ParIC : A Family of Parallel Incomplete Cholesky Preconditioners

Mardochée Magolu monga Made[1]* and Henk A. van der Vorst[2]

[1] Université Libre de Bruxelles
Service des Milieux Continus, CP 194/5
50, avenue F.D. Roosevelt, B-1050 Brussels, Belgium
magolu@ulb.ac.be
http://homepages.ulb/~magolu/

[2] Utrecht University, Mathematical Institute, Mailbox 80.010,
3508 Utrecht, The Netherlands
vorst@math.uu.nl
http://www.math.uu.nl/people/vorst/

Abstract. A class of parallel incomplete factorization preconditionings for the solution of large linear systems is investigated. The approach may be regarded as a generalized domain decomposition method. Adjacent subdomains have to communicate during the setting up of the preconditioner, and during the application of the preconditioner. Overlap is not necessary to achieve high performance. Fill-in levels are considered in a global way. If necessary, the technique may be implemented as a global re-ordering of the unknowns. Experimental results are reported for two-dimensional problems.

1 Introduction

Krylov subspace based iterative methods are quite popular for solving large sparse preconditioned linear systems

$$B^{-1}Au = B^{-1}b \ , \tag{1}$$

where $Au = b$ denotes the original system, and B denotes a given preconditioning matrix (see, e.g., [2, 10]). The main operations within Krylov subspace methods are following:

1. sparse matrix–vector multiplication(s);
2. vector updates;
3. dot products;
4. setting up of the preconditioner;
5. application of the preconditioner: solve w from $Bw = r$, for given r.

* Research supported by the Commission of the European Community, under contract nr. 25009, within the ESPRIT IV project.

As in well-known, the handling of steps involving the preconditioner may be problematic on parallel platforms. In general, there is a trade-off between high level parallelism and fast convergence [8], especially when B is taken as an incomplete Choleky (IC) factorization of A [16, 17]. We shall not review all commonly used techniques for achieving parallelism. We refer to [6, 9] for such surveys. Till recently, most of works (if not all) on parallel *global* IC type methods concentrate on fill level zero preconditionings, for which the sparsity structure of A is preserved (see, e.g., [11, 12, 19]). We propose two techniques that allow high fill levels in a global preconditioner. A key feature of our approach is that adjacent subdomains have to exchange data during the computation of the preconditioning matrix factors, and during the application of the preconditioner. In contrast to classical domain decomposition methods, there is no overlap. A special treatment of interior boundary layers (interfaces) allows to alleviate the degradation of the convergence rate. In each situation, there exists an implicit global (re-)ordering of the unknowns. Experimental results are reported for two-dimensional problems, on a 16-processor SGI Origin 2000, showing that our methods compare favourably with classical overlapping domain decomposition methods.

2 Background

We consider the following self-adjoint second order two-dimensional elliptic PDE

$$\begin{aligned} -(p\,u_x)_x - (q\,u_y)_y &= f(x,y) && \text{in } \Omega = (0,1)\times(0,1) \\ u &= 0 && \text{on } \Gamma \\ u_n &= 0 && \text{on } \partial\Omega\backslash\Gamma \ . \end{aligned} \tag{2}$$

Γ denotes a nonempty part of the boundary $\partial\Omega$ of Ω. The coefficients p and q are positive, bounded and piecewise constant. We discretize (2) over a uniform rectangular grid of mesh size h in both directions with the five-point box integration scheme [18]. The lexicographical ordering in the (x, y)-plane is used to number the unknowns. The resulting system matrix A is a nonsingular block-tridiagonal, irreducibly diagonally dominant, Stieltjes (that is, symmetric positive definite and none of its offdiagonal entries is positive [22]) matrix. A popular method for solving such a system is the preconditioned conjugate gradient (PCG) method, combined with an incomplete factorization as preconditioning technique (see, e.g., [1, 2, 10]). Fig. 1 shows an incomplete LPL^t factorization algorithm, where

$$\mathcal{D} = \{\, (k,i) \mid lev(l_{k,i}) > \ell \,\}$$

stands for the set of discarded fill-in entries. The integer ℓ denotes a user specified maximal fill level. With respect to the notation of Fig. 1, $lev(l_{k,i})$ is defined as following:

