Comput. Methods Appl. Mech. Engrg, **191**, 2095–2109 (2002)

Hierarchical, Adaptive, Material Point Method for Dynamic Energy Release Rate Calculations

Honglai Tan and John A. Nairn*
Material Science & Engineering, University of Utah, Salt Lake City, UT 84112, U.S.A

April 26, 2001

Abstract

A crack-closure method was developed for use in Material Point Method (MPM) calculations. The method can be used for calculation of the dynamic energy release rate in a variety of dynamic fracture mechanics problems. Most previous MPM analyses have used regular grids and a "lumped" mass matrix. For the most accurate energy release rate calculations, the regular grid had to be replaced by an adaptive grid that automatically refined the mesh around the crack tip and the lumped mass matrix had to be replaced by a full mass matrix. Using an adaptive mesh was more important to accuracy than was switching to a full mass matrix. Some sample calculations are given for energy release rate in a double cantilever beam specimen carried out by several different MPM methods.

1. Introduction

The material point method (MPM) has recently been developed as a numerical method for solving problems in dynamic solid mechanics [1–4]. In MPM, a material is represented by a collection of material points. 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 equations of motion for the particles are solved on a background calculational grid; this solution is used to update the particles and the background mesh can be discarded or reused for the next time step in its initial, undistorted form. 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.

MPM has been used to analyze material failure problems [4], but the approach has been by modification of constitutive laws to mimic damage. A problem with this approach is that it contradicts experimental results. For example, the use of constitutive laws to predict transverse matrix cracking in cross-ply laminates would assume that the cracks form when the stress in the transverse plies reaches some critical value. Experimental observations show, however, that the stress in those plies at the formation of a crack is not a constant, but rather a complicated function of composite geometry and proximity of existing cracks [5, 6]. In other words, it is not possible to construct a constitutive law to model damage by such a failure mode. A more rigorous simulation of material failure, albeit a more difficult one to implement numerically, is to do explicit modeling of cracks and use fracture mechanics methods to find the energy release rate for crack growth. For the previously cited example of matrix cracking, explicit cracks and energy methods were found to explain nearly all the experimental results that were modelled incorrectly by a constitutive-law approach [5]. This paper considers modification of MPM for explicit crack calculations. Before such simulations are possible, however, there are three problems that need to be solved. First, fracture calculations might require more mesh refinement than previously used in MPM calculations; second, the standard MPM algorithm has some energy dissipation [7] that might affect energy-based fracture calculations; and third, current MPM does not correctly represent internal crack surfaces. The first two problems are the subject of this paper.

Previous MPM calculations in solid mechanics have typically discretized the material using a regular grid in which all elements in the background calculational grid are the same size. Such a mesh is inefficient for solving large problems while simultaneously capturing all the detail required to find the crack-tip stress state needed for accurate fracture calculations. To solve this problem, we have developed hierarchical, adaptive

*Corresponding Author: Phone:+1-801-581-3413, FAX:+1-801-581-4816, Email:John.Nairn@m.cc.utah.edu

MPM methods. During the simulation, these new MPM cells can adapt automatically to local resolution requirements as judged by tracking gradients in the evolving solution. The use of adaptive cells greatly improved the accuracy of our energy release rate calculations.

In each MPM time step, the grid accelerations are related to the grid forces through a mass matrix. To avoid inverting the full mass matrix, most MPM simulations use a lumped mass matrix in which the masses in each row are summed and placed on the diagonal. This approach gives excellent results for many problems but causes some kinetic energy dissipation [7]. Because fracture calculations rely on energy calculations, it is important to assess the effect of a lumped mass matrix on energy release rate calculations. We found that using a full mass matrix does improve accuracy, but that refinement near crack tips using adaptive cells was more important to accuracy than is the use of a full mass matrix.

The **Theory** sections gives some background information on MPM calculations and then presents one method for calculating energy release rate for crack growth from typical MPM results. For improved accuracy in energy release rate calculations, we used both adaptive cells and full mass matrix calculations. The **Theory** section also describes the techniques for splitting particles, the criterion used for deciding when to refine particles, and an estimation of error reductions associated with refinement or with the use of a full mass matrix. The **Results** section describes energy release rate calculations for a double cantilever beam under mode I loading. The energy release rate was calculated by all combinations of adaptive or non-adaptive cells and lumped or full mass matrix methods.

2. Theory

2.1. Material Point Method Background

In MPM [1–4], a material is discretized into a collection of N material points in a manner similar to representing an image with pixels (see Fig. 1). Each material point is assigned a mass (m_p, $p = 1, \ldots, N$) consistent with the material density and all the variables needed to solve the problem (*e.g.*, position, velocity, strain, stress, *etc.*). To solve the equations of motion for the particles for each time step of the analysis, which requires evaluation of gradients and retention of continuous displacement fields, MPM uses a background calculational grid (see Fig. 1). Using this grid, any function, $\phi(\boldsymbol{x})$ can be expressed as a piecewise continuous function by using standard finite element shape functions

$$\phi(\boldsymbol{x}) = \sum_{i-1}^{n} \phi^{(i)} N_i(\boldsymbol{x}) \tag{1}$$

where n is the number of nodes in the background grid and superscript (i) refers to the nodal values of $\phi(\boldsymbol{x})$. For simplicity, this sum is extended from $i = 1$ to n, but in typical finite element descriptions of the background grid, the only non-zero shape functions will be the ones associated with the element containing the coordinate $\boldsymbol{x}$.

