Available online at www.sciencedirect.com
ScienceDirect
Comput. Methods Appl. Mech. Engrg. 289 (2015) 291–315
Computer methods in applied mechanics and engineering
www.elsevier.com/locate/cma

A mesh-grading material point method and its parallelization for problems with localized extreme deformation☆

Y.P. Lian[a,b], P.F. Yang[a], X. Zhang[a,*], F. Zhang[a], Y. Liu[a], P. Huang[c]

[a] AML, School of Aerospace, Tsinghua University, Beijing 100084, PR China
[b] Department of Mechanical Engineering, Northwestern University, Evanston IL, 60208, USA
[c] Institute of Structure Mechanics, China Academy of Engineering Physics, Mianyang, 621900, PR China

Received 29 November 2014; received in revised form 4 February 2015; accepted 16 February 2015
Available online 23 February 2015

Abstract

As a kind of meshless method, material point method (MPM) applies an Eulerian background grid served as a finite element mesh in each time step, and therefore its accuracy and efficiency are mainly dependent on the cell size setting of background grid. However, the conventional MPM commonly uses a regular background grid with uniform cells, which is not apposite for localized extreme deformation problems from the view point of computation efficiency, where, in fact, a local refined background grid is preferable. Hence, a mesh-grading material point method (MGMPM) is proposed here for such problems to supply MPM with the ability for local refinement simulation. The edge displacement continuity associated with mesh grading is embedded in the nodal shape functions. Besides, the truss element is incorporated into MGMPM to model the steel reinforcement bars in reinforced concrete impacting problems, based on our previous work. Furthermore, the proposed method is parallelized using OpenMP (Open Multi-Processing) to take advantage of PC power with multi-core and hyper threading technologies for large scale engineering problems, where both loop-level parallelism and code-block parallelism are used. Several numerical examples including stress wave propagation, Taylor bar impact, and penetration problems, are studied, which show that the efficiency of MGMPM is much higher than that of conventional MPM, and with lower memory requirement.

Keywords: Material point method; Mesh grading; Parallel; Reinforced concrete; Extreme deformation

1. Introduction

There are many kinds of engineering problems with localized extreme deformations, such as bird impact, penetration and local explosion problems, all of which involve extreme material deformation or failure. Compared with conventional numerical methods such as finite element method, meshless/meshfree methods have shown their advantages for such problems simulation. Up to now, many meshless methods have been proposed, such as the smoothed

☆ Supported by the National Basic Research Program of China (2010CB832701), National Natural Science Foundation of China (11272180, 11102195), and China Postdoctoral Science Foundation (International Postdoctoral Exchange Fellowship Program).
* Corresponding author.
E-mail address: xzhang@tsinghua.edu.cn (X. Zhang).

http://dx.doi.org/10.1016/j.cma.2015.02.020
0045-7825/

particle hydrodynamics (SPH) method [1–3], the element free Galerkin (EFG) method [4,5], the reproducing kernel particle method (RKPM) [6,7], material point method (MPM) [8,9], and the recently developed bond/stated-based peridynamics [10–12], just to name a few.

This paper is focused on material point method, which is an extension of the particle in cell method to solid mechanics problems. The basic idea of MPM is to take a collection of Lagrangian material points (particles) to discretize a given material domain, and to apply an Eulerian background grid to cover the material domain. All the state variables associated with the material, such as displacements, velocities, stresses and others material properties, are assigned to and updated on the particles, while the background grid is used for integrating momentum equations and calculating spacial derivatives, but carries no permanent information when the particles move through it. The particle interval must match the cell size of the background grid to guarantee the accuracy of particles quadrature and to avoid numerical fracture. Usually the particle interval is set as half of the cell size. Except for a few implicit approaches, MPM typically uses an explicit time integration scheme for solid mechanics problems. In contrast to some other meshless methods, MPM is less complex and with better performance on efficiency and tension stability [13]. Due to its advantages, MPM has been attracting more and more attentions and applied to a large range of science and engineering problems, such as impact/penetration, explosion problems [14,15], crack expending problems [16,17], biomechanics problems [18,19], fluid–structure interaction problems [20,21], multi-scale problems [22,23], geotechnical engineering problems [24,25], and so on. In order to suppress the artificial noise when the particles move across the cell boundary, different methods, such as generalized interpolation material point (GIMP) method [9], convected particle domain interpolation technique [26], dual domain material point method [27], were developed. Although MPM can handle no-slip contact without additional treatment, much work has been done [28–30] on the contact/friction/separation algorithm based on the Lagrangian multiplier method. In order to reduce oscillations of the contact force, a penalty function is proposed [25]. Besides, different work has been performed by Zhang's group to couple MPM with FEM [31,32] or with finite difference method (FDM) [33] to take advantage of the strengths of each method.

Although there are many kinds of applications mentioned above, the computation efficiency of conventional MPM can be further improved for large-scale engineering applications. The regular background grid with uniform cells is commonly applied in conventional MPM. Such a type of background grid is not flexible for problems with localized phenomena, where, in fact, a local refinement mesh is needed. For this issue, Lian and Zhang [34] proposed a tied interface grid material point method (TIGMPM), in which the background grid is composed of several individual regular grids with different cell sizes for different sub material domains. Each sub grid has an individual list of grid nodes. The interaction between two adjacent sub grids is implemented by a tied interface method, and then additional steps must be given to implement it based on the original steps of MPM.

Here an alternative method, the mesh-grading material point method (MGMPM), is propounded for MPM still to achieve dramatic computational saving. In contrast to TIGMPM, the background grid constructed by MGMPM has one list of grid nodes instead of several lists. In the view point of geometry, the background grid is divided into several sub grids level by level with half cell size decreasing. In order to build the topology between two adjacent grids with different cell sizes, add additional nodes to the corresponding cells, and then modify the shape functions of the cells with additional nodes to take account of the influence of the refined cells. By this method, the background grid can be refined locally, and therefore the total number of grid cells and particles can be reduced significantly. Considering that the edge displacement continuity associated with mesh grading is embedded in the nodal shape functions, there is no difference between the implementation of MGMPM and that of MPM except for different shape functions applied for different cell nodes. Compared with TIGMPM, this salient characteristic of MGMPM makes it easy to be parallelized based on the precursor work [35,36] which parallelized conventional MPM using OpenMP.

Although the dramatic computational savings can be obtained by MGMPM, development of its parallel algorithm is still desirable. Compared with the MPI, OpenMP is developed for shared memory machines, such as PC with multi-core and hyper threading technologies, and therefore may requires smaller modifications of the original serial code. To fully exert the performance of the common PC available nowadays, the proposed method is parallelized using OpenMP. There are two top difficulties in parallelizing MGMPM, load imbalance and data race. Huang and Zhang [35] proposed two methods to parallelize MPM for the first time. One is the array expansion method. In order to prevent data race, an accessory array for each thread is created; in the end of parallel region, these arrays are added together to form the global array used by the background grid. The other is domain decomposition method. The background grid is decomposed into some uniform patches, and the information of neighbor patches is exchanged through shared variables. After updating nodes in all patches, their nodal variables are assembled to establish the global variables.

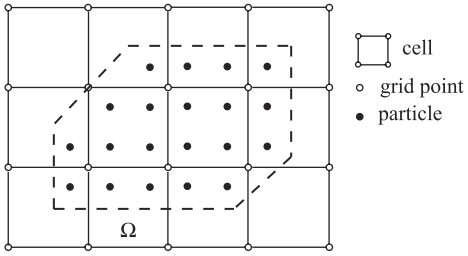

Fig. 1. Discretization scheme of MPM.

The first method is easy to implement without load imbalance, but with significant extra memory consuming when threads increasing. The second method does not consume extra memory but needs a significant modification to the original serial code. Based on the second method, Zhang [36] proposed another domain decomposition algorithm, which needs only small modification to the original code and is much easier to achieve dynamic load balance. Here, we extended this method in MGMPM for arbitrary number of bodies.

In order to simulate penetration problems involving reinforced concrete (RC) by MGMPM, the hybrid finite element–material point (HFEMP) method [37] is applied to enable MGMPM with ability to simulate the steel reinforcement bars (rebar) in RC. In HFEMP, the rebars are discretized by both particles and rebar truss elements, considering the tensile or compressive loading. The particles, serving as the nodes of the truss elements, only carry mass position and velocity variables, whereas truss elements carry material properties. This method is also parallelized using OpenMP.

The proposed method is first validated by a stress wave propagation problem and a Taylor bar impact problem, and then is applied to study the penetration of reinforcement concrete slabs and aluminum targets to show the accuracy and efficiency of MGMPM and its parallel algorithm. The remaining part of this article is organized as follows. In the next section, we provide a brief review of MPM and HFEMP. In Section 3, the mesh-grading background grid algorithm is given, including its numerical implementation. Section 4 gives the parallel MGMPM using OpenMP. The numerical examples are presented in Section 5, and some conclusions are drawn in Section 6.

2. Brief review of MPM and HFEMP

Both MPM and HFEMP are introduced briefly here for completeness. More detailed descriptions of MPM and HFEMP are available in the literature [8,30,37].

In MPM, the material domain Ω is discretized by a set of particles and covered by an Eulerian background grid, as shown in Fig. 1. In each time step, in order to calculate momentum equations on background grid, the mass and momentum of the particles are mapped to grid nodes around them by shape functions, separately

$$m_I = \sum_{p=1}^{n_p} m_p N_{Ip} \tag{1}$$

$$p_{iI} = \sum_{p=1}^{n_p} m_p v_{ip} N_{Ip} \tag{2}$$

where the subscript I denotes the variable of grid node, n_p is the total number of particles, m_p and v_{ip} are the mass and velocities of particle p. N_{Ip} is shape function associated with the grid node I evaluated at the site of particle p. Then the nodal velocity of background grid can be obtained by

$$v_{iI} = \frac{p_{iI}}{m_I}. \tag{3}$$

With leap frog central difference algorithm, the momentum of grid node can be updated by

$$p_{iI}^{k+1/2} = p_{iI}^{k-1/2} + (f_{iI}^{k,\text{ext}} + f_{iI}^{k,\text{int}})\Delta t^k \tag{4}$$

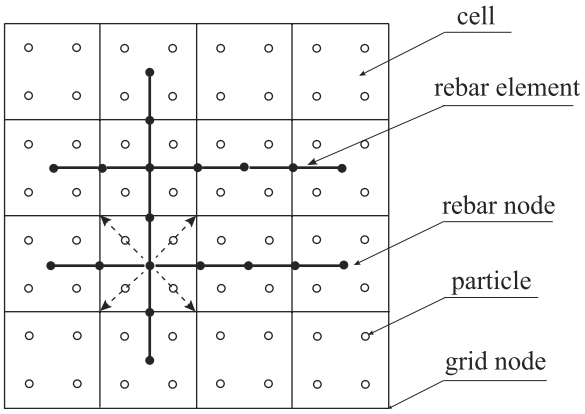

Fig. 2. RC discretization in HFEMP method: hollow dots denote concrete material points, while solid dots denote rebar nodes and solid lines connecting solid dots denote rebar elements.

where, the superscript k denotes the value of variable at time t^k, the internal nodal force and external nodal force are calculated, respectively, as follows.

$$f_{iI}^{k,\text{int}} = -\sum_{p=1}^{n_p} N_{Ip,j}\sigma_{ijp}^k \frac{m_p}{\rho_p} \tag{5}$$

$$f_{iI}^{k,\text{ext}} = \sum_{p=1}^{n_p} m_p N_{Ip} b_{ip}^k \tag{6}$$

where $b_{ip}^k = b_i^k(\boldsymbol{x}_p)$ is the body force per unit mass, $\sigma_{ijp}^k = \sigma_{ij}^k(\boldsymbol{x}_p)$ is the Cauchy stress of particle p.