<u>Initialization:</u> $lev(l_{k,i}) := \begin{cases} 0 & \text{if } l_{k,i} \neq 0 \text{ or } k = i \\ \infty & \text{otherwise} \end{cases}$

<u>Factorization:</u> $lev(l_{k,i}) := \min \{\, lev(l_{k,i})\, ,\, lev(l_{i,j}) + lev(l_{k,j}) + 1 \,\}$.

Compute P and L $(B = LPL^t \quad \text{with} \quad diag(L) = I)$

Initialization phase

$p_{i,i} := a_{i,i} \;, \quad i = 1, 2, \cdots, n$

$l_{i,j} := a_{i,j} \;, \quad i = 2, 3, \cdots, n \;, \qquad j = 1, 2, \cdots, i-1$

Incomplete factorization process

```
do j = 1, 2, ..., n-1
      compute parameter ρ_j
      do i = j+1, j+2, ..., n
            l_{i,i} := l_{i,i} - l_{i,j}^2 / l_{j,j}
            l_{i,j} := l_{i,j} / l_{j,j}
            do k = i+1, i+2, ..., n
                  if  (k,i) ∉ D   l_{k,i} := l_{k,i} - l_{i,j} l_{k,j}
            end do
      end do
end do
```

Fig. 1. Standard incomplete factorization (IC).

Any gridpoint j that is connected, with respect to the graph of L, with two gridpoints i and k such that $j < i < k$ (say, $l_{i,j} \neq 0$ and $l_{k,j} \neq 0$) gives rise to a fill-in entry (or a correction) in position (k, i) of L.

3 Parallel Incomplete Cholesky Preconditioners

3.1 Explicit Pseudo-Overlap

For simplicity we assume that the grid is divided into p stripes, as illustrated on Fig. 2. We impose the following conditions:

(c1) for each subdomain the computation starts at the bottom layer gridpoints, and finishes at the top layer gridpoints. The actual computations start from two sides: for the subdomains in the upper side of the physical domain, the flow of computations is downward;

(c2) immediately after the computations at the bottom layer gridpoints of a subdomain $(\mathcal{P}_j, j \notin \{0, p-1\})$ have been completed, the relevant corrections for