The governing equations for motion of the particles are

$$\rho \operatorname{div} \boldsymbol{\sigma}^{\mathrm{s}} + \rho \boldsymbol{b} = \rho \boldsymbol{a} \tag{2}$$

where $\boldsymbol{\sigma}^s$ is the *specific* stress tensor, $\boldsymbol{b}$ is the specific body force, and $\boldsymbol{a}$ is the acceleration. For linear-elastic materials, the stress and strain further satisfy

$$\boldsymbol{\sigma}^s = \boldsymbol{C}^{(s)} \boldsymbol{\varepsilon} \qquad \text{and} \qquad \boldsymbol{\varepsilon} = \frac{1}{2}\left[(\nabla \boldsymbol{u}) + (\nabla \boldsymbol{u})^T\right] \tag{3}$$

where $\boldsymbol{C}^{(s)}$ is the *specific*, fourth-rank, stiffness tensor, $\boldsymbol{\varepsilon}$ is the strain tensor, and $\boldsymbol{u} = (u_x, u_y, u_z)$ is the displacement field. MPM is well suited for analysis of plastic and viscoelastic materials which would require additional equations. The results in this paper considered only linear elastic materials. In MPM, these equations are solved by writing the density in terms of material point masses (m_p) and locations ($\boldsymbol{x}_p$) and expressing acceleration in terms of the shape functions

$$\rho = \sum_{p=1}^{N} m_p \delta(\boldsymbol{x} - \boldsymbol{x}_p) \qquad \text{and} \qquad \boldsymbol{a}(\boldsymbol{x}) = \sum_{i=1}^{n} \boldsymbol{a}^{(i)} N_i(\boldsymbol{x}) \tag{4}$$

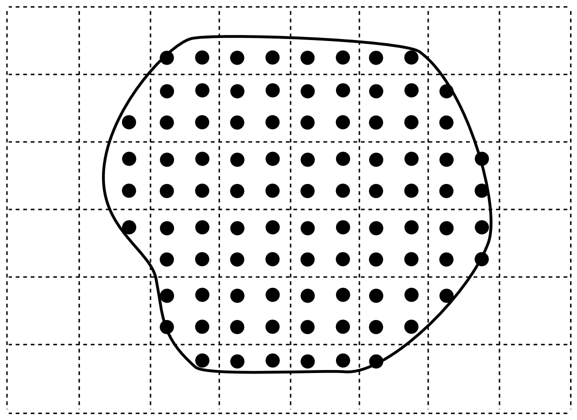

Fig. 1. A schematic view of a two-dimensional MPM calculation. The solid line is a outline of the body analyzed. The black dots are material points. The dashed lines here show a regular, background, calculational grid.

In this representation, the acceleration vector at each grid point is found by solving [1]:

$$\sum_{j=1}^{n} M_{ij}\boldsymbol{a}^{(j)} = \boldsymbol{f}_i^{int} + \boldsymbol{f}_i^{ext} \tag{5}$$

where M_{ij} is the mass matrix:

$$M_{ij} = \sum_{p=1}^{N} m_p N_i(\boldsymbol{x}_p) N_j(\boldsymbol{x}_p) \tag{6}$$

and $\boldsymbol{f}_i^{int}$ and $\boldsymbol{f}_i^{ext}$ are "forces" on node i arising from the current stresses or from body forces and tractions, respectively. They are defined by [1]

$$\boldsymbol{f}_i^{int} = -\sum_{p=1}^{N} m_p \boldsymbol{\sigma}^s(\boldsymbol{x}_p)\nabla N_i(\boldsymbol{x}_p) \qquad \text{and} \qquad \boldsymbol{f}_i^{ext} = \hat{\boldsymbol{\tau}}^{(i)} + \sum_{p=1}^{N} m_p \boldsymbol{b}(\boldsymbol{x}_p) N_i(\boldsymbol{x}_p) \tag{7}$$

where $\hat{\boldsymbol{\tau}}^{(i)}$ is the surface traction associated with grid point i.

In each time step, the grid point accelerations are used to calculate the grid point velocities. The grid point acceleration and velocities are then used together with material properties to update the material point positions, velocities, stress, and strain. At this point, the now-deformed background grid is discarded and a new undeformed grid is created for the next time step. In many MPM simulations, the new unformed grid is identical to all previous undeformed grids and thus the previous grid can be reused. But, it is also possible to define a new grid, which means MPM is well suited to adaptive mesh techniques. As explained below, this feature of MPM was used in our fracture calculations to adaptively refine the mesh around a crack tip.

To begin the next time step, the velocities at the grid points of the new grid must be calculated by extrapolation from the material points to the new grid. Because there are always more material points than grid points, this extrapolation is done by a least squares method which results in an equation for grid point velocities in terms of material point masses and velocities ($\boldsymbol{v}(\boldsymbol{x}_p)$) [1, 4]:

$$\sum_{i=1}^{n} M_{ij}\boldsymbol{v}^{(j)} = \sum_{p=1}^{N} m_p \boldsymbol{v}(\boldsymbol{x}_p) N_i(\boldsymbol{x}_p) \tag{8}$$

where M_{ij} is the same mass matrix in (5) for grid point accelerations.

Each time step in MPM requires inversion of the large mass matrix, M_{ij}. This inversion must be repeated each time step because the mass matrix depends on the locations of the particles and thus changes at each time step. For a more efficient algorithm, the mass matrix is typically replaced by a diagonal or "lumped" mass matrix in which the elements of each row of the mass matrix are summed and placed on the diagonal [1, 4]:

$$M_{ij}^L = \begin{cases} \displaystyle\sum_{j=1}^{n} M_{ij} = \sum_{p=1}^{N} m_p N_i(\boldsymbol{x}_p) & \text{for } i = j \\ 0 & \text{for } i \neq j \end{cases} \tag{9}$$

By using a lumped mass matrix, the grid point accelerations and velocities can trivially be found from

$$\boldsymbol{a}^{(i)} = \frac{1}{M_{ii}^L}\left(\boldsymbol{f}_i^{int} + \boldsymbol{f}_i^{ext}\right) \qquad \text{and} \qquad \boldsymbol{v}^{(i)} = \frac{1}{M_{ii}^L}\sum_{p=1}^{N} m_p \boldsymbol{v}(\boldsymbol{x}_p) N_i(\boldsymbol{x}_p) \tag{10}$$

The cost is a small amount of numerical dissipation of kinetic energy [7]. Because fracture calculations rely on accurate energy calculations, we investigated the effect of lumped *vs.* unlumped mass matrices on the accuracy of energy release rate calculations.

