THREE DIMENSIONAL GENERALIZED INTERPOLATION MATERIAL POINT (GIMP) SIMULATIONS IN SAMRAI ENVIRONMENT

By

ROHIT KUMAR RAGHAV

Bachelor of Engineering

MAHARSHI DAYANAND UNIVERSITY

Haryana, India 1999

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
December, 2005

THREE DIMENSIONAL GENERALIZED

INTERPOLATION MATERIAL POINT

(GIMP) SIMULATIONS IN SAMRAI

ENVIRONMENT

Thesis Approved:

Ranga Komanduri

Thesis Adviser

Hongbing Lu

Thesis Co-Adviser

Samit Roy

A. Gordon Emsile

Dean of the Graduate College

ACKNOWLEDGEMENT

I wish to express my sincere thanks to my parents for their confidence in me. I would like to thank them for their encouragement at times of difficulty, love and understanding. Special thanks are due to my father for his inspiration, motivation, love and guidance.

I would like to express my sincere appreciation to my advisor, Dr. Ranga Komanduri for his intelligent supervision, constructive guidance, financial support, inspiration, motivation and friendship. I would also like to express my sincere appreciation to Dr. Samit Roy and Dr. Hongbing Lu for providing encouragement and technical guidance throughout this study. I would like to express my gratitude to Dr. Bo Wang for his invaluable guidance.

This project has been funded by the grant (F49620-03-1-0281) from the Air Force Office of Scientific Research (AFOSR). The author thanks Dr. Craig S. Hartley for his interest and support of this work.

I thank Dr. R. Hornung, A. Wissink of LLNL for helping us with SAMRAI and PETSc support team for their help in installation and implementation of PETSc solver in GIMP 3D code.

I wish to express my sincere gratitude to Jin Ma for his assistance throughout this project. I wish to extend my gratitude to Nitin Dhaplapurkar, Nikhil Joshi and Yang Liu for their encouragement and friendship.

TABLE OF CONTENTS

Chapter Page

1. Introduction …………………………………………………………………….. 1

 1.1 The General Problem ……………………………………… 1

 1.2 Material Point Methods 5

 1.3 Structured Adaptive Mesh Refinement Infrastructure........... 6

 1.4 Outline of the Thesis 7

2. Literature Review.. 8

 2.1 Generalized Interpolation Material Point Method 8

 2.1.1 Particle-In-Cell Methods 8

 2.1.2 Material Point Method ……………………………………... 9

 2.1.3 Generalized Interpolation Material Point Method…...............11

 2.2 Parallel Processing ....…...........12

 2.3 Contact Mechanics ………………………………….........13

3. Generalized Interpolation Material Point Method Parallel Processing……..... 16

 3.1 Introduction ……………………………………………….... 16

 3.2 Governing Equations ……………………………………….. 16

 3.3 Numerical Implementation in 3D.. 19

4. Problem Statement 24

5. Contact Algorithm for Rigid Surface Indenter 27

5.1 Introduction.. 27

5.2 Contact Algorithm 29

5.3 Numerical Implementation………………………………...... 30

3.3.1 Detecting Contacts…...31

3.3.2 Finding outward normal and Contact velocity………………......32

3.3.3 Solving for contact pressure……...32

6. Parallel Computing Scheme Using GIMP with SAMRAI......………………..........34

6.1 Structured Adaptive Mesh Refinement Infrastructure...............34

6.2 Spatial and Temporal Refinements...…....37

6.3 Domain Decomposition……..41

6.4 Solver for contact algorithm……………………………….......45

6.4 Implementation of 3D-GIMP in SAMRAI……………………....48

7. Results and Discussions……..…..………………………………………….......….55

7.1 Reduction in Computational Time with parallel processing...57

7.2 Validation of 3D-GIMP results with parallel processing and advantages over FEM...60

7.3 Validation of 3D-GIMP results with 2 Levels Refinement and with one particle per cell..63

7.4 Results with 3 Levels of refinement consisting 4 Patches each level in 3D-GIMP using 4 Processors and comparison with FEM…………………………………………..........................69

7.5 Validation of 3D-GIMP results considering bilinear plastic material behavior……………………………………………74

8. Conclusions and Future Work ……………………………………...…...80

8.1 Conclusions……………………………………………….....81

8.2 Future Work………………………………………………...82

9. References.......……………………………………...………………........83

LIST OF FIGURES

Figure No. Page

1. Schematic of simulations at various levels……………………………………..3
2. Eight Material points in cell and the weighting function for 1D in GIMP method 20
3. Schematic of contact algorithm between the rigid indenter and the deformable work piece ……………………………………………………………………………28
4. A schematic of Conical and Spherical Indenters………………………………….31
5. Flow charts for detecting contact with indenter …………………………………..32
6. User view of SAMRAI [66] ………………………………………………………36
7. Schematic showing two neighboring coarse and fine grid levels with material points inside ………………………………………………………………………37
8. Schematic of coarse level only as shown in fig. 7 ………………………………..37
9. Schematic of fine level only as shown in fig. 7 ………………………………….38
10. Schematic of a model with Nested multi-level refinement and reduction in the number of cells with number of levels ……………………………………………40
11. A computational domain of two patches in one grid level ………………………..42
12. Flowchart showing advancement of grid levels recursively starting from the coarsest to the finest level in GIMP method [15] …………………………………44
13. Main member methods of class MPMSamrai3d and their functions ……………...49
14. SAMRAI "patch data": index data [66] …………………………………………...50

15. Loading conditions for a simple indentation problem ……………………………..55

16. Loading conditions for a simple indentation problem ……………………………57

17. Time taken per step for number of processors …………………………………..59

18. Comparison of results from GIMP 3D and FEM …………………………………62

19. One Material points in cell and the weighting function for 1D in GIMP method ..63

20. Comparison of results from GIMP 3D and FEM …………………………………69

21. 3D Sectioned view of indentation showing (a) three levels of refinement and (b) the indenter velocity history ……………………………………………………70

22. Comparison of results from GIMP 3D and FEM …………………………………73

23. Comparison of results from GIMP 3D 1-level and FEM coarse ………………….74

24. Schematic of the model used for indentation ……………………………………...75

25. Comparison of normal stresses in Z-direction from FEM and GIMP coarse ……..76

26. Comparison of normal stresses in Z-direction from FEM and GIMP fine ………..76

27. Time histories of important physical variables ……………………………………78

CHAPTER 1

INTRODUCTION

1.1 The General Problem

Material failure at all length scales experiences such processes as elastic deformation, dislocation generation and their propagation, cleavage, crack initiation and growth, and final rupture. Despite significant developments in materials simulation techniques, the goal of reliably predicting the properties of new materials in advance of material characterization and fabrication has yet to be achieved. It is also not possible to predict the properties of the material at nanoscale knowing the properties of the material at macro level or vice versa. This situation exists for several reasons that include a lack of reliable potentials to describe the behavior of the material at the atomistic level, computational limitations, inadequate modeling of the process, absence of scaling laws, and difficulties associated with the experimental measurement of properties even at the micro scale, let alone at the nano scale.

Multiscale simulation technique can be applied to reliably predict the properties of new materials in advance of material characterization and fabrication. Scaling laws governing the mechanical behavior of materials from atomistic (nano), via mesoplastic (micro), to continuum (macro) are very important to numerous DoD applications, such as the development of a new class of aircraft engine materials, or new steels for naval battle ships, or new tank armor materials for the army, or numerous micro electromechanical components for a myriad of applications, for the following three reasons: (1) information on the mechanical behavior of materials at nano level is not presently available as input to nanotechnology for the manufacturing of nano components or microelectro-mechanical systems (MEMS). For example, nanostructures may possess unique properties in view of their very high surface to volume ratio, or the nanostructures might be relatively free of defects with strength approaching the theoretical values, (2) applications where two length scales of different orders of magnitude are involved. For example, one is atomistic (nano) and the other mesoplastic (micro) as in nano indentation, or, one is mesoplastic (micro) and the other continuum (macro) as in conventional indentation, and (3) it may be possible to extend the knowledge accumulated over time on material behavior at the macro (or continuum) level to the atomistic (or nano) level, via mesoplastic (micro) level.

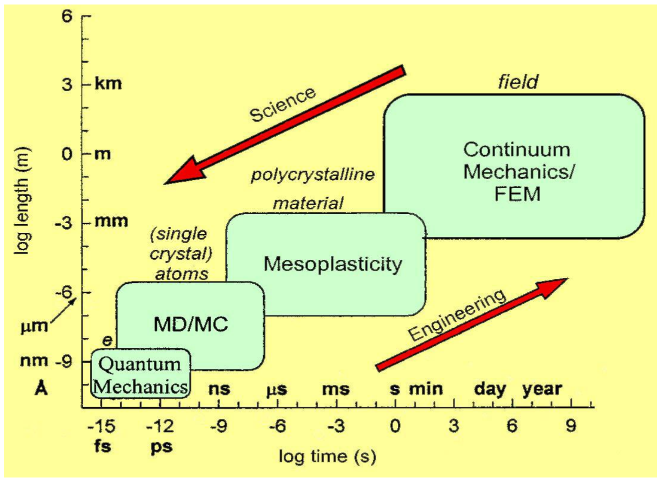


Fig. 1: Schematic of simulations at various levels (Komanduri, 2002)

The work described, herein, is part of a larger on-going research to aid materials modeling and simulation. The main objective of this on-going research is to develop a fundamental understanding of indentation and tension tests at various scales and develop a computer simulation code that would bridge the gap from atomistic to continuum, via. mesoplastic or microscopic behavior. Material multiscale simulations span from electronic structure, atomistic scale, crystal scale, to macro/continuum scale [18, 22]. 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 [18]. This investigation focuses on multiscale simulations e.g., from nanometer to millimeter, based on the continuum mechanics approach, for 3D Indentation problem.

At the continuum level, FEM is the dominant numerical technique used but there are advantages of using material point method (MPM) over FEM in some situations. The major advantage of material point method is that the particles are not treated as a mesh so that mesh entanglement is not a problem and large deformations can be treated with these methods. Creating new meshes and mapping between meshes is eliminated. The required more accurate and reliable solutions in numerical analysis of various large scale structural, fluid mechanics problems and design problems need more computationally intensive process, usage of more efficient methods and here implementation of parallel processing technique is one of the best economic alternatives [23]. Parallel processing in FEM is not that straightforward as it is in MPM, primarily due to the coupling of a large number of simultaneous linear equations.

The material point method (MPM), [25], has already demonstrated the capabilities in simulation of a wide range of mechanics problems including impact/contact/penetration and fracture. But to resolve alternating stress sign and instability problems associated with conventional MPM, Bardenhagen and Kober (2004) [13] introduced recently the generalized interpolation material point method (GIMP) and implemented for one-dimensional simulations. Komanduri, Lu, Roy, Wang, Hornung, Wissink , Ma (2004) [15] implemented it in 2D and simulated simple tension and indentation problems spanning multiple length scales, based on the continuum mechanics approach.

The present investigation implements the extended GIMP method presented by Bardenhagen-Kober to three-dimensional simulations in SAMRAI environment and applies it to solve indentation problems involving rigid surface indenter as an example to demonstrate the capability of 3D GIMP. Structured Adaptive Mesh Refinement Applications Infrastructure (SAMRAI) is used here as a platform for parallel computation. Solving indentation problems involves the contact issues between an indenter and work piece. A contact algorithm, which allows the contact interface to be located in a few computational domains, distributed over multiple processors is introduced in this study. Furthermore, a refinement technique for both temporal and special refinement in areas of interest and a parallel processing scheme are developed using SAMRAI so that the serial GIMP algorithm and code can be extended for parallel computation of large-scale continuum mechanics computations and to reduce computation time and computer memory requirement.

1.2 Material Point Method (MPM)

The material point method (MPM) has demonstrated its capabilities in addressing such problems as impact, upsetting, penetration, and contact [25, 26]. 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). As the dynamic analysis proceeds, the solution is tracked on the material points by updating all required properties such as position, velocity, acceleration, stress state, etc. At each time step the equation of motion for particles are solved on background calculation grid; this solution is used to update the particles and the background mesh can be discarded or reused for next time step in its initial, undistorted form [27]. The use of combination of Lagrangian and Eulerian methods on one hand avoid problems of mesh distortion associated with use Lagrangian mesh (FEM) by using fixed Eulerian mesh and on the other hand problems of numerical dissipation associated with Eulerian mesh is avoided since the variables are defined on material points which advect through Eulerian mesh [35]. 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 [13]. To solve the instability problems associated with the conventional MPM simulations, Bardenhagen and Kober recently proposed the generalized interpolation material point (GIMP) method and implemented for one-dimensional simulations, Ma, Lu, Wang, Roy, Hornung, Wissink and Komanduri [15] implemented the GIMP presented by Bardenhagen and Kober to two-dimensional simulations.

1.3 Structured Adaptive Mesh Refinement Application Infrastructure (SAMRAI)

A platform for parallel computation, namely, the structured adaptive mesh refinement application infrastructure (SAMRAI) [17], has been developed recently 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 computation. SAMRAI has also provided a foundation for parallel adaptive mesh refinement (AMR) with the use of either dynamic or static load balancing [28]. 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. In this investigation with the appropriate use of fine and coarse meshes in different regions, multiscale simulations using GIPM has been conducted with desired computational accuracy and with reduced costs associated with computer memory and computational time.

1.4 Outline of the Thesis

This thesis is divided into seven chapters. Chapter 1 gives an introduction to present investigation by briefly describing multi-scale simulations, Material Point Method (MPM), advantages of MPM over Finite Element Method, Some of Disadvantages of conventional MPM, how Generalized Interpolation Material Point (GIMP) method has evolved, Structural Adaptive Mesh Refinement Infrastructure (SAMRAI) and how their roles in the present investigation. In Chapter 2, a review of literature on topics of interest to the present investigation, namely Generalized Interpolation Material Point (GIMP) method, parallel processing and contact mechanics has been presented. Chapter 3 gives an introduction to the GIMP method, describe the governing equations involved in GIMP method and finaly their numerical implementation. Chapter 4 discusses the problem statement for this investigation. Chapter5 gives the detailed description of contact algorithm developed for GIMP for 3D Indentation problem using rigid surface Indenter. Chapter 6 discusses the parallel computation scheme for GIMP method using SAMRAI for solving 3Dimensional solid mechanics problems. This also involves refinement techniques for temporal and spatial refinement and parallel solver scheme for solving contact problems involving contact interface spanning over multiple processors.

Chapter 7 presents results and discussion of 3D-GIMP simulation for indentation using parallel processing and multi-level refinement scheme. These results are compared with FEM simulation using ABAQUS/Explicit code when possible. Chapter 8 gives conclusions arising out of the present investigation and offers some suggestions for future work.

CHAPTER 2

LITERATURE REVIEW

2.1 Generalized Interpolation Material Point (GIMP) Method

GIMP is a technique developed recently by Bradenhagen and Kober [13] to resolve some problems associated with MPM. MPM, as a computational algorithm for material continuum, was evolved from a method called particle in cell method. Based on the historical developments of these methods, we review the particle in cell method, the material point method in this chapter and GIMP in Chapter 3.

2.1.1 Particle-In-Cell Methods (PIC)

According to Harlow [1] the Particle-in-Cell (PIC) method was developed in 1955 at Los Alamos National Laboratory for the solution of complex fluid dynamics problems. PIC method is a combination of Lagrangian and Eulerian methods that naturally can handle no slip interfaces between materials and large slippage and distortions. Amsden [3] has developed a PIC code and explained the code in detail.

The general idea behind the PIC method is to solve the governing equations on an Eulerian grid where derivatives can be conveniently defined. Information is then transferred from the grid to Lagrangian material particles using mapping functions. The material particles move or convect and carry with them certain physical properties.

Variations on the method can occur by changing the mapping method involving the use of different mapping functions. In Harlow's classical version of PIC, velocities were mapped from the grid to the particle and in a less dissipative version called FLIP (FLuid-Implicit-Particle) [54, 55] material particle velocities are only updated from the grid solution.

The basic ideas behind the PIC or FLIP methods have been adapted recently by Sulsky, Zhou, Schreyer [25] to solid mechanics by revising FLIP-type algorithm. When the particle quantities are mapped to grid in preparation for solution at next time step the field variables such as velocity gradient, strain and stresses are evaluated at material points as opposed to grid in FLIP type algorithm, and the resulting approach is applied to impact problems with elastic and elastic-plastic constitutive equations.

2.1.2 Material Point Method (MPM)

Sulsky and Schreyer [26] presented the general description of the material point method (MPM), along with special considerations relevant to axisymmetric problems. The method utilizes a material or Lagrangian mesh defined on the body under investigation, and a spatial or Eulerian mesh defined over the computational domain. The set of material points making up the material mesh is tracked throughout the deformation history of the body and these points carry with them a representation of the solution in a Lagrangian frame. Interactions among these material points are computed by projecting information they carry onto a background mesh where equations of motion are solved. They reported that the material point method does not exhibit locking or an overly stiff response in simulations of upsetting.

The material point method (MPM) has used to solve various dynamic problems in solid mechanics [2, 25, 26]. In MPM, a solid body is discretized into a collection of points much like a computer image is represented by pixels. As the dynamic analysis proceeds, the solution is tracked on the material points by updating all required properties, such as position, velocity, acceleration, stresses strains, etc. At each time step, the particle information is extrapolated to a background grid which serves as a calculational tool to solve the equations of motions. Once the equations are solved, the grid-based solution is used to update all particle properties. This combination of Lagrangian and Eulerian methods has proven useful for solving solid mechanics problems including those with large deformations or rotations and involving materials with history dependent properties such as plasticity or viscoelasticity effects [2, 25, 26]. MPM is amendable to parallel computation [4], implicit integration methods [5] and alternative interpolation schemes that improve accuracy [6].

Although MPM uses a background grid and is frequently compared to finite element methods, a new derivation of MPM [4] presents it as a Petrov-Galerkin method that has similarities with meshless methods, such as Element-Free Galerkin (EFG) methods [65] and Meshless-Local Petrov-Galerkin (MLPG) methods [7, 8, 9]. The "meshless" aspect of MPM, despite the use of a grid, derives from the fact that the body and the solution are described on the particles while the grid is used solely for calculations. The body can translate through the grid. Furthermore, the grid can be discarded at each time step and redrawn which makes MPM suitable for adaptive mesh methods. It is essential for any extension to MPM, such as presented here, to preserve the separation between the grid and the particles. MPM, EFG, and MLPG differ in their approach to derive shape functions and in their selection of test functions during numerical implementation [12, 38]. One potential application of MPM is its use for dynamic fracture modeling. It was recently shown that MPM could accurately calculate fracture parameters, such as energy release rate [10, 64].

MPM has found application in the solution of a wide variety of problems, including silo discharge [38], membrane stretching [45], landfill settlement [42], elastic vibrations [40], collisions [25, 12, 38, 39], and the response of granular materials [7, 12, 39], just to name few.

2.1.3 Generalized Interpolation Material Point Method (GIMP)

The MPM algorithm derivation has been recently formulated in variational, or weak form [2, 25]; providing standard setting for the discretization of the governing equations [65] but advantage of generalizing the MPM discretization technique has not been taken [13]. PIC methods were developed for providing alternative representation of solution variables on grid by using particles, so that these variables can advect through the grid avoiding difficult interface tracking. For appropriate correspondence between particle and grid, the nature of transferring information needs to be carefully attended [13]. The same governing equations presented by , Chen, and Scheryer [2]; Sulsky, Zhou, and Scheryer [25] can be derived , without reference to variational form, by conserving momentum on computational grid and by conserving mass and momentum in the interpolation between particles and grid [54]. But the variational form of governing equation provides a consistent framework for generalization of MPM discretization technique and identifying similarities to other meshless methods [6,59].

In Generalized interpolation material point method full generality of the variational formulation is exploited. The result of this generalization is smoother representation of particle data on the computational grid, which has significantly removed the numerical artifacts inherent in MPM formulation, which develops when material points fail to register in self-similar fashion on computational grid [13]. The use of smoother representations of discrete material point data has allowed an entire family of methods to be developed and the nature of general derivation suggests denoting the family of Material Point Method discretization schemes developed by Bradenhagen and Kober, as the Generalized Interpolation Material Point (GIMP) methods [13].

