 CMES, vol.8, no.2, pp.135-152, 2005

Multiscale Simulations Using Generalized Interpolation Material Point (GIMP) Method And SAMRAI Parallel Processing

J. Ma[1], H. Lu[1], B. Wang[1], S. Roy[1], R. Hornung[2], A. Wissink[2] and R. Komanduri[1,3]

Abstract: In the simulation of a wide range of mechanics problems including impact/contact/penetration and fracture, the material point method (MPM), Sulsky, Zhou and Shreyer (1995), demonstrated its computational capabilities. To resolve alternating stress sign and instability problems associated with conventional MPM, Bardenhagen and Kober (2004) introduced recently the generalized interpolation material point (GIMP) method and implemented for one-dimensional simulations. In this paper we have extended GIMP to 2D and applied to simulate simple tension and indentation problems. For simulations spanning multiple length scales, based on the continuum mechanics approach, we present a parallel GIMP computational method using the Structured Adaptive Mesh Refinement Application Infrastructure (SAMRAI). SAMRAI is used for multi-processor distributed memory computations, as a platform for domain decomposition, and for multi-level refinement of the computational domain. Nested computational grid levels (with successive spatial and temporal refinements) are used in GIMP simulations to improve the computational accuracy and to reduce the overall computational time. The domain of each grid level is divided into multiple rectangular patches for parallel processing. This domain decomposition embedded in SAMRAI is very flexible when applied to GIMP. As an example to validate the parallel GIMP computing scheme under SAMRAI parallel computing environment, numerical simulations with multiple length scales from nanometer to millimeter were conducted on a 2D nanoindentation problem. A contact algorithm in GIMP has also been developed for the treatment of contact pair between a rigid indenter and a deformable workpiece. GIMP results are compared with finite element results on indentation for validation. A GIMP nanoindentation problem with five levels of refinement was modeled using multi-processors to demonstrate the potential capability of the parallel GIMP computation.

keyword: Material point method (MPM), Generalized interpolation material point method (GIMP), Tension, Nanoindentation, Parallel computing, SAMRAI, Multilevel refinement, Contact problem.

1 Introduction

The material point method (MPM) has demonstrated its capabilities in addressing such problems as impact, upsetting, penetration, and contact (e.g. Sulsky, Zhou and Schreyer (1995); Sulsky and Schreyer (1996)). In MPM, two descriptions are used – one based on a collection of material points (Lagrangian) and the other based on a computational background grid (Eulerian), as proposed by Sulsky, Zhou and Schreyer (1995). A fixed structured mesh is generally used in the background throughout the MPM simulations. The material points are followed throughout the deformation of a solid to provide a Lagrangian description and the governing field equations are solved at the background grid nodes so that MPM is not subject to mesh entanglement. Compared to the finite element method (FEM), MPM takes advantage of both Eulerian and Lagrangian descriptions and possesses the capability of handling large deformations in a more natural manner so that mesh lockup problems present in FEM are avoided. Additionally, for problems involving contact, MPM provides a naturally non-slip contact algorithm to avoid the penetration between two bodies based on a common background mesh (Sulsky, Zhou and Schreyer (1995), Sul-

[1] School of Mechanical and Aerospace Engineering, Oklahoma State University, Stillwater, OK 74078, U.S.A

[2] Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, Livermore, CA 94551, U.S.A

[3] Correspondence author, e-mail: ranga@ceat.okstate.edu; Tel: 405-744-5900; Fax: 405-744-7873.

This work was performed under the auspices of the U.S. Department of Energy by University of California, Lawrence Livermore National Laboratory under Contract W-7405-Eng-48. UCRL-JRNL-208241.

sky and Schreyer (1996)). One drawback of the conventional MPM is that when the material points move across the cell boundaries during deformation, some numerical noise/errors can be generated, Bardenhagen and Kober (2004). To solve the instability problems associated with the conventional MPM simulations, Bardenhagen and Kober recently proposed the generalized interpolation material point method (GIMP) and implemented for one-dimensional simulations.

The present investigation extends the GIMP presented by Bardenhagen-Kober to two-dimensional simulations and applies it to simple tension and indentation problems. Furthermore, a refinement technique and a parallel processing scheme are developed so that the serial GIMP algorithm and code can be extended for parallel computation of large scale computations based on the continuum mechanics approach.

Parallel processing has been used successfully in numerical analysis using different methods, such as FEM and boundary element method (BEM), Mackerle (2003) and molecular dynamics (MD), Kalia and Nakano (1993). The computational time on parallel processors can be reduced to a small fraction of the time consumed by a single processor at the same speed. Parallel processing generally involves issues, such as domain decomposition/partitioning, load balancing, parallel solver/algorithms, parallel mesh generation, and multi-grid, Mackerle (2003). Domain decomposition has been widely applied in parallel processing in FEM, Hsien (1997). With partitioning of the overall computational domain, sequential FEM algorithm usually cannot be used directly in parallel processing without some modification, primarily due to the coupling of a large number of simultaneous linear equations. Remeshing is sometimes needed in each sub-domain. The interfaces of neighboring sub-domains must be meshed identically for subsequent communications, Mackerle (2003). These problems are intrinsic to certain numerical methods, such as FEM; however, they can be totally or partially avoided if other appropriate computational methods are used. For example, the domain decomposition is more straightforward for structured meshes, and large systems of coupled equations can be avoided, if explicit time integration is used.

Recently a platform for parallel computation, namely, the structured adaptive mesh refinement application infrastructure (SAMRAI), Hornung and Kohn (2002), has been developed by the Center for Applied Scientific Computing at the Lawrence Livermore National Laboratory. SAMRAI has provided interfaces for user-defined data types so that material points carrying physical variables (mass, displacement, velocity, acceleration, stress, strain, etc.) can be readily defined. As a result, SAMRAI is very suitable for handling material points and their physical variables in MPM or its variant, GIMP. In this investigation SAMRAI is used for parallelizing GIMP. SAMRAI has also provided a foundation for parallel adaptive mesh refinement (AMR) with the use of either dynamic or static load balancing, Wissink, Hysom and Hornung (2003). This function allows SAMRAI to process both spatial and temporal refinements in areas of interest, typically with high gradients in some physical variables (e.g., strains), and to use coarse mesh in the remaining areas. With the appropriate use of fine and coarse meshes in different regions, multiscale simulations using MPM can provide desired computational accuracy with reduced costs associated with computer memory and computational time.

Material multiscale simulations span from electronic structure, atomistic scale, crystal scale, to macro/continuum scale, Horstemeyer, Baskes, Prantil, Philliber and Vonderheide (2003); Komanduri, Lu, Roy, Wang and Raff (2004). Appropriate simulation algorithms can be used at various scales, e.g., *ab initio* computation for electronic structure, molecular dynamics at the atomistic scale, crystal plasticity or mesoplasticity at the crystal scale, and continuum mechanics at the macro scale, Horstemeyer, Baskes, Prantil, Philliber and Vonderheide (2003). At the continuum scale, FEM is generally used. Recently, the meshless local Petrov-Galerkin (MLPG) method (Shen and Atluri (2004a, 2005)) and the continuum/lattice Green function method (Tewary and Read (2004)) have been used to couple with molecular dynamics seamlessly. The MLPG method is a simple, less-costly alternative approach to FEM, Atluri and Shen (2002). For the purposes of providing the insights into the discrete atomistic system and coupling with continuum, an equivalent continuum was defined in the MD region to compute the atomic stress based on the Smoothed Particle Hydrodynamics (SPH) method, Shen and Atluri (2004b). The atomic stress tensor computed using the SPH method is more natural than other atomic stress formulations because it is in the nonvolume-averaged form and rigorously

satisfies the conservation of linear momentum. Hence, it is applicable to both homogeneous and inhomogeneous deformations. A tangent stiffness formulation was developed for both MLPG and MD regions and the displacements of the nodes and atoms are solved in one coupled set of linear equations. The MLPG/MD coupling has been demonstrated to be capable of enforcing the local balance equations in the handshaking region between continuum mechanics and molecular dynamics, Shen and Atluri (2005).

The simulation using parallel GIMP computing scheme in this investigation will focus on multiscales, e.g., from nanometer to millimeter, based on the continuum mechanics approach, namely, 2D GIMP. An example used for validating the simulation at several length scales at the continuum level is nanoindentation. It involves the contact issue between a rigid indenter and a deformable workpiece. A contact algorithm, which allows the contact interface to be located in a few computational domains, is introduced in this study. The contact pressure is determined from solving a set of equations from multiple processors. Parallel GIMP results on nanoindentation are compared with FEM results using the ABAQUS/Explicit code. A nanoindentation model with five levels will be used; this model allows simulation from nanometer to millimeter scales.