2.2. Energy Release Rate Calculation

In finite element analysis, energy release rate for crack growth is usually found using crack closure methods [9]. By making use of the background calculation grid, a similar method can be implemented in MPM. The mode I energy release rate for growth of a traction-free crack can be expressed in terms of a virtual crack-closure integral [8]:

$$G_I = \lim_{\delta a \to 0} \frac{1}{2\delta a} \int_0^{\delta a} \sigma_{yy}(x) \Delta u_y(x - \delta a) dx \tag{11}$$

where $\sigma_{yy}(x)$ is the stress normal to the crack surface and $\Delta u_y(x - \delta a)$ is the crack-opening displacement normal to the crack surface (see Fig. 2). In finite element calculations, this integral can be approximated using nodal displacements and the nodal force at the crack-tip node [9]. MPM does not have a comparable nodal force, but at any time step, the stresses and displacements on the particles can be extrapolated to the grid nodes by the same equation used to extrapolate velocities (see (8), or the lumped mass matrix version in (10)). Writing the stresses as a linear interpolation between nodal values at nodes (2) and (3) and the displacements as linear interpolations between nodes (1) or (1′) and (2), and substituting into the crack closure integral leads to

$$G_I = \frac{u_y^{(1)} - u_y^{(1')}}{12}\left(2\sigma_{yy}^{(2)} + \sigma_{yy}^{(3)}\right) \tag{12}$$

where superscripts (i) refer to nodal values of displacement or stress at the nodes show in Fig. 2. A similar analysis for mode II energy release rate gives

$$G_{II} = \frac{u_x^{(1)} - u_x^{(1')}}{12}\left(2\sigma_{xy}^{(2)} + \sigma_{xy}^{(3)}\right) \tag{13}$$

All calculations in this paper were for mode I loading of a crack at a symmetry plane. We thus only analyzed the half plane above the crack in Fig. 2. The y-direction displacements at nodes (2), (3) (*etc.*, for $x > 0$) were constrained to be zero. By symmetry, the displacements at node (1') were $u_y^{(1')} = -u_y^{(1)}$ and $u_x^{(1')} = u_x^{(1)}$. The energy release rates were thus calculated from

$$G_I = \frac{u_y^{(1)}}{6}\left(2\sigma_{yy}^{(2)} + \sigma_{yy}^{(3)}\right) \qquad \text{and} \qquad G_{II} = 0 \tag{14}$$

2.3. Hierarchical Refinement

The accuracy of crack-closure calculations depends on the size of the elements at the crack tip (δa) and is best when δa is small relative to the size of the specimen. Most previous MPM calculations for solid mechanics have used a regular calculational grid of equally spaced nodes. To get small δa would require the inefficient use of small elements throughout the mesh. Instead, we developed adaptive, hierarchical elements

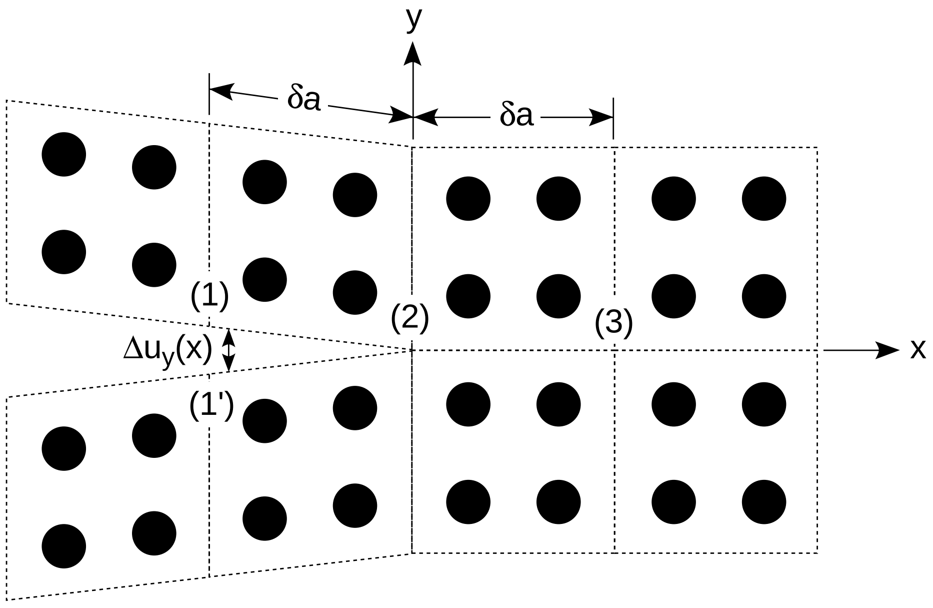

Fig. 2. MPM elements around a crack tip. The background calculational grid is shown deformed for clarity. The numbers in parentheses are node numbers in the background grid. The crack tip is at node (2) and is the origin of the (x, y) axes.

that automatically refine into smaller elements and more material points in regions of high gradients such as around a crack tip.

Consider a single material point representing an area Ω_p (or volume in 3D calculations). Because the mass of the particle must be consistent with the density of the material, the area Ω_p can be described by an equivalent square of dimension

$$c = \sqrt{\frac{m_p}{\rho t}} \tag{15}$$

where t is the thickness of a 2D analysis. If the original particle is at location (x_0, y_0) when there is no deformation, it could be split, or replaced, by four particles at the quarter points of the equivalent square. In other words, the coordinates of the new particles would be at $(x_0 \pm c/4, y_0 \pm c/4)$. But, when particles are split during an MPM calculation, the original particle will be under strain and the locations of the four new particles should account for the current deformation field. Our approach was to split the strained particle into four new particles such that if the deformations were removed, the new particles would return to the undeformed locations of $(x_0 \pm c/4, y_0 \pm c/4)$. Under deformation field $\boldsymbol{u}$, the original particle and split particles will have moved from the undeformed locations to

$$\text{Original Particle}: \quad \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} + \begin{pmatrix} u_x \\ u_y \end{pmatrix} \tag{16}$$

$$\text{SplitParticles}: \quad \begin{pmatrix} x_i \\ y_i \end{pmatrix} = \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} + \begin{pmatrix} u_x \\ u_y \end{pmatrix} + \left(\mathbf{I} + \nabla \boldsymbol{u}\right) \begin{pmatrix} \pm c/4 \\ \pm c/4 \end{pmatrix} \tag{17}$$

where $\mathbf{I}$ is the identify matrix. Thus, to retain the current deformation field, a deformed particle at position (x, y) should be split into four particles located at

$$\begin{pmatrix} x_i \\ y_i \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 1 + \varepsilon_{xx} & \varepsilon_{xy} + \omega_{xy} \\ \varepsilon_{yx} + \omega_{yx} & 1 + \varepsilon_{yy} \end{pmatrix} \begin{pmatrix} \pm c/4 \\ \pm c/4 \end{pmatrix} \tag{18}$$

where $\boldsymbol{\omega}$ is the rotation tensor defined by

$$\boldsymbol{\omega} = \frac{1}{2}\left[(\nabla \boldsymbol{u}) - (\nabla \boldsymbol{u})^T\right] \tag{19}$$

