Parallel Computing Techniques for Metal Plasticity Applications

K. Inal[a], K.W. Neale[a] & P.D. Wu[b]

[a]Faculty of Engineering, University of Sherbrooke, Sherbrooke, Quebec, Canada.
[b]Alcan International Limited, Kingston R&D Centre, Kingston, Ontario, Canada.

Parallel computing algorithms for the finite element method have been developed to simulate multiscale large deformation plasticity problems. Due to the nature of these problems and the computational methods involved, different parallel computing techniques must be employed for the Taylor-type crystal plasticity (TFE) and the so-called "finite element per single crystal" (FESC) models. For the FESC model, the global stiffness matrix is formed in parallel and a parallel solver employing a number of Krylov iterative methods is used to solve the systems of equations. For the TFE model, a data parallel implementation technique is employed. This technique provides a large reduction in the storage required on a single processor and thus allows for an effective solution of computationally demanding crystal plasticity problems. The efficiency and the performance of the parallel finite element models that have been developed are investigated by comparing simulations performed on an IBM POWER3 SP parallel supercomputer.

Nous avons développé des algorithmes parallèles pour la méthode des éléments finis dans le but de simuler des problèmes de grandes déformations plastiques. En raison de la nature des problèmes étudiés et des méthodes de calcul employées, des schémas de parallélisation différents doivent être utilisés pour la plasticité cristalline de type Taylor (TFE) et pour les modèles à un élément par monoscristal (FESC). Dans le modèle FESC, la matrice de rigidité gloable est construite en parallèle et un code parallèle utilisant des méthodes itératives de Krylov est utilisé pour résoudre le système d'équations. Dans le modèle TFE, une méthode de parallélisation des données est utilisée plutôt. Celle-ci permet de réduire la mémoire requise sur chaque processeur et permet ainsi la résolution efficace de problèmes cristallins très exigeants. L'efficacité et la performance de ces deux algorithmes sont étudiées en comparant des simulations exécutées sur un superordinateur parallèle IBM SP3.

1 Introduction

Metal forming has, for a long time, been an important processing operation. In many manufacturing areas such as the automotive, aerospace, packaging and electronic industries, the optimization of metal processes has become a key factor to reduce product development time and final cost. In general, metal forming involves large strains due to stretching, drawing, bending or various combinations of these basic deformation modes. From the view point of mechanics, the analysis of metal working involves nonlinearities in geometry, material and contact. In an effort to better understand metal forming processes, various research works have been carried out using diverse technologies involving experimental, analytical and computational methods.

Essentially two classes of models have been developed for numerical simulations of metal forming operations: phenomenological (macroscopic) models and polycrystal (microscopic) models. Metal forming can be modelled accurately based on crystal plasticity theories where the initial texture and its evolution, as well as the anisotropy induced by the evolution of microstructure and microscopic properties, are accounted for [1, 2, 3, 4, 5].

The mathematical modelling of material behaviour is a very effective way of reducing time and costs involved in optimizing manufacturing processes. Indeed, numerous complex forming operations have been simulated using numerical methods in order to predict critical parameters. Among these methods, the finite element method (FEM), has been widely applied to the study of metal forming because of its flexibility, accuracy and efficiency. Considering the rapid advancement of computer capabilities, the finite element method has become a powerful tool in modelling metal forming operations.

Up to the 1980's, most applications involving the finite element method have been based on phenomenological constitutive models since microscopic models involve significantly more demanding computations. However the introduction of parallel computers has rendered metal forming modelling based on crystal plasticity feasible since they offer more computational power and storage than serial computer architectures. With proper parallelization techniques, realistic applications based on crystal plasticity can be performed on parallel computers such as the IBM SP3. Beaudoin et al. [6] presented a data parallel formulation for the analysis of polycrystalline solids where parallel computing algorithms were considered for both crystal equations and the iterative solution procedure. Later on, Sorensen and Andersen [7] developed a parallel finite element method suitable for the analysis of 3D quasi-static crystal plasticity problems. This method was based on substructuring of the original mesh into a number of substructures which were treated as isolated finite element models related via the interface conditions. Their method showed a good speedup with increasing number of processors.

In this paper, parallel computing algorithms for both the finite element method and crystal plasticity theory have been developed to simulate the plastic deformation of polycrystalline solids. First the parallel computing algorithms that have been developed for the Taylor-type crystal plasticity (TFE) and the so-called finite element per single crystal (FESC) models are described respectively. Then the efficiency of the parallel computing algorithms that we have developed are discussed by comparing simulations performed on the parallel computer IBM RS/6000 POWER3 (WinterHawk2) SP.