2 Generalized Interpolation Material Point (GIMP) Method

The governing equations in both conventional material point method (MPM), Sulsky, Zhou and Sheryer (1995), Hu and Chen (2003), Bardenhagen (2002) and generalized interpolation material point (GIMP) method, Bardenhagen and Kober (2004), are briefly summarized in this section. The weak form of the momentum conservation equation in the conventional MPM is given by

$$\int_\Omega \rho \mathbf{w}\cdot\mathbf{a}d\Omega = -\int_\Omega \rho \mathbf{s}^{\mathrm{S}} : \nabla \mathbf{w} d\Omega + \int_{\partial\Omega} \rho \mathbf{c}^{\mathrm{S}}\cdot\mathbf{w}dS + \int_\Omega \rho \mathbf{w}\cdot\mathbf{b}^s d\Omega \quad (1)$$

where $\mathbf{w}$ is the test function, $\mathbf{a}$ is the acceleration, and $\mathbf{s}^s$, $\mathbf{c}^s$ and $\mathbf{b}^s$ are the specific stress, specific traction, and specific body force, respectively. Ω is the current configuration and $\partial\Omega$ is the surface with applied traction. The material density, ρ, can be approximated as the sum of material point masses using a Dirac delta function $\rho(\mathbf{x},t) = \sum_{p=1}^{N_p} M_p \delta(\mathbf{x}-\mathbf{x}_p^t)$, where N_p is the total number of material points and M_p is the mass of the material point. Upon discretization of Eq. (1) using the shape functions $N_i(\mathbf{x}_p^t)$, the governing equations at the background grid nodes become (see Sulsky, Zhou and Schreyer (1995))

$$m_i^t \mathbf{a}_i^t = (\mathbf{f}_i^t)^{int} + (\mathbf{f}_i^t)^{ext}. \quad (2)$$

where the lumped mass matrix is given by

$$m_i^t = \sum_{p=1}^{N_p} M_p N_i(\mathbf{x}_p^t), \quad (3)$$

and the internal and external forces are given by

$$(\mathbf{f}_i^t)^{int} = -\sum_{p=1}^{N_p} M_p \mathbf{s}_p^{s,t}\cdot\nabla N_i|_{\mathbf{x}_p^t}, \quad (4)$$

$$(\mathbf{f}_i^t)^{ext} = -\sum_{p=1}^{N_p} M_p \mathbf{c}_p^{s,t} h^{-1} N_i(\mathbf{x}_p^t) + \sum_{p=1}^{N_p} M_p \mathbf{b}_p^t N_i(\mathbf{x}_p^t), \quad (5)$$

where h is the thickness of a boundary layer. At each time step, all variables for each material point, such as mass, velocity, and force are extrapolated to the grid nodes of the cell in which the material point resides. New nodal momenta are computed and used to update the physical variables carried by the material points. Thus, material points move relative to each other to represent deformation in a solid. A spatially fixed background grid is used throughout the MPM computation. MPM has already demonstrated its capabilities in solving a number of problems involving impact/contact/penetration. In case of large deformation, however, numerical noise, or errors have been observed, especially when material points have just crossed cell boundaries resulting in instability problems in the MPM simulations (see, e.g., Sulsky, Zhou and Schreyer (1995), Hu and Chen (2003), Bardenhagen and Kober (2004)). The primary cause for the problem has been attributed to the discontinuity of the gradient of the shape functions across the cell boundaries (see, e.g., Hu and Chen (2003), Bardenhagen and Kober (2004)). To resolve this problem, Bardenhagen and Kober (2004) proposed a generalized interpolation material point (GIMP) method. In GIMP, the interpolation between node i and material point p is given by the volume averaged weighting function

$$\overline{S}_{ip} = \frac{1}{V_p}\int_{\Omega\cap\Omega_p} \chi_p(\mathbf{x}) S_i(\mathbf{x}) d\mathbf{x}, \quad (6)$$

where V_p is the current volume of the material point, $\chi_p(\mathbf{x})$ is the characteristic function of the material point, and $S_i(\mathbf{x})$ is the node shape function. The role of the weighting function is the same as the shape function in conventional MPM. The modified equation of momentum conservation, Bardenhagen and Kober (2004), can be written as

$$\begin{aligned}&\sum_p \int_{\Omega\cap\Omega_p} \frac{\dot{p}_p\chi_p}{V_p}\cdot\delta\mathbf{v}d\mathbf{x}+\sum_p \int_{\Omega\cap\Omega_p} \sigma_p\chi_p:\delta\mathbf{v}d\mathbf{x}\\&=\sum_p \int_{\Omega\cap\Omega_p} \frac{m_p\chi_p}{V_p}\mathbf{b}\cdot\delta\mathbf{v}d\mathbf{x}+\int_{\partial\Omega}\mathbf{c}\cdot\delta\mathbf{v}d\mathbf{x}\end{aligned}, \tag{7}$$

where $\delta\mathbf{v}$ is an admissible velocity field, $\dot{p}_p$ is the rate of change of the material point momentum. Eq. (7) can be further discretized and solved at the grid nodes, Bardenhagen and Kober (2004). Herein, the weighting function $\overline{S}_{ip}$ is C^1 continuous under the spatially fixed background grid. Consequently the noises associated with material point crossing cell boundaries in the conventional MPM can be minimized.

In this paper, we have implemented GIMP presented by Bardenhagen-Kober for two-dimensional simulations. We have also developed a refinement technique and a parallel processing scheme to extend the serial GIMP algorithm to code large scale parallel computing. The capability of parallel GIMP computing has been demonstrated by modeling nanoindentation problem. A contact algorithm has been developed to address the contact problem between the rigid indenter and the deformable workpiece. We proceed next to describe the contact algorithm developed in this investigation.

2.1 Contact Algorithm in GIMP

Nanoindentation involves a contact pair of a rigid indenter and a deformable workpiece. The contact interaction between these two surfaces is governed by the Newton's third law and Coulomb's friction law as well as the boundary compatibility condition at the contact interface, Oden and Pires (1983), Zhong (1993). While MPM can prevent the penetration at the interface automatically, it uses a single mesh for the two bodies. At the contact surface, all components of the variables are interpolated to the nodes from both bodies using Eqs. (3)-(5). As a result, MPM using a single mesh tends to induce early contact in approaching and late separation when two parts move away from each other. So, MPM cannot model the contact behavior between two parts correctly. Hu and Chen (2003) proposed a multi-mesh MPM algorithm to release the no-slip constraint inherent in the MPM using a single mesh. In the multi-mesh MPM, in addition to a common mesh for all objects, there is an individual mesh for each of the objects under consideration. All meshes are identical, i.e. nodal locations are the same. The multi-mesh can be generated by creating multiple nodal fields for each node. Each nodal field corresponds to an object. In multi-mesh MPM scheme, the nodal masses and forces are mapped from the material points of each object to its own mesh. The nodal values are transferred to the corresponding nodes in the common mesh. When the values at a node of the common background mesh involve contributions from two parts, the contact between two parts occurs so that this node is defined as an overlapped node. Otherwise, two parts move independently. This multi-mesh algorithm can handle sliding and separation for the contact pair. However, in using the multi-mesh for contact problem in GIMP, the interaction at the overlapped nodes is still activated too early before the actual contact of the material points occurs.

Fig. 1 is an example illustrating early contact when Part 1 is moving toward Part 2. The four bottom particles of Part 1, labeled in hollow circles, have come into the cells of Part 2, the nodes of which cells are labeled in three dashed circles. Physical variables (e.g., normal force, and velocity component normal to the contact surface) in Part 1 will be interpolated onto these three overlapped nodes. The physical variables in the three overlapped nodes will be further interpolated into material points within the top layer of cells in Part 2, and contribute to the stress and deformation in the entire Part 2. With this treatment in the previous multi-mesh algorithm, even though Parts 1 and 2 are not in physical contact, the particles in Part 2 will contribute to the physical variables of particles in Part 1, leading to numerical early contact, and vice versa, through the overlapped nodes. Similar situation occurs when Part 1 is retracting from Part 2, resulting in late separation of two parts. Unless other measures are taken to prevent these physically incorrect early contact and late separation problems, they could cause large errors in GIMP and must be corrected in contact problems.

In this paper, a new contact algorithm is developed for GIMP simulations. Fig. 2 illustrates the contact algorithm for the contact pair between a rigid indenter and a deformable workpiece. Although circular points are

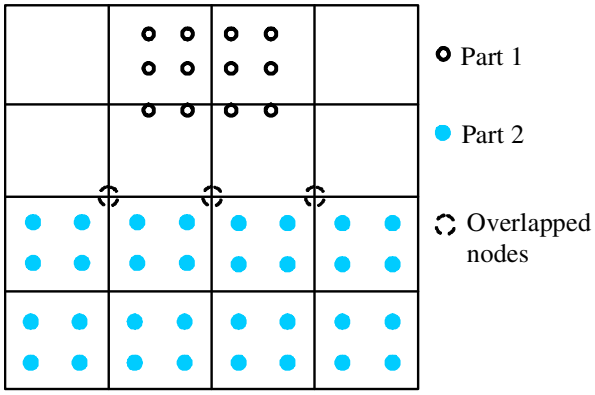


Figure 1 : Illustration of early contact in multi-mesh

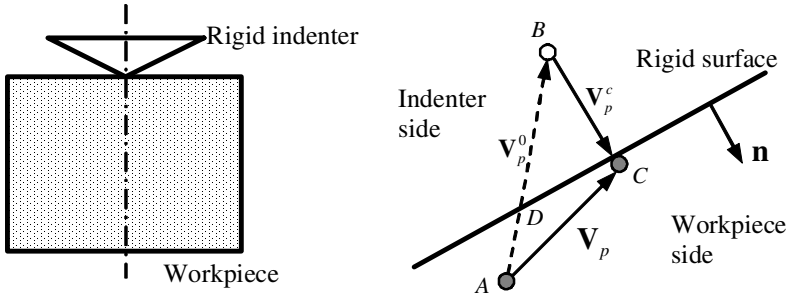


Figure 2 : Schematic of contact algorithm between the rigid indenter and the deformable workpiece

used in this schematic diagram, it should be noted that the points are representations of areas occupied by these points, based on the GIMP algorithm. A frictionless contact is assumed in this investigation. At the beginning of a time step, a material point is located at point A. At the end of this time step, the material point moves to B, if there is no contact interaction.

