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 .

This is a preprint of the article published in its final form in: Laurent Champaney, Jean-Yves Cognard, David Dureisseix, Pierre Ladevèze. Numerical experimentations of parallel strategies in structural non-linear analysis. *Calculateurs Parallèles* 8(2):245-249, 1996, Hermes.

Mots clés : simulation, viscoplasticité, parallélisme, sous-structuration

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 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 Ω 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 "VISCOLATIN" that we have built, uses the database of the finite element code CASTEM 2000 ([7]) and 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 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 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 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 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 Ω^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 CRAY C90 of the 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 PVM show that taking into account the intrinsic parallelism of the 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

[1] J.-Y. Cognard. Simulation sur ordinateur à architecture parallèle de structures viscoplastiques sous chargements cycliques. 5(1):101–119, 1996.

[2] J.-Y. Cognard, D. Dureisseix, P. Ladevèze, and P. Lorong. Expérimentation d'une approche parallèle en calcul de structures. 5(2):197–220, 1996.

[3] C. Farhat and F.-X. Roux. Implicit parallel processing in structural mechanics. In J. T. Oden, editor, *Computational Mechanics Advances*, volume 2. North-Holland, June 1994.

[4] A. Geist, A. Berguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam. *PVM: A user's guide and tutorial for networked parallel computing.* MIT Press, Tennessee, 1994.

[5] P. Ladevèze. *Mécanique non-linéaire des structures — Nouvelle approche et méthodes de calcul non incrémentales.* Hermès, Paris, 1996.

[6] P. Ladevèze and P. Lorong. A large time increment approach with domain decomposition technique for mechanical non linear problems. In R. Glowinski, editor, *Comput. Meths. Appl. Sc. Engng.*, pages 569–578, New York, 1992. INRIA, Nova Science.

[7] P. Verpeaux, T. Charras, and A. Millard. CASTEM 2000 : une approche moderne du calcul des structures. In J.-M. Fouet, P. Ladevèze, and R. Ohayon, editors, *Calcul des Structures et Intelligence Artificielle*, volume 2, pages 261–271. Pluralis, 1988.

host program	node program

Loop on processes
  Distribution of the calculations
  messages to node programs →
End of loop
	Recovery of the data
	→ message from host program
	Loop on each local element
	  Loop on each integration point of the element
	  Computation over [0,T]
	  Contribution of each integration point
	End of loop
	End of loop
	Transfer of the results
	← messages to host program

Same computation as node programs	

Loop on processes
  Recovery of the results
  messages from node programs ←
End of loop
Last part of the calculation	

Table 1: host and node algorithms for viscoplastic computations

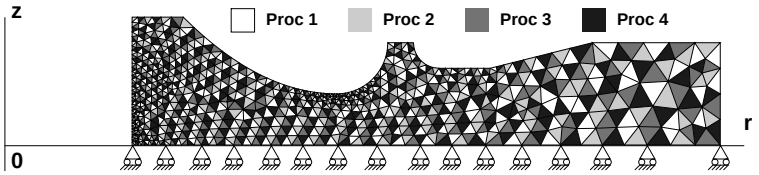


Figure 1: model for an aircraft turbine disk (biaxial fatigue disk)