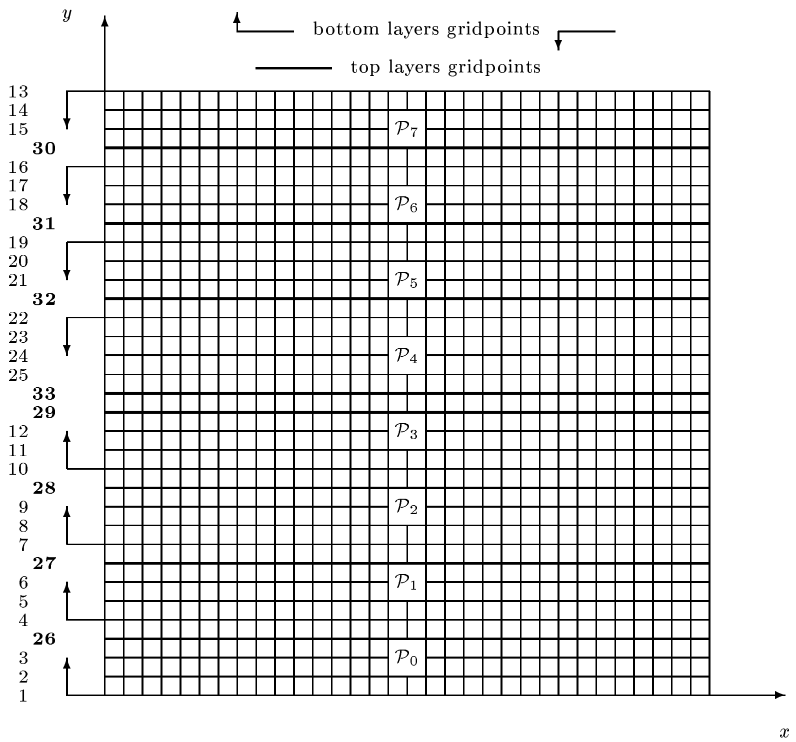

Fig. 2. Decomposition of the grid into stripes,and assignment of subdomains to processors, for $n_x = 32$, $n_y = 33$ and $p = 8$. Arrows indicate the progressing direction of the line numbering per subdomain. Numbers along the y-axis give an example of global (line) ordering, which satisfy all the required conditions. Within each horizontal line, gridpoints are ordered lexicographically.

the top layer gridpoints of the (appropriate) adjacent subdomain are packed and sent (preferably by means of some nonblocking communication);

(c3) the numbering decreases or increases in the same way for neighbouring points, for the bottom layer gridpoints and the top layer gridpoints of adjacent subdomains. This facilitates the implementation (communication). Each top layer gridpoint has "to know" where corrections come from.

Definition 1. *Given that communication only involves the gridpoints in the bottom and top layer, the union of adjacent layers is referred to as the pseudo-overlapping region (or simply the pseudo-overlap). Equivalently, if $\mathcal{P}_r$ has to send data to $\mathcal{P}_s$ during the incomplete factorization process, we will say that $\mathcal{P}_s$ is pseudo-overlapped by $\mathcal{P}_r$.*

The rate of convergence of a parallel IC(0), executed under the conditions as described above, will degrade as the number of subdomains increases ($p > 2$

for stripes type partitionings [3, 15], and $p > 4$ for more general partitionings [8, 23, 24]; see also [11, 12, 19]). In order to explain why this occurs, we make, in Fig. 3, a zoom of an interface between two adjacent subdomains. We use a stencil graph notation [13]: a diagonal entry $a_{i,i}$ is represented by circle number i; the edge $\{i, j\}$ (here horizontal and vertical lines) corresponds to a nonzero offdiagonal entry $a_{i,j}$. Oblique line represent (rejected) level-1 fill-in entries that are not significantly different from the case of $p = 1$. Thick lines (the arcs) are the (neglected) level-1 fill-in entries that would not arise if a global *natural* ordering (or some other equivalent ordering) were used. Such level-1 fill-in entries are responsible for the deterioration of the convergence. For IC(0), rejected level-1 fill entries determine the remainder matrix $R = B - A$. In the terminology of Doi and Lichnewsky [4, 5], the bottom layer gridpoints of $\mathcal{P}_{s+1}$ (see gridpoints marked with $\star$ in Fig. 3), which induce the additional level-1 fill entries, are called *incompatible nodes*.

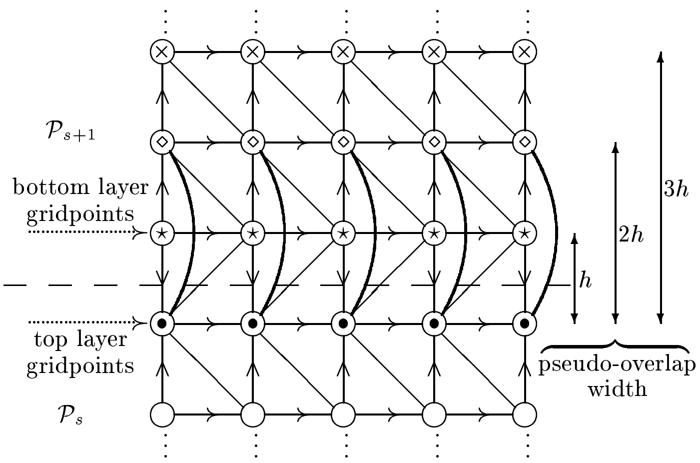

Fig. 3. Part of graph of matrix A assigned to two adjacent subdomains ($\mathcal{P}_s$ and $\mathcal{P}_{s+1}$). Oblique lines and thick lines are (rejected) level 1 fill-in entries.