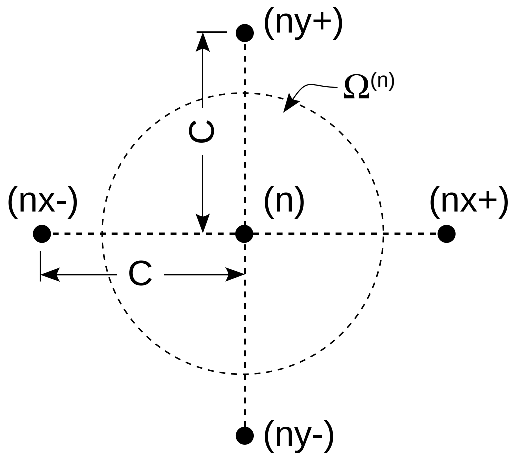

Fig. 3. A node n in the background grid and four nodes located around that node that are used to calculate gradients needed for the refinement parameter $R^{(n)}$. The nodes are separated by a distance c. The area $\Omega^{(n)}$ indicates any area centrally located on node n.

For small-strain calculations, the differences between splitting undeformed and deformed particles are small, but they are easy to implement. The above method is for 2D problems. The extension to 3D problems where a single particle is split into 8 particles is straightforward.

To adaptively refine material points during an MPM calculation, we used a dimensionless refinement parameter based on the gradient of a current calculational result. Any calculation result can be used, but for the fracture calculations in this paper, we used the gradient of the energy density. Let $\phi(\boldsymbol{x})$ be any scalar function of position. The refinement parameter for node n in the calculational grid is ideally calculated from

$$R^{(n)} = \left| \frac{\int_{\Omega^{(n)}} \left(\phi(\boldsymbol{x}) - \phi^{(n)}\right) dV}{\int_{\Omega^{(n)}} \phi^{(n)}\, dV} \right| \tag{20}$$

Unfortunately, we do not know the exact $\phi(\boldsymbol{x})$. To get a result for $R^{(n)}$, we assumed $\phi(\boldsymbol{x})$ can be expressed as a quadratic Taylor expansion:

$$\phi(\boldsymbol{x}) - \phi^{(n)} = \phi^{(n)}_{,i}\left(\boldsymbol{x}_i - \boldsymbol{x}^{(n)}_i\right) + \frac{1}{2}\phi^{(n)}_{,ij}\left(\boldsymbol{x}_i - \boldsymbol{x}^{(n)}_i\right)\left(\boldsymbol{x}_j - \boldsymbol{x}^{(n)}_j\right) \tag{21}$$

where repeated indices are summed $(i, j = x, y)$, subscript "$,i$" indicates partial differentiation with respect to that coordinate, and superscript (n) indicates the value at node n. If the integration volume is selected as any area centrally located on node n, the linear terms and the $\phi^{(n)}_{,xy}$ and $\phi^{(n)}_{,yx}$ terms are zero by symmetry. The final result thus only depends on the second derivatives which we evaluated from the neighboring nodal values shown in Fig. 3 using

$$\phi^{(n)}_{,xx} = \frac{\phi^{(nx+)} + \phi^{(nx-)} - 2\phi^{(n)}}{c^2} \qquad \text{and} \qquad \phi^{(n)}_{,yy} = \frac{\phi^{(ny+)} + \phi^{(ny-)} - 2\phi^{(n)}}{c^2} \tag{22}$$

Integration over any centrally located area leads to

$$R^{(n)} \propto \left| \frac{\langle \phi^{(\pm)} \rangle}{\phi^{(n)}} - 1 \right| \tag{23}$$

where $\langle \phi^{(\pm)} \rangle$ is the average value of $\phi(\boldsymbol{x})$ at the four neighboring nodes and the proportionality constant is a function of the integration area selected. For simplicity, the constant can be ignored and $R^{(n)}$ is calculated from (23).

In our MPM calculations, we evaluated the value of $R^{(n)}$ for each time step. Then, for each element in the calculation grid, we examined the average value of $R^{(n)}$ from all nodes for that element. If $\langle R^{(n)} \rangle > R_{crit}$,

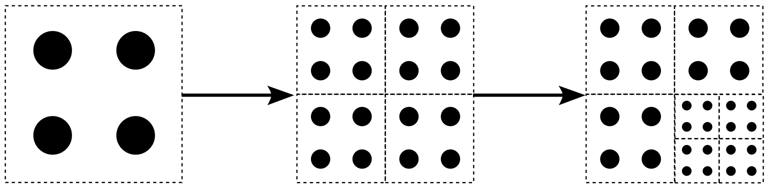

Fig. 4. The left side shows the initial material points in a single element. If $\langle R^{(n)} \rangle > R_{crit}$, any element can refine by splitting all material points in that element into 4 new material points. The refinement can continue for more levels as required by the problem being analyzed.

where R_{crit} is some predetermined critical value, each material point in that element was split into four points with new coordinates as discussed above. The adaptivity is hierarchical because at the next step the four new elements are examined individually and one or more may continue to refine. The refinement process is illustrated in Fig. 4. Our calculations only included refinement. In other problems, such as problems with crack propagation, it should be possible to recombine material points when refinement is no longer needed. In other words, if $\langle R^{(n)} \rangle < R_{crit}$ for some previously refined elements, the material points can be recombined. If necessary, the critical values for refining and recombining can be selected differently. Recombination should be straight forward for elastic materials but might require some approximations for history-dependent materials. Finally, notice that after the second refinement in Fig. 4 that the elements that do not refine have five nodes in the background grid instead of four. Our adaptive mesh shape functions accounted for such elements by allowing background elements with four to eight nodes.