2 Parallel Computing Algorithms for Finite Element Analysis Based on Crystal Plasticity

In this section parallel computing algorithms developed for the TFE and the FESC models will be described. Due to the nature of these problems and the computational methods involved, different parallel computing techniques had to be developed. For the TFE model, parallelization focuses on the crystal aggregate, while for the FESC model, the finite element domain is divided into subdomains.

2.1 Constitutive Model

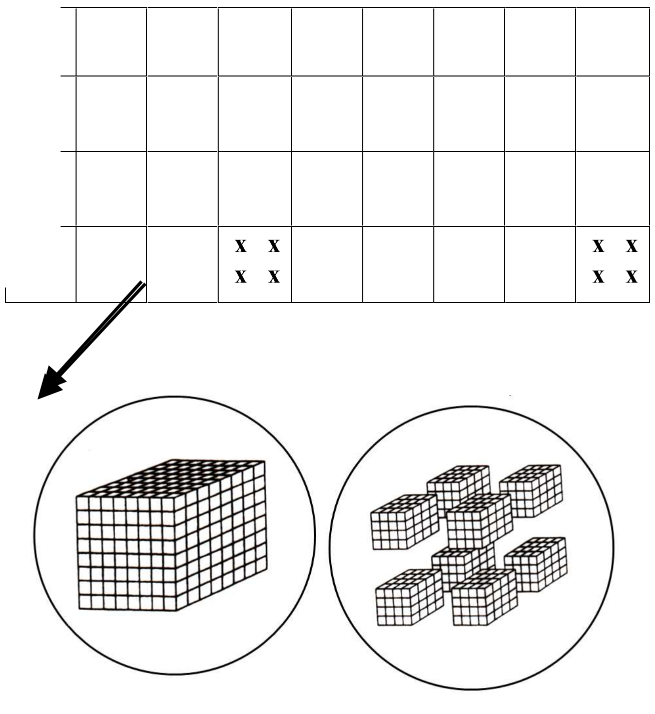
A rigorous framework for the kinematics of the finite plastic deformation of a crystal has been firmly established for some time. This basic formulation has been incorporated into a rate-dependent description of crystal plasticity constitutive relations. For details of this formulation we refer to [8].

2.2 Parallel Computing Algorithms for the TFE Model

For polycrystalline simulations based on the Taylor hypothesis (TFE), the microstructural calculations for the stress and crystal reorientations are essentially independent. At a material point representing a polycrystal of N grains, the deformation in each grain is taken to be identical to the macroscopic deformation of the continuum. Furthermore, the macroscopic values of all quantities, such as stresses, stress-rates and elastic moduli, are obtained by averaging their respective values over the total number of grains at the particular material point.

In general, Taylor-type polycrystal models are ideally suited for the parallelization of the computational procedures. Especially, when CPU time is considered, the simulations fall in the category of “embarrassingly parallel” (e.g., Sorensen [7], Inal [9]) applications, and they provide significant computational improvements. However, such “embarrassingly parallel” applications are strictly feasible only if the total program size fits within a single processor of the parallel computer. This is not always the case for simulations with TFE models since they involve extremely demanding computational resources due to the amount of information at the grain level that must be tracked.

The parallel algorithms employed in the TFE model are designed to distribute data on the microscopic level (crystal data) over the processors of the IBM SP3 parallel computer. By this method, the global size of the simulation is distributed between the processors of the parallel computer. To illustrate this, consider a simulation with a total number N of crystals (Fig.1). The basic idea in the finite element formulation is that each material point is representing a polycrystal comprised of N crystals and the constitutive response is given through the Taylor polycrystal model (Fig. 1a). The global crystal data is distributed between the processors (Fig. 1b) such that each processor runs a part of the global program for $B = N / A$ crystals where A is the total number of processors used in the simulation. (Note that the processors read only the crystal data to which they are assigned and all arrays containing microscopic quantities have the maximum size of B instead of N.) Thus all processors compute microscopic arrays (for the set of crystals that they have assigned) independently. However, to compute the global stiffness matrix, the macroscopic values of stresses, stress rates and the moduli are required (eqn. 1). These values are obtained by collective communication between the processors using the Message Passing Interface (MPI) Library. Note that,



(a) (b)

Figure 1 (a) Polycrystal aggregate comprised of N crystals, (b) the distribution of this polycrystal aggregate between processors that, after this communication, although macroscopic values of stresses, stress rates and the moduli are computed, microscopic values of these tensors are still stored since the microscopic quantities are updated each increment. As a result, each processor has its own microscopic data; however, all of the processors have the same macroscopic data. We should mention that the total number of crystals should be distributed as evenly as possible between the processors. Otherwise the large difference in the number of crystals per processor adversely affects the CPU time per processor, and in such cases blocking commands are required during communication between processors. To avoid this the parallel algorithms developed here distribute the crystals as evenly as possible among the available processors.