To satisfy the displacement compatibility condition, the material point has to be brought to the indenter surface and kept in contact with the indenter. The contact velocity correction $\mathbf{V}_p^c$ can be determined based on the rigid surface orientation indicated by its unit outward normal vector $\mathbf{n}$. The final location of the material point is set to C by a contact pressure. Hence, the velocity of a material point p under contact can be determined by

$$\begin{aligned}\mathbf{V}_p &= \sum_i^N \mathbf{V}_i \overline{S}_{ip} = \sum_i^N \frac{\mathbf{p}_i^0 + (\mathbf{F}_i^0 + \mathbf{F}_i^c)\Delta t}{m_i} \overline{S}_{ip} \\ &= \sum_i^N \frac{\mathbf{p}_i^0 + \mathbf{F}_i^0 \Delta t}{m_i} \overline{S}_{ip} + \sum_i^N \frac{\mathbf{F}_i^c \Delta t}{m_i} \overline{S}_{ip}\end{aligned} \tag{8}$$

where N is the number of nodes contributing to material point p, $\mathbf{F}_i^c$ is the contact force on node i, $\mathbf{p}_i^0$ and $\mathbf{F}_i^0$ are the nodal momentum and force without consideration of the contact, respectively. The velocity $\mathbf{V}_p^0$ of the material point without the consideration of contact is given by

$$\mathbf{v}_p^0 = \sum_i^N \frac{\mathbf{p}_i^0 + \mathbf{F}_i^0 \Delta t}{m_i} \overline{S}_{ip} \tag{9}$$

The contact force $\mathbf{F}_i^c$ on node i is the resultant of the contact pressures on the neighboring particles, and can be computed in terms of contact pressures using the approach given by Bardenhagen and Kober (2004), i.e.,

$$\mathbf{F}_i^c = \sum_{q=1}^{Q} \int_{\partial\Omega_q} S_i(\mathbf{x}) \mathbf{P}_q^c ds, \tag{10}$$

where Q is the total number of material points in contact with the indenter. If the contact pressure $\mathbf{P}_q^c$ is assumed to be constant in the contact area occupied by material point q, we have $\mathbf{F}_i^c = \sum_{q=1}^{Q} T_{iq} \mathbf{P}_q^c$, where $T_{iq} = \int_{\partial\Omega_q} S_i(\mathbf{x}) ds$. Since $\mathbf{V}_p = \mathbf{V}_p^0 + \mathbf{V}_p^c$, the contact velocity $\mathbf{V}_p^c$ for material

point p is given by

$$\mathbf{V}_p^c = \Delta t \sum_i^N \frac{\overline{S}_{ip}}{m_i} \sum_q^Q T_{iq} \mathbf{P}_q^c \tag{11}$$

Eq. (11) can be established for each material point in contact. At each material point there is an unknown contact pressure $\mathbf{P}_q^c$. Therefore, the number of unknown pressures, $\mathbf{P}_q^c$, is equal to the number of points in contact. In parallel computing, points in contact might be located in different domains processed by different processors. Consequently, a parallel solver is needed to solve Eq. (11) in this investigation. Since contact can only occur on the outer surface of an object, Eq. (11) is solved analytically under the physical contact condition $\mathbf{P}_q^c \cdot \mathbf{n} < 0$ to find the contact pressure at all material points in contact with the indenter. The contact pressure is then extrapolated to the nodes from the contact material points to update the total nodal forces.

2.2 Numerical Implementation

We consider the case where initially there are four material points in a cell for which the 2D weighting function is depicted in Fig. 3. To compute the weighting function, we take $\chi_p(\mathbf{x})$ to be one in the current region occupied by the material point p and zero elsewhere. In this figure, one node is at the origin and the horizontal axes give material point positions normalized by the cell size. Fig. 3 is based on the same material point characteristic function and node shape function as in Bardenhagen and Kober (2004). It is noted that the computation of the weighting function in the deformed state involves some practical difficulties because the integration boundaries in Eq. (7) can be difficult to obtain. To circumvent this problem, we assume that the shape of the region occupied by the four material points remains rectangular without rotation, so that Eq. (6) can be evaluated analytically. This assumption leads to significant saving in the computational time while introducing only small errors. Using this assumption, GIMP is extended to 2D simulations and the results are presented in Section 4.

3 Parallel Computing Scheme Using GIMP with SAMRAI

3.1 Structured Adaptive Mesh Refinement Application Infrastructure (SAMRAI)

The Structured Adaptive Mesh Refinement Application Infrastructure (SAMRAI), a scientific computational package for structured adaptive mesh refinement and parallel computation, is used with the GIMP for parallel computation of large-scale simulations. SAMRAI is chosen because of its similarity in grid structure with GIMP. In GIMP, the computation is usually independent of the background grid mesh so that a structured spatially-fixed mesh can be used throughout the entire simulation process. This advantage makes GIMP highly suitable for parallel computation, as the domain decomposition for structured mesh can be easily performed and no remeshing is required. Thus the complexity and inefficiency associated with parallel processing can be avoided.

In SAMRAI, the computational domain is defined as a hierarchy of nested grid levels of mesh refinement, Berger and Oliger (1984), as shown in Fig. 4. Each grid level is divided into non-overlapping, logically-rectangular patches, each of which is a cluster of computational cells. Indices are used extensively in SAMRAI to manage grid levels and patches. For example, patch connectivity is managed by the cell indices. The organization of the computational mesh into a hierarchy of levels of patches allows data communication and computation to be expressed in geometrically-intuitive box calculus operations. Communication patterns for data dependencies among patches can be computed in parallel without inter-processor communications, since the mesh configuration is replicated readily across processor memories. Inter-processor communications, i.e., data communications between patches on the same as well as neighboring levels, are pre-defined by SAMRAI communication schedules. Problem-specific communication interfaces are also provided by SAMRAI.

SAMRAI supports several data types defined in a patch, such as cell-centered data, node-centered data, and face-centered data. These data are stored as arrays to allow numerical subroutines to be separated easily from the implementation of mesh data structures. User-defined data structures over a patch, which can be accessed through cell index, are supported by SAMRAI. These characteristics make SAMRAI a very flexible parallel com-

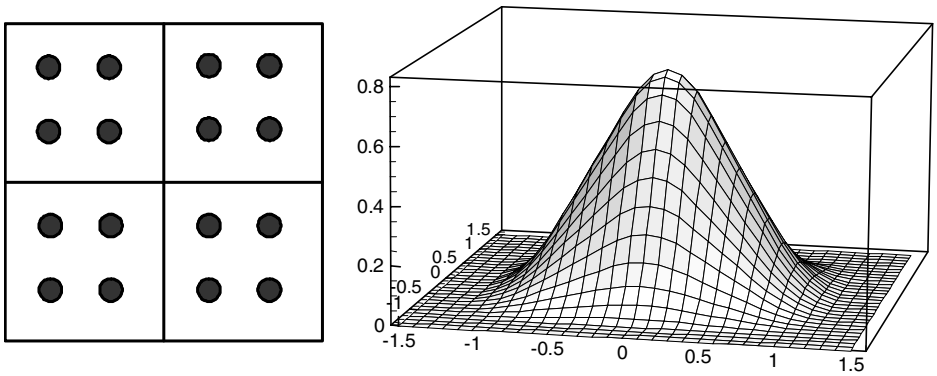


Figure 3 : Material points in cells and the weighting function in 2D GIMP

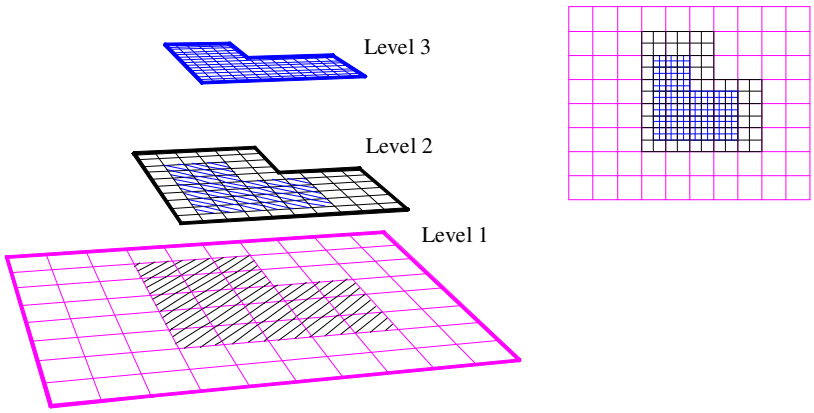


Figure 4 : Illustration of a hierarchy of three nested grid levels of mesh refinement

puting environment for numerous physics applications, Wissink, Hysom, and Hornung (2003).

