Parallel asynchronous Richardson method for the solution of obstacle problem

Pierre Spitéri
ENSEEIHT-IRIT-LIMA
2 rue Charles Camichel
31071 Toulouse Cédex, France
Pierre.Spiteri@enseeiht.fr

Ming Chau
ENSEEIHT-IRIT-LIMA
2 rue Charles Camichel
31071 Toulouse Cédex, France
Ming.Chau@enseeiht.fr

Abstract

The present study deals with parallel asynchronous iterations applied to the numerical solution of the obstacle problem defined in a three-dimensionnal domain. For the considered problem, the convergence analysis of the algorithm is made. Finally, the implementation of the algorithms are presented and computational experiments on IBM-SP3 are analysed.

Keywords: parallel iterative algorithms, asynchronous iterations, obstacle problem

1. Introduction

The obstacle problem occurs in many applications such as mechanics and financial derivative. In the stationnary case, the obstacle problem can be formulated as follows:

$$\begin{cases} \text{Find } u^* \text{ such that} \\ A.u^* - f \geq 0, u^* \geq \phi \text{ everywhere in } \Omega \\ (A.u^* - f)(\phi - u^*) = 0 \text{ everywhere in } \Omega \\ \text{B.C.} \end{cases}$$

where $\Omega \subset \mathbb{R}^2$ (or $\mathbb{R}^3$) is an opened set, A is an elliptic operator and ϕ a given function. B.C. describes the boundary conditions on $\partial\Omega$.

In literature, there are many equivalent formulations of the obstacle problem and the reader is referred to [6] for complements. Indeed, such problem can be equivalently formulated as a complementary problem

$$\begin{cases} \sup(A.u^* - f, \phi - u^*) = 0 \\ \text{B.C.} \end{cases}$$

and also as a variationnal inequality

$$\begin{cases} \text{Find } u^* \in K \text{ such that} \\ \forall v \in K, \langle A.u^* , v - u^* \rangle \geq \langle f , v - u^* \rangle \end{cases}$$

where K is a closed convex set defined by

$$K = \{v \mid v \geq \phi \text{ everywhere in } \Omega\}$$

and $\langle . , . \rangle$ denotes the dot product $\langle u , v \rangle = \int_\Omega u\, v\, dx$. The obstacle problem can also be formulated as a constrained optimization problem where the cost function is

$$J(v) = \frac{1}{2}\langle A.v , v \rangle - \langle f , v \rangle$$

Then it is proved (see [6]) that the solution u^* of the variationnal inequality is the solution of the constrained optimization problem

$$\begin{cases} \text{Find } u^* \in K \text{ such that} \\ \forall v \in K, J(u^*) \leq J(v) \end{cases}$$

This optimization problem can be solved using a projected relaxation method on the convex set. The numerical solution of the considered problem leads to the solution of large scale algebraic systems. Owing to the great size of such system, parallel numerical algorithms may be used in order to reduce computation time. We will consider in the sequel an asynchronous parallel adaptation of the Richardson projected method.

The goal of the present study is to analyse the parallel asynchronous iterative method applied to a test problem. Asynchronous algorithms reduce idle time due to synchronizations between the processors. In the present study, the method considered is derived from Richardson's projected method, wich corresponds to a fixed point mapping. The convergence is studied by contraction techniques in a theoretical framework well adapted to distributed computation. Implementation is studied on a distributed memory multiprocessor. Communications are managed with M.P.I. Asynchronous and synchronous versions of the parallel algorithms are compared; their effciency is analysed.

In section 2, an asynchronous projected Richardson method is presented and its convergence analysed. In section 3, the implementation of the algorithms and experimental results are presented.

Proceedings of the 16th Annual International Symposium on High Performance Computing Systems and Applications (HPCS'02)
0-7695-1626-2/02 $17.00 © 2002 IEEE

2. Convergence analysis

2.1. Preliminaries and background

Let α be a positive integer. Assume that V_i is a Hilbert space where $\langle\,.\,,\,.\,\rangle_i$ denotes the scalar product and $|\,.\,|_i$ the associated norm, for all $i \in \{1,\ldots,\alpha\}$.