A way to improve the performance consists in accepting *enough* fill-in entries generated by the parallelization strategy: increase both the pseudo-overlap width ($\varpi = h, 2h, 3h, \ldots$) and the fill level ($\ell_\varpi \geq 0, 1, 2, \ldots$) in the pseudo-overlapping region(s).

Definition 2. *We denote by* $\mathrm{ParIC}(\ell; \varpi, \ell_\varpi)$ *any standard* $\mathrm{IC}(\ell)$ *combined with the parallelization strategy as described above. This reads parallel* IC *with interior fill level* ℓ*, pseudo-overlap width* ϖ*, and pseudo-overlap fill level* ℓ_ϖ*. In the specification of* ϖ*,* k *will stand for* kh*, in order to include variable mesh size problems and (graphs of) matrices that do not stem from discretized PDEs.*

3.2 Implicit Pseudo-Overlap

The parallelization technique discussed so far may be rather tedious to apply, when some subdomains have a high number of neighbours and the grid is not well structured. The alternative method, with an ordering induced pseudo-overlapping strategy, that we shall describe now may be easily used to tackle intricate geometries and partitionings.

For the purposes of our exposition, let us think of each small (grid) square of Fig. 2 as a finite element, and assume that the finite elements have been partitioned into p subdomains by means of some automatic mesh partitioning algorithm [7]. Then, *in each subdomain*, the (local) unknowns are re-numbered class by class, consecutively, as following:

1. class 1: all interior gridpoints are numbered (firstly);
2. class 2: next follow all gridpoints, if any, that belong to two subdomains;
3. class 3: next follow all gridpoints, if any, that belong to three subdomains;
4. etc ...

In doing so, we obtain a (generalization of a) reverse variant of an ordering discussed in [11] (see also [19]). A global renumbering of the gridpoints may be achieved in a similar way. The computation and the exchange of data is performed, class by class, as follows:

1. compute class 1 gridpoints;
2. exchange data for class 2 gridpoints updates; compute class 2 gridpoints;
3. exchange data for class 3 gridpoints updates; compute class 3 gridpoints;
4. exchange data for class 4 gridpoints updates; etc ...

Any step that involves an empty class must be skipped. The computation and the exchange of data should be organized in such a way that, at each gridpoint shared by two or more subdomains, each subdomain involved obtains the same value, up to round-off errors, during the incomplete factorization process, and during the preconditioning steps. This requires to drop any connection between two gridpoints of the same class, but which belong to two different interfaces. An illustration is provided in Fig. 4 where we give a partitioning of the physical domain into 2×4 boxes. In the case where the connection to be dropped corresponds to some entry $a_{i,j}$ of the original system matrix, the dropped value may be added to the diagonal entries $a_{i,i}$ and $a_{j,j}$. This technique, which preserves the rowsum of a matrix, is known as *diagonal compensation* [1].

Definition 3. *Now the pseudo-overlap will be implicitly determined by the local numberings of unknowns, and the fill level taken inside each subdomain. We shall denote this more general parallel IC simply by ParIC(ℓ). It can be easily extended to include the case of subdomains with different fill levels.*