3.2 Spatial and Temporal Refinements

In the application of SAMRAI to large-scale GIMP simulations, the techniques for refinement, both spatial and temporal, have to be developed to achieve high accuracy in areas of high stress/strain gradients while reducing the overall computational time by using coarse mesh in regions of low stress/strain gradients. Since a structured mesh is used in GIMP, the refinement can be implemented by imposing fine levels of sub-grids at locations of interest, using the approach adopted by Berger and Oliger (1984) in SAMRAI. The scheme for the structured grid refinement is illustrated in Fig. 4. The cell size ratio, also called the refinement ratio, of two neighboring levels is always an integer for convenience. The advantage of this refinement technique is that nesting relationships between different levels can be handled. A material point in GIMP can be split into several small material points. Tan and Nairn (2004) proposed a criterion to split material points based on local deformation gradient. If the refinement ratio is two in each direction, one coarse material point can be split into four material points in the next fine level in 2D case. However, this splitting technique can become complicated when conservation of energy and momentum have to be enforced. In this paper, a more natural refinement approach is developed to avoid direct splitting and merging processes by using material points of the same size and mass in overlapped region (called ghost region) between two neighboring levels.

Fig. 5 shows two neighboring coarse and fine grid levels in 2D GIMP computations with a refinement ratio of two. The thick line represents the physical bound-

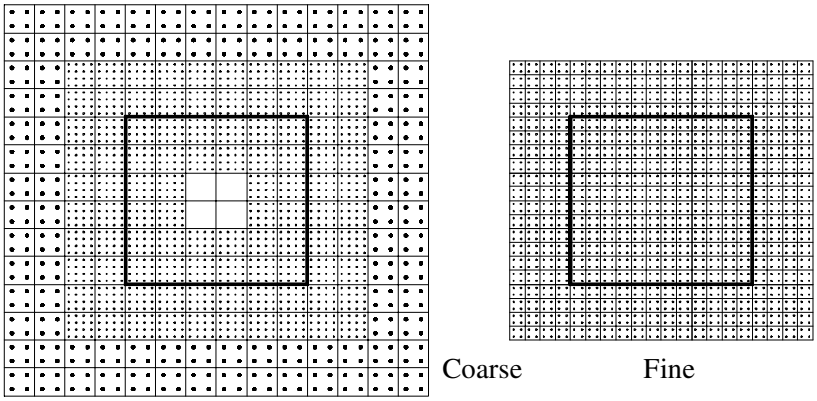


Figure 5 : Two neighboring coarse and fine grid levelsin 2D GIMP computations

ary of the fine level with four layers of ghost cells. Initially, four material points are assigned to each cell at the fine level. At the coarse level, the portion overlapped by the fine level is assigned with 16 material points per cell. Hence, these material points have the same size and initial positions as those at the fine level. The rest of the coarse level is assigned with four material points per cell. GIMP provides a natural coupling of the material points with different sizes at the same grid level. This is because the weighting function depends on the characteristic size of the material points and cell length and the interpolation between the nodes and material points is weighed by the mass of the material point. In GIMP computation, each level is computed independently with the physical variables communicated through the ghost regions between neighboring levels. Two data exchange processes, namely, refinement and coarsening are used in the communication. Refinement process passes information from the coarse level to the immediate fine level, while coarsening process will pass information from fine level to the next coarse level. In the refinement process, physical variables at the fine material points inside the thick lines in Fig. 5 are copied directly to replace the material points at the coarse level. In the coarsening process, the physical variables at coarse material points are copied to the ghost cells of the immediate fine level.

In the refinement, the material points located in the ghost cells at the fine level are eliminated first, and the material points in the corresponding region of the coarse level (with the same size as points in the immediate fine level) are copied to ghost cells at the fine level. In the copying process, if some (small) material points in the coarse level fall into the interior (inside the square on the right of Fig. 5) of the immediate fine level, these points will not be copied to this fine level, as points in the fine level will be able to carry over all computations in the interior of the fine level already. In coarsening, the material points of the coarse level located in the region overlapped by the fine level (inside the square on the left of Fig. 5) are eliminated first, and the material points in the fine level are then copied to the immediate coarse level. With these refinement/coarsening operations, the material points can move around freely, including moving outside the original level in large deformations. To ensure that this coarsening process can still be performed reliably during deformation, sufficiently wide region of cells should be assigned with refined material points at the coarse level so that the ghost cells of the fine level always stay within the region with fine material points on the coarse level. At the coarse level, the interior cells covered by the fine level do not participate in the computation and there are no material points inside (see Fig. 5).

The refinement techniques can be applied for multiple times at the regions of interest, such as the stress concentration regions. A fixed refinement ratio of two between two neighboring levels is very effective in reducing the total number of computational cells. Fig. 5 shows nested multi-level refinement and its corresponding relation between the total number of cells and the number of grid levels. The cell percentage represents the ratio of the total number of cells with multi-level refinement mesh to the total number of cells with one-level finest mesh. If each fine level occupies one quarter of the neighboring coarser level, as shown in Fig. 6 (a), the cell percentage as a

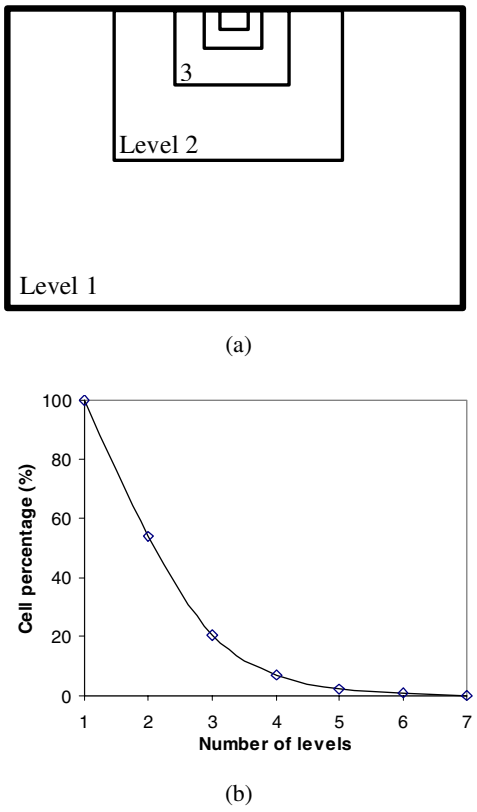


Figure 6 : Nested multi-level refinement and reduction in the number of cells with number of levels

function of the number of grid levels can be calculated, as shown in Fig. 6 (b). For example, when totally four levels of successive refinements are used the total number of cells is about 8% of that of one uniform fine mesh. A reduction in the number of computational cells leads to a reduction in the number of material points. Hence, the total amount of computational time can be reduced significantly. However, refinement and coarsening communications will cost additional computational time, as will be discussed in Section 4.

Another advantage of the multi-level refinement is that it allows for temporal refinement. Since the computation at each grid level is conducted independently, different time step increments can be used for computation at different levels. For example, a smaller time step increment can be used for the fine level to improve computational accuracy, while a larger time step increment can be used for the coarse level. Since the refinement ratio is an integer, the time step increment ratio should also be an integer for convenience in the computation and data communication/synchronization. For example, in Fig. 5, when the refinement ratios in both directions are fixed at two, the time step increment ratio should be set to two as well. As a result, two time step computations are performed at the fine level, and results are passed over to the immediate coarse level to couple with the results at the coarse level.

3.3 Domain Decomposition

GIMP uses structured mesh, consistent with SAMRAI, so that domain decomposition is straightforward and no remeshing, in general, is necessary. Fig. 7 (a) shows a two-dimensional computational domain decomposed into two patches separated by a horizontal dash line. The elliptical solid object with different boundary conditions applied at different regions is inside this domain/grid. After discretization, there are a certain number of material points and part of the boundary in a patch, which will be computed individually. It may be noted that patch boundary does not have to coincide with the boundary of the material continuum. The patch boundary is always chosen to be larger than the region occupied by the material continuum so that there is extra space for the material to deform. This will not cause any additional computational burden as the GIMP computation is only carried out on material points inside the patch. Each patch can be processed by a single processor and the convenience in creating patches will provide great flexibility in parallel processing.

Communication between two neighboring patches is realized through information sharing in the region overlapped by the two patches. The overlapped regions are also called 'ghost' regions, as shown in Fig. 7 (b). The ghost cells are denoted by dash lines. For ease of visualization, only the ghost cells overlapped by the other patch are shown and the ghost cells along the other three sides of a patch are not shown. On one grid level, patches can communicate with each other by simply copying data from one patch to another at the same computation time (Fig. 7 (b)). Using the material point information from the previous time step, and the physical boundary conditions, each patch is ready to advance one more time step. At this time, the material point information in the outermost layer of the ghost cells becomes inaccurate. For instance, one outermost grid node in patch one, marked by the circle, obtains information from eight material points before advancing to the next step. After advancing, it

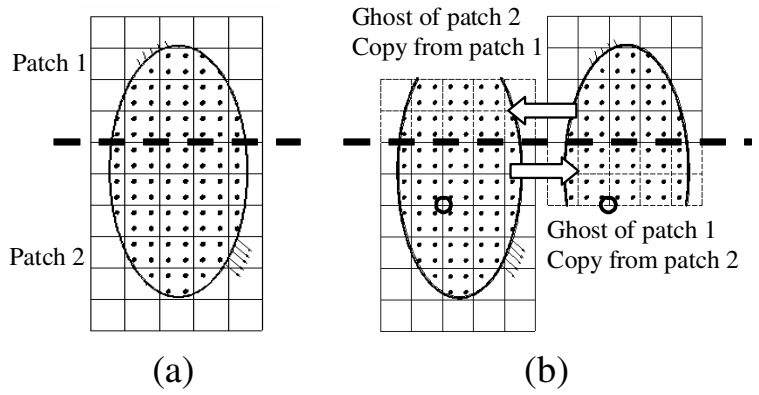


Figure 7 : A computational domain of two patches in one grid level

extrapolates to eight material points. However, in patch two, the grid node at the same location obtains information from sixteen surrounding material points. It extrapolates to these sixteen material points after advancing. Typically, after each step, the material points in the next inner layer in the ghost region become inaccurate as well.