Let $V = \prod_{i=1}^{\alpha} V_i$ and for all $u, v \in V$ denote by $\langle u\,,v\rangle = \sum_{i=1}^{\alpha}\langle u_i\,,v_i\rangle_i$ the scalar product on V and $\|\,.\,\|$ its associated norm.

In the sequel, we consider the following fixed point problem

$$\left\{\begin{array}{l}\text{Find } u^* \in V \text{ such that}\\ u^* = F(u^*)\end{array}\right. \tag{1}$$

where $v \mapsto F(v)$ applies from V to V. Let $v \in V$ and consider the following block decomposition of v and the corresponding decomposition of F:

$$\begin{array}{rcl} v & = & (v_1,\ldots,v_\alpha)\\ F(v) & = & (F_1(v),\ldots,F_\alpha(v))\end{array}$$

In order to solve problem (1), let us consider now the parallel asynchronous iterations (see [1] [2] [3], [8]) defined as follows. Let $u^0 \in V$ be given, and for $p \in \mathbb{N}$ assume that we could get $u^1,\ldots,u^p$; then u^{p+1} is defined by

$$\left\{\begin{array}{lcl} u_i^{p+1} & = & F_i(u_1^{\rho_1(p)},\ldots,u_j^{\rho_j(p)},\ldots,u_\alpha^{\rho_\alpha(p)})\\ & & \text{if } i \in s(p)\\ u_i^{p+1} & = & u_i^p\\ & & \text{if } i \notin s(p)\end{array}\right. \tag{2}$$

where

$$\left\{\begin{array}{l}\forall p \in \mathbb{N}, s(p) \subset \{1,\ldots,\alpha\} \text{ and } s(p) \neq \emptyset\\ \forall i \in \{1,\ldots,\alpha\}, \{p \mid i \in s(p)\} \text{ is denombrable}\end{array}\right. \tag{3}$$

and $\forall j \in \{1,\ldots,\alpha\}$,

$$\left\{\begin{array}{l}\forall p \in \mathbb{N}, \rho_j(p) \in \mathbb{N} \text{ and } 0 \leq \rho_j(p) \leq p\\ \lim_{p\to\infty}\rho_j(p) = +\infty\end{array}\right. \tag{4}$$

Remark: Such asynchronous iterations describe various classes of parallel algorithms, such as parallel synchronous iterations if $\forall j \in \{1,\ldots,\alpha\}$, $\forall p \in \mathbb{N}, \rho_j(p) = p$. In the synchronous context, for particular choice of $s(p)$, then (2) and (3) describe sequential algorithms, among them, a successive approximation method, Gauss-Seidel method, etc ... (see [8]).

2.2. Asynchronous parallel Richardson method

Let A a continuous and linear operator from V to V, such that $A.v = (A_1.v,\ldots,A_\alpha.v)$ Consider the following assumption:

$$\forall i \in \{1,\ldots,\alpha\}, \forall v \in V, \langle A_i.v\,,v_i\rangle_i \geq \sum_{j=1}^{\alpha} n_{i,j}|v_i|_i|v_j|_j \tag{5}$$

where

$$N = (n_{i,j})_{1\leq i,j\leq\alpha} \text{ is an } \alpha\times\alpha \text{ M-matrix} \tag{6}$$

The reader is referred to [11] for the definition of M-matrix.
Remark: In the classical formulation of the Richarson algorithm, it is assumed that

$$\forall v \in V, \langle A.v\,,v\rangle \geq c\,\|v\|^2 \tag{7}$$

where c is a positive real number. The assumption (5) is actually weaker than (7). In addition, (5) takes into account the decomposition needed by parallel computation.