2.2 Parallel Processing

The required more accurate and reliable solutions in numerical analysis of various large scale structural, fluid mechanics problems and design problems need more computationally intensive process, usage of more efficient methods and here implementation of parallel processing technique is one of the best economic alternatives [23]. Parallel processing has been used successfully in numerical analysis using different methods, such as FEM and boundary element method (BEM) [23] and molecular dynamics (MD) [21]. The computational time on parallel processors can be reduced to a small fraction of the time consumed by a single processor at the same processor speed. Parallel processing generally involves issues, such as domain decomposition/partitioning, load balancing, parallel solver/algorithms, parallel mesh generation, and multi-grid [23]. Domain decomposition has been widely applied in parallel processing in FEM [19]. 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 [23]. 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 as in GIMP method.

2.3 Contact Mechanics

Contact problems are central to solid mechanics, because contact is the principal method of applying loads to a deformable body and resulting stress concentration is often the most critical point in the body. Contact is characterized by unilateral inequalities, describing the physical impossibility of tensile contact tractions and of material interpenetration [31]. Historically, the development of the subject stems from the famous Investigation of Heinrich Hertz (1882) giving the solution for the frictionless contact of two elastic bodies of ellipsoidal profile. Hertz' analysis still forms the basis of the design procedures used in many industrial situations involving elastic contact. Since 1882, the subject of contact mechanics has seen considerable development and two major threads can be distinguished – from mathematical standpoint, emphasis has been placed on the extension of uniqueness of solution, whereas engineers have focused on the solution of particular problems in an attempt to understand and influence phenomena that occur in practical contacting systems, both on the macro and the micro scale [31].

The essence of a contact problem lies in the fact that any point on the boundary of each body must either be in contact or not in contact. If it is not in contact, the gap g between it and the other body must be positive ($g>0$), whereas if it is in contact, $g=0$, by definition. A dual relation involves the contact pressure p between the bodies which must be positive ($p>0$) where there is contact and zero where there is no contact. The inequalities serve to determine which points will be in contact and which not [31]. Fichera (1964, 1972) proved that the complete problem, including the inequalities, has unique solution, when the material is linear elastic and many related proofs have since been advanced for other classes of contact problem (Duvaut and Lions, 1976).

Algorithms to handle the contact inequalities are now routinely included in most commercial finite element packages, usually based on smoothing of the discontinuities involved or the Lagrange multiplier technique. However, body fixed meshes can be difficult and time consuming to generate for 3D complex structures and distortion associated with large deformation can compromise accuracy [13]. These difficulties have spurred the development of methods such as MPM to avoid mesh distortion. Indentation involves a contact pair of a rigid indenter and a deformable work piece. 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 [24, 29]. While MPM can prevent the penetration at the interface automatically, it uses a single mesh for the two bodies and single-valued mapping functions between background grid nodes and material particles, so that a natural no-slip contact constraint will occur. To release the inherent no-slip contact constraint, a contact/sliding/separation scheme was proposed by introducing multi-mesh environment as background mesh by

Hu and Chen [20]. In multi-mesh MPM, in addition to a common mesh for all objects, there is an individual mesh for each of the object under consideration. All meshes are identical, i.e. nodal locations are same. The nodal masses and forces are mapped from the material points of each object to its own mesh each step. The nodal values are transferred to the corresponding nodes in the common mesh. When values at a node of the common background mesh involves contributions from two parts, the contact between two parts occurs so that this node is defined as 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 method, the interaction at overlapped node is activated too early before the actual contact of material points occurs. This is physically incorrect and cause large errors in GIMP method. A new contact algorithm was proposed by Komanduri, Lu, Roy, Wang, Hornung, Wissink, and Ma [15] for GIMP to overcome these problems and implemented that in 2D. In this investigation that algorithm has been extended to 3D.

CHAPTER 3

Generalized Interpolation Material Point Method Parallel Processing

3.1 Introduction

The Generalized Interpolation Material Point (GIMP) is a family of methods resulted from generalization of the Material Point Method (MPM) discrete solution procedures for computational solid mechanics using a variational form and Petrov-Galerkin discretization scheme [13]. In MPM calculations a numerical artifact noise is expected due to lack of smoothness of interpolation functions; it occurs when material points cross computational grid boundaries during their coarse of deformation. This noise results in non-physical local variations at the material points, where constitutive response is evaluated, and seriously degrades the explicit solution in case of large deformations. GIMP method developed by Bradenhagen and Kober [13] provide next degree of smoothness (C^1 continuous) of interpolation functions which is capable of significantly eliminating cell crossing noise. In the next section GIMP methods are derived from variational form using Petrov-Galerkin discretization scheme and specialization to MPM algorithm is shown.

3.2 Governing Equations

The governing equations in both conventional material point method (MPM) [20, 25] and generalized interpolation material point method (GIMPM), Bardenhagen and Kober (2004) [13], are briefly summarized in this section. The weak form of the momentum conservation equation in MPM is given by

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

where **w** is the test function, **a** is the acceleration, and $\mathbf{s}^{s}$, $\mathbf{c}^{s}$ and **b** are the specific stress, specific traction, and specific body force, respectively. Ω is the current configuration and ∂Ω 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 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 [25]

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

Where the lumped mass matrix is given by

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

and the internal and external forces are given by

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

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

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 [13, 25, 20]. The primary cause for the problem has been attributed to the discontinuity of the gradient of the shape functions across the cell boundaries [20,13]). To resolve this problem, Bardenhagen and Kober [13] proposed a generalized interpolation material point (GIMP) method. In GIMP method, the interpolation between node i and material point p is given by the volume averaged weighting function

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

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. $\chi_p(\mathbf{x})$ is one in the current region occupied by the material point p and zero elsewhere. 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 [13], can be written as

$$\sum_{p}\int_{\Omega\cap\Omega_p}\frac{\dot{\mathbf{p}}_p\chi_p}{V_p}\cdot\delta\mathbf{v}d\mathbf{x}+\sum_{p}\int_{\Omega\cap\Omega_p}\boldsymbol{\sigma}_p\chi_p:\delta\mathbf{v}d\mathbf{x}=\sum_{p}\int_{\Omega\cap\Omega_p}\frac{m_p\chi_p}{V_p}\mathbf{b}\cdot\delta\mathbf{v}d\mathbf{x}+\int_{\partial\Omega}\mathbf{c}\cdot\delta\mathbf{v}d\mathbf{x} \qquad (7)$$

where $\delta\mathbf{v}$ is an admissible velocity field, $\dot{\mathbf{p}}_p$ 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 [13]. Herein, the weighting function $\overline{S}_{ip}$ is C1 continuous under the spatially fixed background grid. Consequently the noises associated with material point crossing cell boundaries in the conventional MPM can be minimized. Bardenhagen and Kober have implemented GIMP method in one-dimensional simulations, further Komanduri, Lu, Roy, Wang, Hornung, Wissink , Ma [15] implemented GIMP in two-dimensional simulations.

In this Investigation, GIMP presented by Bardenhagen-Kober and extended for three-dimensional simulations has been implemented in SAMRAI environment. A refinement technique and a parallel processing scheme have been developed using SAMRAI to extend the serial GIMPM algorithm to code large scale parallel computing. The capability of parallel GIMP computing has been demonstrated by modeling three-dimensional indentation problem. A contact algorithm has been developed to address the contact problem between the rigid indenter and the deformable work piece and a parallel solver scheme has been developed for contacts spanning over multiple processors.

3.3 Numerical Implementation for 3D

We consider the case where initially there are eight material points in a cell for which the 1D weighting function is depicted in fig.2. In this figure, one node is at the origin and the horizontal axes give material point positions normalized by the cell size.

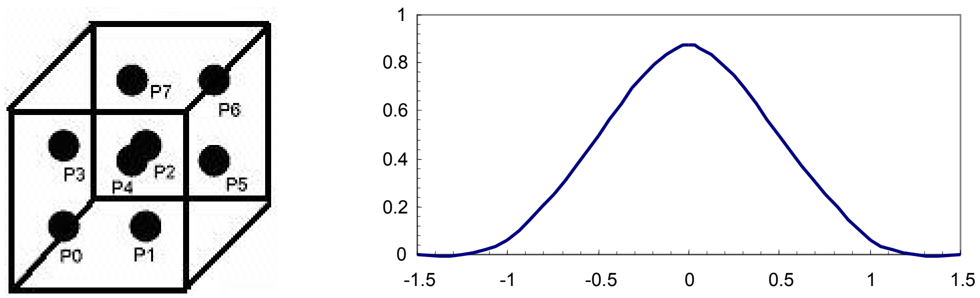


Fig. 2: Eight Material points in cell and the weighting function for 1D in GIMP method

Fig. 2 is based on the same material point characteristic function and node shape function as in Bardenhagen and Kober [13]. To get weighing function in 3D for a material point first the values of weighing function along each horizontal, transverse and vertical axis is determined using 1D weighing function depicted in Fig.2 and then 3 values are multiplied to get 3D weighing function value. 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 eight material points remains cubical 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 3D simulations. In case of large rotations, rotation has to be accounted in computations for better accuracy.

In the processing stage, first a background grid is created based on the input data and then material points are placed in cell based upon the shape functions derived from the natural co-ordinates. The complete algorithm follow the following steps.

1. Particle mass can be found as :

$\mathrm{M_p}$ = particle mass = $\dfrac{(\mathrm{Cell}_{\mathrm{volume}})\times(\mathrm{density})}{(\text{total number of particles in each cell})}$

Weighing functions $\overline{S}_{ip}$ are calculated for each particle as described above and then lumped mass and momentum cab be mapped from particles to nodes using equation (3) and equation (8) below respectively using weighing function calculated in place of shape function $N_i(\mathrm{x}_p^t)$.

$$\mathrm{Pk_i} = \mathrm{momentum_{node}} = \sum_{p=1}^{N_p} M_p V_p \overline{S}_{ip} \tag{8}$$

Where, V_p is the particle velocity

2. Get total grid point forces:

$$(\mathrm{F}_i^t)^{TOTAL} = \left((\mathrm{f}_i^t)^{\mathrm{int}} + (\mathrm{f}_i^t)^{ext} + (\mathrm{f}_i^t)^{damping}\right)$$

Where $(\mathrm{f}_i^t)^{\mathrm{int}}$ and $(\mathrm{f}_i^t)^{ext}$ can be found using equations (4) and (5) and damping force for artificial damping can be used to reduce numerical noise. Then Zero $(\mathrm{F}_i^t)^{TOTAL}$ boundary condition is imposed on nodes having fixed displacement boundary conditions.

3. Update Information

- Update grid momentum (for all the nodes):

$$pk_i^{t+\Delta t} = pk_i^t + \left((\mathrm{F}_i^t)^{TOTAL} \times timestep\right)$$

- Find new velocity at particles:

$$\Delta V_p = \sum_{i=1}^{N_n} \frac{pk_i^{t+\Delta t}\overline{S}_{ip}}{m_i^t}$$

- Find new particle acceleration:

$$\Delta\ \mathrm{A_p} = \sum_{i=1}^{N_n} \frac{(\mathrm{F}_i^t)^{TOTAL}\,\overline{S}_{ip}}{m_i^t}$$

- Update particle position:

$$\mathrm{Pos} = \Delta\ V_p\ \text{x (timestep)}$$

- Update particle velocity:

$$\text{Velocity particle} = \Delta\ \mathrm{A_p}\ \text{x (timestep)}$$

Final step is to update strain using the suitable constitutive law for the case analyzed for example elastic only or elastic perfectly plastic etc and then find stresses. For example for elastic only case 9 strain components in 3D can be found as following:

Straintime = (timestep)/2

$$\varepsilon_p^{xx} = \left[\sum_{i=1}^{N_n} \mathrm{v}_n^x \frac{\partial \overline{S}_{ip}}{\partial \xi}_p\right] \text{x Straintime}$$

$$\varepsilon_p^{yy} = \left[\sum_{i=1}^{N_n} \mathrm{v}_n^y \frac{\partial \overline{S}_{ip}}{\partial \eta}_p\right] \text{x Straintime}$$

$$\varepsilon_p^{zz} = \left[\sum_{i=1}^{N_n} \mathrm{v}_n^z \frac{\partial \overline{S}_{ip}}{\partial \zeta}_p\right] \text{x Straintime}$$

$$\varepsilon_p^{xy} = \left[\sum_{i=1}^{N_n} \mathrm{v}_n^x \frac{\partial \overline{S}_{ip}}{\partial \eta}_p\right] \text{x Straintime}$$

$$\varepsilon_p^{yx} = \left[\sum_{i=1}^{N_n} \mathrm{v}_n^y \frac{\partial \overline{S}_{ip}}{\partial \xi}_p\right] \text{x Straintime}$$

$$\varepsilon_p^{xz} = \left[\sum_{i=1}^{N_n} \mathrm{v}_n^x \frac{\partial \overline{S}_{ip}}{\partial \zeta}_p\right] \text{x Straintime}$$

$$\varepsilon_p^{zx} = \left[\sum_{i=1}^{N_n} \mathrm{v}_n^z \frac{\partial \overline{S}_{ip}}{\partial \xi}_p\right] \text{x Straintime}$$

$$\varepsilon_p^{yz} = \left[\sum_{i=1}^{N_n} \mathrm{v}_n^y \frac{\partial \overline{S}_{ip}}{\partial \zeta}_p \right] \text{x Straintime}$$

$$\varepsilon_p^{zy} = \left[\sum_{i=1}^{N_n} \mathrm{v}_n^z \frac{\partial \overline{S}_{ip}}{\partial \eta}_p \right] \text{x Straintime}$$

And then, specific stiffness matrix below can be used to find stress from strain.

$E/(1-\nu^2)*\rho$	nu* $E/(1-\nu^2)*\rho$	0
Nu* $E/(1-\nu^2)*\rho$	$E/(1-\nu^2)*\rho$	0
0	0	G/ρ

For the next step again a new grid is created or same grid can be used without any deformation with all variables at node made zero and then the above 3 steps are repeated again. This process will continue until the total time of simulation is reached.

CHAPTER 4

PROBLEM STATEMENT

The material point method (MPM) proposed by Sulsky et al. [2, 7, 25, 26] has received increased applications, though FEM has been the dominant numerical technique to simulate dynamic problems in solid mechanics. MPM has demonstrated capabilities in the simulation of impact/contact, penetration, and interfacial crack growth problems. As compared to FEM, MPM has the following advantages:

(1) MPM is able to handle large deformation in a more natural manner so that mesh lock-up present in FEM is avoided;

(2) MPM can easily couple with molecular dynamics (MD) simulations because of the use of material points (similar to atoms used in MD) instead of arbitrary sized elements in FEM;

(3) Parallel computation is more straightforward in MPM because of the use of a grid structure that is consistent with parallel computing grids; and

(4) Use of the background grid in MPM enables structured adaptive refinement for local interested region.

Hence for multi-scale large deformation solid mechanics problems, MPM appears as the most suitable method but 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 [13]. Additionally, for problems involving contact [20], MPM is able to provide a naturally non-slip contact algorithm to avoid the penetration between two bodies based on a common background mesh. While MPM can prevent the penetration at the interface automatically, it uses a single mesh for the two bodies and with the use of single-valued mapping functions between background grid nodes and material particles a natural no-slip contact constraint will occur. To release the inherent no-slip contact constraint a contact/sliding/separation scheme was proposed by introducing multi-mesh environment as background mesh [20]. In multi-mesh MPM, in addition to a common mesh for all objects, there is an individual mesh for each of the object under consideration. All meshes are identical, i.e. nodal locations are same. The nodal masses and forces are mapped from the material points of each object to its own mesh each step. The nodal values are transferred to the corresponding nodes in the common mesh. When values at a node of the common background mesh involves contributions from two parts, the contact between two parts occurs so that this node is defined as 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 MPM, the interaction at overlapped node is activated too early before the actual contact of material points occurs. This is physically incorrect and cause large errors in MPM. One more disadvantages of Material point method is their relatively high computational cost.

To solve the instability problems associated with the conventional MPM simulations, Bardenhagen and Kober recently proposed the generalized interpolation material point (GIMP) method and implemented for one-dimensional simulations, Ma, Lu, Wang, Roy, Hornung, Wissink and Komanduri implemented the GIMP presented by Bardenhagen and Kober to two-dimensional simulations. Parallel processing has been used successfully in numerical analysis using different methods, such as FEM and boundary element method (BEM) [23] and molecular dynamics (MD) [21]. 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. A new contact algorithm was proposed by Komanduri, Lu, Roy, Wang, Hornung, Wissink , Ma [15] for GIMP to overcome the problems associated with conventional MPM for handling contact problems and implemented that in 2D. Also after years of scientific development 2D codes are of common use by engineer while 3D codes well fitted for practical applications are scares but because of improvement of resolution algorithms and computer technology are now in continuous progress [32].

Considering all the points discussed above to solve 3 dimensional multi-scale large deformation problems in solid mechanics based on the continuum mechanics approach, GIMP method algorithm presented by Bardenhagen-Kober and extended to 3D has been implemented in SAMRAI environment and a refinement technique and a parallel processing scheme using SANRAI have been developed to extend the serial GIMP algorithm to be used for parallel computation. Also contact algorithm proposed by Komanduri, Lu, Roy, Wang, Hornung, Wissink, Ma [15] for GIMP method has been extended to 3D to solve contact problems involving 3D indentation. A parallel solver scheme has also been developed for contact interface spanning over multiple processors.

CHAPTER 5

CONTACT ALGORITHM FOR RIGID SURFACE INDENTER

5.1 Introduction

Contact problems are critical in solid mechanics, because contact is the principal method of applying loads to a deformable body and resulting stress concentration is often the most critical point in the body [31]. Indentation involves a contact pair of a rigid indenter and a deformable work piece. 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). The essence of contact problem lies in the fact that if bodies are in contact then gap g between the bodies is zero (g=0) otherwise it should be positive (g>0). Also the contact pressure p at contact should be positive (p>0) and zero where there is no contact [31]. While MPM can prevent the penetration at the interface automatically, it uses a single mesh for the two bodies and with the use of single-valued mapping functions between background grid nodes and material particles a natural no-slip contact constraint will occur. To release the inherent no-slip contact constraint a contact/sliding/separation scheme was proposed by introducing multi-mesh environment as background mesh [20]. In multi-mesh MPM, in addition to a common mesh for all objects, there is an individual mesh for each of the object under consideration. All meshes are identical, i.e. nodal locations are same. The nodal masses and forces are mapped from the material points of each object to its own mesh each step. The nodal values are transferred to the corresponding nodes in the common mesh. When values at a node of the common background mesh involves contributions from two parts, the contact between two parts occurs so that this node is defined as 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 method, the interaction at overlapped node is activated too early before the actual contact of material points occurs. This is physically incorrect and cause large errors in GIMP method. A new contact algorithm was proposed by Komanduri, Lu, Roy, Wang, Hornung, Wissink , Ma [15] for GIMP method to overcome these problems and implemented that in 2D and this investigation is extending that approach to 3D. Next section describe the contact algorithm in detail.

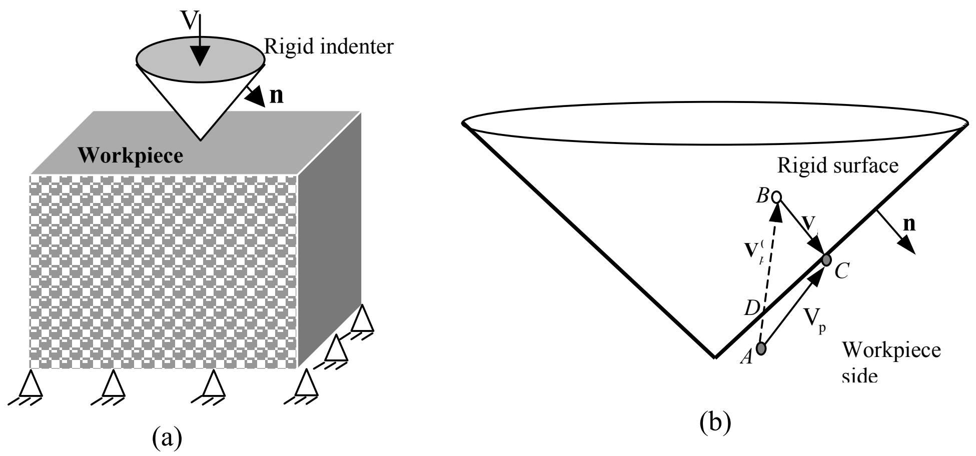


Fig. 3: Schematic of contact algorithm between the rigid indenter and the deformable work piece

5.2 Contact Algorithm

Fig. 3 illustrates the contact algorithm for the contact pair between a rigid indenter and a deformable work piece. A frictionless contact is assumed in this investigation. In this schematic diagram, even though points are used to represent material points but they must be understood to be areas occupied by material points in GIMP. 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 **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

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

where $\mathbf{F}_i^c$ 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

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

If the contact pressure $\mathbf{P}_q^c$ on the contact material point q is assumed to be constant at each material point, we have $\mathbf{F}_i^c = \sum_{q=1}^{Q} T_{iq} \mathbf{P}_q^c$, where $T_{iq} = \int_{\partial\Omega_q} S_i(\mathbf{x}) ds$ and Q is the total number of material points in contact with the indenter. 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

$$\mathbf{V}_p^c = \Delta t \sum_i^N \frac{\overline{S}_{ip}}{m_i} \sum_q^Q T_{iq} \mathbf{P}_q^c \,. \tag{10}$$

Eq. (10) can be 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.

