Numerical experimentations of parallel strategies in structural non-linear analysis
Laurent Champaney, Jean-Yves Cognard, David Dureisseix, Pierre Ladevèze

To cite this version:

HAL Id: hal-00321320
https://hal.archives-ouvertes.fr/hal-00321320
Submitted on 15 Dec 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Numerical experimentations of parallel strategies in structural non-linear analysis

L. Champaney, J.-Y. Cognard, D. Dureisseix, P. Ladevèze

Laboratoire de Mécanique et Technologie,
(E.N.S. de Cachan / C.N.R.S. / Université Paris 6),
61, Avenue du Président Wilson, 94235 CACHAN CEDEX, FRANCE

Abstract

La simulation de structures à comportement non linéaire conduit souvent à des coûts numériques élevés. Pour les réduire, nous utilisons une stratégie adaptée aux calculateurs parallèles. Nous présentons alors deux applications (disque de fatigue biaxiale et éprouvette biaxiale), pour des structures de type industriel à grand nombre de degrés de liberté. Les simulations ont été réalisées sur un ordinateur IBM SP2 à 16 processeurs utilisant PVM.

Numerical simulation of structures with non-linear behaviour often leads to high numerical costs. In order to reduce these costs, we use a strategy well-suited to parallel computers. Two applications (biaxial fatigue disk and biaxial specimen) are presented in the case of industrial-type problems with a large number of degrees of freedom. They have been performed on a 16-processor IBM SP2 using PVM.


Keywords: simulation, viscoplasticity, parallelism, substructuring

1 Introduction

In structural mechanics, models for materials are numerous and describe with increasing accuracy the materials’ real behaviour for more and more complex loadings. In order to predict the life expectancy of industrial structures, the classical step-by-step methods lead to the resolution of a costly non-linear, time-independent problem at each increment of the loading path. Speed-up techniques have been developed in order to reduce the computational cost, but complex simulations do often give rise to large numerical costs.

An approach called the LATIN method (LArge Time INcrement method), suited to parallel computers and whose goal is to reduce the numerical costs, has been proposed; its general presentation can be found in [5].

Two applications using PVM ([4]) on a MIMD parallel machine, an IBM SP2 with 16 processors at the CEA of Saclay, France, illustrate the numerical behaviour of this approach for large-scale industrial-type problems.

2 Viscoplastic simulations under cyclic loading

2.1 A two time scale representation suitable for cyclic phenomena

The LATIN method is an iterative procedure which takes into account the entire loading process at each iteration. It only confronts one of the main difficulties at each stage of the iterative scheme ([5]). Non-linear relationships are solved locally in space, and global-in-space problems are linear. The unknowns are space-time functions and a key point of the method is to choose appropriate space-time representations: the corrections are defined as a sum of products of space fields by scalar time functions. Moreover, for cyclic loadings, the time functions are represented over the whole loading time from their value over a few selected cycles ([1]).
Since the \textsc{latin} algorithms require many independent calculations for each element, the use of parallel computers is expected to reduce the computational costs. At the local stages, the problem is a small non-linear, time-dependent one over the studied time interval \([0, T]\), and it can be solved at each integration point concurrently. At the global stages, integrals over the body \(\Omega\) at each time \(t\) and integrals over \([0, T]\) at each integration point have to be evaluated; they can also be computed concurrently. At the end of the loop on the elements, the host program has to finish the calculation (Table 1), ([1]). The code “\textsc{viscolatin}” that we have built, uses the database of the finite element code \textsc{castem} 2000 ([7]) and \textsc{pvm} system ([4]).

2.2 Model of an aircraft turbine disk under cyclic loading

We consider a model of an aircraft turbine disk (figure 1) submitted to centrifugal force, which has been used many times in studies on viscoplasticity and failure ([1]). The material behaviour is described by a modified Chaboche’s viscoplastic model. The used mesh contains 10 208 elements (three-node elements) and 10 714 d.o.f. The computation was carried out over 500 cycles in a single increment. A two time-scale approximation uses only 18 cycles to represent the time functions. A high level of accuracy is reached after only 18 iterations (figure 2). Speed-ups are shown on figure 3. These quite encouraging results can be explained with a well-balanced load on the different processors. Figure 1 shows the distribution of elements among the processors when 4 processors are used on a 1 137 elements mesh. For such kind of problems, the influence of the size of the problem is reported in Figure 4 using \textsc{xpvm} flow trace, when changing both the number of elements and the number of loading cycles. The portion of time spent in synchronisations and in exchange of messages only slowly increases with the size of the problem; so, the efficiency increases with it.

3 The \textsc{latin} method and a substructuring technique

3.1 Decomposition into sub-structures and interfaces

Domain decomposition methods allow parallel-oriented algorithms along with a reduction of the size of stiffness matrices ([3]), and thus of the size of the problems. Such a substructuring is used with the \textsc{latin} approach in order to manage with “massive” parallelism. The structure is seen as an assembly of sub-structures and interfaces ([6], [2]), each having its own behaviour equations. We assume that sub-structures remain elastic; each communicates only with its neighbouring interfaces. An iterative scheme based onto the \textsc{latin} method leads to the resolution of independent linear problems on each sub-structure, and independent local in space variable problems on the interface. The linear problems consist, here, in satisfying the behaviour of each sub-structure \(\Omega^{E}\), onto which it looks like an elasticity-type problem with a constant matrix \([K^{E}]\). Table 2 shows the differences between the sequential and parallel implementations when one node program manages only one sub-structure.