It is possible to estimate the benefits of refinement by comparing the errors in shape-function interpolations for a known function before and after refinement. Assume some scalar function, $\phi(\boldsymbol{x})$, is again given exactly by the expansion in (21) around node (0) at the center of Fig. 5A. Before refinement, the error in calculating $\phi(\boldsymbol{x})$ is

$$\Delta\phi_{coarse}(\boldsymbol{x}) = N_1(\boldsymbol{x})\phi^{(1)} + N_2(\boldsymbol{x})\phi^{(2)} + N_3(\boldsymbol{x})\phi^{(3)} + N_4(\boldsymbol{x})\phi^{(4)} - \phi(\boldsymbol{x}) \tag{24}$$

where $N_i(\boldsymbol{x})$ and $\phi^{(i)}$ are the shape function and nodal value associated with node (i) in Fig. 5A. The nodal values can be calculated from (21) using the nodal coordinates. Because the shape functions for 4-noded elements are linear, the error is zero for the constant and linear terms in (21). The final error is a function of position and the magnitudes of $\phi_{,ij}$. After refinement of the four material points into 16 material points, the error in calculating $\phi(\boldsymbol{x})$ in the upper-right quadrant is

$$\Delta\phi_{fine}(\boldsymbol{x}) = N_0(\boldsymbol{x})\phi^{(0)} + N_5(\boldsymbol{x})\phi^{(5)} + N_3(\boldsymbol{x})\phi^{(3)} + N_6(\boldsymbol{x})\phi^{(6)} - \phi(\boldsymbol{x}) \tag{25}$$

It is thus simple to compare the errors at any location. For example, at the material point (p) in Fig. 5A, the results are

$$\Delta\phi_{coarse}(\boldsymbol{x}) = \frac{15}{32}c^2\left(\phi_{,xx} + \phi_{,yy}\right) \qquad \text{and} \qquad \Delta\phi_{fine}(\boldsymbol{x}) = \frac{3}{32}c^2\left(\phi_{,xx} + \phi_{,yy}\right) \tag{26}$$

In other words, refinement improves the interpolation of a quadratic function by a factor of 5. The specific error depends on position. At all new material points the error is reduced by at least a factor of 7/3.

2.4. Lumped vs. *Unlumped Mass Matrix*

Another source of error and a known cause of energy dissipation [7] is the use of a lumped mass matrix instead of a full mass matrix. The error of using a lumped mass matrix can be approximately evaluated by a method similar to that used above to estimate the benefits of refining the particles. Assume that the i^{th} component of acceleration near node (0) is given exactly by a quadratic expansion

$$\boldsymbol{a}_i = \boldsymbol{a}_i^{(0)} + \boldsymbol{a}_{i,i}^{(0)}\left(\boldsymbol{x}_i - \boldsymbol{x}_i^{(0)}\right) + \frac{1}{2}\boldsymbol{a}_{i,ij}^{(0)}\left(\boldsymbol{x}_i - \boldsymbol{x}_i^{(0)}\right)\left(\boldsymbol{x}_j - \boldsymbol{x}_j^{(0)}\right) \tag{27}$$

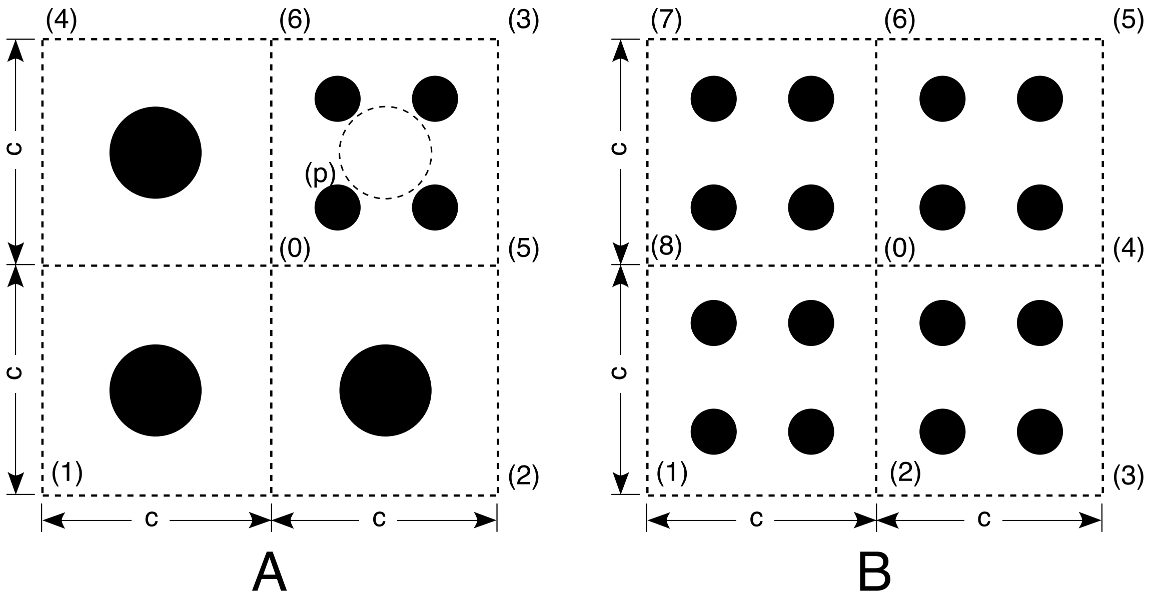

Fig. 5. **A.** Before refinement, the four large particles are in the element with nodes 1, 2, 3, and 4. After refinement, the four points are split into 16 material points. For clarity, only the new material points in the upper-right quadrant are shown. The refined material points are in the element with nodes 0, 5, 3, and 6. **B.** A full mass matrix calculation for the acceleration at node (0) requires mass matrix terms with all neighboring nodes and a sum of all particles in elements that include node (0).