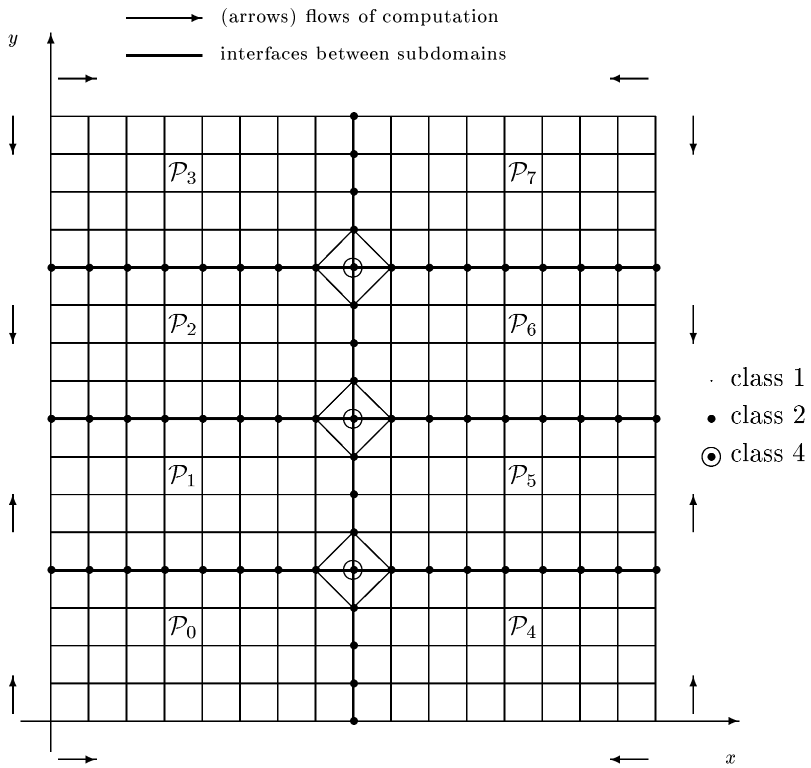

Fig. 4. Decomposition of the grid into 2×4 subdomains, assignment of subdomains to processors, and partitioning into classes. Oblique lines correspond to forbidden level-1 fill-in entries.

4 Numerical Experiments

The zero vector is used as initial guess, and the PCG is stopped as soon as the residual vector r satisfies $\|r\|_2 \,/\, \|b\|_2 \leq 10^{-6}$. The test is performed only once $\|B^{-1}r\|_2 \,/\, \|B^{-1}b\|_2 \leq 10^{-6}$ is satisfied. The computations are carried out in double precision Fortran on a 16-processor SGI Origin 2000 (195 MHz), using the MPI library for communications. The preconditionings include: ParIC($\ell; \varpi, \ell_\varpi$), $\ell_\varpi \geq \varpi - 1$, with the stripes (or $1 \times p$) partitionings; ParIC(ℓ) with $2 \times p$ partitionings; and the additive Schwarz with overlap AS($\ell; \varpi$), each local problem is handled with one IC(ℓ) solve, ϖ stands here for the actual overlap width. We use $\varpi = h_0,\ h,\ 2h$, where h_0 means that only one line of nodes is shared by adjacent subdomains. No global coarse grid correction has been added, as suggested in [20, 21]), to improve the performance of the preconditionings involved.

Problem 1. $p = q = 1$, $\Gamma = \Omega$, $u(x, y) = x(x-1)y(y-1)e^{xy}$, and $h = 1/(n_y + 1)$.

Problem 2. $\Gamma = \{(x,y);\ 0 \le x \le 1,\ y = 0\}$, $h = 1/n_y$,

$$p = q = \begin{cases} 100 \text{ in } (1/4,3/4)\times(1/4,3/4) \\ 1 \quad \text{elsewhere} \ , \end{cases} \qquad f(x,y) = \begin{cases} 100 \text{ in } (1/4,3/4)\times(1/4,3/4) \\ 0 \quad \text{elsewhere} \ . \end{cases}$$

We first collect in Figs. 5 and 6, and in Table 1, the results of our numerical experiments performed with the stripes partitionings. We use the parallel speed-up, which is the ratio between the execution time of the parallel algorithm on one processor and the time on p processors.

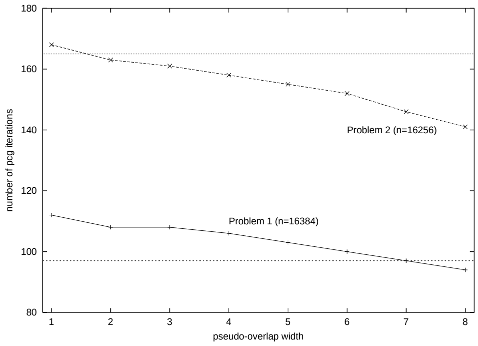

Fig. 5. Effects of pseudo-overlap width ϖ on the number of pcg iterations, for 1×8 processors and ParIC$(0; \varpi, \varpi - 1)$. Horizontal lines display the number of pcg iterations for sequential IC(0).