The size of any array storing microscopic quantities is equal to *M*, where *M* is equal to the product of the number of elements times number of integration points times number of crystals times number of slip systems. For the simulation presented above, the mesh consists of 1792 elements with 4 integration points per element. The initial texture is represented by 400 discrete orientations in Euler space, and with 12 active slip systems per crystal *M* becomes

$$M = 1972 \times 4 \times 400 \times 12$$
$$M = 34406400$$

Table 1 compares the CPU times for the above problem obtained from simulations with various number of processors. Note that we start comparing results with 16 processors, since this is the minimum number of processors required to provide enough memory for our global data. It can be seen that the parallelization employed in the TFE model provides an excellent speed-up with values very close to the theoretical maximum achievable speed-up by parallel computing.

No. of processors	CPU time (s)	Speed-up factor
16	169,200	1
24	113,557	1.49
32	85,454	1.98
56	48,760	3.47

Table 1. CPU time and speed-ups as a function of the number of processors

2.3 Parallel Computing for the FESC Model

The FESC model utilizes finite element methodologies to discretize the crystals of an aggregate using one or more elements for every crystal. In this model, a polycrystalline aggregate is treated as a continuum where both compatibility and equilibrium among the individual grains are automatically satisfied.

Parallel computing algorithms developed for the FESC model are completely different than the algorithms developed for the TFE model since, due to the basic assumptions of modelling, they require significantly less memory. For the FESC model, parallel computing algorithms focus on the finite element mesh and the solution procedure.

In general, the global finite element equations can be written in the form

$$[\mathbf{K}]\{\dot{\mathbf{u}}\} = \{\dot{\mathbf{F}}\} \tag{1}$$

where $[\mathbf{K}]$ is the global stiffness matrix, $\{\dot{\mathbf{u}}\}$ is the vector containing the unknown incremental nodal displacements and $\{\dot{\mathbf{F}}\}$ is the vector of applied incremental nodal forces. Note that, due to the nature of finite elements, $[\mathbf{K}]$ is a *n* x *n* sparse matrix where *n* is the total number of degrees of freedom within the system. To reduce storage requirements, the "skyline" storage strategy is adopted since this has the effect of reducing the number of zero terms that would be stored in a vector due to bandwidth variability using conventional methods.

A parallel solver employing a number of Krylov iterative methods is used to solve the systems of equations. The parallel solution process is carried out in two steps:

- The global stiffness matrix is formed in parallel. To illustrate this, consider a simulation with the finite element mesh comprised of 16 elements (Fig. 2a). This finite element mesh is divided into 2 subdomains with 8 elements (Fig.2b) and each subdomain is assigned to a single processor. Once input is read globally, each processor starts to compute the elemental stiffness matrix for the elements in their subdomain. All vector elements are explicitly stored on the processor and are defined by a set of indices referred to as the processor's "main set". The "main set" is further divided into two subsets: "internal" and "border". A component corresponding to an index in the "internal set" performs all computations using only information on the current processor. The "border set" defines elements that would require values from other processors in order to compute correct global values (line A-A' in Fig. 2b). Thus, a component corresponding to an index in the "border set" is computed by point-to- point communication using the MPI Library between the processors that share information. Note that, if the finite element mesh is distributed evenly between the processors, blocking commands are not necessary during these point-to-point communications. However, if one of the subdomains is a contact surface, it is obvious that special attention must be paid to the point-to-point communications between the processors.

- The iterative parallel solver employed in our code was specifically designed to solve the linear systems of equations $Ax = b$ where A is a sparse matrix. However, it has already been mentioned that the global stiffness matrix was stored in 2 vectors: A (contains the non zero terms in the upper diagonal) and C (contains the non zero terms in the lower diagonal). Thus, from these 2 vectors the global stiffness matrix must be reconstructed. This is performed by creating pointers and an integer vector which holds information regarding the position of the diagonal terms and the non zero terms to be stored up to each diagonal term. Note that the global stiffness matrix in never constructed completely; using pointers, it is constructed block by block on the processors. Once the equations are solved, updating of all quantities is performed in parallel without any communication.

To assess the efficiency of the parallel computing algorithms that have been developed for the FESC model, plane strain tension of a commercial aluminum alloy was simulated with 8000 elements. Figure 3 presents the speed-up curve for a simulation where the total number of equations n = 8181. It can be seen that a maximum speed-up of 2.8 (saturation value) was obtained with 8 processors; when more than 8 processors are employed, the speed-up starts to decrease. These results are expected since the communications performed during the parallel iterative solution process involving a sparse matrix are much more costly (in terms of CPU) in comparison to solutions involving a triangular or a diagonal matrix. However, it should be noted that the performance of parallel computing will noticeably increase when the number of the equations to be solved is increased.