In MPM, constitutive calculation is carried on the particles, so σ_{ijp}^k is updated by the corresponding constitutive law, where the strain rate and spin tensor are used and calculated from the velocity field of background grid, respectively.

$$\dot{\varepsilon}_{ijp} = \frac{1}{2}\sum_{I=1}^{n_g}(N_{Ip,j}v_{iI} + N_{Ip,i}v_{jI}) \tag{7}$$

$$\Omega_{ijp} = \frac{1}{2}\sum_{I=1}^{n_g}(N_{Ip,j}v_{iI} - N_{Ip,i}v_{jI}) \tag{8}$$

where n_g denotes the total number of background grid nodes which encompassed the particle within a certain distance.

After integrating the momentum equations on the nodes of background grid, map the results back to update particles' positions and velocities. Therefore, the positions x_{ip}^{k+1} and velocities $v_{ip}^{k+1/2}$ of particle p are obtained, respectively, by

$$x_{ip}^{k+1} = x_{ip}^k + \Delta t^{k+1/2}\sum_{I=1}^{n_g} v_{iI}^{k+1/2} N_{Ip}^k \tag{9}$$

$$v_{ip}^{k+1/2} = v_{ip}^{k-1/2} + \Delta t^k \sum_{I=1}^{n_g} a_{iI}^k N_{Ip}^k \tag{10}$$

where $v_{iI}^{k+1/2} = p_{iI}^{k+1/2}/m_I^k$, and $a_{iI}^k = f_{iI}^k/m_I^k$. After that, all the variables assigned to the grid nodes are reset to zero, which implies that no permanent information is stored on the grid nodes. Then, a new regular background grid is used in the next time step.

HFEMP is a method, which can enable MPM with ability to simulate problems involving reinforce concrete (RC) material. Considering that the main function of the steel reinforced bars in RC is to carry tensile loading, HFEMP [37] discretizes the reinforced bars in RC by rebar elements as shown in Fig. 2. All the rebar element nodes (referred to as

rebar nodes hereafter) are treated as particles for momentum equations updating. Then the rebar nodes move through the background grid together with others particles in the same single-valued velocity field for modeling the interaction between rebars and concrete. However, the constitutive law and axial force are calculated on the Gauss point of rebar element, which are same to that of the truss element in FEM. Then nodal mass, momentum, and force of rebar nodes are mapped to background grid by shape functions as follows.

$$m_I = \sum_{r=1}^{n_r} m_r N_{Ir} \tag{11}$$

$$p_{iI} = \sum_{r=1}^{n_r} m_r v_{ir} N_{Ir} \tag{12}$$

$$f_{iI}^{\text{int}} = \sum_{r=1}^{n_r} N_{Ir} f_{ir}^{\text{int}} \tag{13}$$

$$f_{iI}^{\text{ext}} = \sum_{r=1}^{n_r} m_r N_{Ir} f_{ir}^{\text{ext}} \tag{14}$$

where n_r is the total number of rebar nodes, m_r and v_{ir} are the lumped mass and velocity of rebar node r, respectively. f_{ir}^{int} and f_{ir}^{ext} are the internal force and external force acting on rebar node r. Then the position and velocity of rebar nodes are updated by Eqs. (9) and (10), separately, as was done with the particles.

From above, one can find that the efficiency of conventional MPM can be improved by changing the setting of the background grid. In the current time step, the background grid is embedded in and deformed with the material domain for solving the spatial derivatives and momentum equations in the current time step. The background grid can serve as a finite element mesh. Due to this, the accuracy, efficiency and also memory requirement of MPM is mainly dependent on the cell size setting of the background grid that is used. In MPM, a regular grid with uniform square (for 2D)/cubic (for 3D) cells is usually adopted as the background grid. However, such background grid is not optimal for problems with localized extreme deformation which needs local refinement making the MPM time consuming. In this paper, a mesh-grading background grid is proposed for MPM to obtain dramatic computation and memory savings. The obtained method is referred to as mesh-grading material point method (MGMPM) hereafter.

3. Mesh-grading material point method

In MGMPM, the background grid is locally refined level by level, and the same to the particle interval setting. Considering the accuracy of MPM dependent on the cell size setting, the refined sub grid is used for the material domain undergoing extreme deformation, while the coarse sub grid for the material domain undergoing mild deformation. In order to build the topology between two adjacent grids with different cell sizes, additional nodes are added to the coarse cells located at the interface of the two sub grids. Then new shape functions are constructed for those cells with additional nodes to keep the edge displacement continuity. The particle interval is still set as half of the cell size of the background grid. The particles located in refined grid are referred to as refined particles, while the particles in coarse grid as coarse particles. The particle splitting technique [34] is applied for the particles that enter the refined sub grid from the adjacent coarse grid during the computation process.

3.1. Mesh-grading background grid

Take a 2D problem as an example to show the idea for the mesh-grading background grid. As shown in Fig. 3, there are three levels of grids. The relationship between cell size h_l of level l with that of the level 1 is $h_l = h_1/2^{l-1}$, where the h_1 is the cell size of the coarsest grid.

The topology between the two adjacent grids is built by adding additional nodes to the coarse cell [38]. Due to the mismatch of cell sizes, an additional node, mid-edge node, is added to the relatively coarser cells at the interface, which results in 5-node quadrilateral elements as shown in Fig. 3. For clearness, the mid-edge node is marked with hollow circle, which in fact is common to three cells, one coarse cell and two refined cells. It is clear that the additional node is the corner node for the refined cell (such as node 5 for element b and c), and the mid-edge node for the coarse

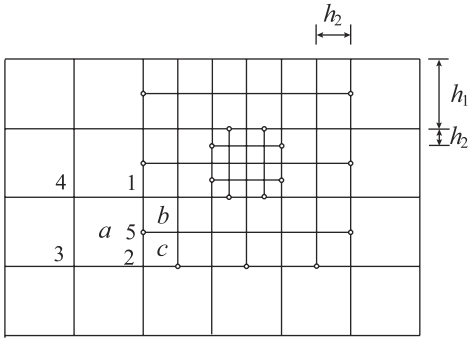

Fig. 3. Mesh-grading background grid.

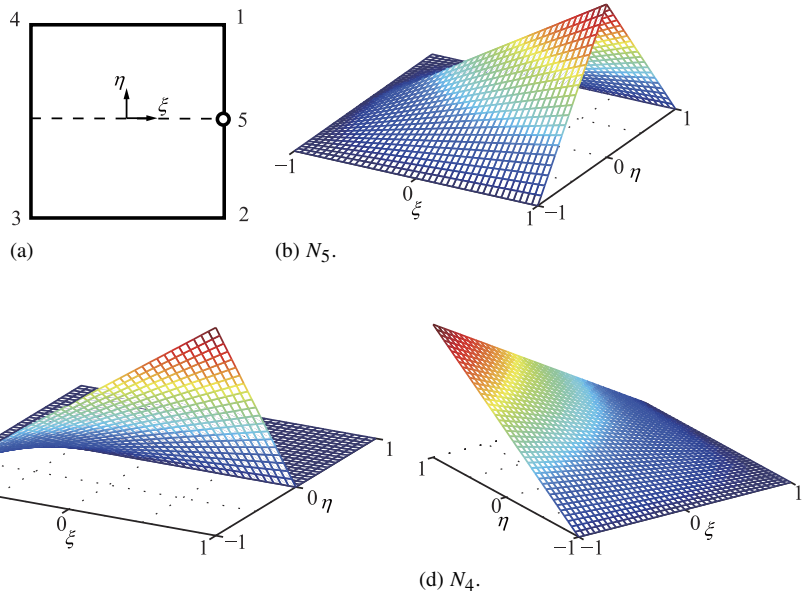

Fig. 4. Shape functions for 5-node linear quadrilateral element.

cells (such as node 5 for element a). The nodes 1, 5, and 2 are used to link elements a to b and c, and the new shape functions for the 5-node quadrilateral element is constructed to take the contribution of the mid-edge node 5. Then the shape functions of the element with additional nodes are important for the mesh-grading background grid.

For 2D problem, there are two kinds of elements, 4-node and 5-node quadrilateral element, respectively. Both of them are linear quadrilateral elements. For a 4-node linear quadrilateral element, the standard shape functions of isoparametric quadrilateral element are used as follows.

$$N_I^S(\xi,\eta) = \frac{1}{4}(1+\xi\xi_I)(1+\eta\eta_I) \quad I = 1,2,\ldots,4 \tag{15}$$

where the superscript S denotes standard shape functions for 4-node linear quadrilateral element. ($\xi \in [-1,1]$, $\eta \in [-1,1]$) are the natural coordinates of particle p, ξ_I and η_I take on their nodal value of ($\pm 1, \pm 1$).

For the 5-node quadrilateral element, the shape functions are constructed based on Eq. (15) and the standard rules for isoparametric elements [38], and are introduced briefly as follows. Take cell a isolated from Fig. 3 as an example, as shown in Fig. 4(a). The mid-edge node 5 creates two smooth sub-domains within the cell via dashed line, and the

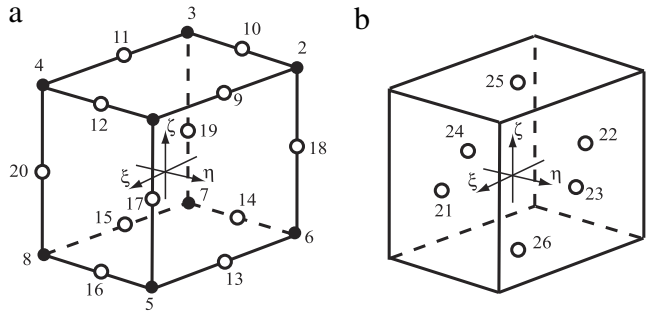

Fig. 5. Node numbering for the 26-node linear hexahedron element: (a) corner and optional mid-edge nodes; (b) optional mid-face nodes.

shape function associated with it is defined as:

$$N_5(\xi,\eta)=\frac{1}{2}(1+\xi)(1-|\eta|). \tag{16}$$

Within each sub-domain, the shape functions should be linear. Therefore N_1 and N_2 must be 0 at node 5. Considering that the value of the linear basis function N_1^S and N_5 is 1/2 and 1 at position of node 5, respectively, both N_1 and N_2 can be given as follows.

$$N_1(\xi,\eta)=N_1^S(\xi,\eta)-1/2N_5(\xi,\eta) \tag{17}$$

$$N_2(\xi,\eta)=N_2^S(\xi,\eta)-1/2N_5(\xi,\eta). \tag{18}$$

Fig. 4(b) shows the value distribution of N_5 over the domain of element a, and Fig. 4(c) for N_2. The mid-edge node will only affect the shape functions of nearby corner nodes 1 and 2. The shape functions of node 3 or 4 are still standard shape functions as in Eq. (15), see Fig. 4(c). By doing this, the edge displacement continuity associated with mesh grading is embedded naturally in the nodal shape functions.