5.3 Numerical Implementation

Numerical Implementation of the contact algorithm described above mainly consists of three steps. First step is to find the particles in deformable work piece which are in contact with rigid surface indenter. Second step is to find the unit outward normal vector **n** to surface of indenter for each contact particle position, which is then used to find the contact velocity $\mathbf{V}_p^c$ for respective particle in contact. Once the contact velocity for contacting particles is known, we can proceed to third step, which is to solve for contact pressure. This contact pressure is then used as external force applied each time step on the contact particles and then rest of the GIMP method computation are done as usual. The three steps that are specific to contact problem are described in detail in next three sections .

5.3.1 Detecting Contact

First step in solving contact algorithm is finding particles, which are in contact with the rigid surface indenter. Let us refer Fig. 4 and assume that material point is at A at the start of current step, then projected position B of material point at the end of step can be found with the help of velocity $\mathbf{V}_p^0$ it is having without consideration of contact. This is done by finding the displacement from A to B by multiplying $\mathbf{V}_p^0$ by the time step**.**

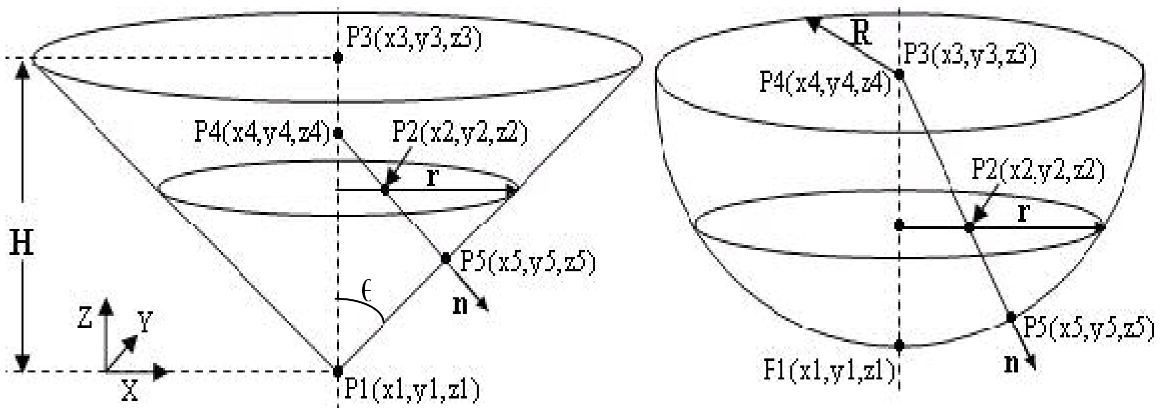


Fig. 4: A schematic of Conical and Spherical Indenters

Fig. 3 is redrawn in fig.4 with more details and also case of spherical indenter is considered, here point P2 is same as point B in fig. 3. Now if point P2 lies inside the indenter then it is in contact, else not. Numerically it can be processed using the flowchart as shown in Fig. 5.

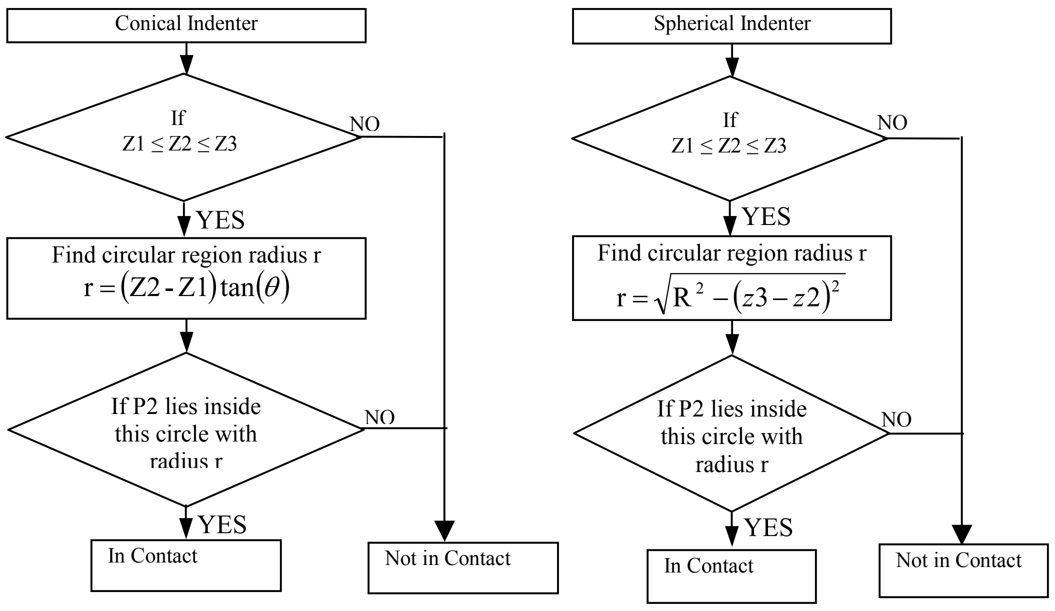

Conical Indenter
If $Z1 \le Z2 \le Z3$
NO
YES
Find circular region radius r
$r = (Z2 - Z1)\tan(\theta)$
If P2 lies inside this circle with radius r
NO
YES
In Contact
Not in Contact
Spherical Indenter
If $Z1 \le Z2 \le Z3$
NO
YES
Find circular region radius r
$r = \sqrt{R^2 - (z3 - z2)^2}$
If P2 lies inside this circle with radius r
NO
YES
In Contact
Not in Contact

Fig. 5: Flow charts for detecting contact with indenter

5.3.2 Finding outward normal vector n and contact velocity $\mathbf{V}_p^c$ for a material point

Now if the position of Indenter tip is at P1 and the particle position is at P2 as shown in the fig. 4 then to find the outward normal unit vector **n** to indenter surface at P2 and also to find contact velocity $\mathbf{V}_p^c$ for material point at P2 we can proceed in following steps.

1. Find a point P4(x4,y4,z4) which is the intersection of axis of indenter and the unit normal vector **n** to indenter surface passing through P2. For spherical indenter P4 is simply P3 center of sphere with x4=x1, y4=y1, and z4= z1+R. For conical indenter P4 can be found as x4=x1, y4=y1, and z4= z2 + ($(x2-x1)^2 + (y2-y1)^2)^{1/2}$ / tan (θ).
2. Second step is find a,b,c and d,

   where a = x4-x2 ; b= y4-y2; c= z4-z2; d= ($a^2 + b^2 + c^2)^{1/2}$.
3. Outward normal unit vector **n** is (a/d , b/d , c/d).

4. Fourth step is find displacement D; particle has to go through to be moved to the surface of Indenter i.e. from P2 to P5 as shown in fig. 4

   D= (| dist1 – dist2 |) **n**

   where dist1 = | (P4) – (P2)| ; for cone dist2 = | z4 – z1 | * sin (**θ**) and for sphere dist2=R.

5. Now contact velocity $\mathbf{V}_p^c$ = -D / (timestep)

1.3.3 Solving for Contact Pressure

Equation-10 above for all the particles in contact can be written in matrix form as:

$$V_k^c = C_{kl}\, P_l^c \tag{11}$$

Where *k* and *l* varies from 1 to total number of contact material points and C*kl* is Coefficient matrix

$$C_{kl} = \Delta t \sum_i^N \frac{\overline{S}_{ik}}{m_i} T_{il} \tag{12}$$

We can find V_k^c as described in previous section and can compute coefficient matrix using equation-12 easily. Now eqution-(11) above can be solved under physical contact condition $\mathbf{P}_q^c \cdot \mathbf{n} < 0$ to find contact pressure P_l^c.

CHAPTER 6

Parallel Computing Scheme Using GIMPM with SAMRAI

6.1 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 method for parallel computation of large-scale simulations. SAMRAI is chosen because of its similarity in grid structure with GIMP method. In GIMP method, 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 method 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 temporal and spatial mesh resolution. Each level in the hierarchy corresponds to single uniform degree mesh spacing [17], as shown in Fig. 6 and Fig. 7. Each grid level is divided into non-overlapping, logically cubical patches, each of which is a cluster of computational cubical cells with overlapping ghost region attached to each patch as shown in fig. 7, 9 and 11. In fig.9 a ghost region outside the thick line shown for Level-2 consisting four ghost cell layers being attached to actual patch region inside the line. Fig. 11 shows the ghost region separated by dashed line from actual patch for Level consisting of 2-Patches. Cell indices that are based on location of cells in total computational domain 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 computing environment for numerous physics applications [28].

Ideally, from application development point of view SAMRAI should be viewed as a collection of classes partitioned into roughly three categories: 1) mesh hierarchy and data structure classes, 2) algorithmic classes, and 3) classes that implement numerical routines on single patches. Classes from each of the categories are then combined in complex ways to build an application. The way in which classes interact is clearly defined through the judicious use of abstract interfaces. Algorithmic and data structure components are used as it is and also after specialization through C++ class derivation. Numerical routines are added for specific application requirements by providing numerical kernels that operate on patches and invoking them through C++ class derivation. Each patch data class is a subclass of a common abstract type that declares the basic operations for creation, data motion, and interprocessor communication Garaizar, Hornung, Kohn(1998). SAMRAI is like a "toolbox" of more then 300 classes, Fig.6. Detailed discussion about actual implementation of 3D-GIMP in SAMRAI is deffered till last section of this chapter when other information necessary for better understanding of that has been provided in the next few sections.

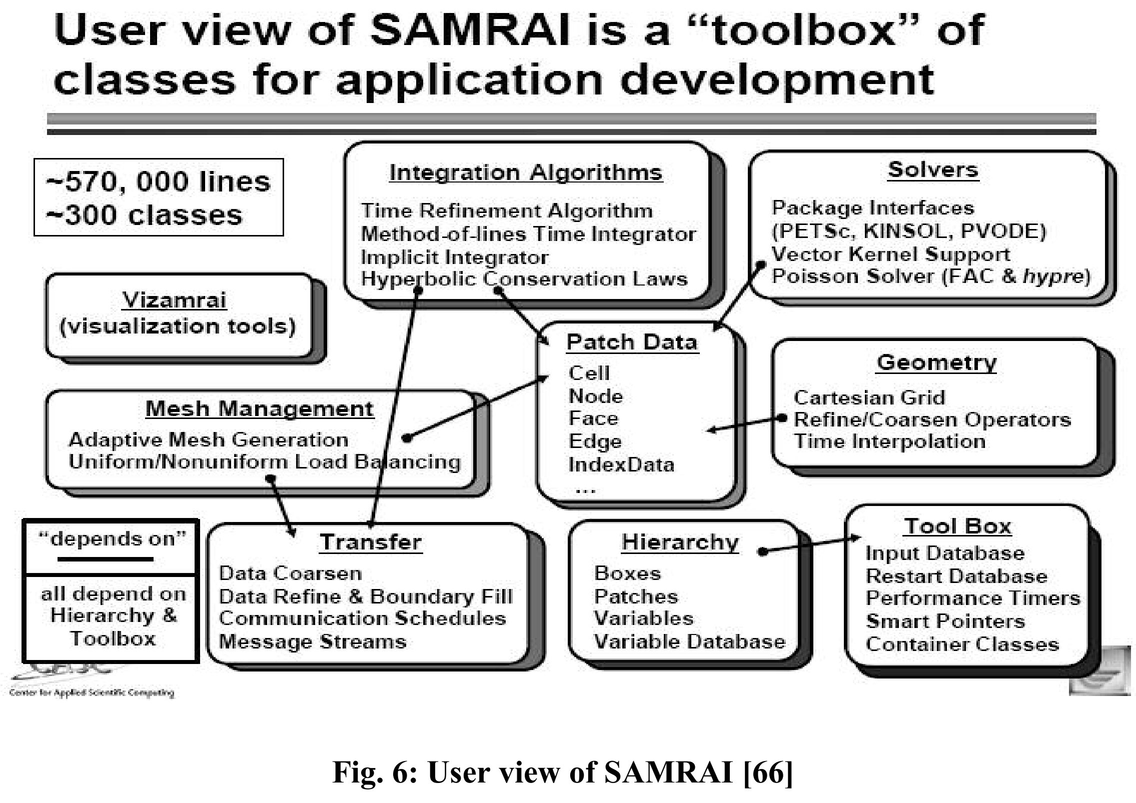


Fig. 6: User view of SAMRAI [66]

6.2 Spatial and Temporal Refinements

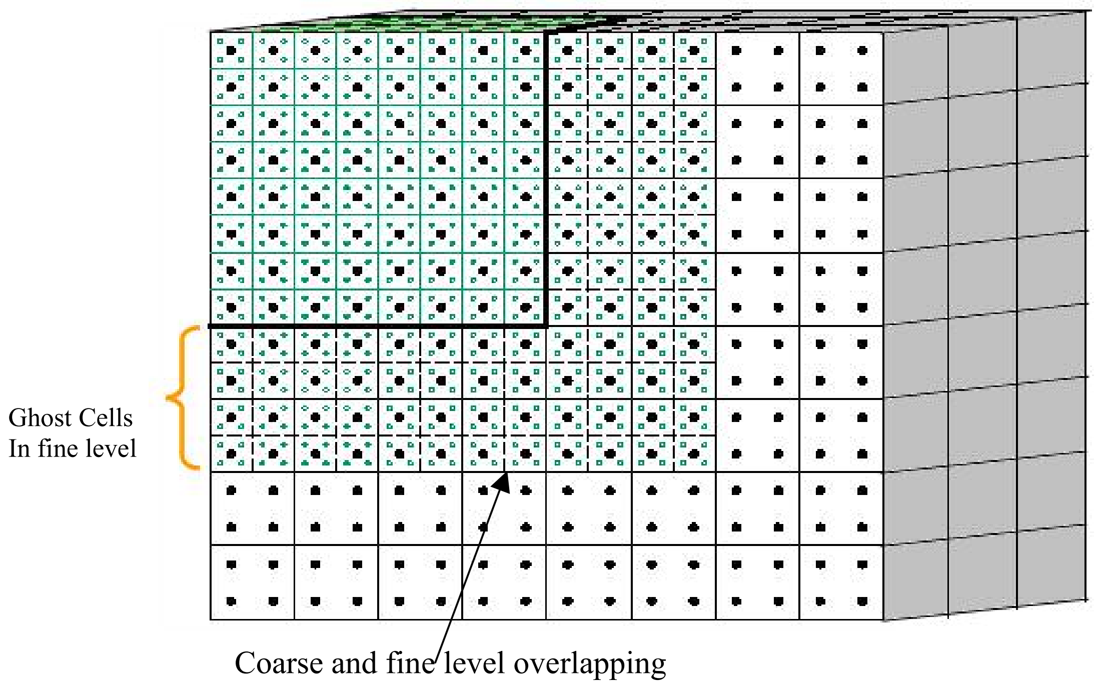


Fig. 7: Schematic showing two neighboring coarse and fine grid levels with material points inside.

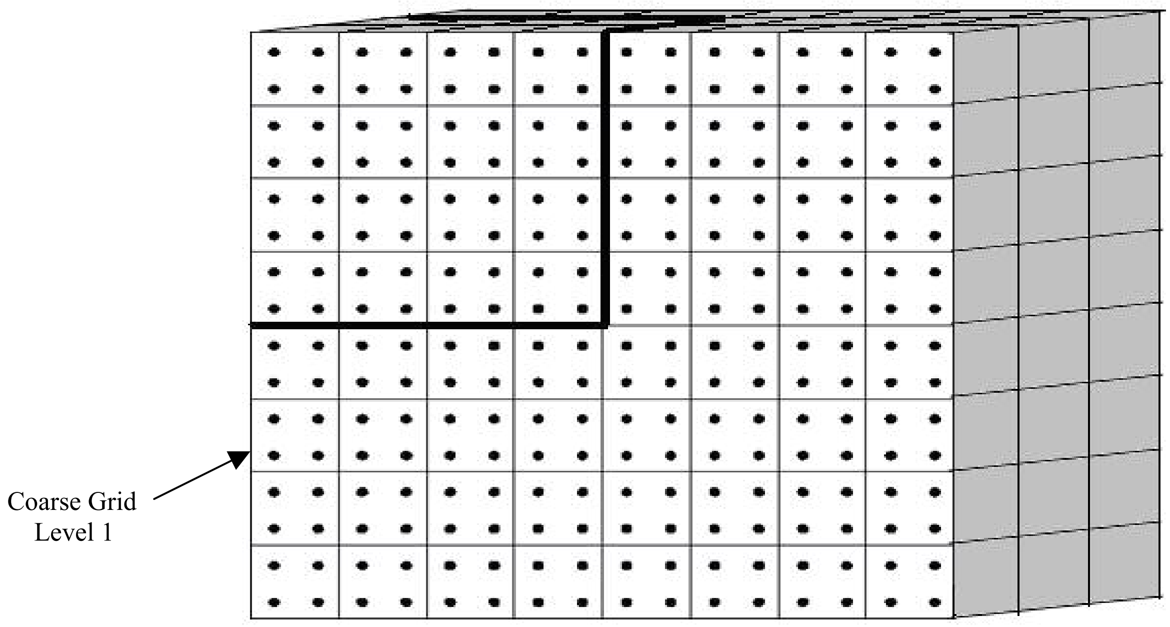


Fig. 8: Schematic of coarse level only as shown in fig. 7.