Ghost cells and material points are attached to each patch to ensure accuracy of the interior. Each patch can be computed independently for one GIMP step since the momentum conservation equation is solved at each node and there are no coupled equations to solve. No data exchange is necessary during the GIMP step. Therefore, different patches can be assigned to different processors for parallel processing. After one GIMP step, the data in the ghost cells will be updated.

Copying material points to ghost cells involves data exchange between processors, which costs additional time. The more the number layers of ghost cells, the longer the time needed for communication, but communication can be performed less frequently. A minimum of two layers of ghost cells are necessary to ensure that computation at the material points inside a patch is always correct. If three levels of ghost cells are chosen, the communication can be performed after every two increments of each patch.

With these refinements and domain decomposition schemes for GIMP, it is possible to implement GIMP into the SAMRAI platform. In this study, the refinement ratio is chosen as two. Four layers of ghost cells are augmented to a patch such that data communications, including both data exchange on the same level and between neighboring levels, are performed every two time-steps for each fine grid level. This is critical because data exchange between levels has to be performed when the two levels are synchronized.

Fig. 8 shows the flowchart advancing all grid levels recursively starting from the coarsest level for one coarsest time step. It may be noted that the sequential GIMP algorithm can be used to advance each patch without modification.

4 Numerical Examples and Results

A Beowulf Linux cluster of 8 identical PCs were used in the simulations. Each PC has a Pentium 4 processor with a 2.4 GHz CPU, 512 MB RAM except that the master node has a memory of 1 GB. A gigabit switch is used to connect the network.

Two examples are used for validation of the 2D parallel GIMP computing under SAMRAI platform. The first example is simple tension of polycrystalline silicon under plane strain conditions. The material is assumed to be homogeneous, isotropic, and linear elastic. The Young's modulus is 170 GPa and the Poisson's ratio is 0.18. One end is constrained along the X- direction while a normal traction is ramped up on the other end. The size of the tensile model is 0.06 mm $\times$ 0.04 mm. The length of a square grid cell is 0.002 mm and the time step is 5×10^4 ps. For verification, the same problem was simulated using both conventional MPM and FEM (ABAQUS/Explicit). Fig. 9 shows GIMP, MPM and FEM simulation results of normal stresses in the X-direction at different increments from a simple tension problem. The simulation using the conventional MPM in

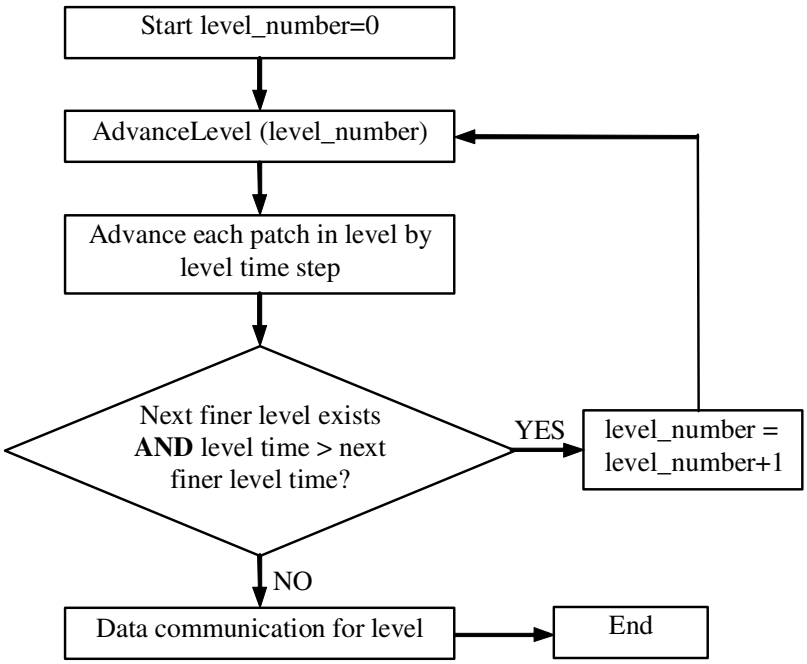
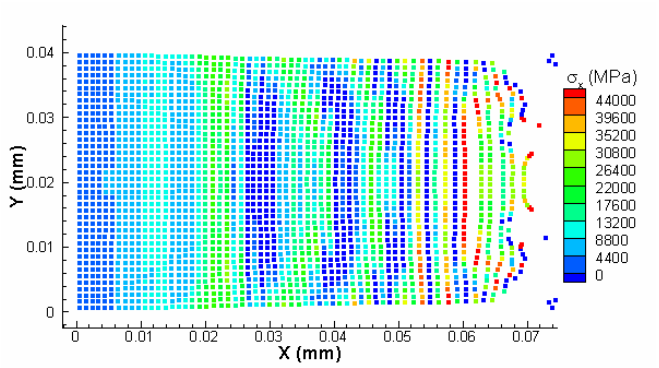


Figure 8 : Flowchart showing advancement of grid levels recursively starting from the coarsest to the finest level in GIMP

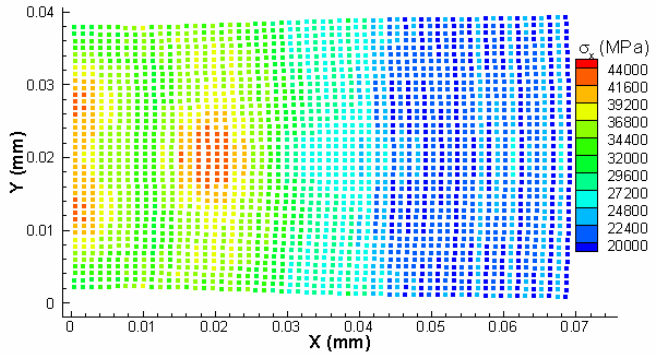
Fig. 9 (a) shows material separation close to the free end with severe numerical instability after 275 increments. Fig. 9 (b) and (c) show the normal stress distribution in the tensile direction and deformation after 500 increments from FEM and GIMP. It may be noted that FEM results show a stress contour plot on the deformed mesh while the GIMP results show a discrete scattered plot of material points. These two results are in good agreement with the difference in the maximum value being less than 10%.

In the second example, indentation on the same silicon material is simulated. The workpiece is subjected to a pressure applied in the middle of the top surface (Fig. 10 (a)) under plain strain conditions with a thickness of 0.001 mm for computing the mass and forces. The magnitude of the pressure increases linearly with time for the first 1500 increments, and is then kept constant (see Fig. 10 (b)). The cell size is 0.001 mm in both directions and the time step is 20 ps for both FEM and GIMP simulations. Due to symmetry, only one half of the workpiece is modeled. This simulation is performed with two patches in one uniform grid level. Two processors are used and one patch is assigned to each processor. Fig. 11 shows GIMP and FEM results of normal stresses in the Y-direction at different increments. The dashed line in Fig. 11 (a) is the boundary between the two patches. Fig. 11 (a) and (b) are plots of normal stresses in Y-direction at 500 time increments for GIMP and FEM simulations. The difference in stress values in Fig. 11 (a) and (b) is less than 5%. It should be noted that the FEM simulation aborted at 1348 increments due to excessive element distortion. The GIMP simulation did not encounter this problem. Fig. 11 (c) shows the GIMP stress result after 2000 increments. This demonstrates the capability of GIMP in handling excessive distortions.

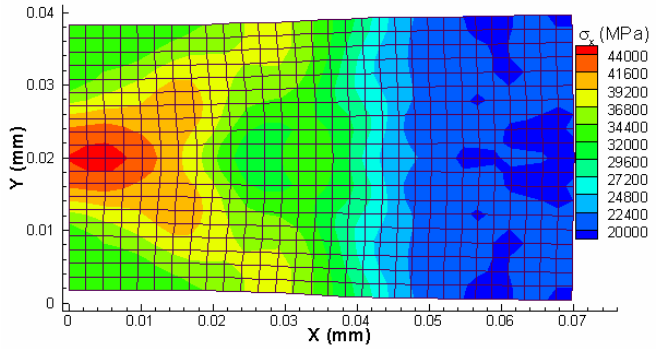
In order to validate the multi-level refinement algorithm and parallel communication, as well as the proposed contact algorithm, a simulation of nanoindentation with a wedge indenter was conducted under 2D plane strain conditions. The workpiece is aluminum and the indenter is assumed to be rigid. Fig. 11 shows the indentation model. The area below the indenter where high stress gradients are expected is refined, as shown in Fig. 12 (a). A prescribed velocity was applied on the indenter, as shown in Fig. 12 (b). The work piece dimensions are 60 μm $\times$ 40 μm. It is fixed in the Y-direction at the



(a) Conventional MPM

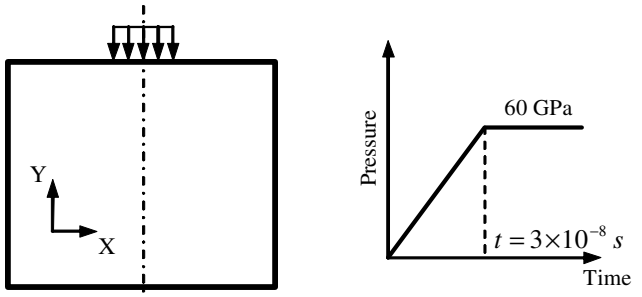


(b) GIMP



(c) FEM

Figure 9 : Simulation results of tensile stress contours for a simple tension problem