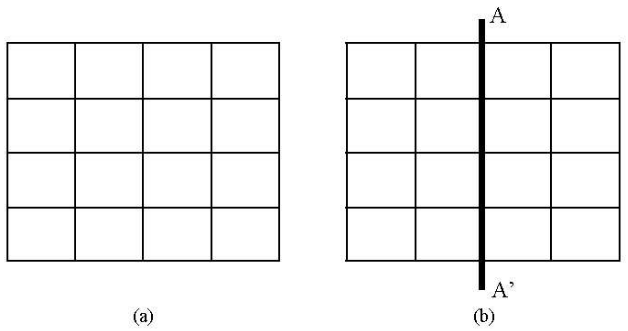


Figure 2 Domain decomposition for the finite element mesh

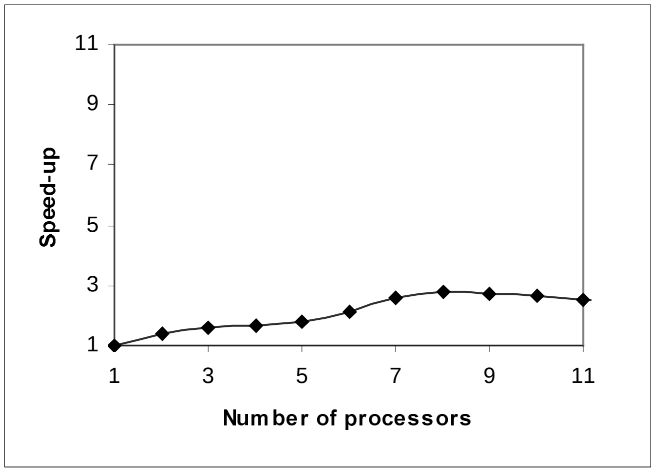


Figure 3 Speed-up as a function of the number of processors

3 Conclusion

In this paper, parallel computing algorithms that have been developed for both the finite element method and crystal plasticity theory to simulate plastic deformation of polycrystalline solids were discussed. The combination of polycrystal plasticity theory and modern parallel processing provides a powerful tool for the analysis of plastic deformation for a large class of engineering materials.

The TFE model demands significant amounts of memory and CPU time due to the amount of information at the grain level that must be tracked. Thus parallel computing algorithms were developed to distribute the global polycrystal aggregate into smaller aggregates between the processors. With this technique, the total capacity of all processors combined can be employed for the numerical simulations.

The FESC model requires significantly smaller memory compared to the TFE model. Thus parallel computing algorithms were developed to improve CPU time where the global stiffness matrix was computed in parallel (employing "skyline" storage), and the global systems of equations were then solved in parallel using an iterative solver. The efficiency of this type of parallelization depends directly on the total number of degrees of freedoms for the finite element system of equations. As the total number of equations to be solved increases, the performance of parallel computing will improve.

References

[1] Inal, K., Wu, P.D. & Neale, K.W., Simulation of earing in textured aluminium sheets. International Journal of Plasticity, **16**, pp. 635-648, 2000.

[2] Inal, K., Wu, P.D. & Neale, K.W., Instability and localized deformation in polycrystalline solids under plane strain tension. *International Journal of Solids and Structures*, **39**, pp. 983-1002, 2002.

[3] Inal, K., Wu, P.D. & Neale, K.W., Large strain behaviour of aluminum sheets subjected to in-plane simple shear. *Modelling and Simulation in Materials Science and Engineering,* **10**, pp. 237-252, 2002.

[4] Inal, K., Wu, P.D. & Neale, K.W., Finite element analysis of localized deformation in fcc polycrystals under plane stress tension. *International Journal of Solids and Structures*, (in press).

[5] Wu, P.D., Inal, K., Neale, K.W., Kenny, L.D., Jain, M. & MacEwen, S.R., Large Strain Behaviour of Very Thin Aluminium Sheets Under Planar Simple Shear. *Journal de Physique IV*, **11**, pp. 229-236, 2001.

[6] Beaudoin, A.J., Mathur, K.K., Dawson, P.R. & Johnson, G.C., Three-dimensional deformation process simulation with explicit use of polycrystal plasticity models. *International Journal of Plasticity*, **9**, pp. 833-860, 1993.

[7] Sorensen, N.J. & Andersen, B.S., A parallel finite element method for the analysis of crystalline solids. DCAMM Report, The Technical University of Denmark, 1995.

[8] Asaro, R.J. & Needleman, A., Texture development and strain hardening in rate dependent polycrystals. *Acta Metallurgica*, **33**, pp. 923-953, 1995.

[9] Inal, K., Modélisation numérique de l'aluminium à grandes déformations plastiques: applications au calcul parallèle," *M.A.Sc. Thesis*, Department of Civil Engineering, University of Sherbrooke, Sherbrooke, Québec, Canada, 1998.