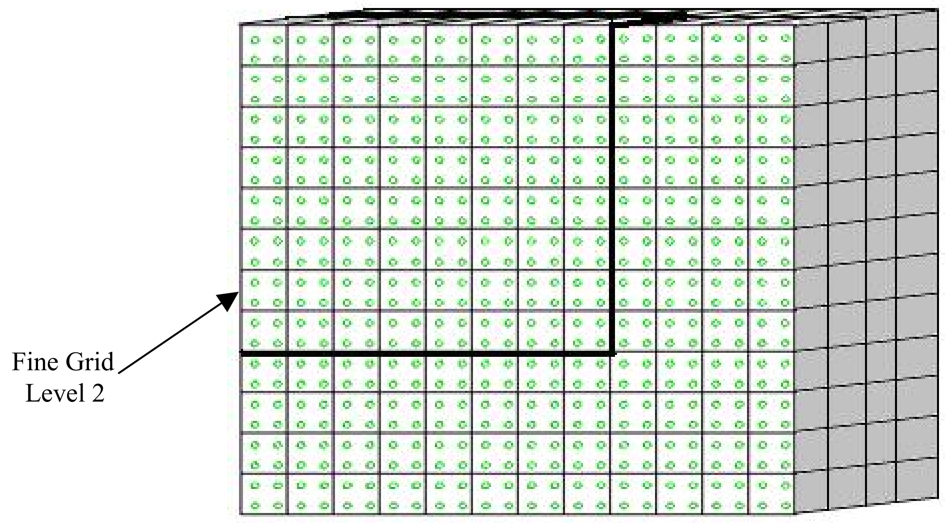
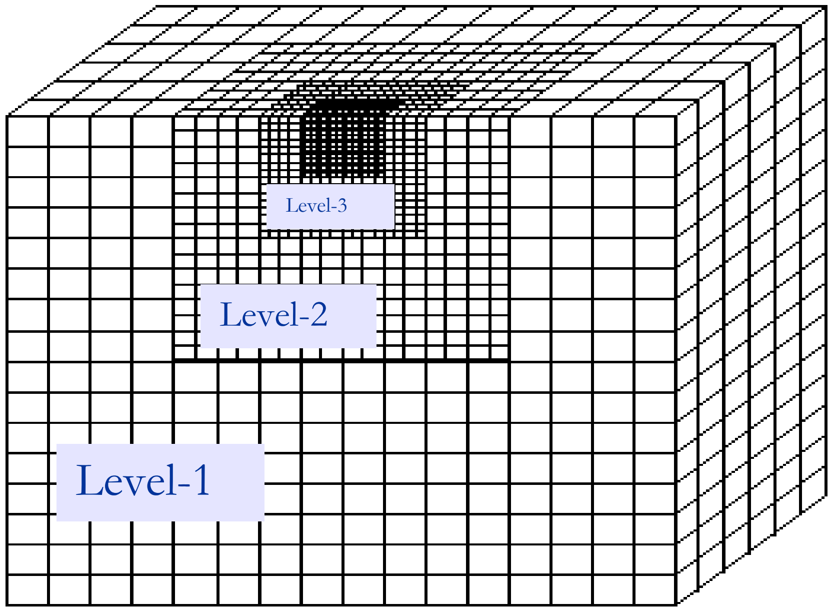


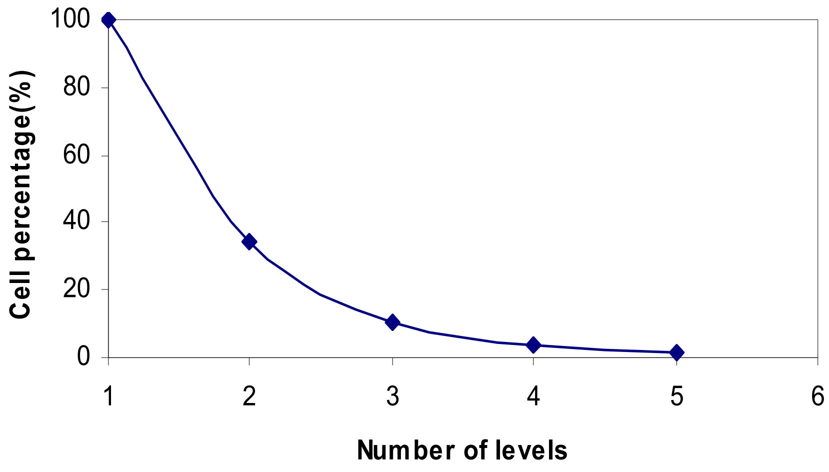
Fig. 9: Schematic of fine level only as shown in fig. 7

In the application of SAMRAI to large-scale GIMP method 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 method, 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. 7. 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 GIMPM can be split into several small material points. Tan and Nairn (2002) [8] 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 eight material points in the next fine level in 3D case. In this investigation, a similar approach is used as Refinement scheme for communication between two neighboring levels in overlapped region (called ghost region). Two data exchange processes, namely Refinement and Coarsening are used in communication. Fig. 7, Fig. 8 and Fig. 9 shows two neighboring coarse and fine grid levels in 3D GIMP method computations with refinement ratio of two. The thick line represents the physical boundary of the fine level with four layers of ghost cells. Initially eight or one material point can be assigned to each cell in coarse as well as fine levels. In GIMP computation each level is computed independently with physical variables communicated through the ghost regions between neighboring levels. Refinement process passes information from the coarse level to the immediate fine level, while coarsening process will pass information from the fine level to next coarse level. During refinement process physical variables at coarse material points are copied to neighboring eight material points in the fine level. Here it is assumed that there will be not much change in values of physical variables in the immediate neighborhood of coarse material points to save computational time and to avoid complicated approach by inducing only small error. In coarsening process, values of physical variables for each coarse material point inside the thick line are replaced by average values of physical variables of neighboring eight fine material points in fine level.

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. 10 shows 3D-section view of 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.



(a)



(b)

Fig. 10: Schematic of a model with Nested multi-level refinement and reduction in the number of cells with number of levels

If each fine level occupies one eighth region i.e. is half-length in each direction of the region occupied by neighboring coarser level, as shown in Fig. 10(a), the cell percentage as a function of the number of grid levels can be calculated, as shown in Fig. 10(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/ coarsening communications will cost additional computational time.

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. 7 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.

6.3 Domain Decomposition

GIMP method uses structured mesh, consistent with SAMRAI, so that domain decomposition is straightforward and no remeshing, in general, is necessary Fig. 11

shows a sectioned view of three-dimensional computational domain decomposed into two patches separated by a horizontal dash line. The ellipsoidal solid object with different boundary conditions applied at different regions is inside this domain/grid.

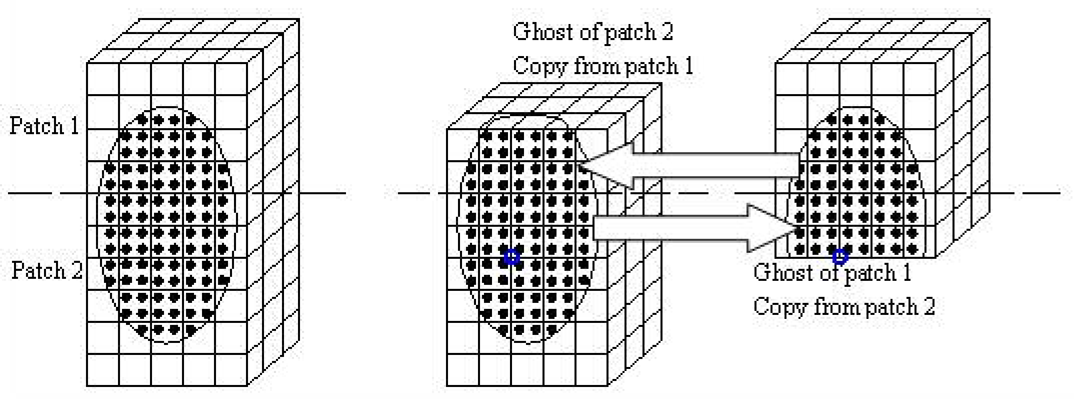


Fig. 11: A computational domain of two patches in one grid level

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 then 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 material points are created only inside material continuum as shown by ellipsoidal region in Fig. 11 and the number of material points remains same throughout the deformation to conserve mass. A single processor can process each patch 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 11. 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. 11. 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 half the number of material points as compared to grid node in patch-2 at same location and after advancing the step it extrapolates to same respective material points in each patch. Hence after each step material points in the outermost layer and 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 method 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 method step. Therefore, different patches can be assigned to different processors for parallel processing. After one GIMP method step, the data in the ghost cells will be updated. Copying material points to ghost cells involves data exchange between processors, which costs small 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 layers 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 method, it is possible to implement GIMP method 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. 12 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 method algorithm can be used to advance each patch without modification.

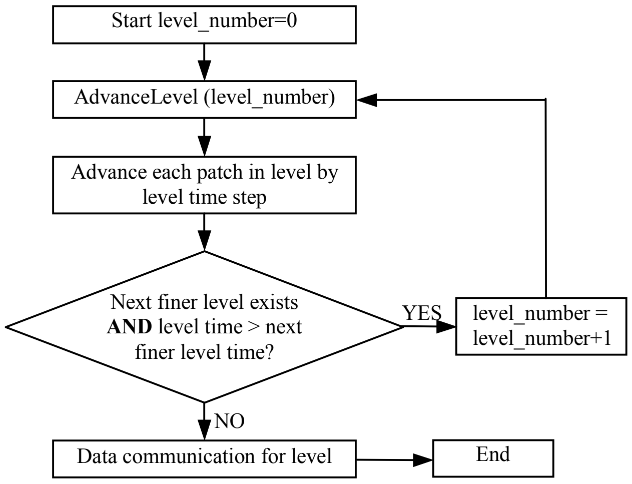

Start level_number=0
AdvanceLevel (level_number)
Advance each patch in level by level time step
Next finer level exists **AND** level time > next finer level time?
YES
level_number = level_number+1
NO
Data communication for level
End

Fig.12: Flowchart showing advancement of grid levels recursively starting from the coarsest to the finest level in GIMP method [15]

6.4 Solver for the Contact Algorithm

In case of indentation problem domain distributed over several processors, contact interface surface can be spanned over more then one processor and will require a parallel solver technique to solve simultaneous contact equations. Equation-11 in contact algorithm cannot be solved without knowing all the material points in contact in all the processors.

In large scale 3-Dimensional contact problem spanning over multiple processors assembling the coefficient matrix and contact velocity matrix to solve simultaneously for contact pressure is not straightforward as it requires information required for this to be gathered from all processors involved in computation. Also assembling the information required to know which contact particles are in neighborhood. Moreover, for parallel processing to be efficient requires minimum data transfer across the processors or lots time will be spend in communication. A very efficient and simplified scheme is developed to address this issue. The scheme involves four steps of solving equation-(11).

1. First step is finding contact material points in each processor and giving them their identification number based on the cell indices of the cell they are in. In this way the contact material points at same location in two patches in overlapping regions will have same identification number. The formula used to determine the identification number is

   Material Point No. =MaxMp*(K*MaxCx*MaxCy + J*MaxCx + I) + p ,

   where MaxMp is any high number assumed so that it will never be less then total number of material points present in cell at any time during course of simulation, MaxCx is the maximum number of cells along horizontal direction and MaxCy is the maximum number of cells along transverse direction in entire computational domain. K,J and I are the cell indices of cell the contact material point is in, along vertical, transverse and horizontal directions. There can be more then one contact material point inside the cell and p will keep increasing by 1 starting from value1 with each new contact material point found in same cell. All the variables used here are integer.

2. Second step is creating coefficient matrix and contact velocity matrix in each individual processor based on the contact material points present in that processor and sending all this information with identification numbers associated with them to a master processor.
3. In the master processor creating a global list of contact material point identification numbers associated with each contact material point from all the processor in ascending order and then assembling the global coefficient matrix and global contact velocity matrix based on the identification in the global list. For example let us assume that global list contains numbers 100, 108 ,116, 200, 208, 248, … and so on and local list of contact numbers from processor-2 contains numbers 200, 248, 280, … and so on. Now if we are assembling global coefficient matrix from the local coefficient matrices of other processors and we require to put value at (6, 4) in global coefficient matrix then we should get the value from coefficient matrix of processor-2 at (2, 1). As number associated with row 6 and 2 in global and local coefficient matrix,

respectively is 248 and with column 4 and 1 in global and local coefficient matrix respectively is 200.

4. Fourth step is solving equation-11 for contact pressure and then sending results to individual processor based on the contact material point identification numbers associated with each processor. After this each processor can do rest of the computation independently as described in Chapter 3. In order to increase speed of computations fastest processor is used as master processor.

Creating coefficients matrix and contact matrix first in individual processor for the contact material points for that processor correctly is only possible due to sufficient layers of ghost cells attached to the patches. Ghost cells facilitate the coupling of coefficient matrix and the contact velocity matrix and assembling them into global matrix by providing the information from neighboring contact particles from overlapping region with neighboring patch. Further, the coefficient matrix obtained is sparse matrix and is diagonally dominant.

The coefficient matrix in equation-11 is sparse so The Portable, Extensible Toolkit for Scientific Computation (PETSc) is being used for solving simultaneous linear equations. The default matrix representation in PETSc is the general sparse AIJ format. To obtain a good performance while assembling AIJ sparse matrix, its is crucial to pre-allocate the memory needed for sparse matrix to avoid very expensive dynamic process of allocating new memory and copying data in PETSc. One can specify expected number of non-zeros for each row in PETSc. In the present investigation number of non-zero for each rows has been calculated while creating local coefficient matrix and then is used to create pre-allocate the memory and setting values in Global sparse matrix for better performance.

6.5 Implementation of 3D-GIMP in SAMRAI

The implementation of 3D-GIMP in SAMRAI starts with creating a class MPMSamrai3d which is inherited from classes RefinePatchStrategy and CoarsenPatchStrategy and then actual implementation of various methods defined as virtual in these parent classes should be done. The strategy pattern is the primary mechanism used to encapsulate complex algorithms and provide well-defined interfaces between software components in SAMRAI, Hornung, Kohn(1998). Most of methods, which are defined virtual in parent class are having empty implementation in class MPMSamrai3d, as are not required for our case except for the following two:

virtual void postprocessRefine(Patch& fine, const Patch& coarse,const Box& fine_box,const IntVector& ratio)

virtual void postprocessCoarse(Patch& coarse, const Patch& fine,const Box& coarse_box,const IntVector& ratio)

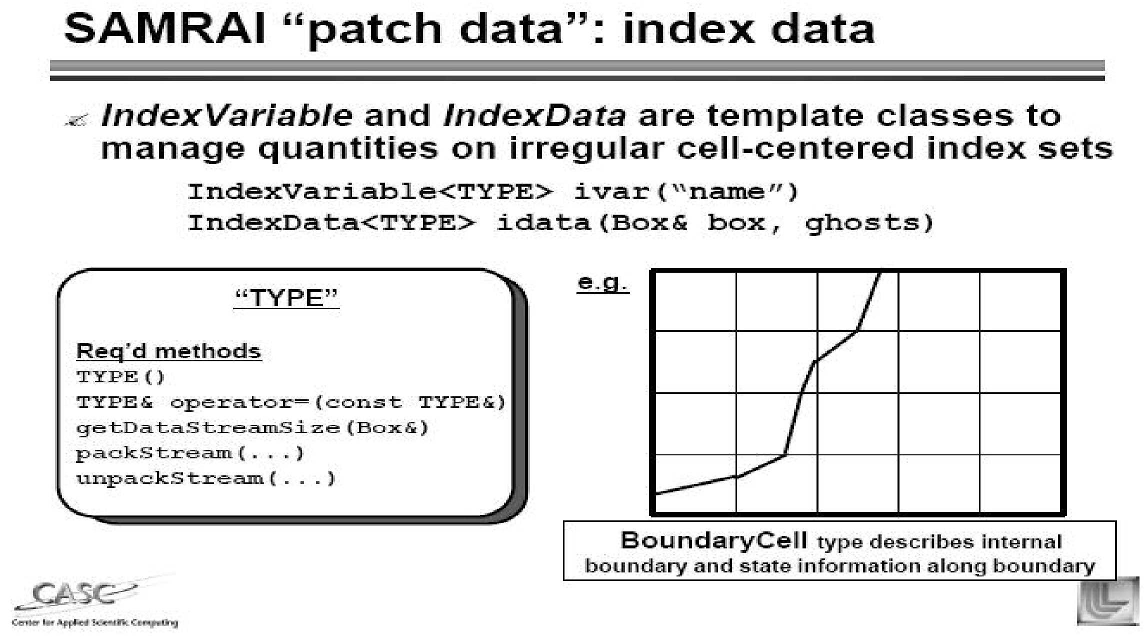
These are required for user defined refinement and coarsening operations during the data communication between the levels by SAMRAI. In class MPMSamrai3d the main member functions implemented are described in Fig. 13.

void setupCommunication()	Setup communication schedules for the functions on different levels
void initiateHierarchy( )	Successively finer levels are created and initialized with creation assigning of patchdata until the maximum allowable number of levels is achieved

void initLevelTimeStep(double)	Time step for Successive finer levels are calculated and assigned to it until the maximum allowable number of levels is achieved
double advanceMPMHierarchy();	Advance data one time step(Time step of coarsest Level) on hierarchy
void recursivelyAdvanceLevel(int)	advancing all grid levels recursively starting from the coarsest level and patches on that level by calling advancePatchMPM( ) for one coarsest time step see Fig. 10
void advancePatchMPM(Pointer<Patch>, int, MPM_Input, double, double,bool)	Advance each patch in level by level time step
virtual void postprocessRefine(Patch& fine, const Patch& coarse,const Box& fine_box,const IntVector& ratio)	Perform user-defined refining operations.
virtual void postprocessCoarse(Patch& coarse, const Patch& fine,const Box& coarse_box,const IntVector& ratio)	Perform user-defined coarseninging operations.
void getFromInput(Pointer<Database> db)	Read input data from specified database and initialize class members and other member variables.
*void outputTecPlot(double time,char *,int,int)*	Print output variables in a format conducive with TecPlot(A visualization software)

Fig. 13 Main member methods of class MPMSamrai3d and their functions

SAMRAI allows users to incorporate new data types, such as particles on the patches without modifying framework with the use of Abstract Factory creational pattern to generate and manage concrete data objects. Each patch data class is a subclass of a common abstract type that declares the basic operations for creation, data motion, and interprocessor communication [17]. The "Patchdata" package provides support for various concrete patch data types that reside on an structured AMR patch hierarchy. The data types found here include a variety of array-based quantities (cell-centered, node-centered, face-centered, etc.) as well as an "index" type for managing data associated with an arbitrary collection of cell indices (e.g., irregular structures like embedded boundaries or **lists of particles**), SAMRAI documentation. Hence, the data type most suitable for GIMP is index as shown in Fig. 14 .



SAMRAI "patch data": index data

- *IndexVariable* and *IndexData* are template classes to manage quantities on irregular cell-centered index sets

```
IndexVariable<TYPE> ivar("name")
IndexData<TYPE> idata(Box& box, ghosts)
```

"TYPE"

Req'd methods

```
TYPE()
TYPE& operator=(const TYPE&)
getDataStreamSize(Box&)
packStream(...)
unpackStream(...)
```

e.g.

BoundaryCell type describes internal boundary and state information along boundary

Fig. 14 SAMRAI "patch data": index data [66]

In Fig. 14 "TYPE" is the user defined data type, for 3D-GIMP, which is "Matpoint3". Matpoint3 is a class written in accordance with requirements specified for "TYPE" which are manily related to data packing and unpacking for data transfer during parallel communication. Matpoint3 represent a cell data and contains a list of variables of data type MPM to represent material points. MPM is defined as struct consisting of physical variables associated with GIMP such as stress, strain, particle position, velocity etc. During the simulation of a solid mechanics problem there can be many material point residing inside a cell where as there might be none inside other cells. The dynamic link list of MPM has been very efficient in handling this problem otherwise if fixed array is used we will either end up wasting some extra memory or may fall short of memory for accommodating all the material points.

The method, which perform main task of performing GIMP simulation in SAMRAI environment is "int main(int argc, char *argv[])" written in file main.C and follows the steps as described below:

1. An instance of tbox_InputDatabase is created to store (key,value) pairs in a database as *Pointer<Database> input_db = new box_InputDatabase("input_db")* and then *InputManager::getManager()->parseInputFile(input_filename,input_db)* is used to parse data from the specified input file into the existing database.
2. A database namely "geometry_db" is created by providing the information about the entire computational grid region, patch grid corresponding to each level from input_db. Also an instance namely grid geometry of CartesianGridGeometry is created by providing the information regarding the cartison coordinates and the indexes of the lower and upper end of diagonal of cubical computational grid of coarsest level.
3. An object of PatchHierarchy namely "hierarchy" is created using grid_geometry and then object of MPMSamrai3d is created namely "MPMSamrai3d_model*"* using these and also providing the database with information such as size of actual work piece, material properties, boundary conditions etc. as:

   MPMSamrai3d MPMSamrai3d_model = new MPMSamrai3d("MPMSamrai3d", hierarchy, grid_geometry, input_db->getDatabase("MPM"));*
4. Mapping is created by assigning each patch on the level to a particular processor in the processors used in simulation. Mapping is array of variable of type Processor

Mapping and contains the information which patch is mapped to which processor for a level.

5. Levels are created in the hierarchy by providing the information of Level domain which is array of boxes representing patches, information regarding ratio to coarsest grid for that level and the mapping information as :

   for (i=0 ; i<numofLevels ; i++)

   hierarchy->makeNewPatchLevel (i, ratioToCoarsest[i], levelDomain[I], mapping[i]);

6. Then communication schedules is setup for the functions on different levels as:

   MPMSamrai3d_model->setupCommunication();

7. Now the hierarchy is initiated by creating material points inside the region actually occupied by problem domain inside the computational domain as :

   MPMSamrai3d_model->initiateHierarchy( );

   and then each level time step is initialized using:

   MPMSamrai3d_model->initLevelTimeStep(Coarse_Time_Step);

   In method initLevelTimeStep(Coarse_Time_Step) timestep for coarsest level is calculated depending upon the wave speed and cell length for that level. After that, time step for successive finer levels can be calculated by using the ratio to the coarsest level. Also user can specify time step for Coarsest level externally by using the variable “Coarse_Time_Step” then time step for successive level can be calculated as usual.