Table 1. Speed-up for stripes partitionings ($1 \times p$).

	Problem 1				Problem 2			
Numb. of procs.	2	4	8	16	2	4	8	16
Precond.								
ParIC(0;1,0)	2.04	4.57	10.66	19.94	1.98	4.73	10.99	19.66
ParIC(4;5,4)	1.88	4.15	9.24	17.96	1.76	3.86	8.37	15.92
AS(4,h_0)	1.32	3.08	7.19	13.12	1.30	2.96	6.46	12.21
AS(4,h)	1.56	3.56	7.38	14.99	1.32	3.11	6.53	11.96
AS(4,2h)	1.58	3.56	7.29	15.46	1.35	3.08	5.78	12.26

From all the observed results, the following trends are evident.

1. It is advantageous to use increased pseudo-overlap. In particular, for difficult (large size) problems (see Fig. 5), the degradation of the performance is

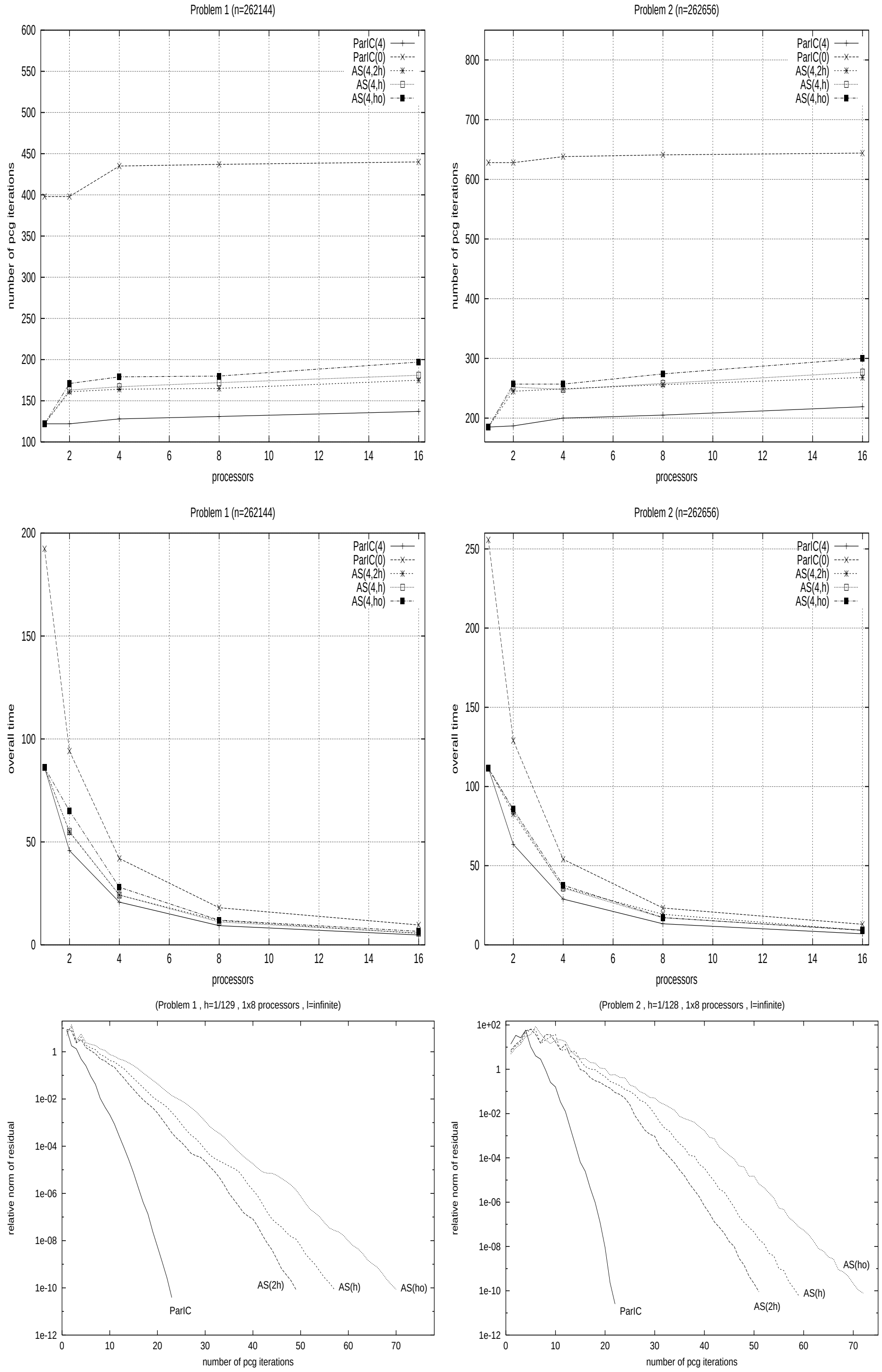

Fig. 6. Number of PCG iterations and overall computational time for ParIC(0;1,0), ParIC(4;5,4), AS(4,h_0), AS(4,h), AS(4,2h), and $1 \times p$ partitionins (stripes). Evolution of the relative residual error for 1×8 processors; the fill-in level $\ell = \infty$ (locally) for each preconditioner involved

(more than) mastered when one accepts some fill-in entries generated by the parallelization strategy.

2. ParIC$(4;5,4)$ is in general twice as fast as ParIC$(0;1,0)$. For both methods, the speed-up is high, and in general better than for AS methods.
3. AS methods must be applied with a sufficiently large overlap width, in order to achieve performance comparable to ParIC methods, which dramatically increases the computational complexity. This holds even if each local problem is solved exactly. Observe that, for $p = 2$, ParIC$(\infty;\varpi_{\max},\infty)$, which is equivalent to ParIC(∞), becomes a direct solver, whereas AS remains an iterative one.

