# Acceleration of finite element analysis by parallel processing

S. Moto Mpong, P. de Montleau, A. Godinas, A.M. Habraken

*Université de Liège, Département M&S,*
*Chemin des Chevreuils, 1, 4000 Liège, Belgique*

*e-mail: smoto@ulg.ac.be ; montleau@ulg.ac.be;*
*godinas@ulg.ac.be ; Anne.Habraken@ulg.ac.be*

ABSTRACT: The Finite Element code LAGAMINE developed since 1982 has been adapted to numerous finite elements and constitutive laws. The present researches on micro-macro modelling increase the requirement of CPU time. The parallelization of software is often divided into two main approaches: the coarse grain approach and the fine grain approach. For Finite Element software, the coarse grain method is usually characterized by the use of the mesh partitioning[4]. In the fine grain approach, there is no partition of the mesh. The parallel program is made by the parallelization of instructions issued of the analysis of dependences between the instructions, and the loops. The efficiency of all these approaches depends on the organization of the memory and on the used exchange protocol (MPI, PVM, or OpenMP). Our approach is intermediate between these two main categories of parallelization. It is characterized by the parallelization of the assembly of the stiffness matrix using a coarse grain approach and the use of a parallel direct solver. Because of the architecture of the available parallel computers, which are shared memory, we decided to use OpenMP protocol.

Keywords: parallel computing; finite element; metal forming simulation.

## 1 INTRODUCTION

The paper is dealing with the design of a parallelization model for the non-linear large deformation finite element software LAGAMINE. The results are presented on an example of deep drawing and on an example of upsetting of an elastic cube. At first, the equilibrium equations solved by the software are recalled as well as the finite element analysis and the time integration method. Then, the identification of the most expensive parts of the algorithm referring to the computational time is done. According to this analysis, the assembly loop and the solver of the linear system have been parallelized using the OpenMP protocole.

## 2 CONSTITUTIVE EQUATIONS

### *2.1 Equilibrium equations*

Relation (1) presents the equilibrium equation in the simple case, when neither volume and nor external forces are applied. Solid metallic alloys are assumed deformable bodies. Their governing equations consist in the elasto-plastic constitutive equations (1-3), defining the Cauchy stress tensor $\underline{\sigma}$, with respect to the strain rate $\dot{\underline{\varepsilon}}$, with $\underline{v}$ as the velocity field, $\underline{\varepsilon}$ as the strain tensor, $K_0$, $p_1$, $p_2$, $p_3$, $p_4$ as temperature dependant parameters. The Von Mises equivalent stress, strain and strain rate are identified by $\bar{\sigma}$, $\bar{\varepsilon}$, $\dot{\bar{\varepsilon}}$.

$$\nabla.\underline{\sigma} = 0 \qquad (1)$$

$$\bar{\sigma} = K_0 \bar{\varepsilon}^{p_4} \cdot exp\left(-p_1 \bar{\varepsilon}\right) p_2 \cdot \sqrt{3}\left(\sqrt{3} \cdot \dot{\bar{\varepsilon}}\right)^{p_3} \qquad (2)$$

$$\dot{\underline{\varepsilon}} = \frac{1}{2}\left(\nabla \underline{v} + (\nabla \underline{v})^T\right) \qquad (3)$$

### *2.2 Time discretisation*

For the computation of the solution of the non-linear equation system, an incremental method is used. The load is applied step by step. If F is the external load applied, we apply $\lambda F$ with $0<\lambda<1$. On each step, the constitutive equations are integrated in function of the strain rate. The iterations are done on the choice of the displacement until the convergence of the Newton-Raphson algorithm used to solve the equation system. Then the load is increased to $\lambda_2 F$ with $\lambda_1 < \lambda_2 \leq 1$ until $\lambda_n = 1$. The equilibrium is then verified when the internal nodal loads are equal to the nodal applied loads.