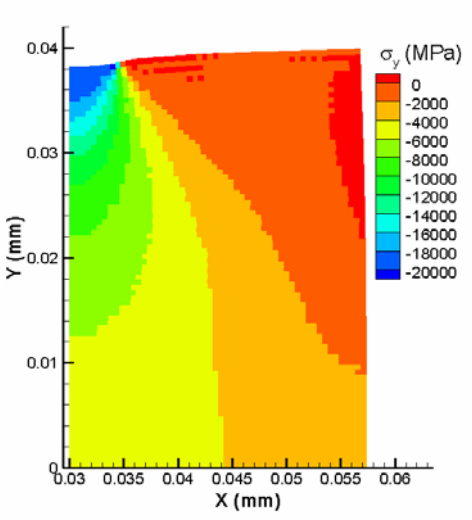
(a) (b)

Figure 10 : Loading conditions for a simple indentation problem

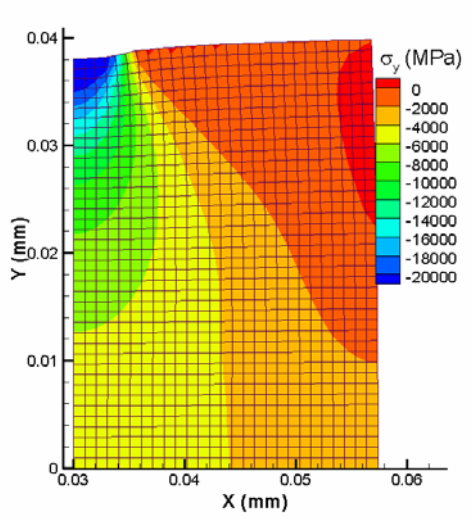
bottom. Only half of the model is simulated because of symmetry. The cell sizes are 500 nm, 250 nm and 125 nm for levels 1, 2 and 3, respectively. Each level is divided into four patches with approximately the same size. The maximum indentation depth in the simulation is about 450 nm. The dotted lines in Fig. 12 (a) illustrate the four patches in level 1. For comparison, an explicit FEM simulation (using ABAQUS/Explicit) was carried out under the same conditions. The FEM element size is uniform and is the same size as the finest GIMP background grid size. In this example, the maximum indentation depth (450 nm) is relatively small compared to the finest cell size, so that FEM simulation has not encountered excessive mesh distortion.

Fig. 13 shows a comparison of contours of normal stresses in the Y-direction at the maximum depth for both FEM and parallel GIMP simulations. The axis of symmetry of the workpiece is located at X=0.03 mm. For FEM, the plot is the contour of nodal stresses with deformed positions, and for GIMP, it is a discrete scattered plot of stress at deformed material points. The area below the indenter with high stress gradients is refined as shown in Fig. 12 (a) for parallel GIMP computation. The borders of grid levels 2 and 3 can barely be seen in Fig. 13 (b) due to the use of high material point density. Fig. 14 is a close-up view of shear stresses in which the three grid levels are shown. Results show that the normal and shear stresses from both parallel GIMP and FEM simulations agree very well. The difference of the normal stresses in the Y-direction for the material point in contact with the indenter tip and the stress of the FEM node at the same location is 4.4%. It may be noted that some non-smoothness in the GIMP stresses around the level boarders can be seen. This non-smoothness is caused by the refinement and coarsening and the error associated with this is negligible for these simulations.

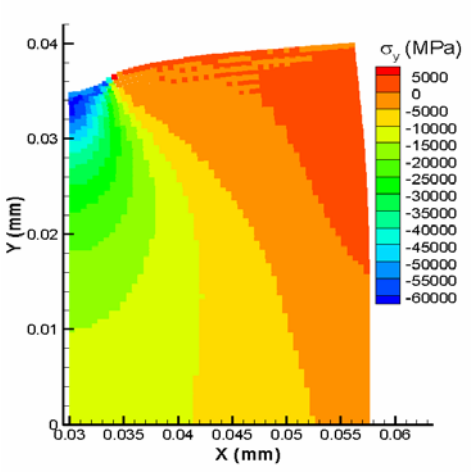
GIMP simulations using a uniform cell size of 500 nm and 125 nm were performed under the same conditions as in Fig. 12 to further verify the refinement/coarsening algorithm. Fig. 15 shows normal stresses in the Y-direction



(a) GIMP at 500 increments

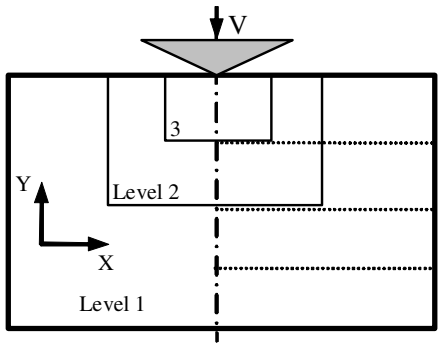


(b) FEM at 500 increments

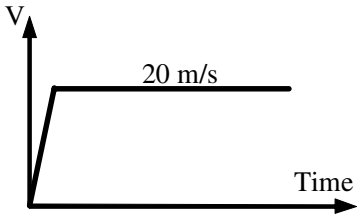


(c) GIMP at 2000 increments

Figure 11 : GIMP and FEM results of normal stress variation in the Y-direction at different increments



(a)

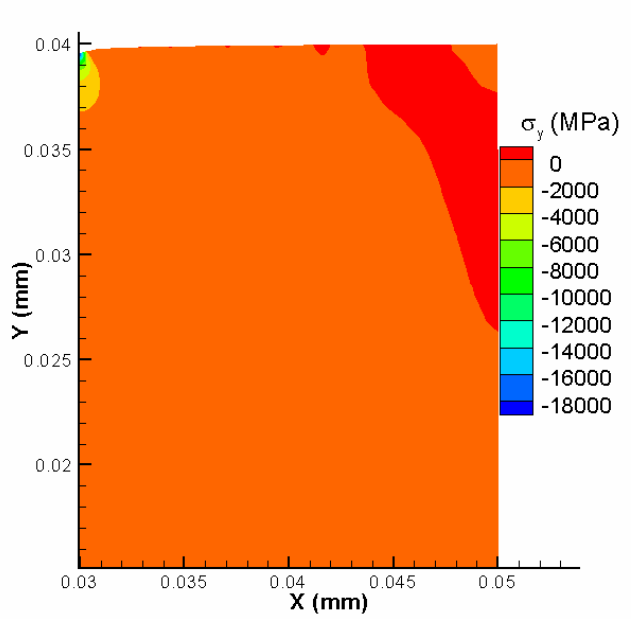


(b)

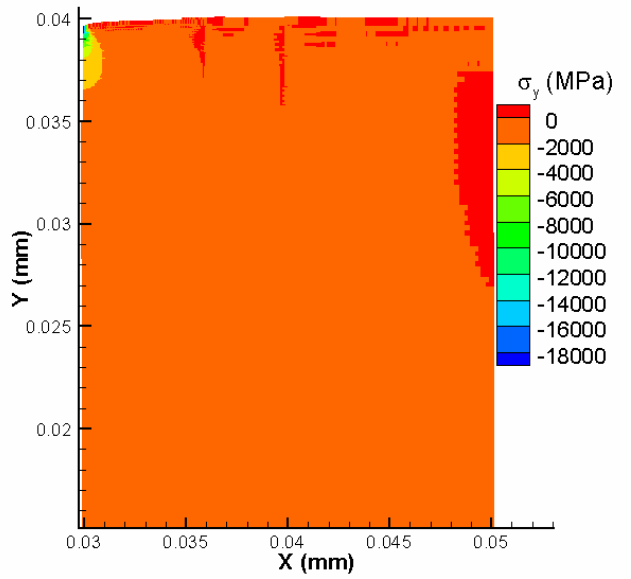
Figure 12 : Schematic of 2D indentation showing (a) three levels of refinement and (b) the indenter velocity history

and shear stresses in GIMP simulations using 500 nm uniform cells. In this case the material points in contact with the indenter are 16 times (4 times in each direction) larger than those with two levels of refinements. In general, the stress magnitudes agree with those in Fig. 13 and Fig. 14.