Table 2 shows the performance of ParIC(4) combined with various partitionings. For stripes (or $1 \times p$) partitionings, ParIC(4) is mathematically equivalent to ParIC(4;5,4). It appears that, for difficult problems, it would be interesting to use partitionings better than the simple stripes ones.

Table 2. Problem 1, $h^{-1} = 513$, $n = 262144$. Problem 2, $h^{-1} = 512$, $n = 262656$. Comparison of various partitionings (part). Number of PCG iterations (iter); elapsed time in seconds for: the computation of the preconditioning matrix (fact), the solver, and overall time; for ParIC(4).

	Problem 1	Time			**Problem 2**	Time		
part	iter	fact	pcg	overall	iter	fact	pcg	overall
1	122	4.76	80.59	86.20	185	4.84	106.17	111.61
1×2	122	2.48	42.84	45.82	187	2.48	60.60	63.45
2×1	122	2.44	42.97	45.88	150	2.34	47.54	50.14
1×4	128	1.24	19.30	20.78	200	1.22	27.48	28.88
2×2	127	1.42	19.68	21.32	159	1.44	23.05	24.61
1×8	131	0.65	8.50	9.32	205	0.62	12.60	13.33
2×4	135	0.74	9.23	10.09	167	0.74	10.59	11.40
1×16	137	0.37	4.26	4.80	219	0.33	6.49	7.01
2×8	137	0.41	4.54	5.18	172	0.40	5.11	5.61

References

1. O. Axelsson: *Iterative Solution Methods* (Cambridge University Press, Cambridge, 1994).
2. R.F. Barret, M. Berry, T.F. Chan, J. Demmel, J. Donato, J.J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. van der Vorst: *Templates for the Solution of Linear Systems : Building Blocks for Iterative Methods* (SIAM, Philadelphia, 1994).
3. R. Beauwens, L. Dujacquier, S. Hitimana and M. Magolu monga Made: MILU factorizations for 2-processor orderings. In: I.T. Dimov, Bl. Sendov and P.S. Vassilevski (eds.), Advances in Numerical Methods and Applications $\mathcal{O}(h^3)$ (World Scientific, Singapore, 1994) 26–34.
4. S. Doi and A. Lichnewsky: A graph-theory approach for analyzing the effects of ordering on ILU preconditioning. INRIA report 1452, INRIA-Rocquencourt, France, 1991.

