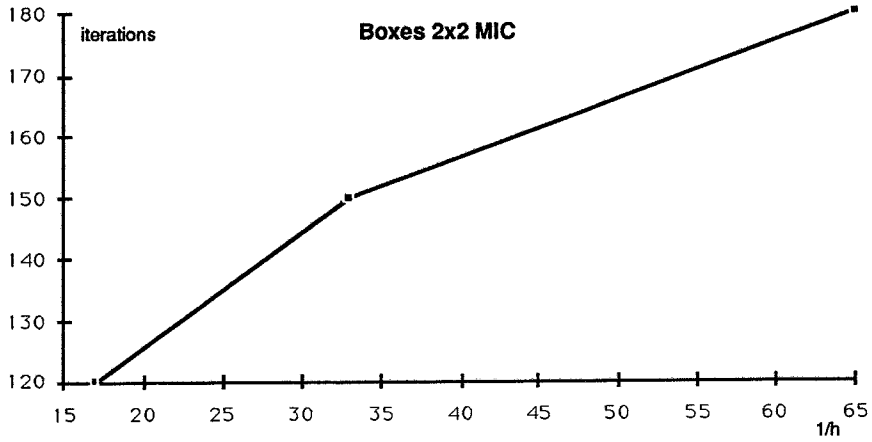


Figure 14
Number of iterations with MIC as a function of 1/h

8 processors as a function of the overlapping for the exact solves. The best speed up we get here is 6.6.

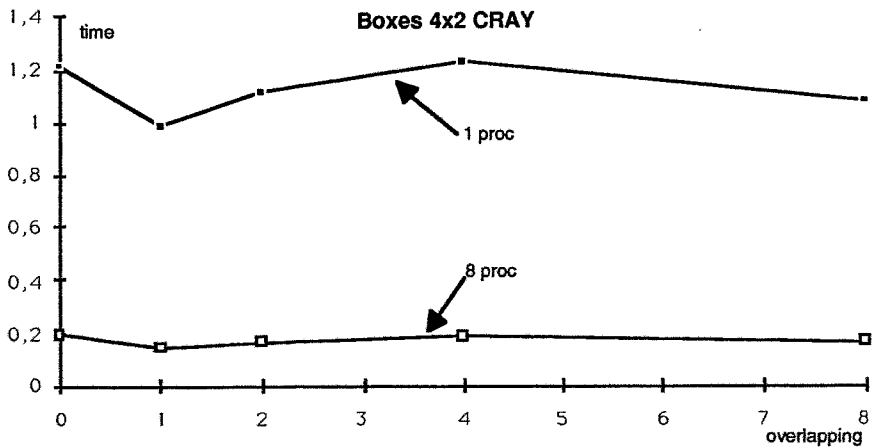


Figure 15
Cray Y-MP elapsed times

Figure 16 gives the same information for the approximate MIC solve. Here the speed up is only 5.6, probably because the granularity of the computation is smaller and the problem we are solving is not large enough. The times for 8 processors are also shown in figure 17. The optima!

overlapping is 4, but the time with an overlapping of 2 is not much different. It can be said that the overall behaviour of the methods is the same on the two shared memory parallel computers we used.