For 3D problem, there are 8- to 26-node linear hexahedron elements in the mesh-grading background grid. Optional nodes could be mid-edge, mid-face nodes or both used in any desired combination depending on the mesh-grading cases. Therefore, the total number of nodes for an element may be up to 26, which is composed of 8 mandatory nodes at the 8 corners of the cube, 12 optional mid-edge nodes at the mid of 12 edges of the cube, and 6 optional mid-face nodes at the center of 6 faces of the cube, as shown in Fig. 5. For the regular 8-node linear hexahedron, the standard shape functions of isoparametric hexahedron element are used as follows.

$$N_I^S=\frac{1}{8}(1+\xi\xi_I)(1+\eta\eta_I)(1+\zeta\zeta_I)\quad I=1,2,\ldots,8. \tag{19}$$

Similar to the 5-node quadrilateral element mentioned above, it is straightforward to construct the shape functions of the 9- to 26-node linear hexahedron elements. The basis functions for the mid-face nodes and mid-edge nodes, as well as their adjacent corner nodes, are creased at the junction of the sub-domains created by them. Take an element with 2 optional nodes in the position of nodes 10 and 25 as an example to derive the new shape functions. Assuming the basis functions are linear within each sub-domain, N_{25} must take value 1 at node 25, and 0 at others nodes. Therefore, N_{25} is given as

$$N_{25}=1/2(1-|\xi|)(1-|\eta|)(1+|\zeta|). \tag{20}$$

Due to the presence of node 25, N_{10} is given as

$$N_{10}=1/4(1-\xi)(1-|\eta|)(1+|\zeta|)-1/2P_{25}. \tag{21}$$

Due to the presence of node 25 and node 10, N_3 can be obtained by modifying N_3^S to take account the influence of N_{25} and N_{10}, namely

$$N_3=1/8(1-\xi)(1-\eta)(1+\zeta)-1/2P_{10}-1/4P_{25}. \tag{22}$$

Table 1
Shape functions for the 3D grading element.

Node	$N(\xi, \eta, \zeta)$
26	$1/2(1-\lvert\xi\rvert)(1-\lvert\eta\rvert)(1-\zeta)$
25	$1/2(1-\lvert\xi\rvert)(1-\lvert\eta\rvert)(1+\zeta)$
24	$1/2(1-\lvert\xi\rvert)(1-\eta)(1-\lvert\zeta\rvert)$
23	$1/2(1-\lvert\xi\rvert)(1+\eta)(1-\lvert\zeta\rvert)$
22	$1/2(1-\xi)(1-\lvert\eta\rvert)(1-\lvert\zeta\rvert)$
21	$1/2(1+\xi)(1-\lvert\eta\rvert)(1-\lvert\zeta\rvert)$
20	$1/4(1+\xi)(1-\eta)(1-\lvert\zeta\rvert)-1/2(N_{21}+N_{24})$
19	$1/4(1-\xi)(1-\eta)(1-\lvert\zeta\rvert)-1/2(N_{22}+N_{24})$
18	$1/4(1-\xi)(1+\eta)(1-\lvert\zeta\rvert)-1/2(N_{22}+N_{23})$
17	$1/4(1+\xi)(1+\eta)(1-\lvert\zeta\rvert)-1/2(N_{21}+N_{23})$
16	$1/4(1+\xi)(1-\lvert\eta\rvert)(1-\zeta)-1/2(N_{21}+N_{26})$
15	$1/4(1-\lvert\xi\rvert)(1-\eta)(1-\zeta)-1/2(N_{24}+N_{26})$
14	$1/4(1-\xi)(1-\lvert\eta\rvert)(1-\zeta)-1/2(N_{22}+N_{26})$
13	$1/4(1-\lvert\xi\rvert)(1+\eta)(1-\zeta)-1/2(N_{23}+N_{26})$
12	$1/4(1+\xi)(1-\lvert\eta\rvert)(1+\zeta)-1/2(N_{21}+N_{25})$
11	$1/4(1-\lvert\xi\rvert)(1-\eta)(1+\zeta)-1/2(N_{24}+N_{25})$
10	$1/4(1-\xi)(1-\lvert\eta\rvert)(1+\zeta)-1/2(N_{22}+N_{25})$
9	$1/4(1-\lvert\xi\rvert)(1+\eta)(1+\zeta)-1/2(N_{23}+N_{25})$
8	$1/8(1+\xi)(1-\eta)(1-\zeta)-1/2(N_{15}+N_{16}+N_{20})-1/4(N_{21}+N_{24}+N_{26})$
7	$1/8(1-\xi)(1-\eta)(1-\zeta)-1/2(N_{14}+N_{15}+N_{19})-1/4(N_{22}+N_{24}+N_{26})$
6	$1/8(1-\xi)(1+\eta)(1-\zeta)-1/2(N_{13}+N_{14}+N_{18})-1/4(N_{22}+N_{23}+N_{26})$
5	$1/8(1+\xi)(1+\eta)(1-\zeta)-1/2(N_{13}+N_{16}+N_{17})-1/4(N_{21}+N_{23}+N_{26})$
4	$1/8(1+\xi)(1-\eta)(1+\zeta)-1/2(N_{11}+N_{12}+N_{20})-1/4(N_{21}+N_{24}+N_{25})$
3	$1/8(1-\xi)(1-\eta)(1+\zeta)-1/2(N_{10}+N_{11}+N_{19})-1/4(N_{22}+N_{24}+N_{25})$
2	$1/8(1-\xi)(1+\eta)(1+\zeta)-1/2(N_{9}+N_{10}+N_{18})-1/4(N_{22}+N_{23}+N_{25})$
1	$1/8(1+\xi)(1+\eta)(1+\zeta)-1/2(N_{9}+N_{12}+N_{17})-1/4(N_{21}+N_{23}+N_{25})$

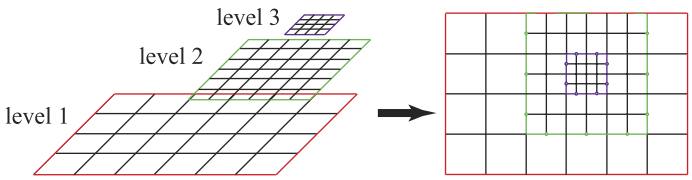

Fig. 6. Generating process of mesh-grading background grid.

From above, one can find out that it is convenient to evaluate the basis functions in decreasing order, from P_{26} to P_1. Table 1 [38] lists all the shape functions for hexahedron element with 26 nodes. If one optional node is absent, its basis function and all subsequent references to it should be removed from Table 1.

3.2. Generate mesh-grading background grid and particles

As shown in Fig. 6, the mesh-grading background grid is generated level by level in increasing order. At first, one generates the grid of level 1, which is the largest grid covering the whole domain and also including the domain covered by other sub grids. Then one generates the grid of level 2. The cells of level 1 grid that are covered by level 2 grid are replaced by the new cells of level 2 grid, but their nodes are reserved as the nodes of the new cells. Meanwhile, optional nodes will be added to the cells of level 1 grid at the interface between the two grids, and the corresponding cells will be labeled so that new shape functions are used in the calculation. Similarly, one can generate the remaining level grids. During this generation, a single list of nodes is maintained with all sub grids grouped together. Due to this reason, the background grid is grouped as a whole instead of composed by several individual grids. However, a list of all sub grids is also formed. Using the latter list, it is easy to find a cell in which a particle is located, level by level in decreasing grid order. It is straightforward to generate a 3D background grid in the same way.

In MGMPM, the particle interval should match the cell size of the sub grid, in which the particle is located as shown in Fig. 7. The interaction and connection between particles are implemented by the background grid. If there is

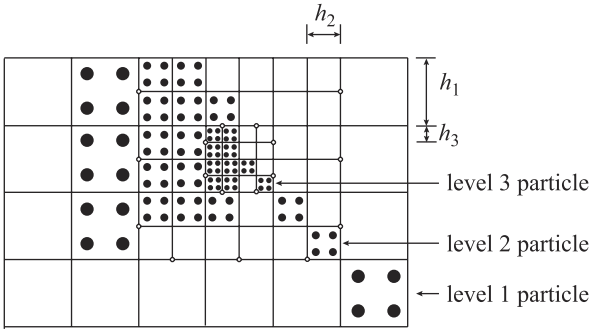

Fig. 7. Particles distribution in mesh-grading background grid.

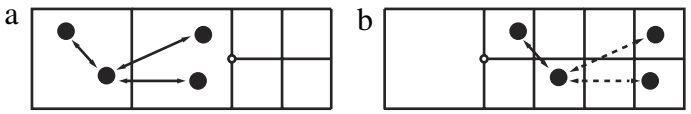

Fig. 8. Numerical fracture due to particle moving: (a) particles position at initial time, (b) particles position after moving.

an empty cell or more between two particles, they will not have influence on each other, see Fig. 8(b) as an example. Assuming a material discretized by 4 particles, there is an empty cell between them, so there is no connection between them, which is referred to as numerical fracture hereafter. Based on this point, the particle interval must match the cell size of the sub grid to avoid numerical fracture. In common, it is often set as half of the cell size of the grid at the initial discretization. With such setting, the relationship of particle intervals dp between level l and 1 is $dp_l = dp_1/2^{l-1}$.

3.3. Particle splitting scheme

Particle splitting scheme is needed. At the initial discretization, there are several levels of particle intervals that match the corresponding cell sizes. However, during the calculation process, the particles could move relatively to the background grid, while the background grid is fixed in space. So numerical fracture may still occur when there is a mismatch between particles and cells, such as coarse particles moving from lower level grid to higher level grid, as shown in Fig. 8. For this case, the coarse particle should be split to refined particles, so that the empty cells can be filled out by the new refined particles, as shown in Fig. 9.

The particle splitting method proposed by Lian and Zhang et al. [34] is applied here. Since the cell size ratio between two adjacent grids is 2, a coarse particle is split into four particles for 2D or eight particles for 3D problems. Take a 2D problem as an example. The mass, volume, and internal energy of the coarse particle are distributed to the four new particles evenly, while the stress, strain and other material variables of the new particles are set to those of the coarse particle. Assuming that the configuration domain of each particle is a cuboid, the new particles are uniformly distributed in the material domain occupied by the coarse particle, as shown in Fig. 9. In the current configuration, the length of the coarse particle domain can be simply obtained by its strain as follows,

$$L_i = L_0(1 + \varepsilon_i) \tag{23}$$

where $L_0 = \sqrt[3]{m/\rho}$ is the particle initial length for 3D problem, and ε_i indicates the accumulated strain of the particle in ith direction.