Fig. 16 (a) shows indentation load versus depth curves from FEM and GIMP with different grid sizes. From GIMP computations, the load versus depth curves with three levels of refinement agree very well with the results from a uniform finest mesh. The load versus depth curve from the FEM simulation with a uniform cell size of 125 nm under the same boundary conditions is plotted for comparison. It can be seen from Fig. 16 (a) that the trend of the load versus depth plots from FEM and GIMP simulations are similar. The difference in indentation load at the end of loading, which corresponds to 450 nm of indentation depth, is 5.9% between FEM and GIMP with 3 grid levels. When the depth is less than 100 nm, there is only one material point in contact with the rigid indenter. The assumption of constant pressure causes large differences under this circumstance. How-

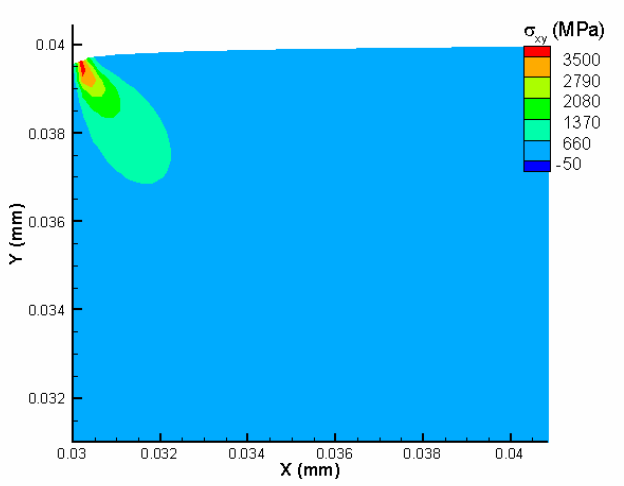


(a) FEM

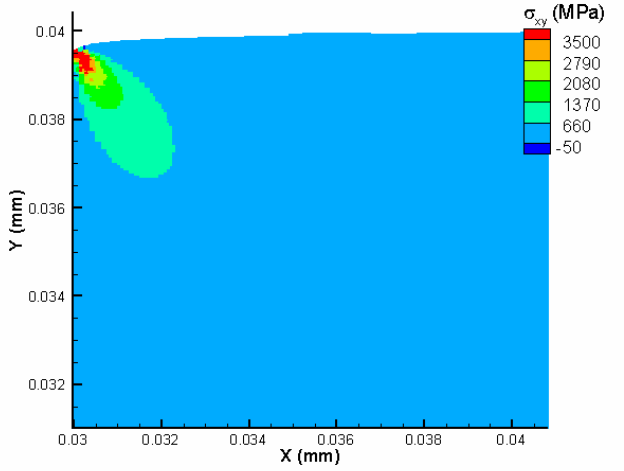


(b) GIMP

Figure 13 : Comparison of normal stresses in Y-direction from FEM and GIMP



(a) FEM

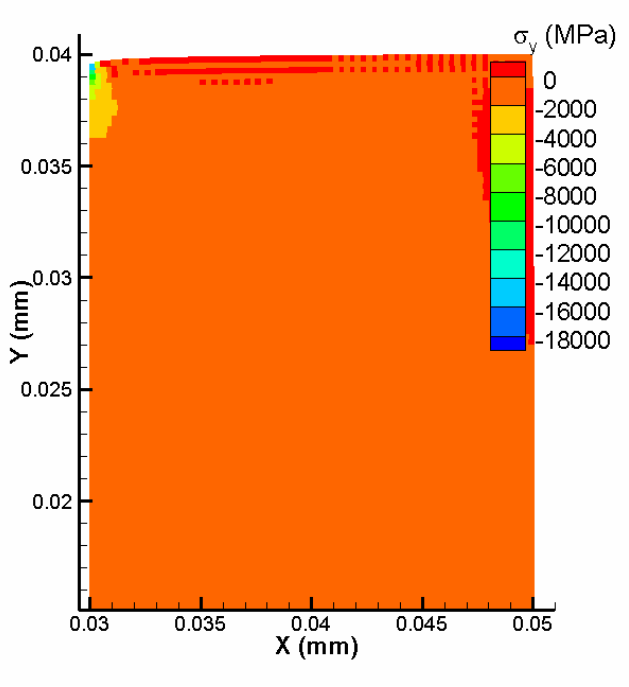


(b) GIMP

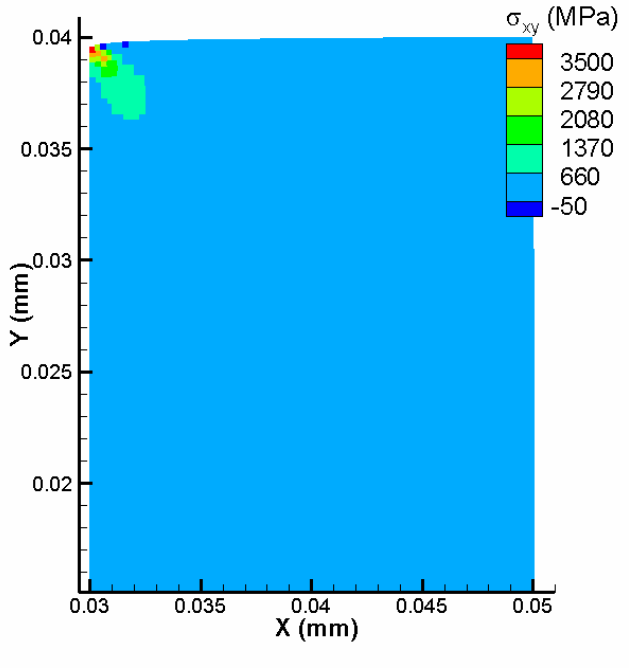
Figure 14 : Comparison of shear stresses from FEM and GIMP

ever, if the GIMP cell size is further refined to 62.5 nm and the size of the material point is 31.25 nm, the difference between GIMP and FEM becomes smaller, as can be seen in Fig. 16 (b).

Other simulations were conducted for the same problem with three levels of refinement using different number of processors to test the efficiency of parallel computing. The number of patches at each level is the same as the number of processors and the size of each patch is approximately the same. The resultant stress distribution and indentation load versus depth plots are the same as the previous results. The average time per computational step is 7.14 sec. when one processor is used and is reduced to 4.26, 3.40, 2.18 sec., respectively when two, three, and four processors are used. When four processors are used, the CPU time per step is only 30.5% of that of one processor. This gives a speed-up by a factor



(a) Normal Stress



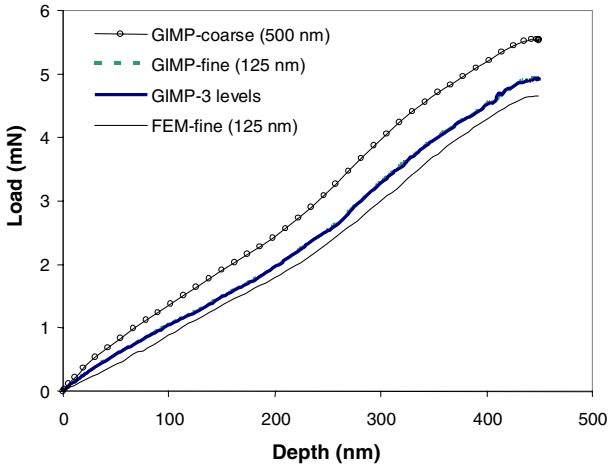
(b) Shear Stress

Figure 15 : Normal and shear stresses of GIMP simulations with a uniform cell size of 500 nm

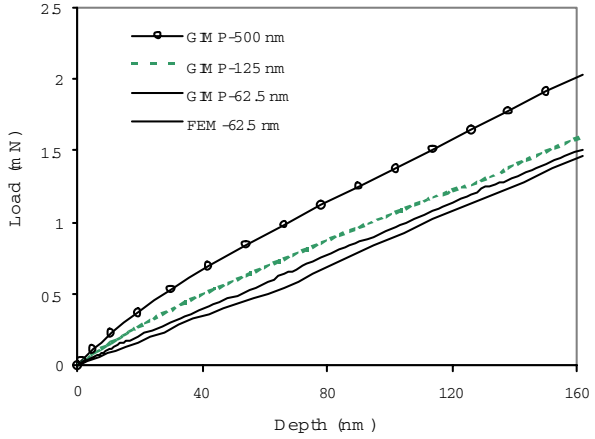
of 3.28. In the ideal case without communication overhead, the speed-up would be 4. The reduction in speed-up from the ideal number is because of the time involved in data communication between processors. It has been observed that the refinement and coarsening algorithm consume most of the communication time. Moreover, in refinement and coarsening, most of the time is taken to search for the corresponding material point in another grid level. This portion of the computational time can be reduced, if improved searching algorithm or more optimized algorithm for the storage of material points can be implemented.

The manual refinement for the indentation problem is adequate since the region of high stress gradient is known to occur below the indenter. The finest level covers the indenter and part of the specimen. With the same initial condition, the results at the finest level is identical to the results in the same area if a uniform fine mesh is used for the entire domain that requires much longer computational time. The computational load of each processor is balanced statically by assigning approximately the same number of material points to each processor. Dynamic load balance is supported by SAMRAI and can potentially improve the efficiency of the simulation.