8. Finally loop can be used starting from time zero till the end time specified in the input for the simulation calling advanceMPMHierarchy( ) each time with increasing time by time step of coarsest level as :

```
while ( (mtime <= end_time)) {
        mtime=MPMSamrai3d_model->advanceMPMHierarchy();
       MPMSamrai3d_model->outputTecPlot(mtime, file_name,
                          TecPlotStep,TrackPtStep);
}
```

Here mtime is computed each time by adding coarsest time step in it. The method advanceMPMHierarchy( ) further calls recursivelyAdvanceLevel(int) method for advancing all grid levels and patches on that level recursively starting from the coarsest level for one coarsest time step see Fig. 10. In Fig. 10 the method used for advancing patches is:

void advancePatchMPM (Pointer<Patch> patch, int indexData_id, MPM_Input inp, double levelTime, double levelTimeStep, bool isFinestLevel)

This method performs the 3D-GIMP computation for that patch by accessing the patch data for that patch using indexData_id and use the levelTimestep and MPM_Input information's. This is described in Chapter 3 in detail. Also for communication between the levels as shown in Fig. 10 user defined methods:

MPMSamrai3d:: postprocessRefine(Patch& fine, const Patch& coarse,const Box& fine_box,const IntVector& ratio) and

MPMSamrai3d:: postprocessCoarsen(Patch& fine, const Patch& coarse,const Box& fine_box,const IntVector& ratio) ;

are used by SAMRAI.

The method outputTecPlot( ) is called to print desired physical variables such as material points position, displacements, stresses, strains etc in a format such that file can be directly input into Tecplot for data visualization.

CHAPTER 7

Results and Discussions

A Beowulf Linux cluster of 10 identical PCs was 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 1GB. A gigabit switch is used to connect the network.

First example has been included here to demonstrate the significant reduction in computational time with parallel processing and then several indentation examples have been included here to demonstrate the capability of 3D GIMP parallel code developed using SAMRAI for solving relatively large deformation multi-scale contact problems in solid mechanics. Fig. 15 is the schematic of indentation model used in the examples involving rigid surface indenter. Fig. 15(a) shows the schematic of complete model with base fixed zero displacement in Z-direction displacement boundary constraint applied at base as shown and indenting at the middle of top surface with indenter with applied velocity V that is varying with time as shown in Fig 15(b). Also it can be seen from Fig.15 (a) that quadrant 1,2,3 and 4 are symmetrically aliened and one of these can be used for simulation considering symmetric boundary condition to save computational time and using computer memory efficiently. It should be noted that this investigation is part of ongoing research in multi-scale simulation spanning from atomistic scale to macro/continuum scale via mesoplastic (micro) scale and the objective here is to bridge the gap between these scales using continuum mechanics approach. This investigation is to be used to be coupled with Molecular dynamics and Meso-plasticity. Which are sensitive to crystal orientation and there for it is essential for this investigation to be able to run complete 3D models. Therefore, examples including both considering symmetric boundary and conditions and complete model has been included. The indenter as shown in Fig. 15(a) is not necessarily conical and spherical indenter has been used. Material for deformable work piece is aluminum for all the cases involving rigid surface indenter discussed here with Young's Modulus 70GPa, Density 2.7g/cm^3 and Poisson's ratio 0.33 and Yield Strength of 270MPa. FEM (ABAQUS) is being used to compare the 3D GIMP results and for validation. Tecplot is used to plot the stresses obtained from results of GIMP 3D and FEM. For few figures triangulation feature in tecplot is used to fill the region between the points with interpolated values.

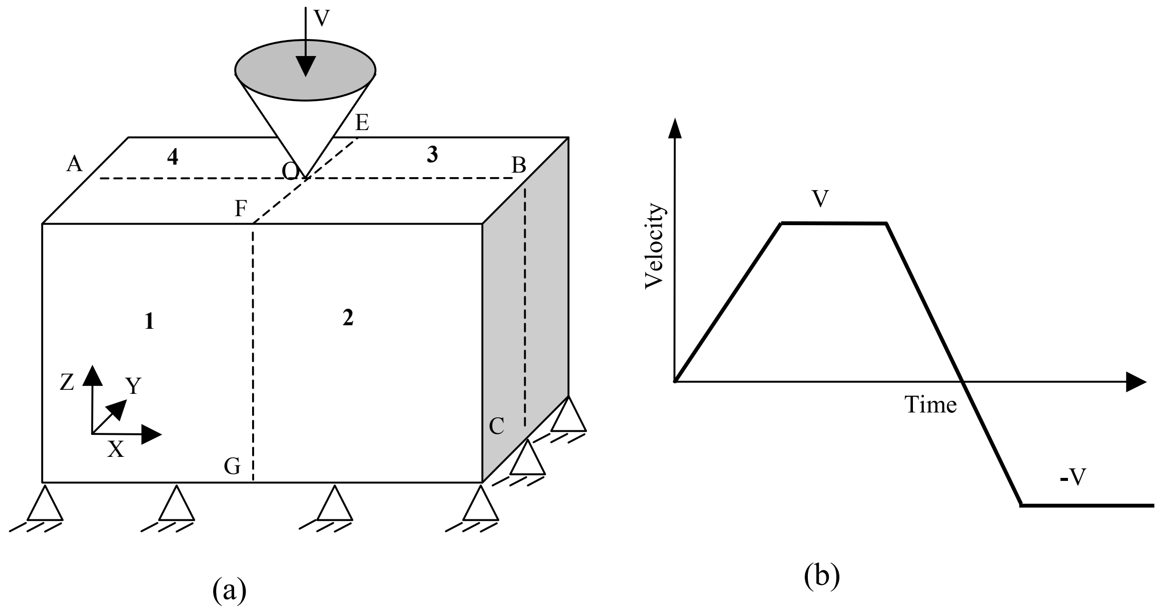


Fig.15: Loading conditions for a simple indentation problem

7.1 Reduction in Computational Time with parallel processing

The main purpose and advantage of parallel processing is to reduce computational time in simulation and it is observed that parallel processing scheme used in this investigation reduces the computational time significantly but for that, some modification in the SAMRAI has to be made.

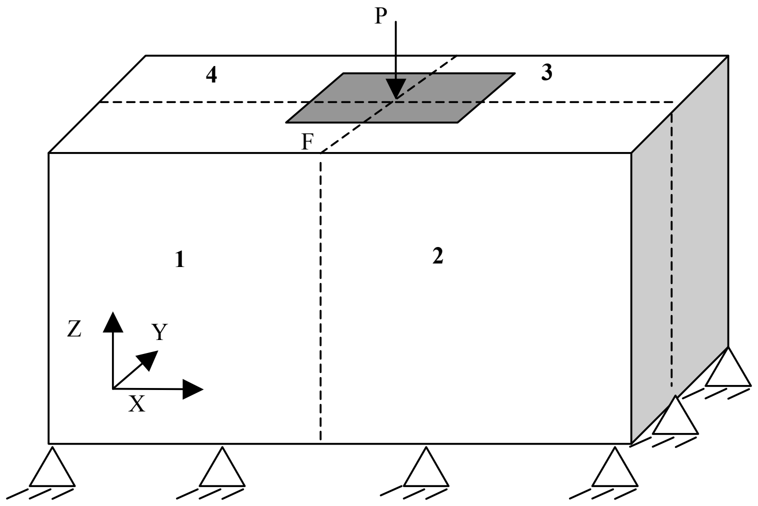


Fig.16: Loading conditions for a simple indentation problem

This is demonstrated in the first example where an indentation problem is simulated, where pressure is applied on an area at the center of top surface of cubical work piece which is fixed at bottom surface to move along vertical direction as shown in Fig. 15. The domain of the work piece can be discretized into patches considering that number of cells in the each patch grid should be nearly same. For example, the work piece domain can be discretized into 4 patches as shown in Fig. 16 above denoted by numbers 1, 2, 3 and 4. Each patch is handled by a single processor in this example. All six simulations are conducted with varying number of patches as 1, 2, 3, 4, 6 and 9 and time taken per step is calculated by taking average of 100 steps during each simulations as shown in Fig. 17.

The work piece dimensions are 5.632x10^{-3}nm × 5.632 x10^{-3} ×1.6x10^{-3}nm with cell size 64×64×64nm^3 for quarter model. Initially it was observed that using parallel processing most of the time computational time was increasing during simulation because of increase in communication time as shown by curve “SAMRAI without Modification” in Fig. 17. During parallel computation while performing data transfer between patches related to overlapping region, when SAMRAI fills the ghost region with correct data from some other patch, it first locates that cell whose data is to be changed based on its indices in the link list of ghost cells local to that processor. This search operation was taking lot of time as each time for new cell it was starting search from the first element in the list. Also time consumed in search was dependent on how big is the list and where in the list that particular cell is located, that is why sometimes with the increase in number of patches the processing time was increasing and sometimes reducing. To resolve this problem SAMRAI was modified so that in each time of new search instead of starting from beginning it starts search where it left, as most of the time for our case the ghost cell indices are in sequence. The search operation was made smart enough to search forward or backward in the list depending upon the increasing or decreasing order of index of cell to be searched, by having an iterator in both directions and for no sequence found it will start search from beginning. It is observed that with this modification in general there is a significant reduction of time with increase in number of processors as is shown in Fig. 17 by curve modified SAMRAI. The amount of reduction of time keeps reducing with increase in number of processors involved, owing to the decrease in amount of cells reduction handled by each processor and increase in communication time though very small. Communication time has been observed not to depend much on the number of processors involved in simulation and was 1 or under 1 sec for most of the cases in example except with 3 processors where it was between 1 and 2 sec's. This is believed to be because of increase in search operation described above due to increase in number of cells not in sequence. The least unit of time here is sec's so further details not possible but it is has been observed that variation in communication time for a simulation problem is never more then 1sec's with increase in number of processors if load balancing is done carefully.

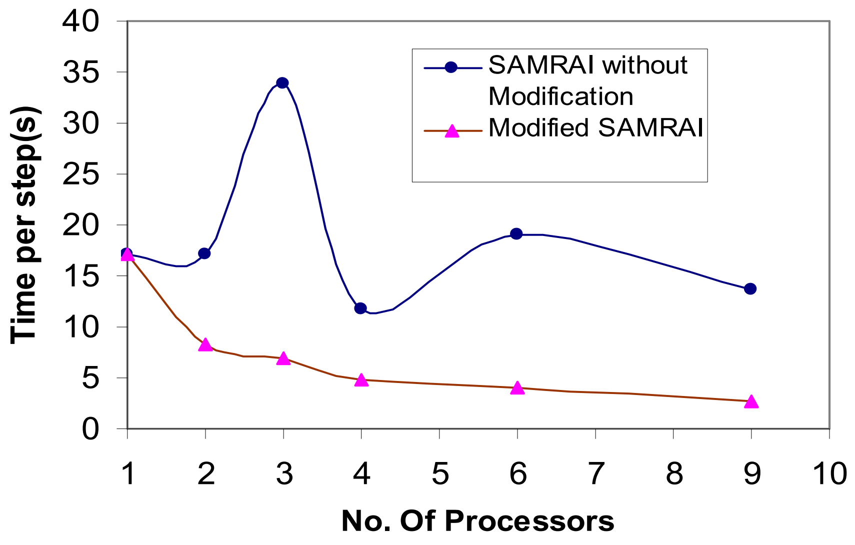


Fig.17: Time taken per step for number of processors

Though parallel processing in 3D-GIMP is very effective in reducing computational time but with contact solver designed to solve the contact equations in one processor, some of the advantages parallel processing is lost and with the increase in number of contact equations, the amount of time saving keeps decreasing. One way two solve this problem is to use processor with high processing speed to solve contact equations and other way is to use parallel processing for solving contact equations.

7.2 Validation of 3D-GIMP results with parallel processing and advantages over FEM

The second example discussed here is to simulate the indentation problem using rigid conical indenter and considering quarter model only using symmetric boundary conditions as shown in Fig. 18(a). This model uses 4 patches spanning over 4 processors with uniform grid with uniform cell sizes of 64 nm. The work piece dimensions are 1920 nm × 1920 nm × 960 nm for quarter model. The material of work piece is aluminum and is assumed linearly elastic throughout the simulation. The time step is 1.03453×10^{-12}s. This model is used to validate results from 3D GIMP using parallel computation scheme and contact algorithm developed for 3D contact problem developed in this investigation. The contact interface at larger depth spans over multiple processors and which proves the capability of parallel contact solver scheme in accurately handling such cases.

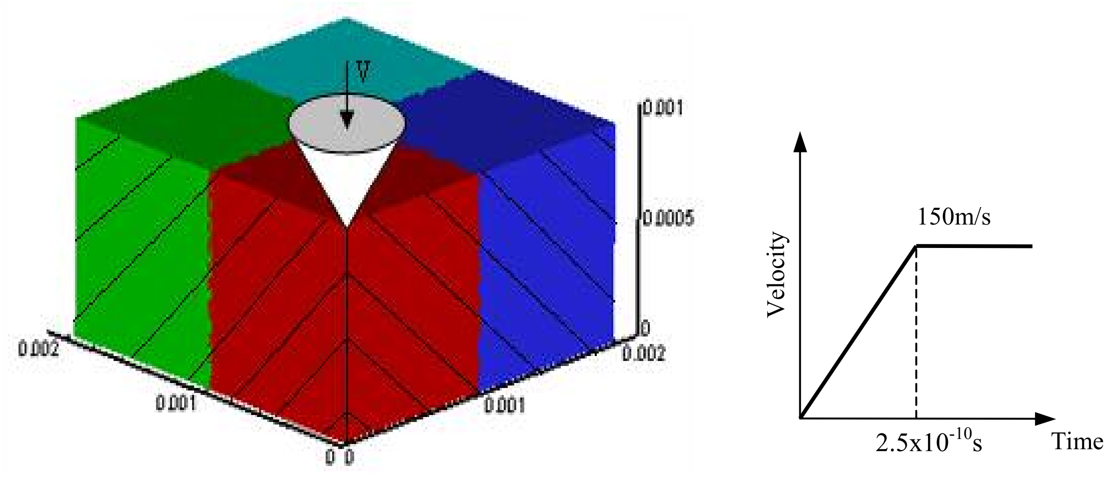


Fig.18 (a): Schematic of indentation Model used with 4 Patches and indenter velocity history

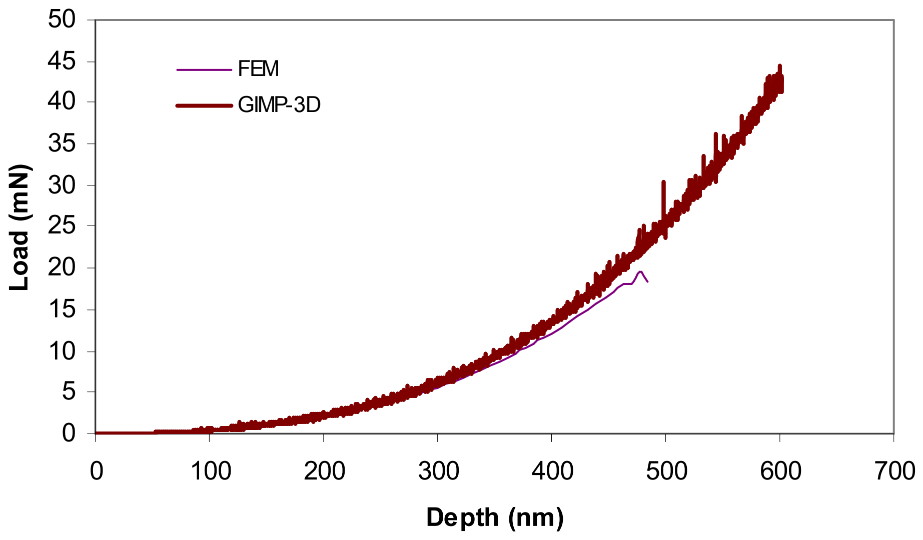


Fig. 18(b): Load Displacement Curves for Indentation problem.

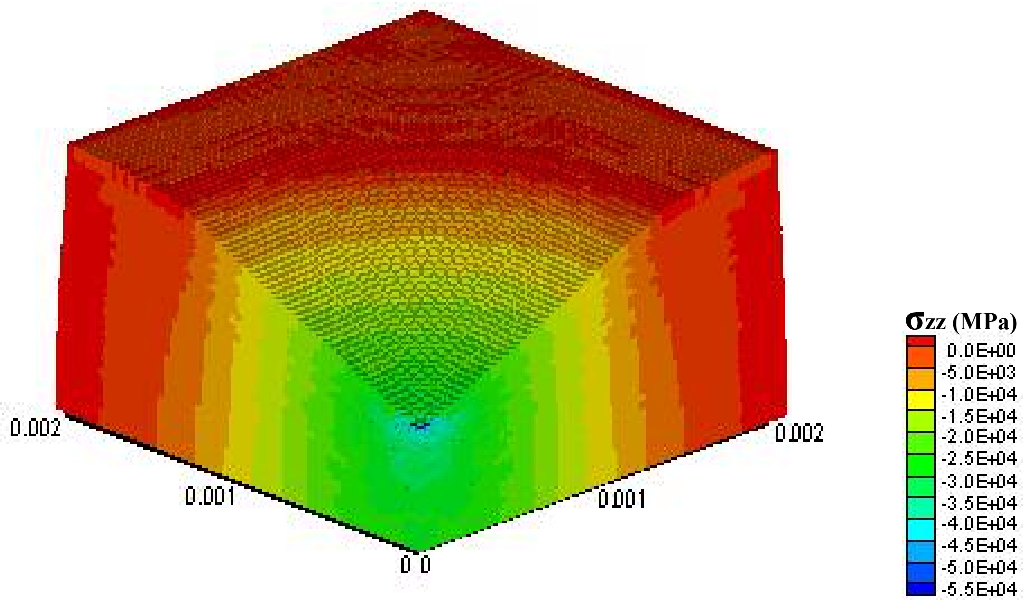


Fig. 18(c): 3D view of model at 3800 step from GIMP

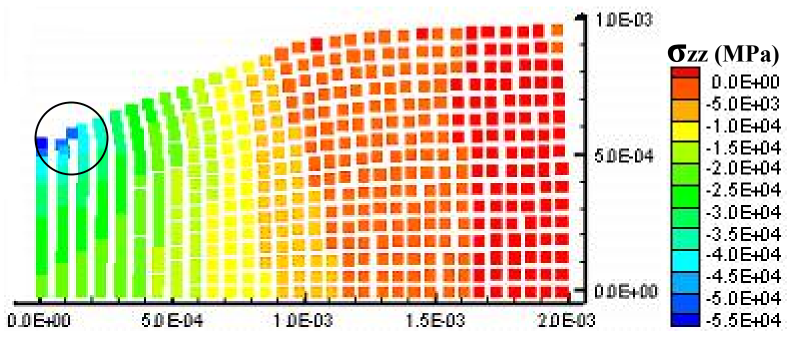


Fig. 18(d): Side view of model at 2800 step from FEM

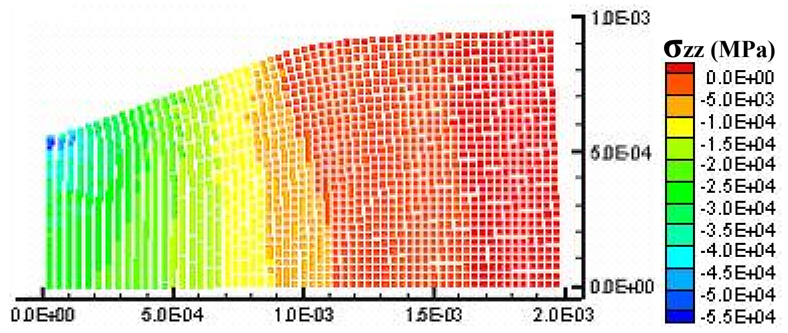


Fig. 18(e): Side view of model at 2800 step from 3D GIMP

Fig.18: Comparison of results from GIMP and FEM

It is observed that at larger depth of Indentation FEM suffers excessive mesh distortions and finally aborts but 3D GIMP runs without any problem as shown by Load – Displacement curves obtained in Fig. 18(a). The Load-Displacement curve obtained from GIMP 3D is having some noises but for FEM, the results are plotted after every 50 steps and look relatively smooth. The noises are present because of discontinuities involved and for FEM results are better because of doing appropriate smoothing of discontinuities [31]. The noises are observed to be reduced by using slower velocities and refinement. The noises can also be reduced by selecting suitable penetration factor and introducing artificial damping (numerical damping) in GIMP 3D.In this example a penetration factor of 0.4 is used which is found to be more effective in eliminating noises for one particle per cell as can be seen in examples that will be followed. This demonstrates the capability of GIMP in handling excessive distortions. Fig. 18(d) and (e) are Side views of the model with contours plots of normal stresses in Z- direction at 2800 time increments for GIMP and FEM simulations for closer comparison. The difference in stress values is less then 10%. The indentation depth reached at this stage is about 415 nm approximately. It should be noted that the FEM mesh has started distorting even at 2800 steps and simulation finally aborted at 3240 increments due to excessive element distortion. The GIMP simulation did not encounter this problem. Fig. 18(c) shows the GIMP stress result after 3800 increments with indentation depth of 571nm.