Moreover, assume that $\forall i \in \{1,\ldots,\alpha\}, K_i \subset V_i$ and K_i is a closed convex, and let $K = \prod_{i=1}^{\alpha} K_i$. Let also $b = (b_1,\ldots,b_\alpha) \in V$ and consider the following variational inequality

$$\left\{\begin{array}{l}\text{Find } u^* \in K \text{ such that}\\ \forall v \in K, \langle A.u^*\,,v-u^*\rangle \geq \langle b\,,v-u^*\rangle\end{array}\right. \tag{8}$$

For any $v \in V$, let $P_K(v)$ be the projection of v on K such that $P_K(v) = (P_{K_1}(v_1),\ldots,P_{K_\alpha}(v_\alpha))$ where $\forall i \in \{1,\ldots,\alpha\}, P_{K_i}$ is the projector from V_i onto K_i.

For any $\delta \in \mathbb{R}, \delta > 0$, let a fixed point mapping F_δ be defined by

$$\forall v \in V, F_\delta(v) = P_K(v - \delta(A.v - b)) \tag{9}$$

which can also be written $F_\delta(v) = (F_{1,\delta}(v),\ldots,F_{\alpha,\delta}(v))$ in the following way

$$\forall v \in V, F_{i,\delta}(v) = P_{K_i}(v_i - \delta(A_i.v - b_i))$$

Proposition 1 *Assume that (5) and (6) hold. Then there exists a value $\delta_0 > 0$, such that $\forall\delta \in]0,\delta_0[$, the asynchonous iterations defined by (2), (3) and (4), applied to F_δ defined by (9), are convergent, the limit being u^*, the unique solution of problem (8).*

Proof: First of all, note that the projector is a contracting operator. Thus, the problem is to find the set of real numbers δ such that the mapping F_δ is contracting. According to the result of El-Tarazi [4], assume that the space V is normed by

$$\|v\|_\gamma = \max_{1\leq i\leq\alpha}\frac{|v_i|_i}{\gamma_i}$$

where $\gamma = (\gamma_1,\ldots,\gamma_\alpha) \in \mathbb{R}^\alpha$ is a real strictly positive vector correspondig to the eigen vector associated with the spectral radius of the Jacobi matrix J of N; note that, according to the Perron-Frobenius theorem [11], there exists

such a vector γ, because N is an M-matrix and hence J is a non negative matrix. So

$$J.\gamma = \rho(J)\gamma \text{ and } \gamma > 0 \tag{10}$$

Let $v \in K$ and $w = v - u^*$, such that, according to the previous notations, $w_i = v_i - u_i^*$; let

$$\begin{aligned} B_i &= \frac{1}{n_{i,i}} \frac{|w_i - \delta A_i.w|_i^2}{\gamma_i^2} \\ &= \frac{1}{n_{i,i}} \frac{|w_i|_i^2 - 2\delta \langle A_i.w\,, w_i\rangle + \delta^2 |A_i.w|_i^2}{\gamma_i^2} \end{aligned}$$

Since $A_i \in \mathcal{L}(V, V_i)$, $\exists M_i \geq 0$, $\forall w \in V$, $|A_i.w|_i \leq M_i \|w\|$. Moreover, according to (5) the above relation can be over-estimated by

$$B_i \leq \frac{1}{n_{i,i}} \frac{|w_i|_i^2 - 2\delta \sum_{j=1}^{\alpha} n_{i,j} |w_i|_i |w_j|_j + \delta^2 M^2 \|w\|^2}{\gamma_i^2}$$

where $M = \max_{1 \leq i \leq \alpha} M_i$. Then

$$\begin{aligned} B_i &\leq \left(\frac{1}{n_{i,i}} - 2\delta \right) \frac{|w_i|_i^2}{\gamma_i^2} \\ &- 2\delta \left(\sum_{j \neq i} \frac{n_{i,j}}{n_{i,i}} \gamma_j \frac{|w_j|_j}{\gamma_j} \right) \frac{|w_i|_i}{\gamma_i^2} \\ &+ \frac{\delta^2 M^2}{n_{i,i}} \frac{\|w\|^2}{\gamma_i^2} \end{aligned}$$