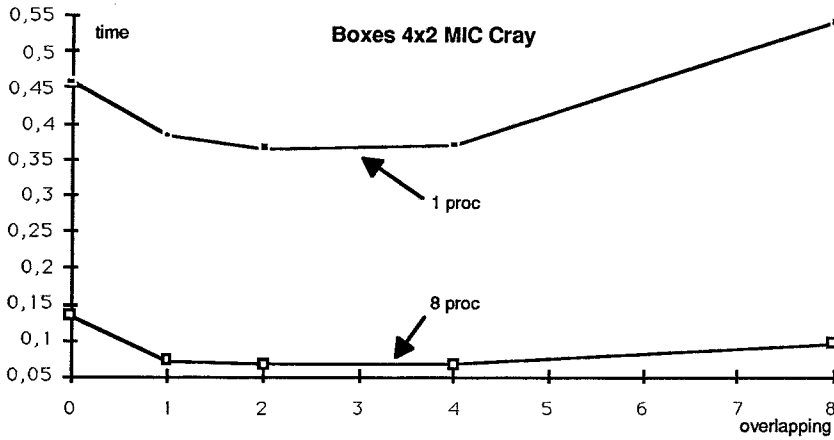


Figure 16
Cray Y-MP elapsed times with MIC

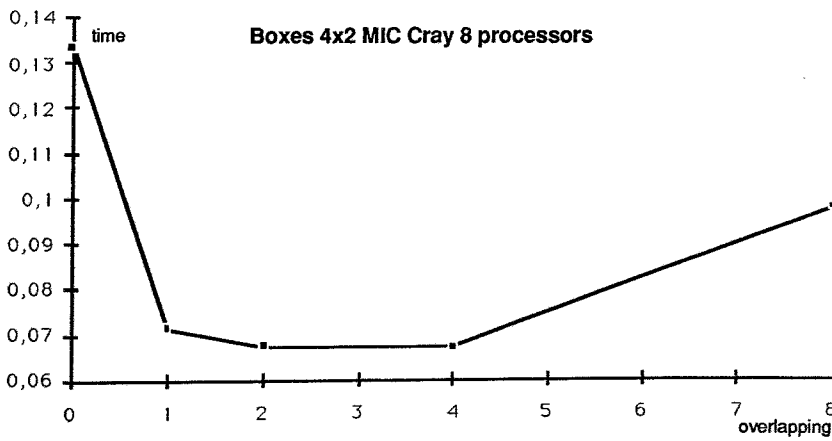


Figure 17
Cray Y-MP elapsed times with MIC and 8 processors

5. Conclusions.

In this paper, we described a domain decomposition preconditioner for linear systems arising from the discretization of parabolic problems. We show in [8] that using some results on inverses of banded matrices, the overlapping of the subdomains can be roughly estimated. A more efficient variant using incomplete factorizations for the subdomains was developed. As a minimal amount of communication is needed between the subdomains, this method is well suited for parallel computers, gives good speed ups as shown by the numerical experiments and is very easy to code.

References

- [1] T. CHAN, R. GLOWINSKI, J. PERIAUX & O. WIDLUND Eds., *Proceedings of the second international symposium on domain decomposition methods for partial differential equations*. SIAM (1989).
- [2] T. DUPONT, R.P. KENDALL & H. RACHFORD, *An approximate factorization procedure for solving self-adjoint elliptic difference equations*. SIAM J. of Numer. Anal v 9 n 3 (1968) pp 559–573.
- [3] R. GLOWINSKI, G.H. GOLUB, G. MEURANT & J. PERIAUX Eds., *Proceedings of the first international symposium on domain decomposition methods for partial differential equations*. SIAM (1988).
- [4] G.H. GOLUB & C. VAN LOAN, *Matrix computations*. Johns Hopkins University Press, second edition, 1989.
- [5] I. GUSTAFFSON, *A class of first order factorization methods*. BIT v 18 (1978) pp 142–156.
- [6] Y. KUZNETSOV, *New algorithms for approximate realization of implicit difference schemes*. Sov. J. Numer. Anal. Math. Modelling v 3 n^o 2 (1988).
- [7] G. MEURANT, *A review on the inverse of symmetric tridiagonal and block tridiagonal matrices*. submitted to SIMAX, (1990).
- [8] G. MEURANT, *A domain decomposition method for parabolic equations*. to appear in Applied Numerical Mathematics (1990).
- [9] G. RADICATI di BROZOLO & Y. ROBERT, *Parallel conjugate gradient-like algorithms for solving sparse non symmetric linear systems on a vector multiprocessor*. Parallel Computing 11 (1989) pp 223–239.