To demonstrate the capability of the algorithm developed in this investigation for multiscale simulation, an indentation model with multiple length scales is simulated with eight processors. The dimensions of the workpiece are 0.25 mm $\times$ 0.125 mm. Initially, the velocity of the indenter increases from 0 to 150 m/s linearly with time and is then kept constant. Five successive levels of refinement are used in this simulation. The smallest material point represents an area of 64 nm $\times$ 64 nm, and the largest material point covers an area of 1 μm $\times$ 1 μm. Each level is divided into 8 equal-sized patches for best load balance. Since the contact surface can evolve into several patches, a parallel solver is implemented to solve Eq. (11) to find the contact pressure based on the Portable, Extensible Toolkit for Scientific Computation (PETSc). An aluminum workpiece is chosen with the Young's modulus and Poisson's ratio of 70 GPa and 0.33, respectively. The maximum indentation depth was 9.8 μm in this simulation (i.e., 153 times the size of the finest material point). It took nine hours to simulate this problem with eight processors. Fig. 17 (a) gives the normal stress distributions and Fig. 17 (b) shows normal stress distribution for the finest two levels. The relative large deformation in this multiscale nanoindentation problem could not be handled by FEM due to excessive distortion

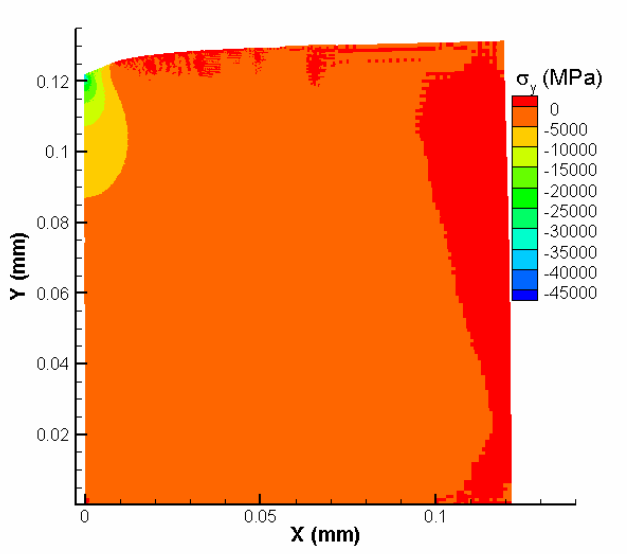


(a) FEM

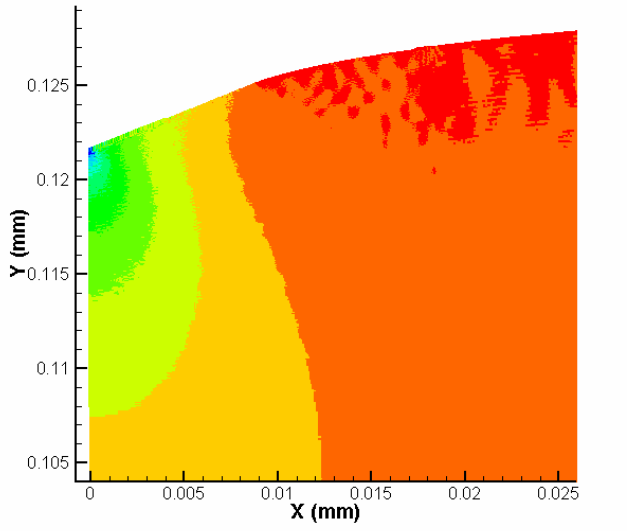


(b) GIMP

Figure 16 : Indentation load versus depth curves from FEM and GIMP with different grid sizes



(a) FEM



(b) GIMP

Figure 17 : Multiscale simulation of nanoindentation with five levels of refinement

in the FEM mesh. However, the parallel GIMP code was able to complete the entire loading/unloading processes without any difficulty. This example shows clearly the advantage of GIMP for multiscale simulations over FEM.

5 Conclusions

The following are specific conclusions based on the results of this investigation:

1. A 2D generalized interpolation material point (GIMP) method has been implemented to address problems, such as particle flying-off and alternating stress sign associated with conventional MPM in case of relatively large deformation.

2. To conduct multiple length scale simulations, a parallel computing scheme has been presented using GIMP under SAMRAI parallel computing environment in which multi-level grids are used for spatial and temporal refinements.

3. A refinement/coarsening algorithm, based on material points of GIMP in two grid levels, has been developed

for communication between neighboring grid levels of different refinements. With increase in the refinement levels, as well as decrease in the time step increments, the computational accuracy is greatly improved in the region of interest while the overall computational time is reduced. The computation at each grid level is performed recursively to ensure that the refinement and coarsening are performed when the two neighboring levels are synchronized.

4. 2D MPM and GIMP were applied to simple tension and indentation problems to validate the GIMP algorithm. GIMP results agree very well with FEM results for these two examples provided that the deformations are small. The noise and instability problems present in conventional MPM are not observed in the GIMP simulations.

5. As the deformation is increased, GIMP continued to execute while FEM aborted due to element distortion. Also GIMP results are stable. Thus GIMP is able to handle relatively large deformation problems.

6. For the nanoindentation problem, a GIMP algorithm for the contact between a rigid indenter and a deformable workpiece was developed. A reasonably good agreement between GIMP and FEM results was reached, validating the contact algorithm presented in this investigation.

7. Another nanoindentation example with multiple length scales from a few nanometers to sub-millimeters was simulated and numerical results validated the parallel GIMP computing with the use of SAMRAI.

Acknowledgement: The work was supported by a grant from the Air Force Office of Scientific Research (AFOSR) through a DEPSCoR grant (No. F49620-03-1-0281). The authors would like to thank Dr. Craig S. Hartley, Program Manager for Metallic Materials Program at AFOSR for his interest and support of this work. The authors also acknowledge Dr. Scott Bardenhagen for his introduction to the GIMP algorithm and valuable comments, and Dr. John A. Nairn for providing the 2D MPM code.

Disclaimer

References

Atluri, S.N.; Shen, S. (2002): The meshless local Petrov-Galerkin (MLPG) method: a simple & less-costly alternative to the finite element and boundary element methods.*CMES: Computer Modeling in Engineering & Sciences,* vol. 3, no. 1, pp. 11-51.

Bardenhagen, S.G. (2002): Energy conservation error in the material point method for solid mechanics. *Journal of Computational Physics,* vol. 180, pp. 383-403.

Bardenhagen, S.G.; Kober, E.M. (2004): The generalized interpolation material point method.*CMES: Computer Modeling in Engineering & Sciences,* vol. 5, no. 6, pp. 477-496.

Berger, M.J.; Oliger, J. (1984): Adaptive mesh refinement for hyperbolic partial differential equations. *Journal of Computational Physics,* vol. 82, pp. 484-512.

Hornung, R.D.; Kohn, S.R. (2002): Managing application complexity in the SAMRAI object-oriented framework. *Concurrency and Computation: Practice and Experience,* vol. 14, pp. 347-368.

Horstemeyer, M.F.; Baskes, M.I.; Prantil, V.C.; Philliber, J.; Vonderheide, S. (2003): A multiscale analysis of fixed-end simple shear using molecular dynamics, crystal plasticity, and a macroscopic internal state variable theory. *Modelling and Simulation in Materials Science and Engineering,* vol. 11, pp. 265-286l.

Hsien, S.H. (1997): Evaluation of automatic domain partitioning algorithms for parallel finite element analysis. *International Journal for Numerical Methods in Engineering,* vol. 40, pp. 1025-1051.

Hu, W.; Chen, Z. (2003): A multi-mesh MPM for simulating the meshing process of spur gears. *Computers & Structures,* vol. 81, pp. 1991-2002.

Kalia, R.K.; Nakano, A.; Greenwell, D.L.; Vashishta, P. (1993): Parallel algorithms for molecular dynamics simulations on distributed memory MIMD machines. *Supercomputer,* vol. 54, pp. 11-25.

Komanduri, R.; Lu, H.; Roy, S.; Wang, B.; Raff, L.M. (2004): Multiscale modeling and simulation for materials processing. *Proceedings of the AFOSR Metallic Materials (2306 AX) Grantees Meeting,* VA, USA.

Mackerle, J. (2003): FEM and BEM parallel processing: theory and applications-a bibliography. *Engineering Computation,* vol. 20, no. 4, pp. 436-484.

Oden, J.T.; Pires, E.B. (1983): Numerical analysis of certain contact problems in elasticity with non-classical friction laws. *Computers & Structures,* vol. 16, no. 1-4, pp. 481-485.

Shen, S.; Atluri, S. N. (2004a): Computational nanomechanics and multi-scale simulation. *CMC: Computers, Materials, & Continua*, vol. 1, no. 1, pp. 59-90.

Shen, S.; Atluri, S. N. (2004b): Atomic-level stress calculation and continuum-molecular system equivalence. *CMES: Computer Modeling in Engineering & Sciences*, vol. 6, no. 1, pp. 91-104.

Shen, S.; Atluri, S. N. (2005): A tangent stiffness MLPG method for atom/continuum multiscale simulation. *CMES: Computer Modeling in Engineering & Sciences*, vol. 7, no. 1, pp. 49-67.

Sulsky, D.; Zhou, S.J.; Schreyer, H.L. (1995): Application of a particle-in-cell method to solid mechanics. *Computer Physics Communications,* vol. 87, pp. 236-252.

Sulsky, D.; Schreyer, H.L. (1996): Axisymmetric form of the material point method with applications to upsetting and Taylor impact problems. *Comput. Methods Appl. Mech. Eng.,* vol. 139, pp. 409-429.

Tan, H.; Nairn, J.A. (2002): Hierarchical, adaptive, material point method for dynamic energy release rate calculations. *Computer Methods in Applied Mechanics and Engineering,* vol. 191, pp. 2095-2109.

Tewary, V. K.; Read D. T. (2004): Integrated Green's function molecular dynamics method for multiscale modeling of nanostructures: application to Au nanoisland in Cu. *CMES: Computer Modeling in Engineering & Sciences*, vol. 6, no. 4, pp. 359-371.

Wissink, A.M.; Hysom, D.; Hornung, R.D. (2003): Enhancing scalability of parallel structured AMRA calculations. *Proc. 17th ACM International Conference on Supercomputing (ICS03),* San Francisco, CA, pp. 336-347.

Zhong, Z.H. (1993): *Finite Element Procedures for Contact-Impact Problems,* Oxford University Press, New York.