*2.3 Weak form of the mechanical equations*

The equilibrium equations are integrated using the principle of virtual power.
According to the above-mentioned time integration scheme, at each time step, equations (1) to (3) have to be solved for $x_{t+\Delta t}$ on the updated geometry $\Omega^{t+\Delta t}$.

The application of the principle of virtual power method to the above mentioned momentum equation (1), to which the external and the volumic loads have been added, leads to the following weak form:

$$\forall \underline{\delta u} \in \vartheta_0$$

$$\int_{\Omega^{t+\Delta t}} \underline{\sigma}^T \underline{\delta\varepsilon}\, dV = \int_{\Omega^{t+\Delta t}} \rho\, \underline{f}^T \underline{\delta u}\, dV + \int_{\partial\Omega^{t+\Delta t}} \rho\, \underline{t}^T \underline{\delta u} dS \quad (5)$$

with $\underline{\delta u}$ *virtual displacement,*

$\underline{\sigma}$ *Cauchy stress vector,*

$\underline{\delta\varepsilon}$ *virtual strain vector compatible with the virtual displacement,*

$\rho$ *density,*

$\underline{f}$ *external volumic load,*

$\underline{t}$ *external surface load,*

$V$ *and* $S$ *respectively the current volume and the current surface.*

*2.4 Matrix form of the equilibrium equations*

Let's $u_i^I$ be the displacement in the direction $i$ of the node $I$,
and $\phi^I$ the interpolation function of the node $I$.
The displacement field can then be discretised in the following form:

$$u_i = \sum_{I=1}^{N} \phi^I(\xi,\eta) u_i^I \quad (6)$$

Also, the virtual displacement can be written in the following form:

$$\delta u_i = \sum_{I=1}^{N} \phi^I(\xi,\eta) u_i^I \quad (7)$$

In matrix form and bi-dimensional case, we have:

$$\underline{\delta u} = \underline{\underline{H}}\, \underline{\delta U} \quad (8)$$

with $\underline{\underline{H}} = \begin{bmatrix} \phi^1 0 \phi^2 0 \ldots \varphi^N 0 \\ 0 \phi^1 0 \phi^2 \ldots 0\ \phi^N \end{bmatrix}$ and

$\underline{\delta U}^T = \left[\delta u_1^1\, \delta u_2^1 \delta u_1^2 \delta u_2^2 \ldots \delta u_1^N \delta u_2^N\right]$ the vector of nodal values.

We also have $\underline{\delta\varepsilon} = \underline{\underline{B}}\, \underline{\delta U}$ (9)

with $\underline{\underline{B}} = \begin{bmatrix} \frac{\partial\phi^1}{\partial x_1} & 0 & \cdots & \frac{\partial\phi^N}{\partial x_1} & 0 \\ \frac{\partial\phi^1}{2\partial x_2} & \frac{\partial\phi^1}{2\partial x_1} & \cdots & \frac{\partial\phi^N}{2\partial x_2} & \frac{\partial\phi^N}{2\partial x_1} \\ 0 & \frac{\partial\phi^1}{\partial x_2} & \cdots & 0 & \frac{\partial\phi^N}{\partial x_2} \end{bmatrix}$

in two-dimensional case.

The discrete equilibrium equation can then be written as:

$$\int_\Omega \underline{\sigma}^T \underline{\underline{B}}\, \underline{\delta U} dV = \int_\Omega \rho\, \underline{f}^T \underline{\underline{H}}\, \underline{\delta U}\, dV + \int_{\partial\Omega} \rho\, \underline{t}^T \underline{\underline{H}}\, \underline{\delta U}\, dS$$

(10)

Equation (10) can also be written as

$$\left(\underline{F}^{int} - \underline{F}^{ext}\right)\underline{\delta U} = 0 \quad (11)$$

With:

$F^{int} = \int_\Omega \underline{\underline{B}}^T \underline{\sigma}\, dV$ vector of internal forces equivalent to the stress,

$F^{ext} = \int_\Omega \rho \underline{\underline{H}}^T \underline{f} dV + \int_{\partial\Omega} \rho \underline{\underline{H}}^T \underline{t} dS$ vector of external forces equivalent to the applied load.