So according to (10),

$$\begin{aligned} B_i &\leq \left(\frac{1 - 2\delta\, n_{i,i}}{n_{i,i}} \right) \frac{|w_i|_i^2}{\gamma_i^2} \\ &+ 2\delta\, \rho(J) \left(\max_{1 \leq j \leq \alpha} \frac{|w_j|_j}{\gamma_j} \right) \frac{|w_i|_i}{\gamma_i} \\ &+ \frac{\delta^2 M^2}{n_{i,i}} \frac{\|w\|^2}{\gamma_i^2} \end{aligned}$$

Then

$$B_i \leq \frac{1}{n_{i,i}} \left((1 - 2\delta\, n_{i,i}(1 - \rho(J)))\, \|w\|_\gamma^2 + \frac{\delta^2 M^2}{\gamma_i^2} \|w\|^2 \right)$$

Let

$$\underline{\gamma} = \min_{1 \leq i \leq \alpha} \gamma_i, \text{ and } \underline{n} = \min_{1 \leq i \leq \alpha} n_{i,i}$$

As $\|w\|^2 = \sum_{i=1}^{\alpha} \gamma_i^2 \frac{|w_i|_i^2}{\gamma_i^2} \leq |\gamma|_2^2 \|w\|_\gamma^2$, where $|\,.\,|_2$ denotes the euclidian norm in $\mathbb{R}^\alpha$, we finally have

$$\|w - \delta A.w\|_\gamma^2 \leq \left(1 - 2\underline{n}(1 - \rho(J))\delta + M^2 \frac{|\gamma|_2^2}{\underline{\gamma}^2} \delta^2 \right) \|w\|_\gamma^2$$

Then, δ being a positive real number, the considered algorithm converges if $\delta \in]0, \delta_0[$, where

$$\delta_0 = \frac{2\underline{n}(1 - \rho(J))\underline{\gamma}^2}{M^2 |\gamma|_2^2}$$

3. Experimental results

3.1. Test problem

The experiments have been carried out when the operator A is the Laplacian operator defined in $\Omega = [0,1]^3$. The problem has Dirichlet boundary conditions (i.e. $u_{|\partial\Omega} = 0$). Using a finite difference method gives rise to the matrix in the discrete system.

The block decomposition of the iterate v is derived from a decomposition of the domain Ω into slices. Such decomposition simplifies and minimizes the communications between the processors. So, if n^3 is the number of discretization points, the iterate vector is decomposed into n blocks of n^2 points.

In such a case, the decomposition of the discrete Laplacian operator can be considered as a block-tridiagonal one. Let I be the $n^2 \times n^2$ identity matrix, and let $h = \frac{1}{n+1}$. Then, $\forall i \in \{1, \ldots, \alpha\}$, $A_{i,i\pm1} = -\frac{1}{h^2} I$, and $A_{i,i} = \frac{2}{h^2} I + B$, where B denotes the discrete Laplacian operator defined in $[0,1]^2$. As B is a definite positive matrix, (5) and (6) are obtained with:

$$N = \frac{1}{h^2} \begin{pmatrix} 2 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{pmatrix}$$

which is known to be an M-matrix (see [11]). Then, convergence is derived from proposition 1. When dealing with large linear systems, the n blocks are distributed to N processors ($N < n$). On each processor, blocks are computed sequentially.

Remark: Convergence of the test problem could also be established in a more general way. Considering a point decomposition of problem (8) and according to [5], assumptions (5) and (6) are verified. Thus, the parallel asynchronous projected Richardson method converges for a point decomposition. Then, according to results of [9], convergence is still ensured for any coarser decomposition.