There is no need to split the rebar element nodes due to their connection based on the element linking instead of background grid.

3.4. Numerical implementation

The detailed implementation of MGMPM is presented here using the USF scheme [39] for one time step in the absence of contact.

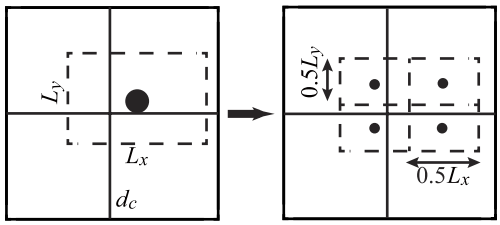

Fig. 9. Split one coarse particle to 4 refined particle.

1. Loop over all bodies to map the mass and momentum of particles and rebar nodes to the background grid via the shape functions as follows,

$$m_I^k = \sum_{p=1}^{n_p} m_p N_{Ip}^k \tag{24}$$

$$p_{iI}^k = \sum_{p=1}^{n_p} m_p v_{ip}^k N_{Ip}^k \tag{25}$$

where the formulation of N_{Ip}^k depends on the type of grid nodes, as shown in Table 1.

2. Loop over all the nodes located at the boundary of the background grid to apply boundary conditions.
3. Loop over the bodies discretized by particles, calculate the strain rate and spin tensor of particles by the velocity field of the background grid as follows,

$$\dot{\varepsilon}_{ijp}^{k-1/2} = \frac{1}{2}\sum_{I=1}^{n_g}[N_{Ip,j}^k v_{iI}^{k-1/2} + N_{Ip,i}^k v_{jI}^{k-1/2}] \tag{26}$$

$$\Omega_{ijp}^{k-1/2} = \frac{1}{2}\sum_{I=1}^{n_g}[N_{Ip,j}^k v_{iI}^{k-1/2} - N_{Ip,i}^k v_{jI}^{k-1/2}] \tag{27}$$

where the superscript n_g denotes the total number of nodes of the cell in which the particle is located. Update particle stresses by a specified constitutive material law.

4. Loop over the bodies discretized by rebar elements, calculate the strain rate of rebar elements, then update their stresses by a constitutive material law, and finally calculate the nodal forces.
5. Loop over all bodies to calculate the nodal force of background grid:
   (a) For bodies discretized by particles, loop over all particles to calculate the internal nodal force and external nodal forces, respectively,

$$f_{iI}^{k,\text{ext}} = \sum_{p=1}^{n_p} m_p N_{Ip}^k b_{ip}^k \tag{28}$$

$$f_{iI}^{k,\text{int}} = -\sum_{p=1}^{n_p} N_{Ip,j}^k \sigma_{ijp} \frac{m_p}{\rho_p}. \tag{29}$$

   (b) For bodies discretized by rebar elements, loop over all rebar element nodes to calculate nodal force of background grid nodes by Eqs. (13) and (14).
6. Loop over all background grid nodes to integrate the momentum equations as

$$p_{iI}^{k+1/2} = p_{iI}^{k-1/2} + (f_{iI}^{k,\text{ext}} + f_{iI}^{k,\text{int}})\Delta t^k \tag{30}$$

where both $f_{iI}^{k,\text{ext}}$ and $f_{iI}^{k,\text{int}}$ take the value 0, if the grid node is fixed in i direction.

7. Loop over all bodies to update the position and velocity of particles and rebar nodes by Eqs. (9) and (10), respectively.
8. Loop over the bodies discretized by particles to split the coarse particles that moved to refined grid using the method presented in Section 3.3.

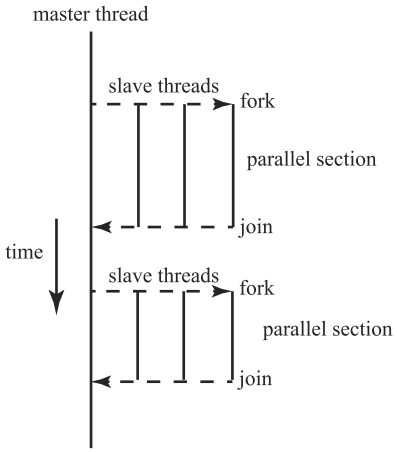

Fig. 10. Fork-join model: at the beginning of a parallel section, the master thread forks some slave threads and runs with them concurrently across the parallel section; at the end of the section, all the threads join together and only the master thread continues to run.

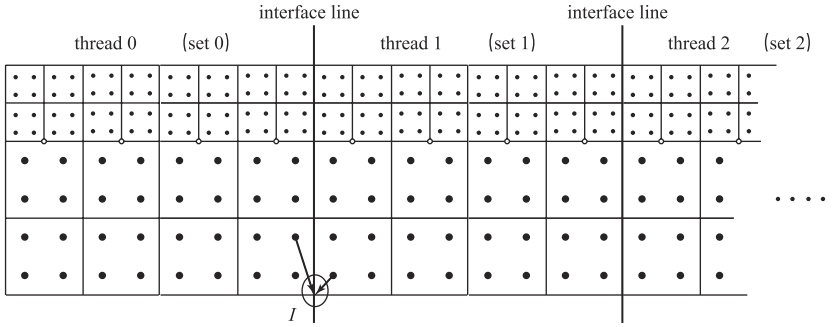

Fig. 11. Decomposition of the computing domain.

4. Parallelizing MGMPM using OpenMP

To expedite the calculation process for large scale engineering problems based on PC with multi-core and hyper threading technologies, the proposed method is parallelized using OpenMP. As shown in Fig. 10, the model of parallelism in OpenMP is a fork-join model, by adding compiler directive to the original code to state a parallel section. There are two common styles of programming, loop-level parallelism and code-block parallelism, respectively. The former parallelism is suitable for loop iterations without data dependence, and is easy to implement without load balance issue. The latter is suitable for loop iterations with data dependence, and needs the user to take charge for both the load balance and data race issues. As listed in Section 3.4, there are 8 computational steps for one time step in MGMPM. Both parallelisms are used for different steps, mainly ascribed to their data dependence.

4.1. Code-block parallelism for steps 1, 4, and 5

The code-block parallelism is applied for steps 1, 4, and 5 due to the data dependence. For steps 1 and 5, different particle or rebar node loops may operate the same grid node at the same time, while for step 4, different rebar element loops may operate simultaneously the same rebar node. Then code-block parallelism is used. Take a 2D problem as an example. As shown in Fig. 11, the computing domain is first divided into several sub-domains in horizontal direction, and the particles in each sub-domain is denoted by a set. Let each thread take one set of particles in each sub-domain. Even in this case, data race will still occur when map particles and rebar nodes information to the same grid nodes, such as grid node I. Zhang [36] has proposed one simple method to solve such data race issue for MPM. Here, this method is applied in MGMPM and extended to more than one particle lists, which is a more flexible data structure.

To avoid data race occurring, additional decomposition for each sub-domain (set) is done. Take two threads as an example. In each sub-domain, the set of particles is further divided into two parts, referred to as L part and R part

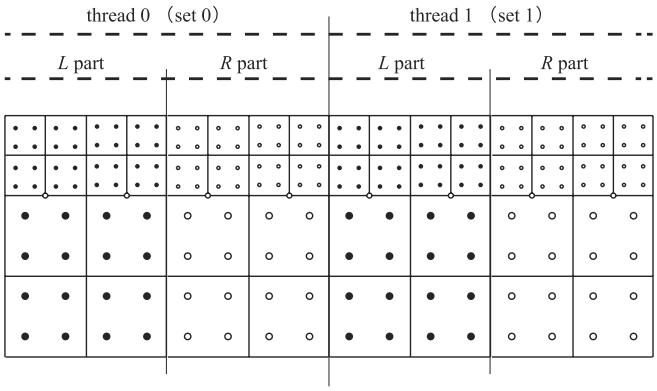

Fig. 12. Decomposition of the computing domain: solid dots denote particles in L part, and hollow dots denote particles in R part.

denoted by solid dots and hollow dots, respectively, as shown in Fig. 12. We can let all threads update its L part first, and then update its R part, namely, the L and R parts are updated alternately. All L parts of set 0 and set 1 do not overlap each other, and the same to the R parts of set 0 and set 1. Therefore, there is no overlap between two adjacent sets for grid nodes information updated at the same time, and no data races emerge.

In our code, a body list is used to manage the whole material domain, and each body object has a particle or rebar node list for the current body discretization. Therefore, in the body class we add two pointer arrays, *LPindex* and *RPindex*, to store the particles or rebar nodes ID for each set. One is for the L part, the other for the R part. A compiler directive, *#pragma omp barrier*, is added between the L part and R part update. Before the barrier, loop over all bodies and let all threads update their L part in the current body in parallel. After the barrier, loop over all bodies again and let all threads update their R part in the current body in parallel. Besides, each thread will loop over its own particles instead of all the particles. The pseudo code for making the index array is given as follows.

```
for(int b=0; b<nb_body; b++)
{
   #pragma omp parallel num_threads(nthread)
   {
      int ithread=omp_get_thread_num();
      int m=0,n=0;
      for(int i=0; i<nb_point; i++)
      {
         Particle = body_list[b]->particle_list[i];
         if(Particle belongs to L part of ithread)
            body_list[b]->LPindex[ithread][m++] = i;
         elseif(Particle belongs to R part of ithread)
            body_list[b]->RPindex[ithread][n++] = i;
       }
   }
}
```

Based on the index arrays made above, steps 1 and 5 can be parallelized with the same method. Here, the pseudo code for step 1 is given as follows.

```
#pragma omp parallel num_threads(nthread)
{
   int ithread=omp_get_thread_num();
   for(int b=0; b<nb_body; b++)
   {
      nb_particle = <The number of particles belong to L part of
```

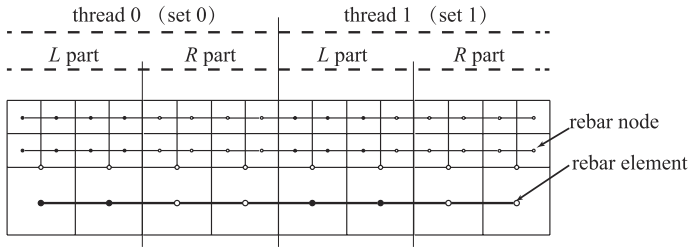

Fig. 13. Decomposition of the computing domain: solid dots denote rebar nodes in L part, and hollow dots denote rebar nodes in R part.

```
                    ithread for the current body>
   for(int i=0; i<nb_particle; i++)
   {
      j = body_list[b]->LPindex[ithread][i];
      Particle=body->particle_list[j];
      < Map variables of particle to background grid nodes>
   }
}
#pragma omp barrier
for(int b=0; b<nb_body; b++)
{
   nb_particle = <The number of particles belong to L group of
                  ithread for the current body>
   for(int i=0; i<nb_particle; i++)
   {
      j = body_list[b]->RPindex[ithread][i];
      Particle=body->particle_list[j];
      < Map variables of particle to background grid nodes>
   }
}
```