The error in using a lumped mass matrix is the difference between calculating the forces using (10) *vs.* the full mass matrix result in (5). Using the mesh in Fig. 5B, that error is

$$\text{Error} = M_{00}^L \boldsymbol{a}_i^{(0)} - \sum_{j=0}^{8} M_{Oj} \boldsymbol{a}_i^{(j)} \tag{28}$$

$$= \sum_p m_p N_0(\boldsymbol{x}_p) \left(\boldsymbol{a}_i^{(0)} - \sum_{j=0}^{8} N_j(\boldsymbol{x}_p) \boldsymbol{a}_i^{(j)} \right) \tag{29}$$

The only non-zero elements of the full mass matrix are those associated with the nodes that share an element with node (0). Thus the full mass matrix sum only needs to include node (0) and the eight nodes around node (0); the sum over particles only needs to include the 16 particles in the elements near node (0). This error estimation depends on particle positions and thus will change during a simulation. To get a result, we assumed that the particles are at their original positions for which the sums can be explicitly evaluated to give

$$|\text{Error}| = \frac{3c^2}{4} \left(\boldsymbol{a}_{i,xx}^{(0)} + \boldsymbol{a}_{i,yy}^{(0)} \right) \tag{30}$$

As in the refinement calculations, a lumped mass matrix induces no error when the acceleration field is constant or linear. If the variation is quadratic (or higher), there will be errors proportional to the second derivative of the accelerations. The error is also proportional to the area of the elements (c^2). Thus, the use of an adaptive mesh will automatically reduce the error of the lumped mass matrix approximation as well. In the following section, we describe some calculations that tried all combinations of adaptive *vs.* non-adaptive meshes together with lumped *vs.* unlumped mass matrix to see which were needed for accurate calculations and which were most effective in reducing errors.

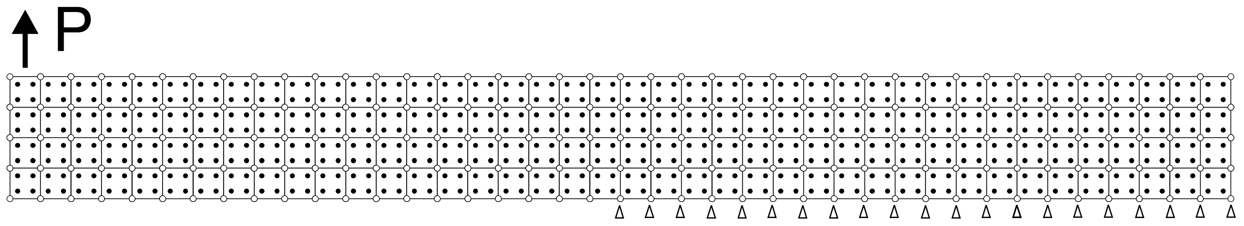

Fig. 6. The initial, unrefined, background grid for the MPM calculations of a double cantilever beam. The mesh is for only half the specimen; the other half is found by symmetry. The specimen is end-loaded with load P. The crack tip is at the first constrained node in the middle of the specimen.

3. Results

To try energy release rate calculations in MPM, we analyzed a double cantilever beam (DCB) specimen. The material points and the initial, unrefined, background grid are shown in Fig. 6. The beam dimensions were 100 mm long with a square cross section 10 mm×10 mm. The crack length was 50 mm long or the crack tip was in the middle of the specimen. The load was applied at the end of the specimen. The displacement of the background grid nodes in the y direction were constrained to zero as indicated. The crack tip is at the left-most constrained node. This grid models only half the specimen; the other half is found by symmetry. The material was assumed to be linear elastic with modulus $E = 70$ GPa and Poisson's ratio of $\nu = 0.3$. A load of 1 N was applied instantaneously at time zero. In a linear elastic material, the stress state and energy release rate would oscillate forever. To eliminate the oscillations, we added a damping force to each particle proportional to the amplitude of the particle's velocity but in the opposite direction. At long times, these damped MPM calculations should converge to static loading of an end-loaded DCB specimen for which all velocities approach zero, the damping term ceases to contribute, and thus the response is linear elastic.

The time step was set sufficiently small as required by the wave speed of the material and the element size in the background grid [10]. For an adaptive analysis in which the smallest element size was a_{min}, we used a time step of

$$\Delta t = \frac{a_{min}}{10c_{rot}} \tag{31}$$

where c_{rot} is the rotational wave speed given by:

$$c_{rot} = \sqrt{\frac{E}{2(1+\nu)\rho}} \tag{32}$$

It may be possible to use larger time steps for the larger elements, but for simplicity, these calculation based the time step on the smallest element size. In adaptive calculations, the time step was adjusted each time smaller elements were created.

We did MPM calculations both with and without adaptive meshes. The grid in Fig. 6 shows the grid at the start of the calculation which uses uniform, square elements. Figures 7–9 shows the crack tip region of the mesh at later stages in the calculation. Each figure is for a different value of R_{crit}. The top half of each figure plots the average value of $R^{(n)}$ for the elements in the plane of the crack. The bottom half of each figure shows the adaptive mesh reflected across the midplane for clarity. The results in Fig. 7 used a large R_{crit} such that there was no adaptation in the grid. This calculation shows that $R^{(n)}$ has a maximum value of about 0.19. Figure 8 shows the result of dropping R_{crit} below this maximum value to $R_{crit} = 0.12$. In this calculation, the elements around the rack tip refine. After one level of refinement,however, all average values of $R^{(n)}$ drop below R_{crit} and adaptation stops. Finally, Fig. 9 show a calculation with a smaller $R_{crit} = 0.03$. The grid now refines several times until all average values of $R_{(n)}$ are below R_{crit}. As R_{crit} decreases, the levels of refinement increase. For all our adaptive mesh calculations described below, we used $R_{crit} = 0.01$.