7.3 Validation of 3D-GIMP results with two levels of Refinement and with one particle per cell

The 3D-GIMP simulation simulations can also be done using initially one material point per cell instead of eight. In principle the only thing changes is the weighing function for this case which can be easily computed using same material point characteristic function and node shape function as in Bardenhagen and Kober [13] , Fig. 19.

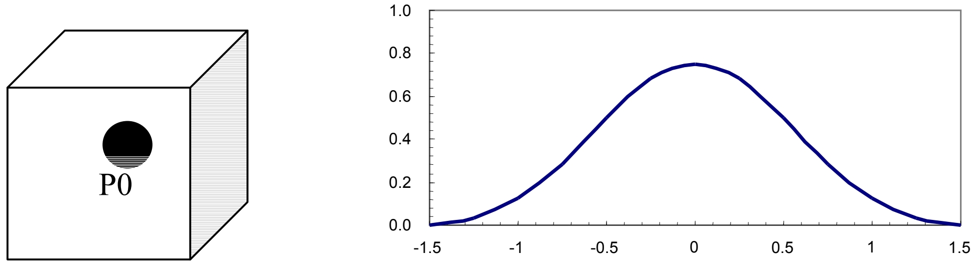


Fig.19: One Material points in cell and the weighting function for 1D in GIMP method

A simulation with one particle per cell takes less computation time and results are generally same as compared to eight particles per cell but have been observed to have problems of separation at relatively large tensile strain for very large deformation but still have been observed to simulate deeper indentation correctly in comparison with FEM without problem. The problem of separation can be resolved by tracking the displacement of the particle corners but has not been incorporated yet. It has been observed that difference in maximum stress is around 10% between the simulation conducted with eight particle per cell and with one particle per cell and that is because of having a point more closer to indenter tip in eight particle per cell as compared to one particle per cell but except that the over all region there is not much difference. There has been observed no change in load displacement curves. Considering this further simulation has been conducted using one particle per cell to save computation time .

This example also demonstrates the advantage of 3D-GIMP over FEM in simulating deep indentation or deep drawing problem but using one particle per cell in GIMP. The simulation conditions are same as above example Fig. 18(a) except the indenter used in this case is rigid surface spherical indenter and instead of 4 patches only 1patch per level is used. The simulation is conducted with first using one level and then using two Levels of refinement and results are compared with FEM using coarse mesh having element size of $1.28\times1.28\times1.28\ \mu m^3$ same as cell in 1 level GIMP grid and with fine mesh having element size of $0.64\times0.64\times0.64\ \mu m^3$ same as cell in 2nd level in GIMP. The dimensions of work piece are $56.32\mu m\times56.32\mu m\times32\mu m$ for quarter model and material is aluminum. The radius of spherical indenter is $3.4\mu m$. Time step value for coarse level is $2.06906\text{x}10^{-}$

^{11}s. FEM quits (Fig. 20(b)) earlier then GIMP and the maximum depth it reaches is around 3.114 μm with fine mesh but valid results can only be obtained until some steps before it. 3D-GIMP simulation was stopped at 3.5 μm indentation depth as indenter is now fully inside the work piece. Though practically for indentation purpose we don't go for such a depth but the important point here is that for simulating other large deformation contact problems such as deep drawing process GIMP can handle larger distortions then FEM. Also it is observed that generally there is not much difference in load-displacement curves with refinement except FEM quits early with coarse mesh but the difference in stresses is significant and is more then 30% for maximum stress values as shown in Fig. 20(f), (g), (h) and (i). For GIMP 3D for one level some noises are observed as seen in Fig. 20(a) but for FEM the results are obtained skipping few steps so it is not possible to tell if same problem is there or not. Stress distribution is much smoother for fine mesh. Here the stress color bars in Figures does not show the maximum stress values and has been adjusted to show transition of stresses from fine level to coarse and has been observed to be very smooth. Hence, in nutshell to better simulate large deformation contact problems fine mesh is preferable. Refinement can be easily achieved in 3D-GIMP without using fine grid for whole region and results in a lot of saving in computational time and computer memory requirement. The results with coarse grid (Fig. 20(i)) and with refinement (Fig. 20(g)) are compared with FEM coarse (Fig. 20(h)) and fine mesh (Fig. 20(f)) and compares very well.

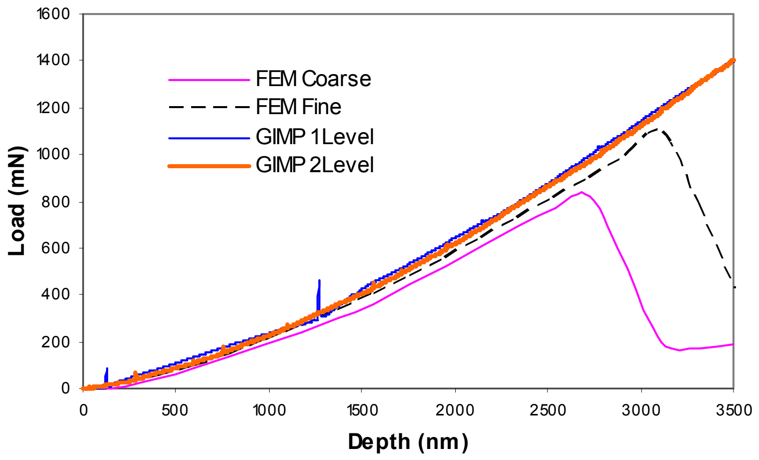


Fig. 20(a): Comparison of Load Displacement curve from FEM and GIMP3D

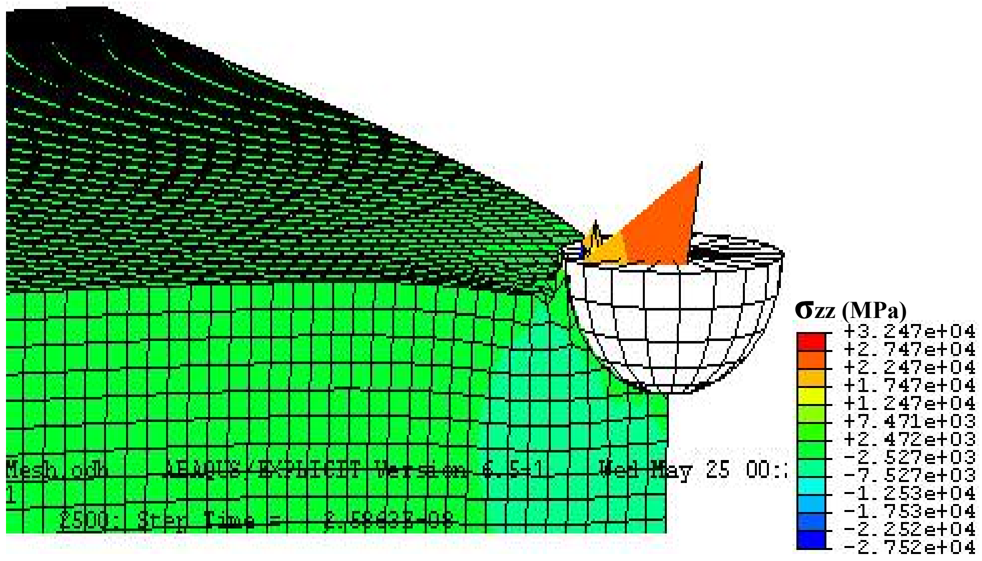


Fig. 20(b): Zoomed view of model from FEM (ABAQUS) at 3.114µm indentation depth

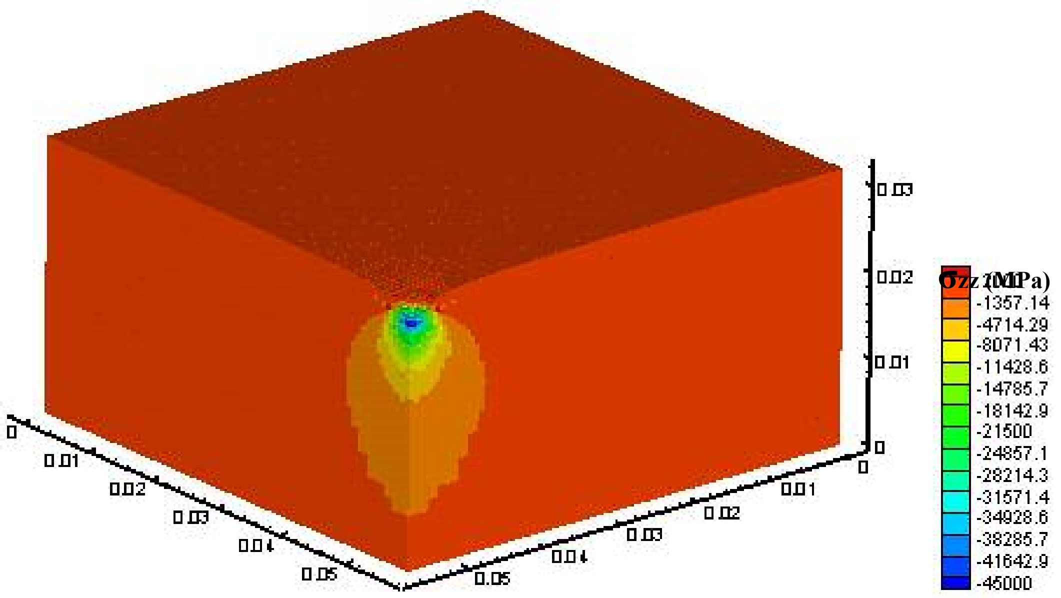


Fig. 20(c): 3D view of model from 3D-GIMP with two levels at 3.114μm indentation depth

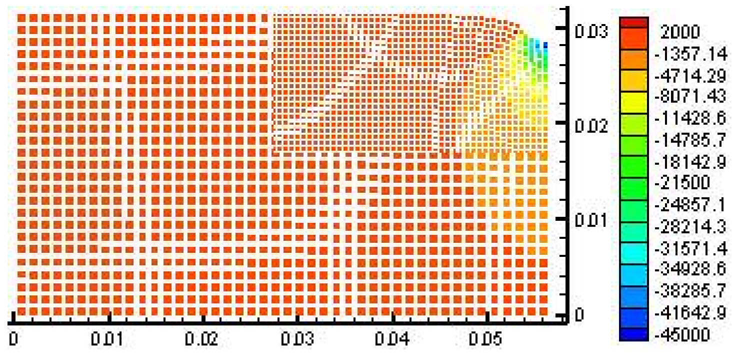


Fig. 20(d): Side view of model (scattered plot) in Fig. 20(b)

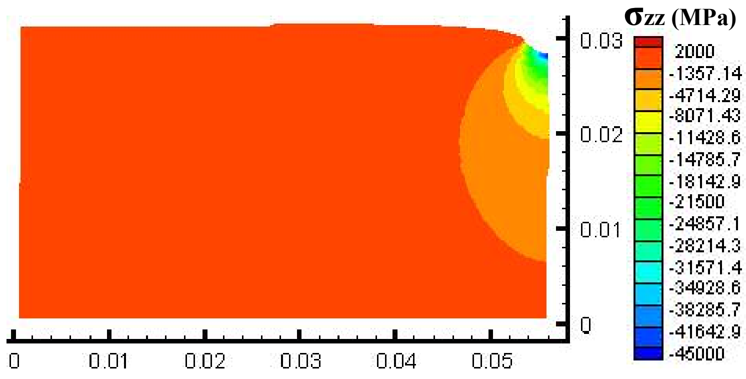


Fig. 20(e): Side view of model in Fig. 20(b)

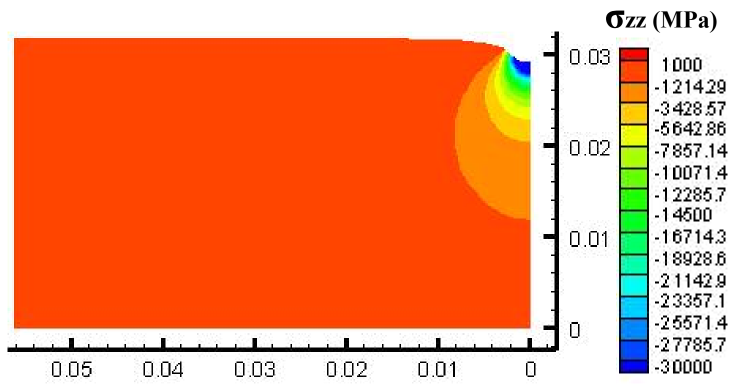


Fig. 20(f): Side view of model from FEM with Fine Mesh at 2.73µm indentation depth

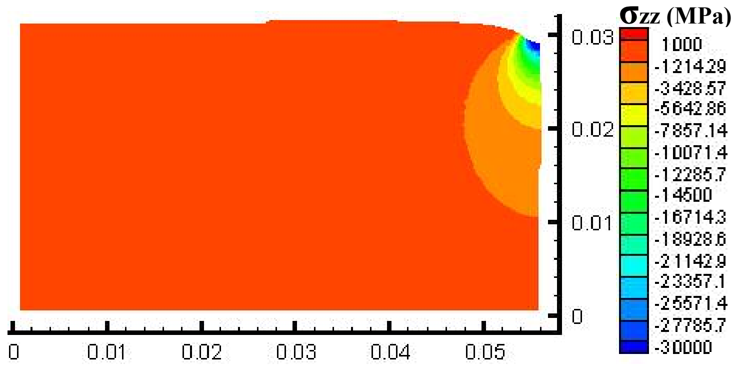


Fig. 20(g): Side view of model from GIMP with 2 Levels at 2.73µm indentation depth

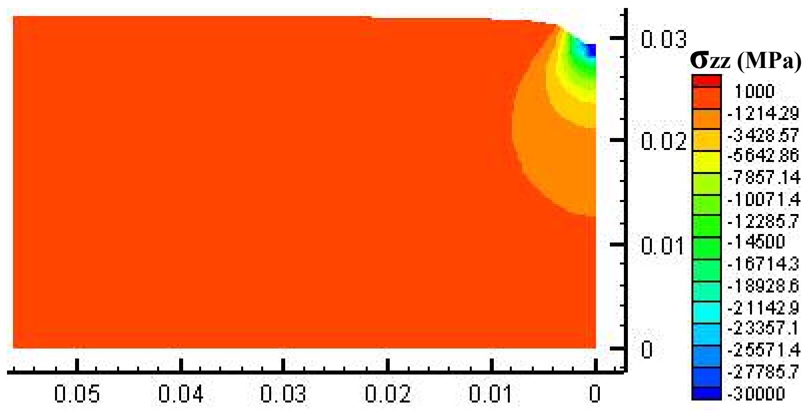


Fig. 20(h): Side view of model from FEM with Coarse Mesh at 2.73µm indentation depth

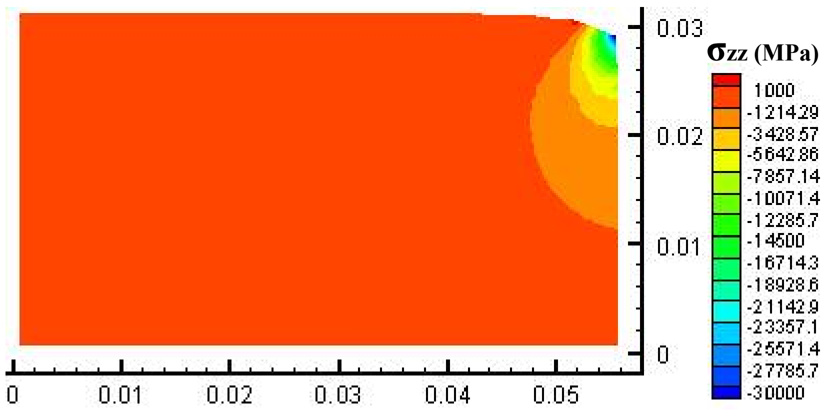


Fig. 20(i): Side view of model from GIMP with 1 Levels at 2.73μm indentation depth

Fig. 20: Comparison of results from GIMP 3D and FEM

Hence, from above two examples it can be summarized that the maximum difference in load-displacement values for Indentation problems has been observed to be less then 10% and difference in maximum stress values has been observed to be around 10 % for simulations using 8 particles per cell and 22% for 1 particle per cell. This high difference in the stress values in case of simulations run with the use of 1 particle per cell can be explained based on the fact that in FEM the node associated with maximum stress value is at the indenter tip whereas in GIMP the nearest particle is about half cell length away from the indenter tip. A penetration factor of 0.4 has been observed to reduce noise in Load-Displacement curve associated with indentation problems in GIMP significantly and has been used in all simulations.

7.4 Results with three levels of refinement consisting four Patches each level in 3D-GIMP using 4 Processors and comparison with FEM

This fourth example included here is to demonstrate the capability and advantages of using multi-level refinement scheme in addition to capabilities demonstrated in previous example. Here full model is considered for simulation as shown in Fig. 15 and Fig. 21 (a) with three levels of refinement as shown in Fig. 21(a) and each level is consisting of 4 patches handled each by single processor from the 4 processors used for the simulation. This model also demonstrates the capability of parallel contact solver in handling computations accurately when the patches are arranged in this configuration. Hence, the first and second examples with this most varied and common arrangement of patches shows the capability of parallel contact solver in handling any kind of such configurations accurately. The material of work piece is aluminum and is assumed linearly elastic throughout the simulation.

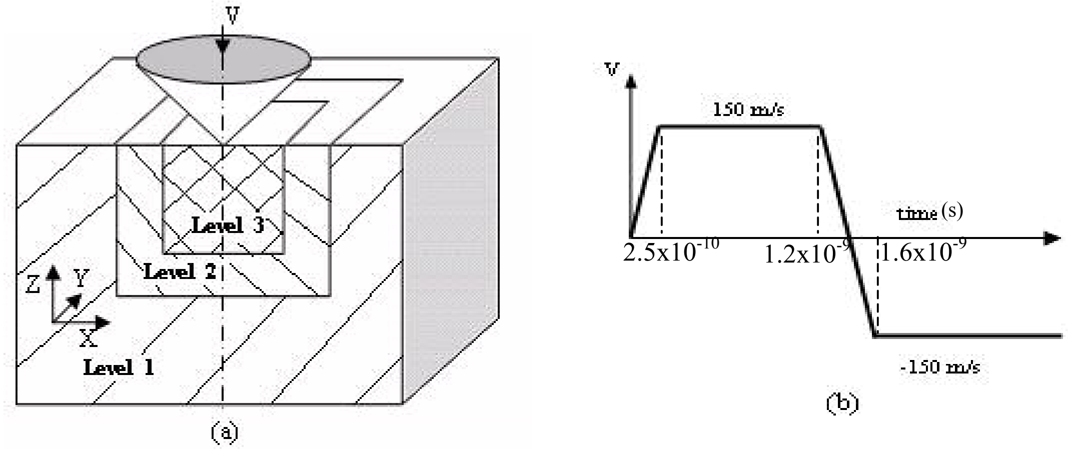


Fig. 21: 3D Sectioned view of indentation showing (a) three levels of refinement and (b) the indenter velocity history

The area below the indenter where high stress gradients are expected is refined, as shown in Fig. 21 (a). A prescribed velocity applied on the indenter, is shown in Fig. 21(b). Work piece size is $5632\times5632\times1600\text{nm}^3$. Fig. 22(c), Fig. 22(d) and Fig. 22(e) shows results at approximately 166.8nm depth from GIMP 3D and Fig. 22(f) shows results from FEM using fine mesh(transitioned). At same depth for comparison. Cell length in all 3 directions for fine, coarse and coarsest levels in GIMP is 16nm, 32nm and 64nm and minimum Element length in FEM is 15nm. The value of coarsest time step is 1.03453×10^{-12} s.