As shown in Fig. 13, the same decomposition method is applied for bodies discretized by rebar elements. Also the same index arrays for rebar nodes and rebar elements of the body are built by the method mentioned above. Then the index arrays *LPindex* and *RPindex* for rebar nodes are used similarly to that of particles in steps 1 and 5, while the index arrays *LEindex* and *REindex* are used for rebar elements in step 4. Because the coordinate is used for the set and part division, let rebar element center position represent it when build the index arrays for rebar elements. Here, the pseudo code for step 4 is given as follows.

```
#pragma omp parallel num_threads(nthread)
{
   int ithread=omp_get_thread_num();
   for(int b=0; b<nb_body; b++)
   {
      nb_element = <The number of rebar elements belong to L part
                    of ithread for the current body>
      for(int i=0; i<nb_element; i++)
      {
         j = body_list[b]->LEindex[ithread][i];
         <LEindex is an index of rebar elements ID
          in the left part of each sub-domain for the current body>
         Element=body->element_list[j];
         ......
```

```
         < Update element stress, based on which calculate
           nodal force of rebar nodes>
      }
   }
   #pragma omp barrier
   for(int b=0; b<nb_body; b++)
   {
      nb_element = <The number of rebar elements belong to R part
                    of ithread for the current body>
      for(int i=0; i<nb_element; i++)
      {
         j = body_list[b]->REindex[ithread][i];
         <REindex is an index of rebar elements ID
          in right part of each sub-domain for the current body>
         Element=body->element_list[j];
         ......
         < Update element stress, based on which calculate
           nodal force of rebar nodes>
      }
   }
```

Another issue associated with code-block parallelism is load balance. If one aims for high efficiency the load assigned to each thread should be equal, otherwise there will be load imbalance which can deteriorate the parallel program efficiency. The load assigned to each thread is dependent on the number of particles and rebar nodes of the set looped on by thread, and therefore the number of particles and rebar nodes in each thread should be kept balanced always. So the decomposition position of the computing domain is dependent on the distribution of the total number of particles and rebar nodes in each layer of the background grid, as shown in Figs. 12 and 13. After several time steps the distribution of particles and rebar nodes will be changed, therefore the computing domain needs to be decomposed dynamically and the index arrays for particles and rebar nodes need to be rebuilt at the same time. However, the decomposition of rebar elements is done only once at the initial time step based on the background grid, as shown in Fig. 13. This is because the rebar elements are in absence of relationship with grid nodes updating.

4.2. Loop-level parallelism for remaining steps

The loop-level parallelism is applied for steps 2, 3, 6 and 7. In step 2, the loop over all grid nodes is used to apply boundary conditions; in step 3, the loop over all particles is used to update their strains and stresses; in step 6, the loop over all grid nodes is used to integrate momentum equations; in step 7, the loop over all particles and rebar nodes is used to update their positions and velocities. It is clear that there is no data dependence in those steps. Therefore, the loop-level parallelism is applied for all these steps. Then, we can simply add compiler directive *#pragma omp for* before those *for* loops. Take steps 2 and 7 for an example. The pseudo code for step 2 is as follows:

```
#pragma omp parallel num_threads(nthread)
{
     #pragma omp for
     for(int i=0; i<nb_gridnode; i++)
     {
        CgridNode *cgd=node_list[i];
        if(!(cgd->FixX|| cgd->FixY || cgd->FixZ))
           continue;
        cgd->Apply_Boundary_Conditions();
     }
}
```

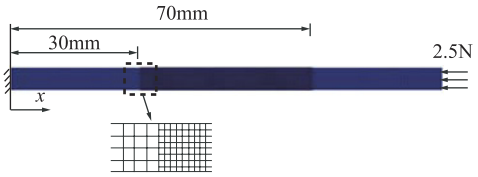

Fig. 14. Discretization model of elastic rod.

The pseudo code for step 7 is as follows:

```
// for particles
#pragma omp parallel num_threads(nthread)
{
     #pragma omp for
     for(int i=0; i<nb_particles/nodes; i++)
     {
        CParticle *p=particle_list[i];
        <Update particle's position and velocity>;
     }
}
// for rebar nodes
#pragma omp parallel num_threads(nthread)
{
     #pragma omp for
     for(int i=0; i<nb_nodes; i++)
     {
        CNode *p=node_list[i];
        <Update rebar nodal position and velocity>;
     }
}
```

In contrast to code-block parallelism, the load balance of the loop-level parallelism is guided automatically by OpenMP.

4.3. Others

In step 8, the particle splitting scheme will add particles to the end of the particle list, which will manipulate memory directly. In OpenMP standard, such manipulations are inherently serial and therefore this step cannot be parallelized. These might decrease the efficiency based on the Amdahl's law.

5. Numerical examples

5.1. Propagation of elastic wave

The propagation of elastic stress wave in an elastic rod is studied to validate the accuracy of MGMPM. As shown in Fig. 14, the rod is fixed at one end, and subjected to a force of 2.5N at the other end. The rod has a length of 100 mm and cross sectional area of 4 mm $\times$ 4 mm. An elastic material with Young's modulus $E = 100$ GPa, density $\rho = 5$ kg/m^3, and Poisson ratio $\nu = 0$ is applied. The sound speed is $c = \sqrt{E/\rho} = 4472.1$ m/s.

The discretization model is also shown in Fig. 14. A background grid with two levels is applied to cover the material domain, while the sub grid with the higher level is predefined at the center of the rod with a length of 40 mm, and the other sub grid covers the material domain elsewhere. Symmetric boundary conditions are applied on the four sides of the rod in length direction to mimic a 1D case. Then, the stress distribution can be obtained analytically from the 1D wave propagation theory. The cell size of the coarse sub grid is set as 0.5 mm, and the coarse particle interval as 0.25 mm. For comparison, this problem is also simulated by MPM with the refined cell size 0.25 mm.

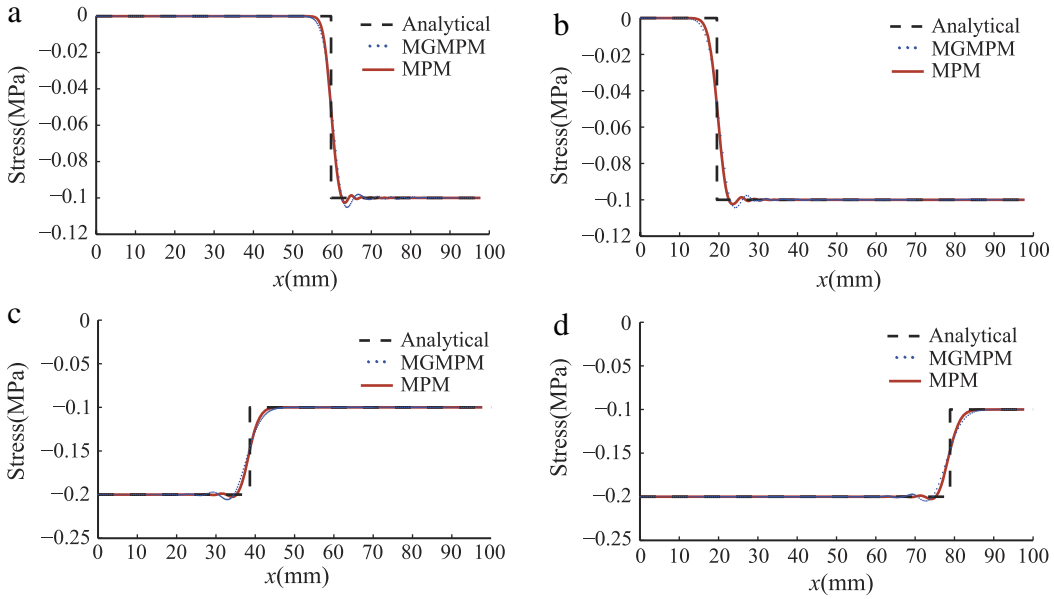

Fig. 15. Elastic rod stress profiles at times of: (a) $t = 0.009$ ms, (b) $t = 0.018$ ms, (c) $t = 0.031$ ms, (d) $t = 0.04$ ms.

Table 2
Material constants of the Taylor bar.

ρ (g/mm^3)	E (MPa)	ν	A (MPa)	B (MPa)	n	C
8.93×10^{-3}	117×10^3	0.35	157	425	1.0	0.0

The stress profiles at four different times obtained by both MGMPM and MPM are compared with the analytical results in Fig. 15. Fig. 15(a) compares the stress profiles at time of $t = 0.009$ ms, where the stress wave front has propagated into the grid of level 2 from the grid of level 1. Fig. 15(b) compares the stress profiles at time of $t = 0.018$ ms. At this time, the stress wave front has propagated into the grid of level 1 from the grid of level 2. Fig. 15(c) and (d) compare the stress profiles at time of $t = 0.031$ ms and $t = 0.04$ ms, respectively. At those times, the stress wave has been reflected from the fixed end forward to the other end, and therefore the magnitude of the stress wave is doubled. The results obtained by both MGMPM and MPM are close to each other and agree well with the analytical results. The small deviation is due to the coarse grid used in MGMPM, which also indicates that the accuracy of MPM is dependent on the cell size of the background grid.

Besides, another compression is given as follows. Two cases of MPM are given, fixing the cell size of MPM as 0.5 mm, and changing the particle interval as 0.25 mm (MPM1) and particle interval as 0.125 mm (MPM1), respectively. The stress profiles at four different times obtained by both MGMPM and such two cases of MPM are compared with the analytical results in Fig. 16. One can find that the results by MGMPM are better than those by both MPM1 and MPM2 when wave front propagates to coarse mesh from refined mesh, for example Fig. 16(b) and (d), while before wave front propagates to refined mesh from coarse mesh, there is no clear difference between the results for all cases, except that the result of MGMPM is marginally better than others in Fig. 16(c), where the wave front is reflected from the fixed side and heads to the other side.

Summarizing, the results obtained by MGMPM are very close to those by MPM with refined cell size. However, MGMPM takes 14 min for the simulation with end time 0.05 ms, but MPM takes 21 min.

5.2. Taylor bar impact

The second example is a typical Taylor bar test conducted by Johnson et al. [40], where the bar with an initial velocity of 190 m/s impacts on a rigid wall. In the test, the bar has a length of $L_0 = 25.4$ mm, a diameter of $D_0 = 7.6$ mm. Johnson–Cook model is applied for the bar and the material constants are listed in Table 2.