3.2 Example involving a large number of d.o.f.

The example concerns a tensile biaxial specimen used in our laboratory. The symmetries allow us to study only one sixteenth of the specimen that has been decomposed into 31 roughly equilibrated sub-structures (Figure 5). For this example (linear elasticity) and for different meshes (Table 3), a valuable solution is obtained after 100 iterations. The computation costs of the direct resolution method and of the proposed iterative approach (sequential version) has been compared on the \textsc{cray} c90 of the \textsc{idris} at Orsay, France, (Table 4). It is important to notice that, when the size of the problem increases, the size of the stiffness matrices and the numerical cost for the substructuring technique increase slower than for the direct one.

4 Conclusion

The first numerical results obtained with \textsc{pvm} show that taking into account the intrinsic parallelism of the \textsc{latin} approach allows good efficiency for “coarse-grain” parallelism. With a substructuring technique, the reduction of storage requirements has been shown; moreover, this approach is better suited to “massive” parallelism. A coupling between the aforementioned sources of parallelism will have to be studied in order to take advantage of both for non-linear complex simulations.

References


<table>
<thead>
<tr>
<th>host program</th>
<th>node program</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loop on processes</td>
<td></td>
</tr>
<tr>
<td>Distribution of the calculations</td>
<td></td>
</tr>
<tr>
<td>messages to node programs $\rightarrow$</td>
<td></td>
</tr>
<tr>
<td>End of loop</td>
<td></td>
</tr>
<tr>
<td>Same computation as node programs</td>
<td></td>
</tr>
<tr>
<td>Loop on processes</td>
<td></td>
</tr>
<tr>
<td>Recovery of the results</td>
<td></td>
</tr>
<tr>
<td>messages from node programs $\leftarrow$</td>
<td></td>
</tr>
<tr>
<td>End of loop</td>
<td></td>
</tr>
<tr>
<td>Last part of the calculation</td>
<td></td>
</tr>
<tr>
<td>Recovery of the data</td>
<td></td>
</tr>
<tr>
<td>$\rightarrow$ message from host program</td>
<td></td>
</tr>
<tr>
<td>Loop on each local element</td>
<td></td>
</tr>
<tr>
<td>Loop on each integration point of the element</td>
<td></td>
</tr>
<tr>
<td>Computation over $[0,T]$</td>
<td></td>
</tr>
<tr>
<td>Contribution of each integration point</td>
<td></td>
</tr>
<tr>
<td>End of loop</td>
<td></td>
</tr>
<tr>
<td>End of loop</td>
<td></td>
</tr>
<tr>
<td>Transfer of the results</td>
<td></td>
</tr>
<tr>
<td>$\leftarrow$ messages to host program</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: host and node algorithms for viscoplastic computations

Figure 1: model for an aircraft turbine disk (biaxial fatigue disk)

Figure 2: results for the disk model

Figure 3: speed-up for the disk model

Figure 4: xpvm flow chart between processors
Sequential (one processor)

Loop on each sub-structure ($\Omega^E$)
- Factorisation of $[K^E]$ for each sub-structure ($\Omega^E$)
- End of loop

Loop on iterations
- Loop on each interface ($L^E_{E'}$)
- Convergence test
- Loop on each sub-structure ($\Omega^E$)
- Factorisation of $[K^E]$ for the sub-structure ($\Omega^E$)
- Loop on the interfaces of the substructure ($L^E_{E'}$)
- Inner interface $\leftrightarrow$ receiving message ($([q^E]_{L^E_{E'}}$)
- End of loop
- Convergence test $\leftrightarrow$ messages (contribution of each sub-structure)
- Global stage [$K^E][q^E] = [f^E]$
- Loop on each sub-structure ($\Omega^E$)
- Global stage

End of loop

Parallel (1 processor — 1 sub-structure)

Loop on each sub-structure ($\Omega^E$)
- Factorisation of $[K^E]$ for the sub-structure ($\Omega^E$)
- Loop on iterations
- Loop on each interface ($L^E_{E'}$)
- Convergence test
- Loop on each sub-structure ($\Omega^E$)
- Factorisation of $[K^E]$ for the sub-structure ($\Omega^E$)
- Loop on the interfaces of the substructure ($L^E_{E'}$)
- Inner interface $\leftrightarrow$ receiving message ($([q^E]_{L^E_{E'}}$)
- End of loop
- Convergence test $\leftrightarrow$ messages (contribution of each sub-structure)
- Global stage [$K^E][q^E] = [f^E]$
- Loop on each sub-structure ($\Omega^E$)
- Global stage

End of loop

Table 2: sequential and parallel algorithms for sub-structuration

<table>
<thead>
<tr>
<th>Direct computation</th>
<th>Sub-structuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>nodes</td>
<td>d.o.f.</td>
</tr>
<tr>
<td>mesh1</td>
<td>58 580</td>
</tr>
<tr>
<td>mesh2</td>
<td>86 359</td>
</tr>
<tr>
<td>mesh3</td>
<td>111 578</td>
</tr>
</tbody>
</table>

Table 3: characteristics of the meshes for the biaxial specimen

<table>
<thead>
<tr>
<th>IBM SP2 (16 processors)</th>
<th>CRAY C90</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direct</td>
<td>Sub-structuration</td>
</tr>
<tr>
<td>total</td>
<td>init. (s-s)</td>
</tr>
<tr>
<td>mesh 1</td>
<td>4 300 s</td>
</tr>
<tr>
<td>mesh 2</td>
<td>-</td>
</tr>
<tr>
<td>mesh 3</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 4: computation costs for the biaxial specimen

Figure 5: sub-structuration and results for the biaxial specimen