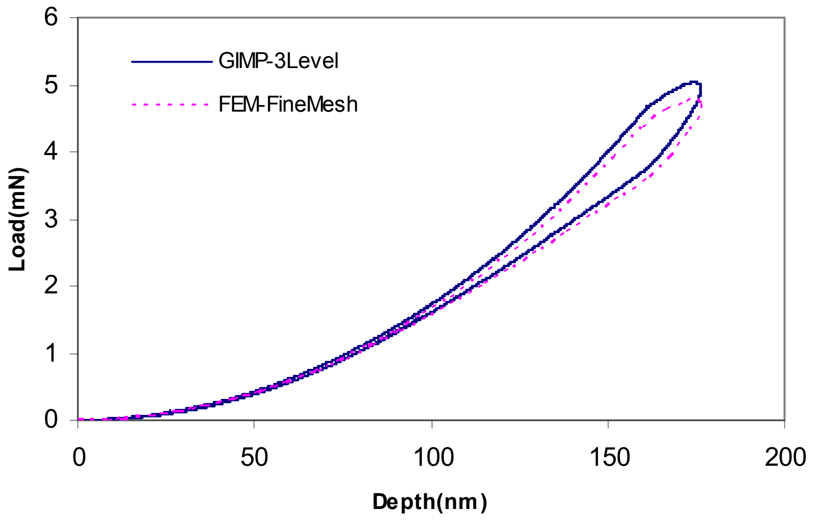


Fig. 22(a): Comparison of Load Displacement curve from FEM fine and GIMP3d 3levels

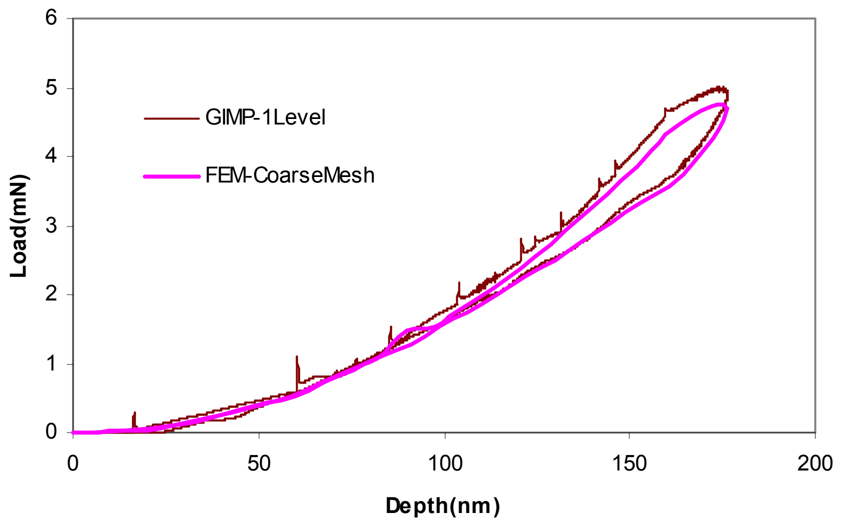


Fig. 22(b): Comparison of Load Displacement curve from FEM and GIMP3d Coarse

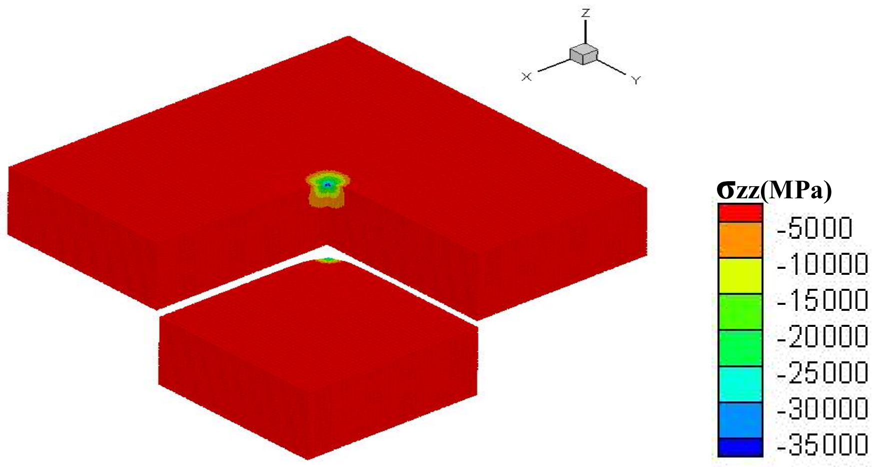


Fig. 22(c): Nanoindentation model from GIMP3d with Normal stresses in Z-axis at 166.8nm indentation depth

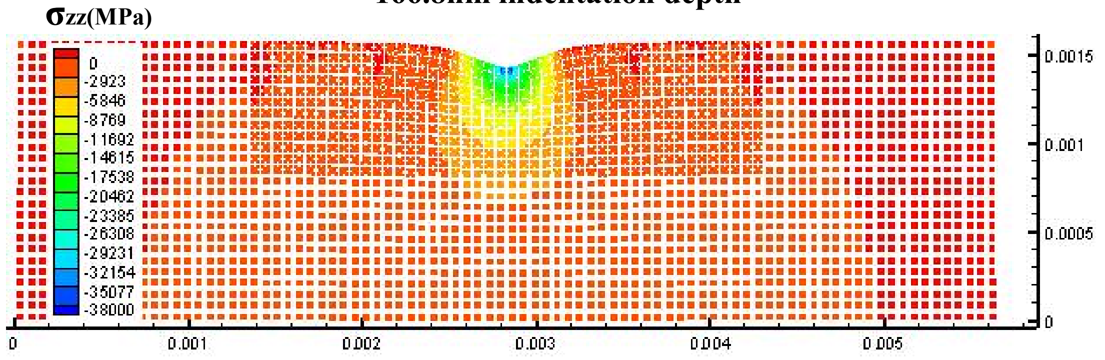


Fig. 22(d): Sectioned view with Normal stresses along Z-axis from 3D GIMP with Three Levels of refinement at 166.8 nm indentation depth, scattered plot

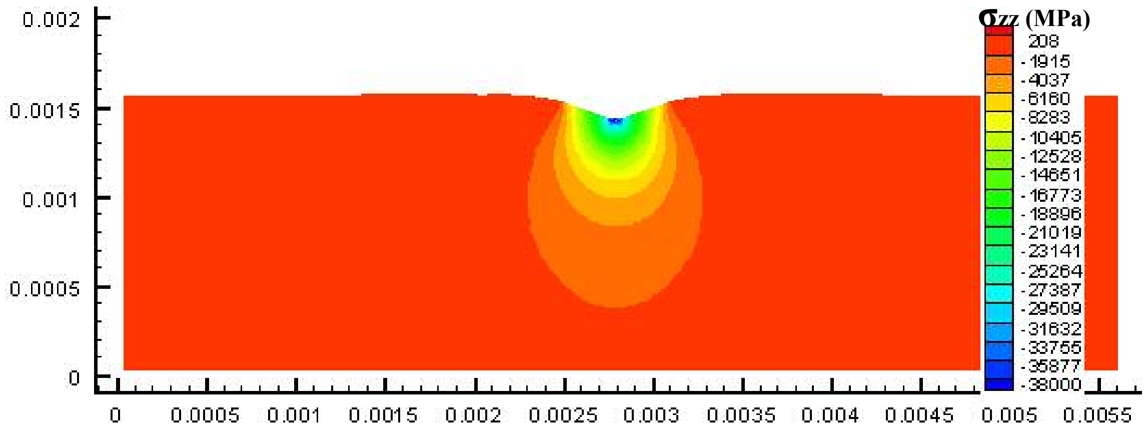


Fig. 22(e): Sectioned view with Normal stresses along Z-axis from 3D GIMP with Three Levels of refinement at 166.8 nm indentation depth

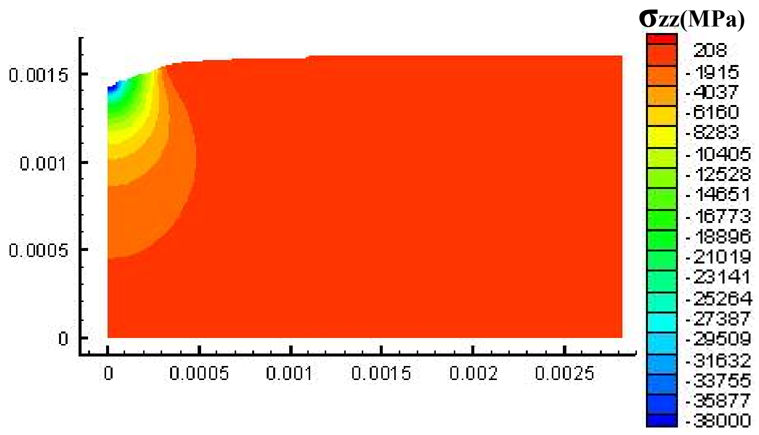


Fig. 22(f): Side view with Normal stresses along Z-axis from FEM with fine mesh at 166.8nm indentation depth.

Fig. 22: Comparison of results from GIMP 3D and FEM

Simulation is conducted using one Particle per cell. The maximum difference in load at a given displacement curve values for Indentation problems has been observed to be less then 8% and difference in maximum stress values has been observed to be 23.5%. This high difference in the stress values is associated with simulations performed using one particle per cell and is not much for the simulations having eight particles per cell initially where difference has been observed to be less then 10%. The rest of the region except for indenter tip stress contours from GIMP compares very well with FEM. Load-displacement curves have been observed to vary very little with refinement as is shown in Fig. 22(a) and 22(b) but are much smoother for fine mesh.

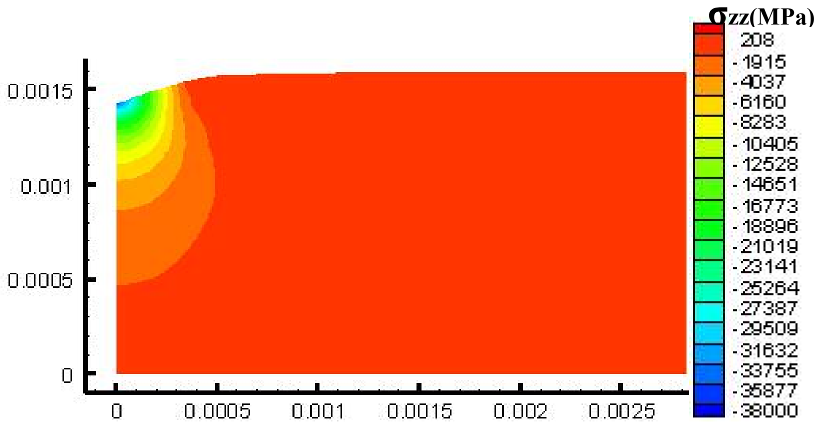


Fig. 23(a): Side view with Normal stresses in Z-direction from FEM with coarse mesh at 166.8nm indentation depth

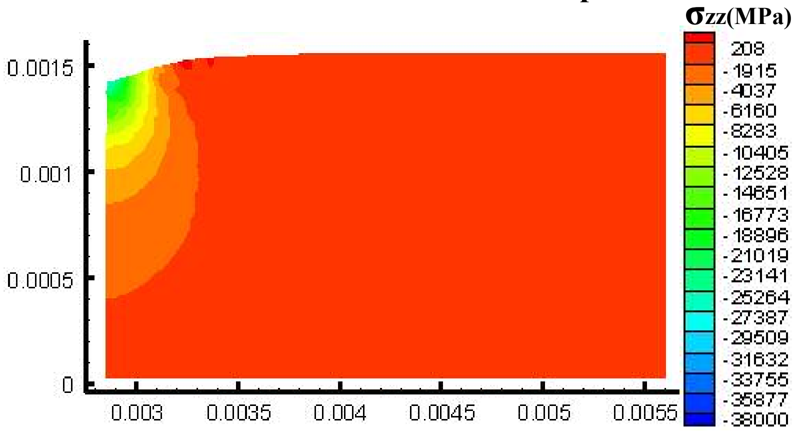


Fig. 23(b): Side view with Normal stresses in Z-direction from 3D GIMP with coarse grid at 166.8nm indentation depth.

Fig. 23: Comparison of results from GIMP 3D 1-level and FEM coarse

Stress has been observed to be sensitive to mesh refinement as is shown in Fig. 22 and 23 with difference in the maximum stress values in fine and coarse mesh to be around 34% in FEM and 34.7% in 3D GIMP. The difference in maximum stress values has been observed to be 24.3% between 3D GIMP one level results and FEM coarse results. The element size of coarse mesh used in FEM and cell size of uniform coarse grid used in 3D

GIMP in one level to simulate this same problem for comparison stresses is $64\times64\times64nm^3$.

7.5 Validation of 3D-GIMP results considering bilinear plastic material behavior

Next example considers elastic-plastic behaviour of material. A plasticity subroutine has been implemented assuming bilinear plasticity and is used with refinement scheme. A quarter model with model size 4x4x7 mm^3 is used for indentation considering symmetric boundary conditions and pressure P= 400MPa is applied on an area 1x1mm^2 as shown in Fig. 24. Material used for work piece has Young's modulus 70GPa, density 2.71g/cm^3 and Poisson's ratio 0.33 and yield strength of 60MPa with hardening modulus of 3.5GPa.

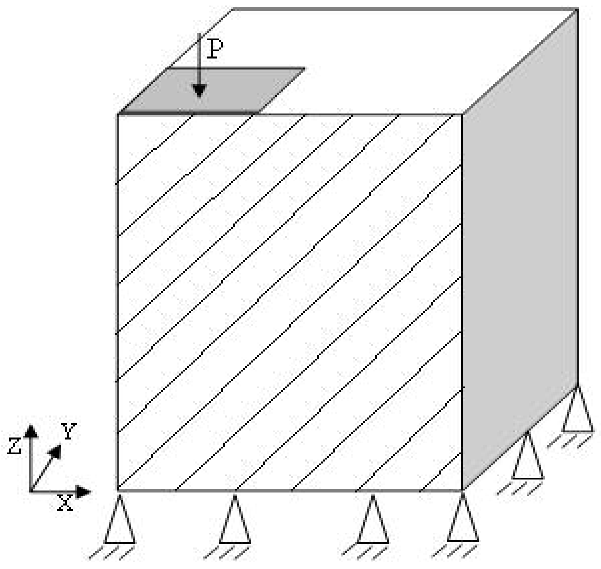


Fig. 24 Schematic of the model used for indentation

Simulations are conducted in FEM and GIMP with first coarse mesh (FEM) or grid(GIMP) with element(FEM) or cell(GIMP) size of $0.5\times0.5\times0.5$ mm^3 and then in FEM with fine mesh of element size $0.25\times0.25\times0.25$ mm^3 and in GIMP with 2-Levels of uniform cell size of 0.5×0.5×0.5 mm^3 in coarse and 0.25×0.25×0.25 mm^3 in fine level. The normal stresses obtained for model along Z-axis obtained compare very well with results from FEM as shown in Fig. 25 and Fig. 26.

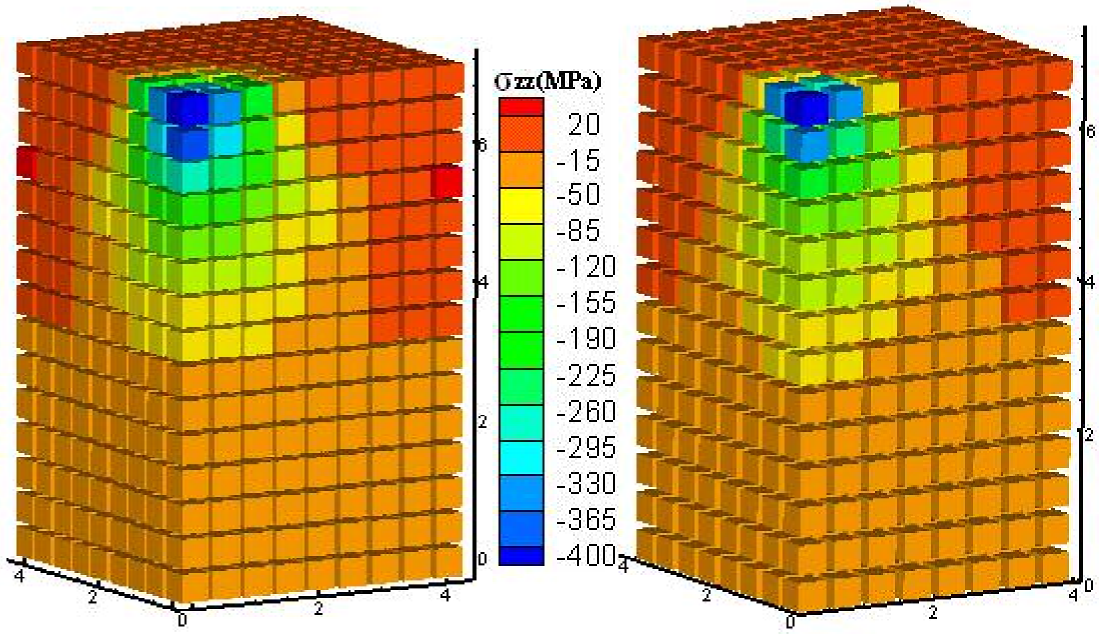


Fig. 25: Comparison of normal stresses in Z-direction from FEM and GIMP coarse

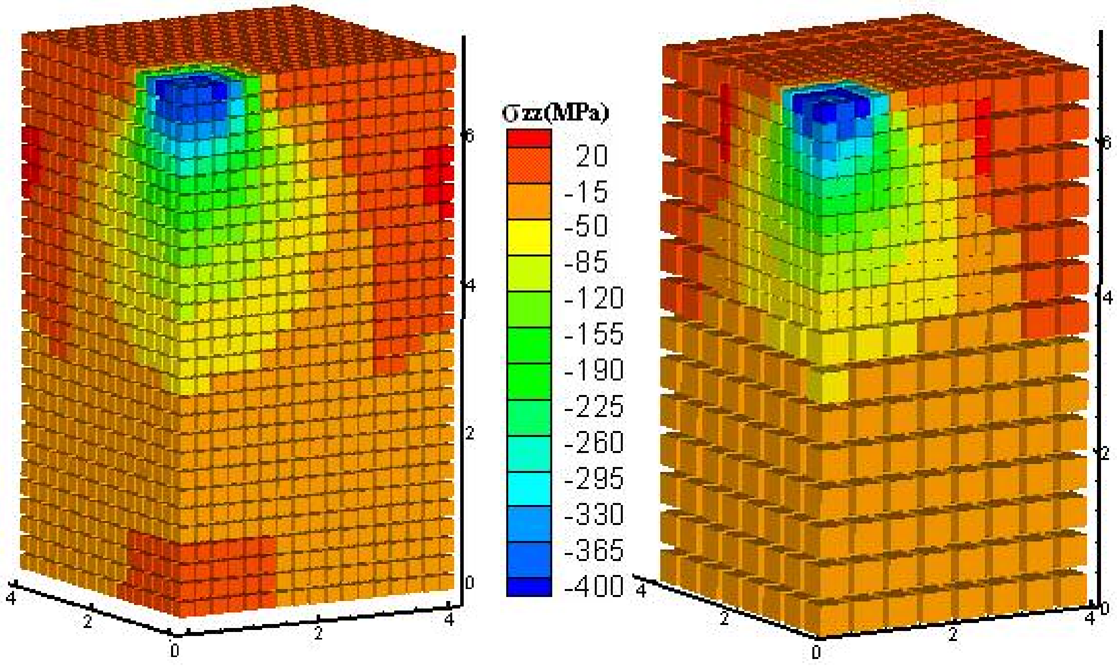


Fig. 26: Comparison of normal stresses in Z-direction from FEM and GIMP fine

For closer comparison the time histories of important physical variables such as Von Misses stress, normal stresses along Z-axis (σzz), equivalent plastic(EQPL) strain and Principle stresses along x-axis(Sxx) which will be equal to principle stresses along y-axis are plotted for a point close to indenter tip in Fig. 27.