## 3 ANALYSIS OF THE MOST EXPENSIVE PARTS OF THE ALGORITHM REFERRING TO THE COMPUTATIONAL TIME

The first thing done was an analysis of the computational time on examples requiring most CPU time. In the non linear process of LAGAMINE, about 99% of the CPU time is spent in the element loop to integrate the constitutive equations and to compute the tangent stiffness matrix, and in the solution of the generated linear system. The balance between these two parts changes seriously with respect to the kind of solved problem. In the general case, the time spent to solve the generated linear system is greater than the time spent for the assembly of the stiffness matrix. However, for the simulations with complicated constitutive laws like micro-macro laws taking into account the evolution of the texture[2], we observed that the time spent in the assembly of the stiffness matrix is greater than the time spent in the linear solver. Also, the first goal has been the parallelization of the element loop[3] by the introduction of OpenMP instructions in the software, and thereafter the direct solver.

## 4 STEPS OF THE PARALLELIZATION

For the parallelization using OpenMP protocol [1], the first thing done was a dependency analysis of the

sequential program. According to this analysis, the sequential order of the program is eventually changed in order to exhibit independent instructions that will be parallelised, but the semantic of the program must be respected.

### *4.1 The assembly loop*

The assembly loop is a loop over all the elements of the mesh. For each element, the associated law is called and the constitutive equations are integrated in order to compute the internal forces and the tangent matrix. This loop is called at each iteration of the Newton Raphson algorithm. Its parallelization consist in the build of a "PARALLEL DO" loop in order to share the iterations of the loop over the available processors. For the sharing of the iteration of the loop, there are two options: the STATIC option which shares the iteration of the loop at once at the beginning of the loop, and the DYNAMIC option which shares the iteration dynamically. Our best results have been obtained with the option DYNAMIC(N) which shares dynamically the iteration by pack of size n. For the attributes of the variables in the parallel zone, we use the PRIVATE, the SHARED and the FIRST PRIVATE attributes. The PRIVATE attribute is for variables which are private to each iteration and remain in the stack memory of each task, while the SHARED attribute is for variables which are shared by all the task. FIRST PRIVATE attribute is for variable private to each task, but initialised at their value in the previously sequential zone. The main difficulties encountered were for variables used as counters and arrays with both SHARED and PRIVATED components. The solution has been the creation of new variables. For variables used as counters that were not used in the element loop, we used the OpenMP ATOMIC directive to successively execute the updating of the variable by each task.

### *4.2 Solver of the generated linear system of equations*

The parallelization of the direct solver is a complex problem, while the parallelization of an iterative solver is quite simpler [4]. However, in regards to the size of the actual problems (about 6000 degrees of freedom), and considering that there won't be more than 100000 degrees of freedom in the nearest future, we chose to keep a direct solver. Moreover, we had the opportunity to use the parallel one developed in the CEPBA department of the Universitat Politénica de Cataluña [5]. That parallel direct solver can be described by the following points:

- the Morse storage is used, so that only non zero elements are stored;
- A METIS software is used for the renumbering of the equations. That renumbering optimises the number of operations that will be performed during the factorisation;
- The symbolic factorisation is performed. This operation gives the structure of the L and U matrices that will be obtained during the factorisation and permits to foresee their storage.

## 5 APPLICATIONS

### *5.1 Deep drawing with evolution of the texture*

We present here the results obtained for the deep drawing simulation example, realised as part of the present PhD thesis of Laurent Duchêne in our department. The material is steel with a small elasticity limit. The mesh has 4020 nodes and 1504 volumic elements, for a total of 7000 degrees of freedom. The simulation is a three-dimensional mechanical computation, which consists in the deformation of the initial steel part by a punch until the final shape. The material is characterised by the anisotropic elasto-plastic Minty-law described in [2]. The texture and its evolution determine the yield locus during the simulation. For this example, the computational time required for the assembly of the stiffness matrix is greater than the computational time required by the solver of the generated linear solver. This is the reason why we decided to parallelise the assembly loop of the stiffness matrix. The computation of this simulation, after 10 days on only one processor was not finished on a Silicon Graphix SG3800 machine, which has 64 processors. On 8 processors of this same machine, it has required 2 days of computation with a speed-up of five. The efficiency was about 93% on two processors, 84% on four processors and 73% on eight processors, as shown on Table1.