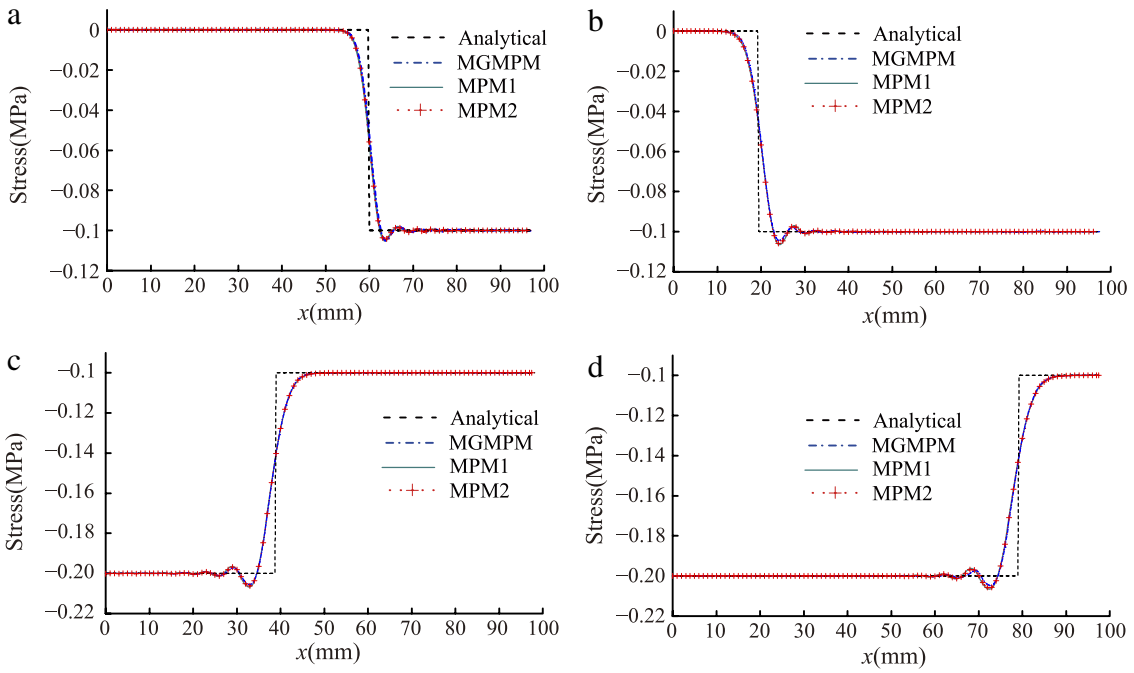

Fig. 16. Elastic rod stress profiles at times of: (a) $t = 0.009$ ms, (b) $t = 0.018$ ms, (c) $t = 0.031$ ms, (d) $t = 0.04$ ms.

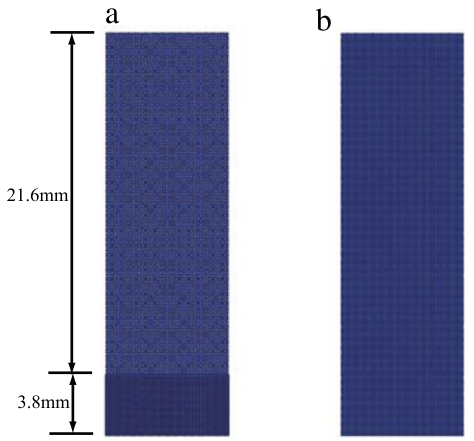

Fig. 17. Discretization models: (a) for MGMPM, (b) for MPM with cell size 0.19 mm.

As shown in Fig. 17, a background grid with two levels is applied, where the cell sizes of two levels grids are 0.19 mm and 0.38 mm, respectively. The space domain of the level 2 grid covers the part of length 3.8 mm from the bottom. For comparison, this problem is simulated by MPM with two cases of cell sizes of 0.19 mm (referred to as MPM1), and 0.38 mm (referred to as MPM2), respectively.

The final configurations of the bar obtained by both MGMPM and MPM are shown in Fig. 18 in color for the equivalent plastic strain. One can find that the color contour in the local domain near the interface obtained by MGMPM is not as smooth as that by MPM1. This is because, during the computation process, more and more coarse particles moving to the bottom refined grid are split and the splitting is a non-consecutive discretization compared with the initial discretization. Moreover, the difference of the color contour is confined in the domain occupied by the splitting particles. Overall, the total configuration shown in Fig. 18(a) is close to that in Fig. 18(b), although coarse grid is used in MGMPM. Besides, the bottom diameter D and the final length L of the deformed bar obtained by MGMPM and MPM are compared with the experimental data in Table 3. The variable D mainly measures the result obtained by

Table 3
The computational results and computation cost comparison.

	DCell (mm)/Case	L (mm)	D (mm)	n_p	CPU (s)
Experiment		16.2	13.5	–	–
MPM1	0.19	16.21	13.28	1 342 408	9488
MPM2	0.38	16.29	13.14	169 376	963
MGMPM	0.19 & 0.38	16.28	13.28	345 056∼565 656	5966

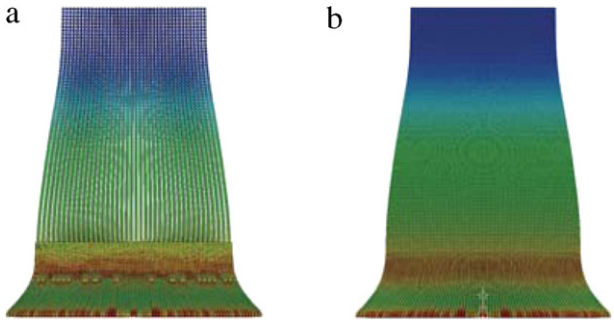

Fig. 18. Final configurations of Taylor bar obtained by: (a) MGMPM, and (b) MPM with cell size of 0.19 mm.

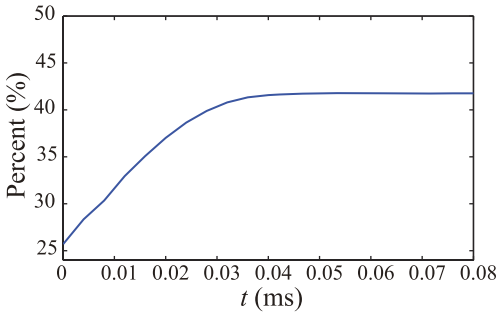

Fig. 19. Profile of ratio of the number of particles used in MGMPM to that in MPM1.

the refined grid, while the variable L measures the result obtained by the whole background grid. Therefore, the value of D obtained by MGMPM is very close to that by MPM1 and better than that by MPM2; the value of L obtained by MGMPM is between those by MPM1 and MPM2 but close to that of MPM2 due to the influence of the coarse grid.

Furthermore, the CPU time and the total number of particles n_p for all cases are also listed in Table 3. Due to the particle splitting, both the numbers of the particles used in MGMPM at the beginning and end times are given in Table 3. From comparison, n_p of MGMPM is no more than 45% of that of MPM1 throughout the computation process, while the cell numbers of MGMPM is about a quarter of MPM, and therefore the memory requirement by MGMPM is smaller than MPM1. However, the CPU time taken by MGMPM is just about 63% of that by MPM1. This is because there are more and more particles split as shown in Fig. 19 and this costs CPU time.

In other words, the computation cost of MGMPM is expensive with some improvement in accuracy compared with the results by MPM2. So, MGMPM should be used for problem with localized extreme deformation such as penetration problems.

5.3. Penetration of reinforced concrete slab

In this example, a penetration of a steel projectile to a reinforced concrete slab [41] is simulated. In the test, the projectiles (see Fig. 20) with several different initial velocities were fired against on RC slab with three layers of square pattern rebars (see Fig. 21). Besides, the targets were aligned so that the projectile did not strike the rebars.

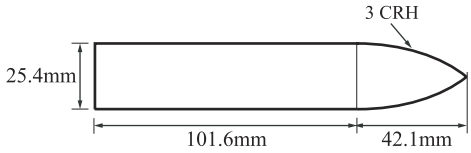

Fig. 20. Projectile geometry (0.5 kg).

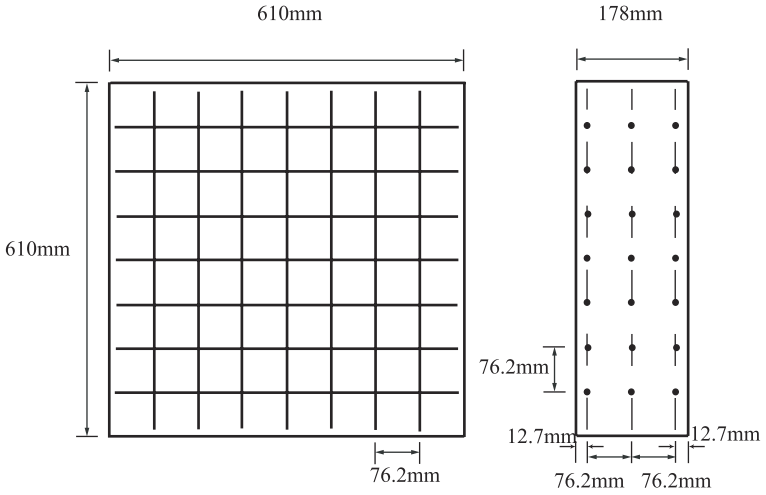

Fig. 21. RC geometry with location of the steel reinforcement bars (5.59 mm diameter).

Table 4
Material constants of the concrete.

Mass/Thermal Constants		Value	Damage Constants		Value
ρ	(kg/m^3)	2440	D_1		0.04
Specific Heat	(J/kg · K)	654	D_2		1.0
			$E_{f\,\min}$		0.01
Strength Constants		Value	Pressure Constants		Value
A		0.79	P_{crush}	(GPa)	0.016
B		1.60	μ_{crush}		0.001
N		0.61	K_1	(GPa)	85
C		0.007	K_2	(GPa)	-171
f_c'	(GPa)	0.048	K_3	(GPa)	208
$S_{\max}$		7.0	P_{lock}	(GPa)	0.80
Shear modulus	(GPa)	14.86	μ_{lock}		0.10
			T	(GPa)	0.004

Then the physical model is symmetric. The recovered projectile showed that there was no permanent deformation but minor nose erosion [41]. Then the projectile can be treated as an elastic material in the modeling.

In the simulation, the projectiles are modeled as elastic material with a density of $\rho = 8.147$ g/cm^3, elastic modulus of $E = 212.42$ GPa, and Poisson's ratio of $\nu = 0.3$. The HJC model [42] is used for concrete with the material parameters listed in Table 4. An ideal elastic–plastic model is used for the rebars with a density of $\rho = 7.5$ g/cm^3, elastic modulus of $E = 210$ GPa, Poisson's ratio of $\nu = 0.284$, and yield stress of 235 MPa. The fracture of the rebar is taken into account by deleting the rebar element when its plastic strain is larger than 0.26 [37]. Once $\varepsilon^p > \varepsilon^p_{max} = 0.26$, this element is treated as an erosion element, while the nodes are reserved in the following computation process to take the effect of inertia.