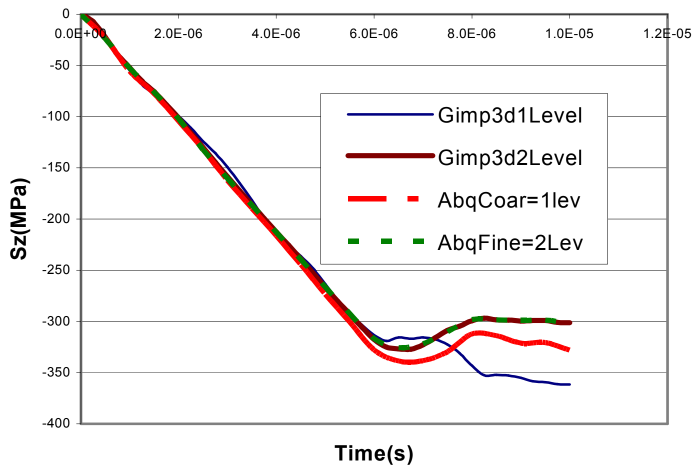


Fig. 27(a): History of Principle stresses along z-axis

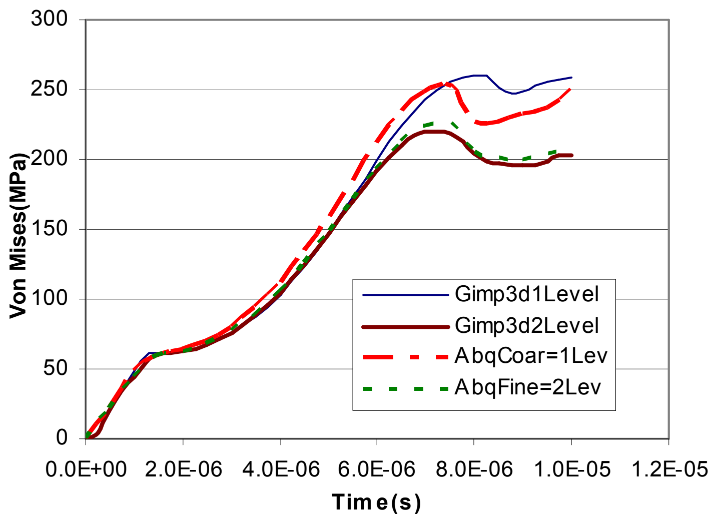


Fig. 27(b): History of Von Messes stresses

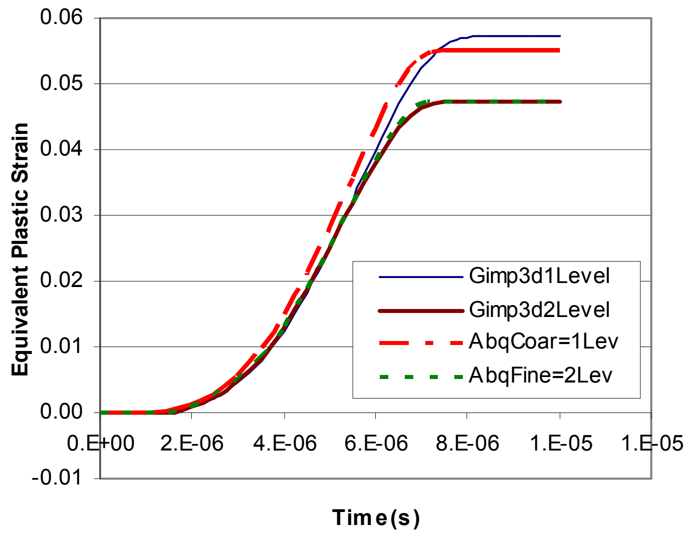


Fig. 27(c): History of Equivalent Plastic strain

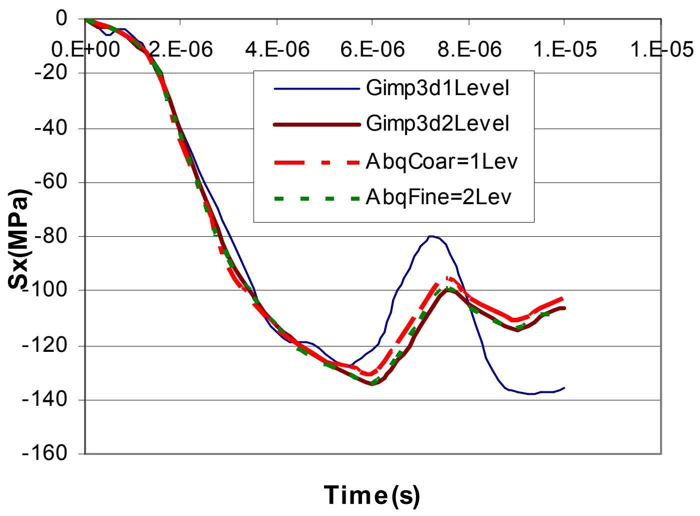


Fig. 27(d): History of Principle stresses along x-axis

Fig. 27: Time histories of important physical variables

It is observed from the plots above that there is a lot of difference in the results obtained from fine and coarse mesh in 3D-GIMP at the later stages as compared to FEM, which shows that 3D-GIMP, is more sensitive then FEM to cell size. Other reason could be that to compare the results at the same location as coarse particle in 3D-GIMP 1-Level only an average of surrounding particles is taken in case of 2-Levels in 3D-GIMP whereas in FEM centroid of the element is used and that can induce differences.

CHAPTER 8

CONCLUSIONS AND FUTURE WORK

8.1 Conclusions

1. To conduct multiple length scale simulations involving large deformations, a parallel computing scheme has been presented using GIMP 3D method under SAMRAI parallel computing environment in which multi-level grids are used for spatial and temporal refinements.

2. 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.

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

4. 3D-GIMP simulations were conducted for a indentation problems using multiple processors in parallel computations to study the effect of parallel computation on computation time. Parallel processing using SAMRAI has proved to be very effective in reducing computation time in simulations with very little increase in communicational time with increase in number of processors.

5. Simulations were conducted to validate the 3D-GIMP results using parallel computations and with a few levels of refinement for 3D indentation problems. In general 3D-GIMP results agree very well with FEM results.

6. Simulations for indentation problem were conducted using one particle per cell and with eight particles per cell; with conical indenter and with spherical indenter, and results agree very well with FEM results for all the cases.

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

8.2 Future Work

1. The present code should be extended to elastic-plastic (multi-linear) and meso-plastic materials.

2. To simulate multi-scale simulation spanning from atomistic to continuum scale via meso-plasticity present code should be coupled with MD code after introducing meso-plasticity in it.

3. To reduce computational time present contact solver should be modified to solve contact equations in parallel.

4. To resolve problems related to separations for simulations involving large strains 3D-GIMP code should be modified to track the displacement of the particle corners.

5. To simulate indentation problems involving other common type of indenter i.e. Bercowich indenter subroutines should be added to the code to find contact normal and detecting contact with this type of indenter.

6. Similarly, other features can be added and GIMP algorithm can be modified to simulate other problems in Solid mechanics without need to modify parallel processing and refinement part.

REFERENCES

1. Harlow, F.H., “The Particle-in-Cell Computing Method for Fluid Dynamics,” Methods in Computational Physics, Editors Alder, B., Fernback,S., and Rotenberg., 3 (1964) 319-343

2. Sulsky, D., Chen, Z., and H. L. Schreyer, “A Particle Method for History-Dependent Materials,” Comp. Meth. in Appl. Mech. and Eng.118 (1994) 179-196

3. Amsden,A.A., “The Particle-in-Cell Method for the Calculation of the Dynamics of Compressible Fluids,” LA-3466, Los Alamos National Laboratory, Los Alamos, New Mexico. (1966)

4. Benson, D., “Computational Methods in Lagrangian and Eulerian hydrocodes,” Comput. Meth. Appl. Mech.Eng., 99 (1992) 235-394

5. Belytschko, T., “An Overview of Semidiscretiztion and Time Integration Procedures,” Computational Methods for Transient Analysis, Vol I, T.Belytschko and T.J.R Hughes, ed.,Elsevire Science Publishers (1983)

6. Belytschko, T., Krongauz, Y., Organ, D., Fleming, M., and Krysl, P., “Meshless Methods:An Overview and Recent Developments,” Computer Methods in Apllied Mechanics and Engineering 139 (1996) 3-4

7. Shen, Y-L., Li, W., Sulsky, D. L., and H. L. Schreyer, “Localization of Plastic Deformation Along Grain Boundaries in a Hardening Material,” Int. J. of Mech. Sci. 42 (2000) 2167-2189

8. Tan H., and J. A. Nairn, “Hierarchical, Adaptive, Material Point Method for Dynamic Energy Release Rate Calculations,” Comput. Methods Appl. Mech. Eng. 191 (2002) 2095-2109

9. Nairn, J. A., “Material Point Method Calculations with Explicit Cracks,” Tech Science Press CMES, (2003).

10. Burgess, D., Sulsky, D., and Brackbill, J.U., “Mass Matrix Formulation of the FLIP Particle-in-cell Method,” Journal of Computational Physics 103 (1992) 1-15

11. Bardenhagen, S. G., “Energy Conservation error in the material point method for solid mechanics,” J. of Computational Physics 180 (2002) 383-403

12. Bardenhagen, S. G., Brackbill, J. U. and D. Sulsky, “The Material Point Method for Granular Materials,” Comput. Methods Appl. Mech. Eng. 187 (2000) 529-541

13. Bardenhagen, S.G.; Kober, E.M, “The generalized interpolation material point method,” Computer Modeling in Engineering & Sciences. vol. 5, n. 6, pp. 477-496 (2004).

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

15. Ma, J.; Lu, H.; Wang, B., Roy, S.; Hornung, R.; Wissink, A.; Komanduri, R. "Multiscale simulations using generalized interpolation material point method(GIMPM) and SAMRAI parallel processing," submitted (2004)

16. Champaney, L.; Cognard, J.Y.; Dureisseix, D.; Ladeveze, P. "Large scale applications on parallel computers of a mixed decomposition method," Computational Mechanics 19 (1997) 253-263

17. 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

18. 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

19. 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

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

21. 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

22. 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.

23. Mackerle, J. (2003): "FEM and BEM parallel processing: theory and applications- a bibliography," *Engineering Computation,* vol. 20, n. 4, pp. 436-484

24. Oden, J.T.; Pires, E.B. (1983): "Numerical analysis of certain contact problems in elasticity with non-classical friction laws," *Computers & Structures,* vol. 16, n. 1-4, pp. 481-485

25. 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

26. 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

27. 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

28. 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

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

30. El-Abbasi, N.; Meguid, S.A. "Large deformation analysis of contact in degenate shell elements," International journal for numerical methods in engineering 43, 1127-1141(1998)

31. Barber, J.R.; Ciavarella, M. "Conatct mechanics" International Journal of solids and structures 37 (2000) 29-43

32. Fourment, L.; Chenot, J.L.; Mocellin, K. "Numerical formulations and algorithms for solving contact problems in metal forming simulation," International journal for numerical methods in engineerin 46, 1435-1462(199)

33. Heiss, H.U.; Dormanns, M. "Mapping large parallel simulation programs to multicomputer systems," High Performance Commputing (1994)

34. Womble, D.E.; Dosanjh, S.S.; Hendrickson, B.; Heroux, M.A.; Plimpton, S.J.; Tomkins, J.L.; Greenberg, D.S. "Massively parallel computing: A Sandia perspective," Parallel Computing 25 (1999) 1853-1876

35. York, A.R.; Sulsky, D.; Schreyer, H.L. "The material point method for simulation of thin membranes," International journal for numerical methods in engineering 44, 1429-1456(1999).

36. Brown, D.; Maigret, B. "Large scale molecular dynamics simulations using the domain decomposition approach," Universite de Nancy I, BO 239, 54506, France

37. Hwang, Y.S.; Das, R.; Saltz, J.H.; Hodoscek, M.; Brooks, B. "Parallelizing molecular dynamics for distributed memory machines: an application of the CHAOS runtime support library," UMIACS and Department of Computer Science

38. Sulsky, D., Ayton, G., Bardenhagen, G.S., McMurtry, P., Voth, G.A. “Interfacing Continuum and Molecular Dynamics: An Application to Lipid Bilayers,” Journal of Chemical Physics, 114 (15) (2001) 6913-6924

39. Cummins, S. J. and J.U. Brackbill, “An Implicit Particle-in-Cell Method for Granular Materials,” J. of Computational Physics 180 (2002) 506-548

40. Chen, Z., Hu, W., Shen, L., Xin, X., and R. Brannon, “An Evaluation of the MPM for Simulating Dynamic Failure with Damage Diffusion,” Engineering Fracture Mechanics 69 (2002) 1873-1890

41. Herrmann, W., and Bertholf, L.D., “Explicit Lagrangian Finite-difference Methods,” Computational Methods for Transient Analysis, Vol. I, T.Belytschko and T.J.R Hughes, ed., Elsevier Science Publishers (1983)

42. Schreyer, H.L., Sulsky, D. L. and S.-J. Zhou, “Modeling delamination as a Strong Discontinuity with the Material Point Method,” Comput. Methods Appl. Mech. Eng. 191 (2002) 2483-2507

43. Chen, Z. and R. Brannon, “An Evaluation of the Material Point method,” SAND Report, SAND2002-0482, (February 2002).

44. Hu, W. and Z. Chen, “A Multi-mesh MPM for Simulating the Meshing Process of Spur Gears,” Computers and Structures 81 (2003) 1991-2002.

45. York, A. R., Sulsky, D. L., and H. L. Schreyer, “The Material Point Method, for Simulation of Thin Membranes,” Int. J. Numer. Eng. 44 (1999) 1429-1456

46. Sulsky, D and A. Kaul, “Implicit Dynamics in the Material-Point Method,” Comp. Meths. Appl. Mechs. Engrg., 193 (2004) 1137-1170

47. Guilkey J. E., and J.A. Weiss, "Implicit Time Integration for the Material Point Method: Quantitative and Algorithmic Comparisons with the Finite Element Method," Int. J. Numer. Meth. Eng.; 57 (9) (May 2003) 1323-1328.

48. Attaway, S.W., Heinstein, M.W., Mello, F.J., and Swegle, J.W. "Coupling of Smooth Particle Hydrodynamics with finite element method," Nuclear Eng. Design, 150 (1994) 199--205

49. Liu, W.K., Jun, S., Li, S.F., Adee, J., Belytschko, T., "Reproducing Kernel Particle Methods for Structural Dynamics," International Journal for Numerical Methods in Engineering 38 (10) (1995) 1655-1679

50. Onate, E., Idelsohn, S., Zienkiewicz, O.C., Taylor, R.L., "A Finite Point Method in Computational Mechanics: Applications to Convective Transport and Fluid Flow," International Journal for Numerical Methods in Engineering 39 (22) (1996) 3839-3866

51. Pipkins, D.S., Atluri, S.N., "Applications of the Three-dimensional Finite Element Alternating Method," Finite Elements in Analysis and Design 23 (2-4) (1996) 133-153

52. Chen, J.S., Pan, C., Wu, C.T., Liu, W.K., "Reproducing Kernel Particle Methods for Large Deformation Analysis of Non-linear Structures", Comp. Meth. in Appl. Mech. and Eng. 139 (1996) 195-228

53. Chen, J.S., Pan, C., Wu, Roque, C.M.O.L., Wang, H.P., "A Lagrangian Reproducing Kernel Particle Method for Metal Forming Analysis," Comp. Mech. 22 (1998) 289-307

54. Brackbill, J.U., and Ruppel, H.M., “FLIP: A Method for Adaptively Zoned, Particle-in-Cell Calculations of Fluid Flows in Two Dimensions,” Journal of Computational Physics, 65 (1986) 314-343

55. Brackbill, J.U., Kothe, D.B., Ruppel, H.M., “FLIP: A Low-Dissipation, Particle-in-Cell Method for Fluid Flow,” Computer Physics Communications, 48 (1988) 25-38

56. Zhou, S.J., “The Numerical Prediction of Material Failure Based on the Material Point Method,” PhD thesis, Department of Mechanical Engineering, University of New Mexico, 1998.

57. Atluri, S.N., Shen, S. “The Meshless Local Petrov-Galerkin (MLPG) method: A Simple & Less Costly Alternative to the Finite Element and Boundary Element Methods,” Computer Modeling in Engineering & Sciences, 3 (2002) 11-52

58. Atluri, S.N., Shen, S.P. “The Meshless Local Petrov-Galerkin (MLPG) Method,” Tech.Science Press.(2002)

59. Atluri, S.N., Zhu, T. “A New Meshless Local Petrov-Galerkin (MLPG) Approach in Computational Mechanics,” Computational Mechanics, 22 (1998) 117-127

60. Libersky, L.D., Petschek, A.G., Carney, A.G., Hipp, T.C., and Allahdadi, F.A. “ High Strain Lagrangian Hydrodynamics – A Three Dimensional SPH Code for Dynamic Material Response,” Journal of Computer Physics 109 (1993) 67 – 75

61. Haines, E. “Graphics Gems IV: Point in Polygon Strategies,” AP Professional, Boston, (1994) 24-46

62. http://mathworld.wolfram.com/NewtonsMethod.html

63. Reddy, J.N. "An Introduction to the Finite Element Method," 2nd Edition, McGraw-Hill series in Mechanical Engineering (1993).

64. Wang, B., Karuppiah, V., Lu, H.B., Roy, S., Komanduri, R. "2D Mixed Mode Crack Fracture Simulation Using Material Point Method," for submission

65. Johnson, C. "Numerical solution of partial differential equations by the finite element method," Cambridge University Press.

66. Hornung, R. "An introduction to SAMRAI Framework: Part1," SAMRAI documentation; www.llnl.gov/CASC/SAMRAI

67. Zhong, Z.H. "Finite Element Procedures For Conatct-Impact Problems," Oxford University Press, New York (1993).

VITA

Rohit Kumar Raghav

Candidate for the Degree of

Master of Science

Thesis: THREE DIMENSIONAL GENERALIZED INTERPOLATION MATERIAL POINT (GIMP) SIMULATIONS IN SAMRAI ENVIRONMET

Major Field: Mechanical Engineering

Biographical:

Education: Received Bachelor of Engineering degree in Mechanical Engineering from Maharshi Dayanand University, Haryana, India in May 1999. Completed the requirements for the Master of Science degree with a major in Mechanical and Aerospace Engineering at Oklahoma State University in December, 2005.

Experience: Assistant Manager in Design department, Steel Strips Wheels Ltd., Chandigarh, India, January 2000 – December 2002

Graduate Research Assistant in Mechanical and Aerospace Engineering Department, Oklahoma State University, Stillwater, Oklahoma, August 2003 -present.

Professional Membership: ASME.

Name: Rohit Kumar Raghav Date of Degree: December, 2005

Institution: Oklahoma State University Location: Stillwater, Oklahoma

Title of Study: THREE DIMENSIONAL GENERALIZED INTERPOLATION MATERIAL POINT (GIMP) SIMULATIONS IN SAMRAI ENVIRONMET

Pages in Study: 91 **Candidate for the Degree of Master of Science**

Major Field: Mechanical Engineering

Scope and Methodology of Study: To resolve alternating stress sign and instability problems associated with conventional Material Point Method (MPM), the generalized interpolation material point method (GIMP) was recently introduced, and implemented for one dimensional simulation by Bardenhagen and Kober. For simulations spanning multiple length scales at continuum level, I present a parallel three dimensional GIMP computational method in the Structured Adaptive Mesh Refinement Application Infrastructure (SAMRAI developed by Lawrence Livermore National Laboratories) environment. 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. A GIMP algorithm for the treatment of contact between a rigid conical indenter and a deformable work piece has also been developed for 3D. As an example to validate the parallel GIMP computing scheme under SAMRAI parallel computing environment, numerical simulations were conducted for a 3D indentation problem and results are compared with finite element results on indentation.

Findings and Conclusions: A flexible domain decomposition technique has been developed for parallel computations. An efficient multi-level refinement technique has been developed and validated with results from FEM (ABAQUS). Expected speed-ups and memory savings has been achieved. GIMP Contact Algorithm has been developed for 3D indentation involving rigid surface indenter and results has been validated with results from FEM (ABAQUS). GIMP 3D code developed in this investigation in general works fine for large deformations but FEM quits early.

ADVISOR'S APPROVAL: Ranga Komanduri