Table 1: Results obtained for the deep drawing case on a SG3800 machine

| Number of processors used | 1 | 2 | 4 | 8 |
|---|---|---|---|---|
| Speed-up | 1 | 1.86 | 3.36 | 5.91 |
| Efficiency(%) | 100 | 93 | 84 | 73 |

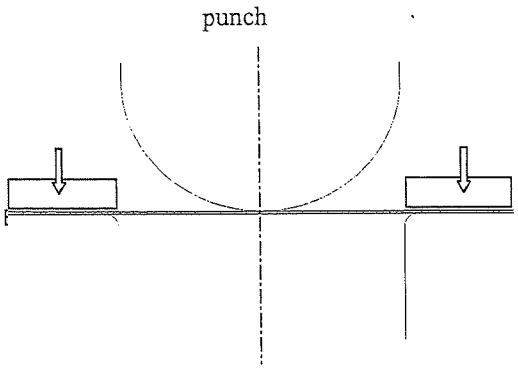


Fig. 1. Experimental set-up of the deep drawing

*5.2 Upsetting of an elastic cube*

The second example is an upsetting of a cube. The mesh of this cube has 4096 finite elements nodes and 3375 volumic elements, for a total of 11500 degrees of freedom. The behaviour of the material is characterised by a linear elastic law. The computation has been done on one processor, two processors, 4 processors and eight processors of a Silicon Graphix SG3800 computer, which has 64 processors. The efficiency was about 80% on 2 processors, 73% on 4 processors and 60% on 8 processors, as shown on Table2.

Table 2: Results obtained for the upsetting of the elastic cube

| Number of processors | 1 | 2 | 4 | 8 |
|---|---|---|---|---|
| Speed-up | 1 | 1.82 | 3.4 | 6 |
| Efficiency(%) | 100 | 91 | 85 | 75 |

## 6 CONCLUSION

We have presented a model of parallelization of a finite element software simple to realise. Our approach was first of all the identification of areas of the software that require a lot of computation time for our big examples. Then, those zones have been parallelised. These zones were the assembly loop of the stiffness matrix and the solution of the generated linear system. For the assembly loop of the stiffness matrix, we analysed and we eliminated the dependencies between the different iterations of the assembly loop. As our parallel computers are shared memory, we used OpenMP protocol. For the solution of the linear system generated, we adapted parallel subroutines from the Universitat Politénica de Cataluña. For the moment, we have result in accordance with the literature, and our simulations can be done within acceptable CPU times.

## REFERENCES

1 J. Chergui, P. F. Lavallée, J.-P. Proux and C. Roult, OpenMP parallélisation multitâche pour machine à mémoire partagée, *IDRIS*, (Janvier 2001).

2 L. Duchêne, A. M. Habraken and A. Godinas, Influence of steel sheet anisotropy during deep-drawing process, *The 4th international ESAFORM conference on Material Forming,* pp 461-464, (April 23-25 2001)

3 C. Farhat, Which parallel finite element algorithm for which architecture and which problem?, *Eng. Comput.*, 7:186-195 (September 1990)

4 S. Marie, Un modèle de parallélisation SPMD pour la simulation numérique de procédé de mise en forme des matériaux, *Thèse de l'Ecole des Mines de Paris* (1997)

5 Jose Maria Cela, Library CESAER, Centro Europeo de Paralelismo de Barcelona (2001)

Acknowledgements:
As Research Associate of National Fund for Scientific Research (Belgium), AM Habraken thanks this Belgian research fund for its support.