Due to the symmetry, a quarter of the model is modeled as shown in Fig. 22. The background grid near the projectile is locally refined. The spacing of the refined grid is set as 80 mm in x direction, 80 mm in y direction, and the same

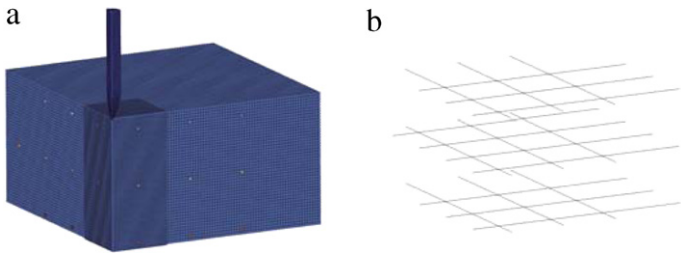

Fig. 22. Discrete model of penetration to RC for MGMPM: (a) projectile and target, (b) steel reinforcement bars isolated from RC target.

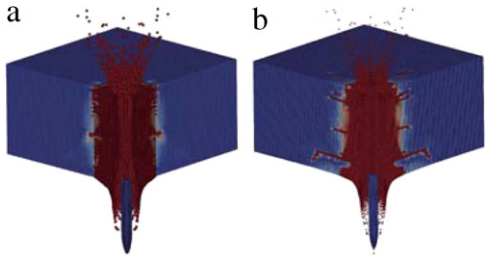

Fig. 23. Comparison of damage contour of targets obtained by: (a) MGMPM, (b) HFEMP.

Table 5
The residual velocities v_r (m/s) and the computation cost comparison.

v_0	v_r			EndTime/(ms)	CPU (min)	
	Experiment	MGMPM	HFEMP		MGMPM	HFEMP
434	214	200	185	2	216	697
606	449	413	400	0.6	52	193
749	615	584	564	0.5	40	166
1058	947	908	899	0.3	31	118

to the coarse domain in z direction, which will cover the projectile throughout the computations process. The cell size for the coarse grid is 0.8 mm. For comparison, this problem is simulated by HFEMP with the refined cell size of 0.4 mm. The particle interval is set as one half of the cell size, and the rebar element size as 0.4 mm.

Four cases with striking velocities of 434 m/s, 606 m/s, 749 m/s, and 1058 m/s, respectively, are simulated. First, the damage contour of the targets obtained by MGMPM and HFEMP is shown in Fig. 23. From the comparison one can find that the damage zone obtained by MGMPM is slightly larger than that obtained by HFEMP. This is because there is wave reflection from the interface between two level grids in MGMPM. Besides, due to the coarse cell size used in level 1 grid, some details are not obtained in the coarse domain compared with the result by HFEMP with refined cell size. However, the result of MGMPM is close to that of HFEMP overall. For such problem, the residual velocity of the projectile is an important variable to prove the accuracy of MGMPM and MPM. Then the residual velocities of the projectile obtained by MGMPM and HFEMP are listed in Table 5, which shows that the results obtained by MGMPM and HFEMP are in good agreement between them and with the experimental data. For this example, the efficiency of MGMPM is about 3 to 4 times that of HFEMP(or MPM), as shown in Table 5.

In addition, this example shows that wave reflection exists in the proposed MGMPM due to different element sizes applied in the background grid. However, compared with the result in this section, the result in Section 5.1 is smoother. This is explained by the fact that a linear elastic material with Poisson's ratio 0 was used, which mimics the problem for 1D problem, while 3D HJC model with multi-parameters was used in the example of this Section. Furthermore, a damage model is included in HJC model.

Finally, the case with striking velocity of 749 m/s is simulated by parallel MGMPM to show its parallel efficiency. This example is accomplished on a shared memory machine with two Quad-Core Xeon 5520 CPUs with four threads each. The operating system is CentOS and gcc 4.2.1 compiler is applied.

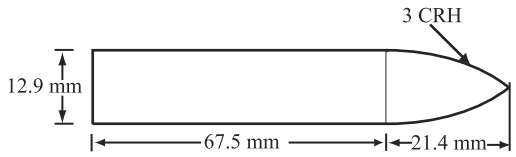

Fig. 24. Schematic of the ogive-nosed projectile.

Table 6
Parallel Efficiency for the case with striking velocity of 749 m/s.

Threads	1	2	3	4	5	6	7	8
Efficiency	1	0.86	0.79	0.69	0.48	0.45	0.39	0.34

Table 7
Material constants of Projectile.

ρ (g/mm^3)	E (GPa)	μ	σ_y (GPa)	E_T (GPa)
0.00785	202	0.3	1.43	14.759

Table 8
Material constants of A6061-T651 for strength model and EOS.

ρ (g/m^3)	E (MPa)	μ	A (MPa)	B (MPa)	n	C	m
0.0027	69	0.3	262	52.1	0.41	0	0.859
c_0 (mm/ms)	s	γ_0	T_{melt}(K)	T_{room}(K)			
5,350	1.34	2.0	875	293			

We decomposed the computing domain along the direction of the striking velocity. Table 6 lists the overall parallel efficiency with respect to the thread number used, which shows that the overall efficiency is decreasing with increasing the number of threads. When the number of threads increases, load balance will deteriorate because the computing domain is decomposed into two times the number of threads. Besides, in this simulation, adaptive particle splitting method and contact method based on local multi-background grid [43] are used. However, both methods cannot be parallelized in OpenMP standard because they will manipulate memory directly. These might decrease the efficiency with the number of threads increasing based on Amdahl's law. However, it can be found from Table 6 that a good efficiency of about 70% can be achieved when using 4 threads.

5.4. Penetration of thick plate

This example is about a penetration of a high strength steel projectile against a A6061-T651 thick plate conducted by Piekutowski et al. [44]. In the test, the projectile has a length of 88.9 mm and a diameter of 12.9 mm with a 3.0 caliber-radius-head as shown in Fig. 24, while the target has a thickness of 26.3 mm and an area of 110 mm × 110 mm. The projectile impacted the target obliquely at an angle of 30°.

In the simulation, the projectiles are modeled as an elastic–plastic material, while the targets are modeled by Johnson–Cook constitutive model for deviatoric stress, and Mie–Grüneisen equation of state for pressure. The material constants for the projectile and target are listed in Tables 7 and 8 [44,45], respectively. For the target, material failure is taken into account by setting the deviatoric components of the stress tensor to zero when the effective plastic strain reaches the value of $\varepsilon_{\text{fail}} = 1.6$. The friction coefficient between projectile and target is set to zero.

Due to symmetry, one half of the model is studied as shown in Fig. 25. A background grid with two levels is applied. The cell size of the grid of level 1 is 2 mm. The space domain of the grid of level 2 is set as length of 48 mm in x direction, 24 mm in y direction, and 110 mm in z direction, which covers the projectile throughout the computation process. For comparison, this problem is also simulated by MPM with the refined cell size of 1 mm.

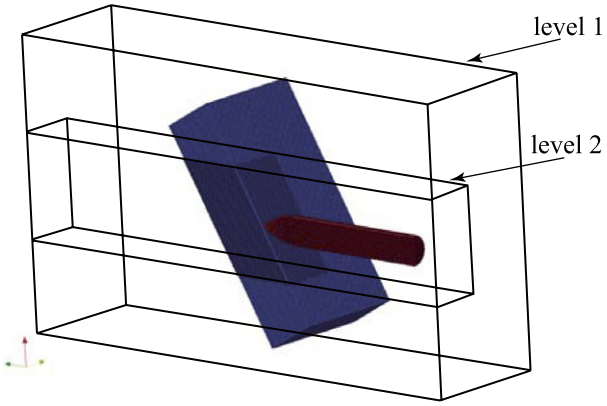

Fig. 25. The discretization model of penetration to thick plate for MGMPM.

Table 9
Residual velocities v_r (m/s) and the computation cost comparison.

v_0	v_r			EndTime/(ms)	CPU (min)	
	Experiment	MGMPM	MPM		MGMPM	MPM
446	288	299	306	0.4	252	467
575	455	472	471	0.3	177	391
730	655	651	652	0.21	127	258

Table 10
Parallel efficiency for the case with striking velocity of $v_0 = 575$ m/s.

Threads	1	2	3	4	5	6	7	8
Efficiency	1	0.82	0.63	0.53	0.39	0.34	0.3	0.29

The numerical results obtained by MGMPM and MPM are compared with experiment data. First, the projectile–target interactions at three impact times are studied for the case in which the striking velocity of projectile is 575 m/s. Fig. 26(a) shows a sequence of experimental X-ray photographs, Fig. 26(b) shows the numerical results by MGMPM, and Fig. 26(c) by MPM, which show that the projectile's shapes obtained by both MGMPM and MPM are in good agreement between them and the experimental results. Then, the residual velocities of the projectile obtained by MGMPM and MPM are compared in Table 9, which shows that the numerical results agree well with the experimental data. The difference between the residual velocities by two methods is due to the coarse grid used in MGMPM and is negligible, while the efficiency of MGMPM is much higher than MPM as shown in Table 9.

We also studied the case with striking velocity of 575 m/s by parallel MGMPM to show its parallel efficiency. We decomposed the computing domain along the direction of the initial velocity. The overall parallel efficiency with respect to thread number used is listed in Table 10, which shows that the efficiency in this simulation is not better than that of example 3. Besides the reason mentioned above, the particles distribution in this simulation is not as good as that in the example 3, and therefore the position of particles in the particle index does not match their spacial position [36]. This causes "cache miss", which also can deteriorate the overall efficiency. Therefore, further optimization of the original serial program to avoid cache miss is also needed. All the results were obtained by using the release version of MGMPM with -O3 optimization.

6. Conclusion

In this paper, a mesh-grading material point method (MGMPM) is proposed for problems with localized extreme deformation with efficiency higher than that of conventional material point method (MPM). In this method, the background grid as a whole is divided into several sub grids level by level with half of the coarse cell size decreasing. Then, the material domain undergoing localized extreme deformation can be covered by locally refined background

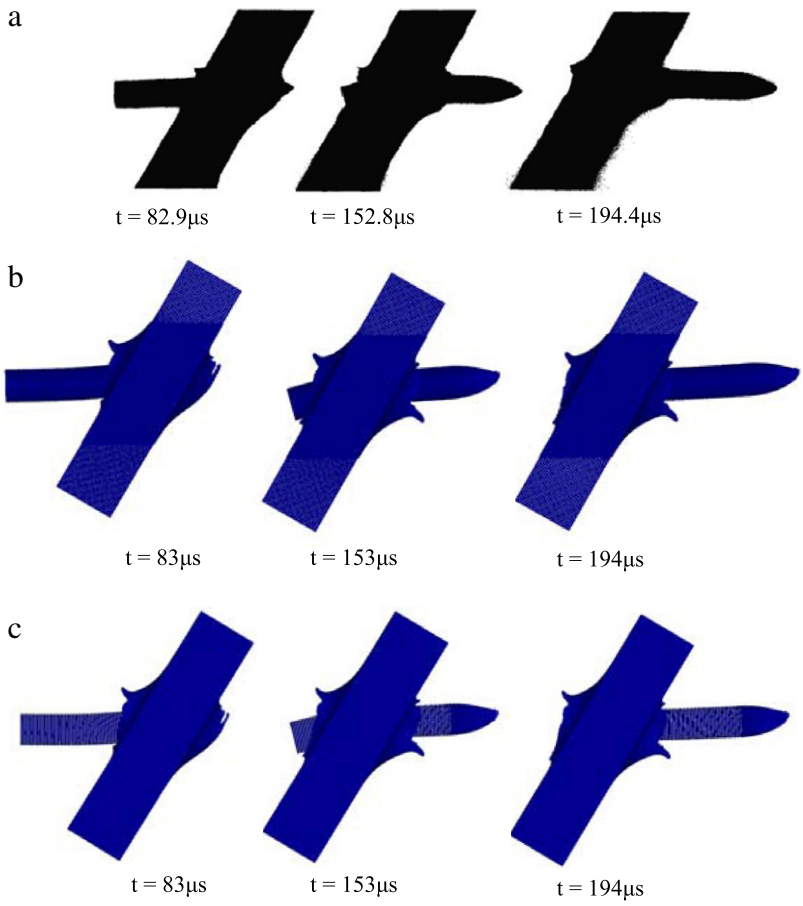

Fig. 26. Comparison of projectile–target interactions for the case with striking velocity of $v_0 = 575$ m/s.