Now, we can give a basic parallel algorithm that does not deal with the asynchronism yet. Let $k \in \{1, \ldots, N\}$. Then the k-th processor is to compute the blocks $U_{l(k)}, U_{l(k)+1}, U_{l(k)+2}, \ldots, U_{q(k)}$. Namely, $U_{l(k)}$ stands for the first block on the k-th processor and $U_{q(k)}$ stands for the last block on the k-th processor. The algorithm is shown in figure 1. The cases $k = 1$ and $k = N$ are not taken in account.

```
send U_{q(k)} to proc k+1                    [send 1]

do until convergence
   i ← l(k)
   receiv eU_{i-1} from proc k-1             [recv 1]
   U_i ← F_{i,δ}(U_{i-1}, U_i, U_{i+1})
   send U_i to proc k-1                      [send 2]

   do i = l(k)+1, q(k)-1
      U_i ← F_{i,δ}(U_{i-1}, U_i, U_{i+1})
   end

   i ← q(k)
   receiv eU_{i+1} from proc k+1             [recv 2]
   U_i ← F_{i,δ}(U_{i-1}, U_i, U_{i+1})
   send U_i to proc k+1                      [send 1]
end
```

Figure 1. The basic parallel algorithm

3.2. Implementation

All block exchanges have been implemented with persistant communication requests. As large messages are exchanged, minimizing data copy is necessary. Therefore, synchronous mode sendings are used in both synchronous and asynchonous versions. All messages are explicitly packed into contiguous buffers before being sent. Requests are handled following this policy:

- Complete or test the requests just before reading or writing in the buffers
- Start the requests as soon as the use of the buffer has finished

The algorithms shown below (see figure 2 and 3) focus on the implementation of communication. The asynchronous algorithm is derived from the synchronous one by substituting non-blocking operations for blocking ones. In addition, the message receipt is implemented with a multiple completion operation in order to restart as many requests as possible.

The main trouble in asynchronous algorithms is the termination. The evaluation of the global state becomes difficult without synchronization nor message acknowledgement. The termination test that is currently used is still experimental and adapted from [2]:

- Processors must not send any message if a local convergence is reached.
- During the hole computation time, a token collects on all processors information about local convergence, counts the number of sent messages and completed receipts.

```
pack the last block
start sending the last block

start receiving both boundaries

do until convergence
   wait for the receipt of the first boundary
   unpack the first boundary
   start the receive operation

   compute the first block

   wait for the completion of the former sending
   pack the first block
   start the send operation

   compute the next blocks

   wait for the receipt of the second boundary
   unpack the last boundary
   start the receive operation

   compute the last block

   wait for the completion of the former sending
   pack the last block
   start the send operation
end
```

Figure 2. The synchronous implementation

A master processor decides that global convegence is reached when all processors have reached local convergence and all messages have been received. The token is still exchanged asynchronously. This test remains easy to implement since no complex data structures are handled.

At the end of the computation, the persistant requests are cancelled before gathering the blocks on the master processor.

3.3. Numerical experiments

Experiments have been carried out on an I.B.M. SP-3[1]. Each node of the calculator has 16 processors.

Computation of efficiency and speed-up are based on a sequential algorithm with no data packing or unpacking. In

[1]We would like to thank IDRIS (Institut du Développement et des Ressources en Informatique Scientifique) for computation hours allocated to our project.

```
pack the last block
start sending the last block

start receiving both boundaries

do until convergence
   test the completion of some receive operation
   unpack updated receive buffers
   start completed operations

   compute the first block

   test the completion of the former sending
   if completed
      pack the first block
      start the send operation
   end

   compute the next blocks

   test the completion of some receive operation
   unpack updated receive buffers
   start completed operations

   compute the last block

   test the completion of the former sending
   if completed
      pack the last block
      start the send operation
   end
end
```

Figure 3. The asynchronous implementation

the asynchronous version, the number of iterations is the mean of the relaxations that each processor has performed.

Efficiencies greater than 1 are due to cache defaults generated by large data in the sequential algorithm.

At first sight, asynchronous and synchronous versions have comparable speeds when fewer than 16 processors are involved. Asynchronous algorithm generally runs a faster than the synchonous one because it is not penalized by blocking communication operations. However, convergence is slower as "fresh data" is not necessarily available while using non-blocking communications. In other words, the additionnal iterations show that $\rho_j(p) < p$. Furthermore, as the termination test is performed with a token that shifts asynchronously, termination is detected later. These are the reasons why the asynchronous algorithm is sometimes slower.