We next calculated dynamic energy release rate using crack closure. Figure 10 plots the MPM results for $G(t)$ calculated with a regular grid or an adaptive grid and calculated with a lumped mass matrix or a full mass matrix. At long times, the numerical results should approach the exact static result. The exact static

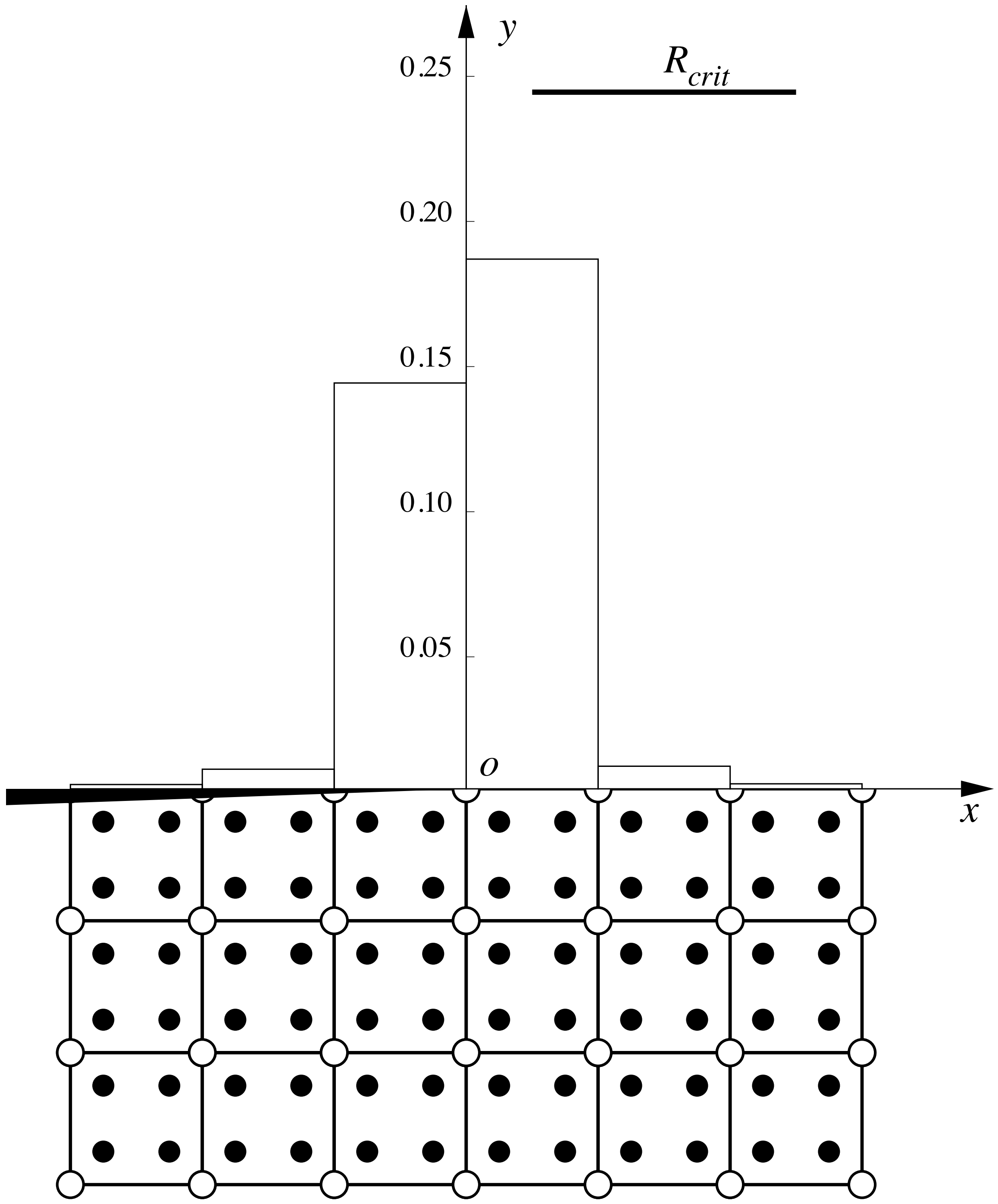

Fig. 7. An adaptive mesh calculation when $R_{crit} = 0.24$. The top half of the plot gives the average value of $R^{(n)}$ in the elements along the crack. The bottom half shows the adapted mesh reflected about the midplane for clarity. In this result there was no adaptation because R_{crit} was large.