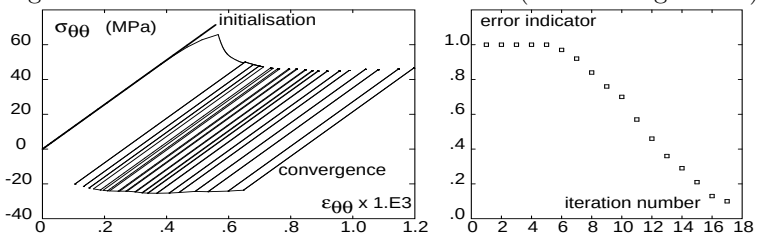


Figure 2: results for the disk model

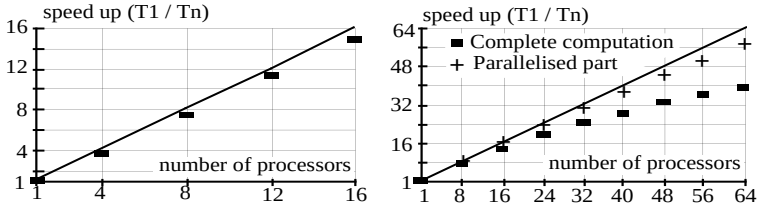


Figure 3: speed-up for the disk model

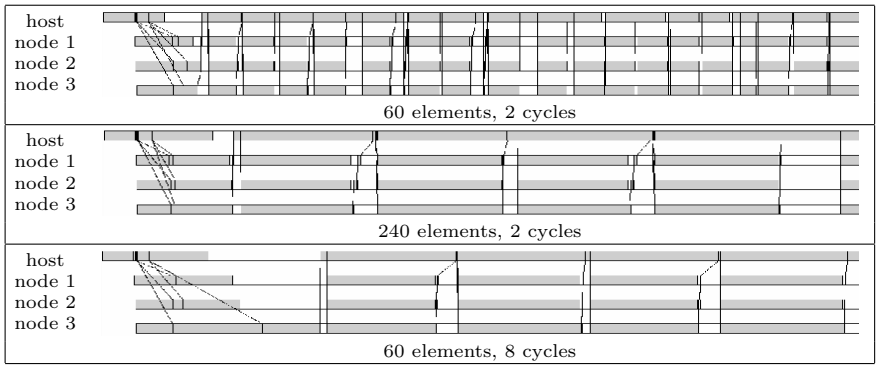


Figure 4: xpvm flow chart between processors

sequential (one processor)	**parallel** (1 processor—1 sub-structure)
Loop on each sub-structure (Ω^E) | Factorisation of $[K^E]$ End of loop Loop on iterations | Loop on each interface ($L^{EE'}$) | **Local stage (error estimation)** End of loop Convergence test Loop on each sub-structure (Ω^E) | **Global stage** $[K^E][q^E] = [f^E]$ End of loop End of loop	Factorisation of $[K^E]$ for the sub-structure (Ω^E) Loop on iterations | Loop on the interfaces of the substructure ($L^{EE'}$) | Inner interface ← receiving message ($[q^E]_{\mid L^{EE'}}$) End of loop Convergence test ↔ messages (contribution of each sub-structure) **Global stage** $[K^E][q^E] = [f^E]$ Loop on the interfaces of the substructure ($L^{EE'}$) | Inner interface → sending message ($[q^E]_{\mid L^{EE'}}$) End of loop End of loop

Table 2: sequential and parallel algorithms for sub-structuration

	Direct computation				Sub-structuration				
	nodes	d.o.f.	elements	size	nodes (s-s)	d.o.f. (s-s)	elements (s-s)	size (s-s)	total size
mesh1	58 580	175 740	11 961	2 160 Mb	2 191	6 573	364	25 Mb	750 Mb
mesh2	86 359	259 077	18 471	4 000 Mb	3 337	10 011	600	45 Mb	1 350 Mb
mesh3	113 578	340 734	24 764	6 400 Mb	4 229	12 687	800	71 Mb	2 130 Mb

Table 3: characteristics of the meshes for the biaxial specimen

	IBM SP2 (16 processors)					CRAY C90			
	Direct	Sub-structuration				Direct	Sub-structuration		
	total	init.(s-s)	iteration (s-s)	total (s-s)	total (100 it.)	total	init.	iteration	total (100 it.)
mesh 1	4 300s	37 s	0.6 s	97 s	2 910 s	1 100 s	316 s	5.9 s	935 s
mesh 2	-	70 s	1.0 s	170 s	5 100 s	2 100 s	548 s	10 s	1 548 s
mesh 3	-	130 s	1.6 s	290 s	8 700 s	3 050 s	861 s	15.5 s	2 411 s

Table 4: computation costs for the biaxial specimen

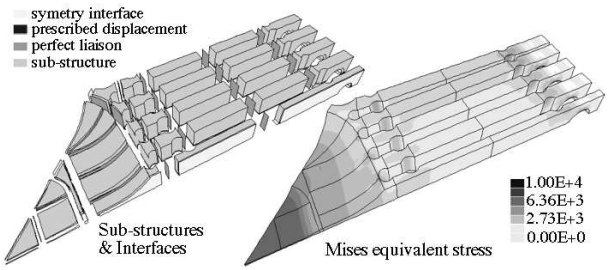


Figure 5: sub-structuration and results for the biaxial specimen