When 32 processors are used, there is a significant differ-

Class	*procs*	*iterations*	*time (sec.)*
Sequential	1	5502	332
Synchronous	2	5502	148
	4	5502	68
	8	5502	36
	16	5502	21
Asynchronous	2	5722	149
	4	5624	63
	8	5693	33
	16	5764	18

Table 1. n=80 blocks

Class	*procs*	*speed-up*	*efficiency*
Synchronous	2	2.26	1.13
	4	4.88	1.22
	8	9.22	1.15
	16	15.81	0.99
Asynchronous	2	2.23	1.11
	4	5.67	1.32
	8	10.06	1.26
	16	18.44	1.15

Table 2. n=80 blocks

Class	*procs*	*iterations*	*time (sec.)*
Sequential	1	7662	708
Synchronous	2	7662	358
	4	7662	161
	8	7662	80
	16	7662	46
	32	7662	35
Asynchronous	2	7740	349
	4	8099	163
	8	7863	76
	16	7877	40
	32	8012	23

Table 3. n=96 blocks

Class	*procs*	*speed-up*	*efficiency*
Synchronous	2	1.98	0.99
	4	4.40	1.10
	8	8.85	1.11
	16	15.39	0.96
	32	20.23	0.63
Asynchronous	2	2.03	1.01
	4	4.34	1.09
	8	9.32	1.16
	16	17.70	1.11
	32	30.78	0.96

Table 4. n=96 blocks

ence between the two versions. The synchronous algorithm has suffered a loss of efficiency because:

- Each processor has only 3 blocks to deal with. As messages and blocks have the same size, the overhead due to communication reaches a significant proportion. The computation collapses.
- Some processors are involved in inter-node communication which are slower. On these processors, the overhead due to communication is increased because of longer idle time. This leads to load unbalance.

The asynchronous algorithm is less sensitive to communication issues than the synchronous one. A loss of efficiency can also be noticed when running on 32 processors but performance remains good.

We may conclude that asynchronism is well adapted to mass parallelism since no complex tuning is required to keep efficiency.

References

[1] BAUDET G., "Asynchronous iterative methods for multiprocessors", *Journal Assoc. Comput. Mach.,* vol. 25, pp. 226- 244, 1978.

[2] BERTSEKAS D., TSITSIKLIS J., *Parallel and Distributed Computation, Numerical Methods*. Prentice Hall Englewood Cliffs N.J., 1989.

[3] CHAZAN D., MIRANKER W., "Chaotic relaxation", *Linear Algebra Appl.,* vol. 2, pp. 199–222, 1969.

[4] EL TARAZI M., "Some Convergence Results for Asynchronous Algorithms", *Numerische Mathematik,* vol. 39, pp. 325-340, 1982.

[5] GIRAUD L., SPITERI P., "Résolution parallèle de problèmes aux limites non linéaires", *M2 AN,* vol. 25, pp. 579–606, 1991.

[6] LIONS J.L., "Quelques méthodes de résolution des problèmes aux limites non linéaires", *DUNOD* 1969.

[7] LIONS P., MERCIER B., "Approximation numérique des équations de Hamilton-Jacobi-Bellman", *R.A.I.R.O. Analyse numérique,* vol. 14, pp. 369-393, 1980.

[8] MIELLOU J., "Algorithmes de relaxation chaotique à retards", *RAIRO Analyse numérique,* vol. R1, pp. 55–82, 1975.

[9] MIELLOU J., SPITERI P., "Un critère de convergence pour des méthodes générales de point fixe", *M2AN,* vol. 19, pp. 645–669, 1985.

[10] MIELLOU J., SPITERI P., "Two criteria for the convergence of asynchronous iterations", *in Computers and computing,* P. Chenin et al. ed., Wiley Masson, Paris, pp. 91–95, 1985.

[11] VARGA R., *Matrix Iterative Analysis*, Prentice Hall 1962.