grid, and correspondingly discretized by refined particles, while elsewhere by coarse background grid and particles. Cells with mid-face nodes and/or mid-edge nodes are used to link two adjacent sub grids. The edge displacement continuity associated with the linking cells is embedded in the nodal shape function. Besides, the truss element is incorporated into MGMPM to enable it with ability to model the steel reinforcement bars (rebars) in reinforced concrete impact problems. Furthermore, parallelization is implemented via OpenMP. Several numerical examples including stress wave propagation, Taylor bar impact, and penetration problems, are given, which show that the proposed method is a better method to solve localized extreme deformation problems with higher efficiency and lower memory requirement than MPM. The parallelization scheme for MGMPM performs well, and the overall parallel efficiency will deteriorate when the number of threads increases due to the load balance issue, serial code section, and catch miss. Therefore, further optimization of the original serial program is still needed in the future.

Acknowledgment

The first author would like to acknowledge Miguel Bessa, for his helpful suggestions concerning this manuscript.

References

[1] J.J. Monaghan, An introduction to SPH, Comput. Phys. Comm. 48 (1988) 89–96.
[2] M.B. Liu, G.R. Liu, Smoothed particle hydrodynamics (SPH): an overview and recent developments, Arch. Comput. Methods Eng. 17 (2010) 25–76.
[3] A. Rafiee, K.P. Thiagarajan, An SPH projection method for simulating fluid-hypoelastic structure interaction, Comput. Methods Appl. Mech. Engrg. 198 (33–36) (2009) 2785–2795.
[4] T. Belytschko, Y.Y. Lu, L. Gu, Element free Galerkin methods, Internat. J. Numer. Methods Engrg. 37 (1994) 229–256.

[5] X.F. Pan, H. Yuan, Computational algorithm and application of element-free Galerkin methods for nonlocal damage models, Eng. Fract. Mech. 77 (2010) 2640–2653.
[6] W.K. Liu, Y.J. Chen, Wavelet and multiple scale reproducing kernel methods, Internat. J. Numer. Methods Fluids 21 (10) (1995) 901–931.
[7] J.S. Chen, C. Pan, C.M.O.L. Roque, H.P. Wang, A Lagrangian reproducing kernel particle method for metal form analysis, Comput. Mech. 22 (3) (1998) 289–307.
[8] D. Sulsky, Z. Chen, H.L. Schreyer, A particle method for history-dependent materials, Comput. Methods Appl. Mech. Engrg. 118 (1–2) (1994) 179–196.
[9] S.G. Bardenhagen, E.M. Kober, The generalized interpolation material point method, CMES Comput. Model. Eng. Sci. 5 (6) (2004) 477–495.
[10] S.A. Silling, M. Epton, O. Weckner, J. Xu, E. Askari, Peridynamic states and constitutive modeling, J. Elasticity 88 (2007) 151–184.
[11] J.T. Foster, S.A. Silling, W.W. Chen, Visoplasticity using peridynamics, Internat. J. Numer. Methods Engrg. 81 (2010) 1242–1258.
[12] M.A. Bessa, J.T. Foster, T. Belytschko, W.K. Liu, A meshfree unification: reproducing kernel peridynamics, Comput. Mech. 53 (2014) 1251–1264.
[13] S. Ma, X. Zhang, X.M. Qiu, Comparison study of MPM and SPH in modeling hypervelocity impact problems, Int. J. Impact. Eng. 36 (2009) 272–282.
[14] W.Q. Hu, Z. Chen, Model-based simulation of the synergistic effects of blast and fragmentation on a concrete wall using the MPM, Int. J. Impact. Eng. 32 (12) (2006) 2066–2096.
[15] Y.P. Lian, X. Zhang, X. Zhou, S. Ma, Y.L. Zhao, Numerical simulation of explosively driven metal by material point method, Int. J. Impact. Eng. 38 (2011) 237–245.
[16] J.A. Nairn, On the calculation of energy release rates for cracked laminates with residual stresses, Int. J. Fract. 139 (2006) 267–293.
[17] P.F. Yang, Y. Liu, X. Zhang, X. Zhou, Y.L. Zhao, Simulation of fragmentation with material point method based on Gurson model and random failure, CMES Comput. Model. Eng. Sci. 85 (3) (2012) 207–236.
[18] J.E. Guilkey, J.B. Hoying, J.A. Weiss, Computational modeling of multicellular constructs with the material point method, J. Biomech. 39 (2006) 2074–2086.
[19] S.Z. Zhou, X. Zhang, H.L. Ma, Numerical simulation of human head impact using the material point, Int. J. Comput. Methods 10 (04) (2013) 1350014.
[20] J.E. Guillkey, T.B. Harman, B. Banerjee, An Eulerian–Lagrangian approach for simulating explosions of energetic devices, Comput. Struct. 85 (2007) 660–674.
[21] Y. Gan, Z. Chen, S.M. Smith, Improved material point method for simulating the zona failure response in piezo-assisted intracytoplasmic sperm injection, CMES Comput. Model. Eng. Sci. 73 (1) (2011) 45–76.
[22] Y. Liu, H.K. Wang, X. Zhang, A multiscale framework for high-velocity impact process with combined material point method and molecular dynamics, Int. J. Mech. Mater. Des. 9 (2013) 127–139.
[23] Z. Chen, S. Jiang, Y. Gan, H.T. Liu, T.D. Sewell, A particle-based multiscale simulation procedure within the material point method framework, Comp. Part. Mech. (2014).
[24] S. Andersen, L. Andersen, Modelling of landslides with the material-point method, Comput. Geosci. 14 (2010) 137–147.
[25] J. Ma, D. Wang, M.F. Randolph, A new contact algorithm in the material point mehtod for geotechnical simulations, Int. J. Numer. Anal. Methods Geomech. (2014).
[26] A. Sadeghirad, R.M. Brannon, J. Burghardt, A convected particle domain interpolation technique to extend applicability of the material point method for problems involving massive de-formations, Internat. J. Numer. Methods Engrg. 86 (2011) 1435–1456.
[27] D.Z. Zhang, X. Ma, P.T. Giguere, Material point method enhanced by modified gradient of shape function, J. Comput. Phys. 230 (2011) 6379–6398.
[28] S.G. Bardenhagen, J.U. Brackbill, D. Sulsky, The material-point method for granular materials, Comput. Methods Appl. Mech. Engrg. 187 (3–4) (2000) 529–541.
[29] S.G. Bardenhagen, J.E. Guilkey, K.M. Roessig, J.U. Brackbill, W.M. Witzel, J.C. Foster, An improved contact algorithm for the material point method and application to stress propagation in granular material, CMES Comput. Model. Eng. Sci. 2 (4) (2001) 509–522.
[30] P. Huang, X. Zhang, S. Ma, X. Huang, Contact algorithms for the material point method in impact and penetration simulation, Internat. J. Numer. Methods Engrg. 85 (4) (2011) 498–517.
[31] Y.P. Lian, X. Zhang, Y. Liu, Coupling of finite element method with material point method by local multi-mesh contact method, Comput. Methods Appl. Mech. Engrg. 200 (2011) 3482–3494.
[32] Y.P. Lian, X. Zhang, Y. Liu, An adaptive finite element material point method and its application in extreme deformation problems, Comput. Methods Appl. Mech. Engrg. 241–244 (1) (2012) 275–285.
[33] X.X. Cui, X. Zhang, X. Zhou, Y. Liu, F. Zhang, A coupled finite difference material point mehtod and its application in explosion simulation, CMES Comput. Model. Eng. Sci. 98 (2014) 565–599.
[34] Y.P. Lian, X. Zhang, F. Zhang, X.X. Cui, Tied interface grid material point method for problems with localized extreme deformation, Int. J. Impact Eng. 70 (2014) 50–61.
[35] P. Huang, X. Zhang, S. Ma, Shared memory OpenMP parallelization of explicit mpm and its application to hypervelocity impact, CMES Comput. Model. Eng. Sci. 38 (2008) 119–147.
[36] Y.T. Zhang, X. Zhang, Y. Liu, An alternated grid updating parallel algorithm for material point method using OpenMP, CMES Comput. Model. Eng. Sci. 69 (2) (2010) 143–165.
[37] Y.P. Lian, X. Zhang, X. Zhou, Z.T. Ma, A FEMP method and its application in modeling dynamic response of reinforced concrete subjected to impact loading, Comput. Methods Appl. Mech. Engrg. 200 (17–20) (2011) 1659–1670.
[38] J.M. McDill, J.A. Goldak, A.S. Oddy, M.J. Bibby, Isoparametric quadrilaterals and hexahedrons for mesh-grading algorithms, Int. J. Numer. Methods Bio. 3 (2) (1987) 155–163.
[39] S.G. Bardenhagen, Energy conservation error in the material point method for solid mechanics, J. Comput. Phys. 180 (2002) 383–403.

[40] G.R. Johnson, T.J. Holmquist, Evaluation of cylinder-impact test data for constitutive model constants, J. Appl. Phys. 64 (8) (1988) 3901–3910.
[41] S.J. Hanchak, M.J. Forrestal, E.R. Young, J.Q. Ehrgott, Perforation of concrete slabs with 48 MPa (7ksi) and 140 MPa (20ksi) unconfined compressive strengths, Int. J. Impact Eng. 12 (1) (1992) 1–7.
[42] T.J. Holmquist, G.R. Johnson, W.H. Cook, A computational constitutive model for concrete subjected to large strains, high strain rates, and high pressures, in: 14th International Symposium on Ballistics Quebec, Candan, 26–29 September 1993, 1993.
[43] Z. Ma, X. Zhang, P. Huang, An object-oriented MPM framework for simulation of large deformation and contact of numerous grains, CMES Comput. Model. Eng. Sci. 55 (1) (2010) 61–87.
[44] A.J. Piekutowski, M.J. Forrestal, K.L. Poormon, T.L. Warren, Perforation of aluminum plates with ogive-nose steel rods at normal and oblique impacts, Int. J. Impact Eng. 18 (1996) 877–887.
[45] M.A. Meyers, Dynamic Behavior of Materials, John Wiley & Sons, New York, 1994, p. 133.