5. S. Doi and T. Washio: Ordering strategies and related techniques to overcome the trade-off between parallelism and convergence in incomplete factorizations. Parallel Comput. **25**, (1999) 1995–2014.
6. J.J. Dongarra, I.S. Duff, D.C. Sorensen and H.A. van der Vorst: *Numerical Linear Algebra for High-Performance Computers* (SIAM, Philadelphia, 1998).
7. R. van Driessche: Algorithms for static and dynamic load balancing on parallel computers. Ph.D. thesis, Dept of Computer Science, Katholieke Universiteit Leuven, Belgium, 1995.
8. I.S. Duff and G.A. Meurant: The effect of ordering on preconditioned conjugate gradients. BIT **29**, (1989) 635–657.
9. I.S. Duff and H.A. van der Vorst: Developments and Trends in the Parallel Solution of Linear Systems. Parallel Comput. **25**, (1999) 1931–1970.
10. G.H. Golub and C.F. van Loan: *Matrix Computations* (third ed.) (The John Hopkins University Press, Baltimore, Maryland, 1996).
11. G. Haase: Parallel incomplete Cholesky preconditioners based on the nonoverlapping data distribution. Parallel Comput. **24**, (1998) 1685–1703.
12. M. Magolu monga Made: Implementation of parallel block preconditionings on a transputer-based multiprocessor. Future Generation Computer Systems **11**, (1995) 167–173.
13. M. Magolu monga Made: Taking advantage of the potentialities of dynamically modified block incomplete factorizations. SIAM J. Sci. Comput. **19**, (1998) 1083–1108.
14. M. Magolu monga Made and B. Polman: Efficient planewise like preconditioners to cope with 3D problems. Numer. Linear Algebra Appl. **6**, (1999) 379–406.
15. M. Magolu monga Made and H.A. van der Vorst: Parallel incomplete factorizations with pseudo-overlapped subdomains. Technical Report, Service des Milieux Continus, Université Libre de Bruxelles, February 2000.
16. J.A. Meijerink and H.A. van der Vorst: An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix. Math. Comp. **31**, (1977) 148–162.
17. J.A. Meijerink and H.A. van der Vorst: Guidelines for the usage of incomplete decompositios in solving sets of linear equations as they occur in practical problems. J. Comp. Physics **44**, (1981) 134–155.
18. S. Nakamura, *Computational Methods in Engineering and Science* (J. Wiley & Sons, New York, 1977).
19. Y. Notay: An efficient Parallel Discrete PDE Solver. Parallel Comput. **21**, (1995) 1725–1748.
20. Y. Notay and A. Van de Velde: Coarse grid acceleration of parallel incomplete preconditioners. In: S. Margenov and P. Vassilevski, (eds.), Iterative Methods in Linear Algebra II. IMACS series in Computational and Applied Mathematics **3**, (1996) 106–130.
21. B.F. Smith, P.E. Bjorstad and D. Gropp: *Domain Decomposition : Parallel Multilevel Methods for Elliptic Partial Differential Equations* (Cambridge University Press, Cambridge, 1996).
22. R.S. Varga, *Matrix Iterative Analysis* (Prentice Hall, Englewood Cliffs, 1962).
23. H.A. van der Vorst: Large tridiagonal and block tridiagonal linear systems on vector and parallel computers. Parallel Comput. **5**, (1987) 54–54.
24. H.A. van der Vorst: High performance preconditioning. SIAM J. Sci. Statist. Comput. **10**, (1989) 1174–1185.