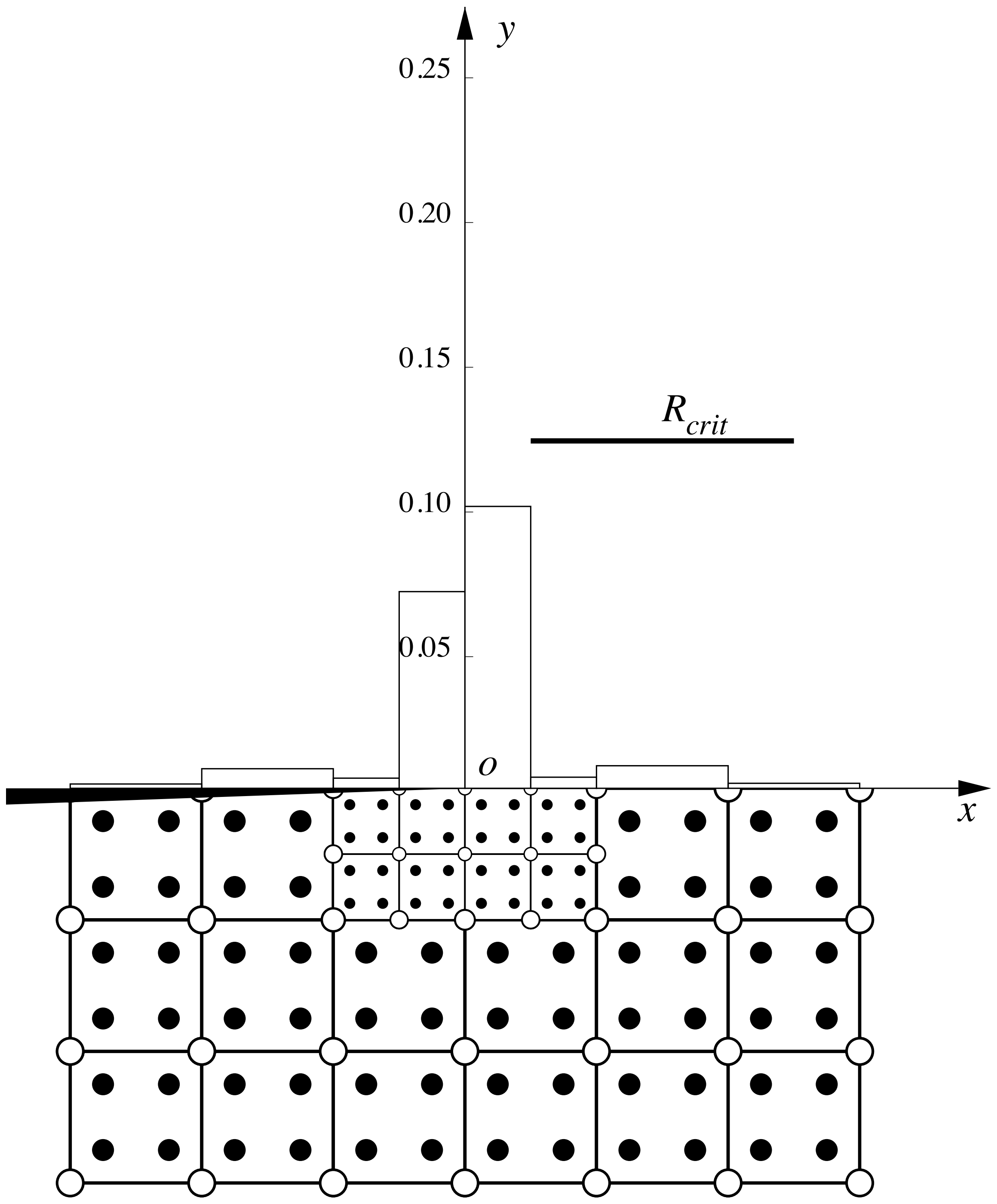

Fig. 8. An adaptive mesh calculation when $R_{crit} = 0.12$. The top half of the plot gives the average value of $R^{(n)}$ in the elements along the crack. The bottom half shows the adapted mesh reflected about the midplane for clarity.

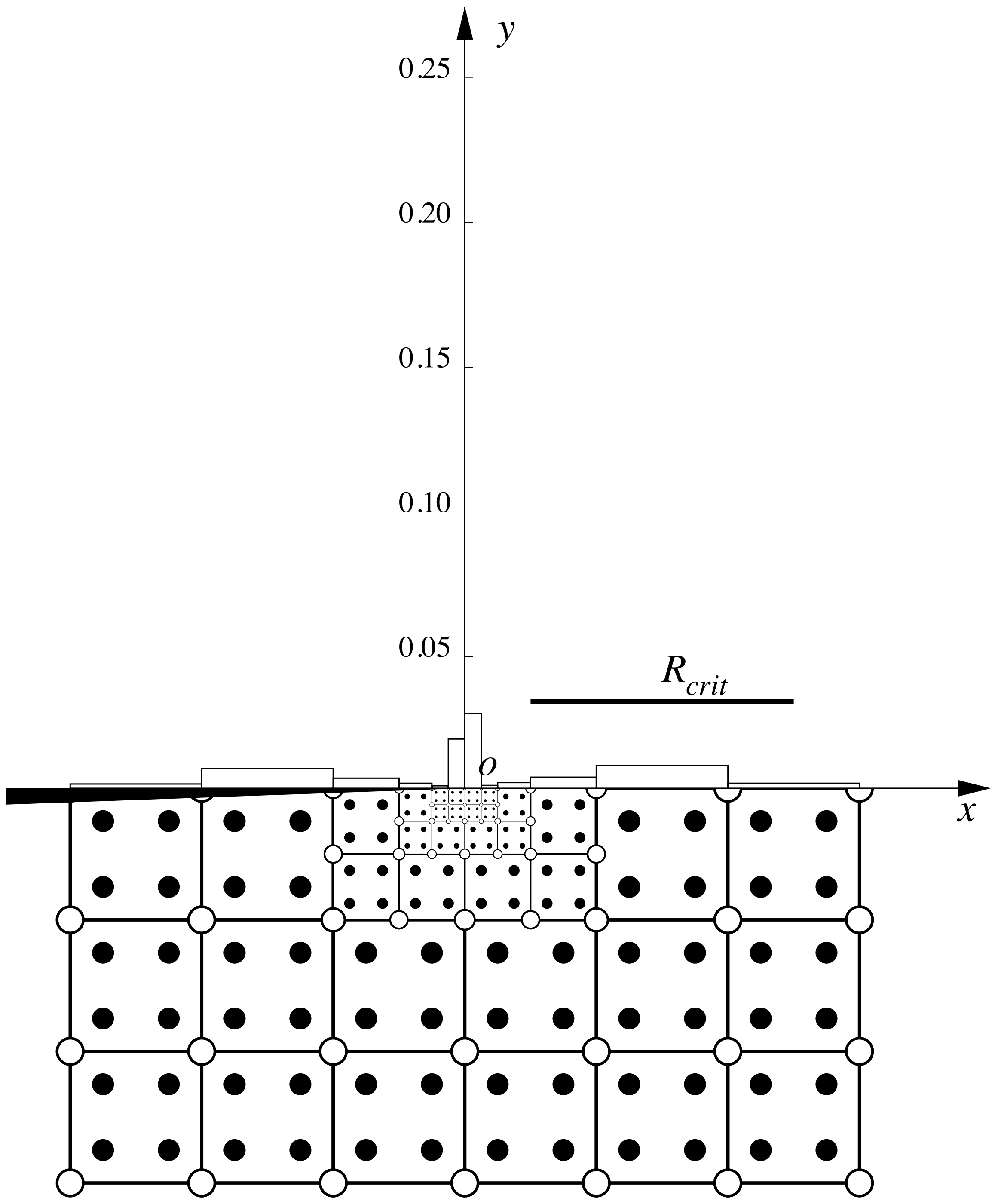

Fig. 9. An adaptive mesh calculation when $R_{crit} = 0.03$. The top half of the plot gives the average value of $R^{(n)}$ in the elements along the crack. The bottom half shows the adapted mesh reflected about the midplane for clarity.

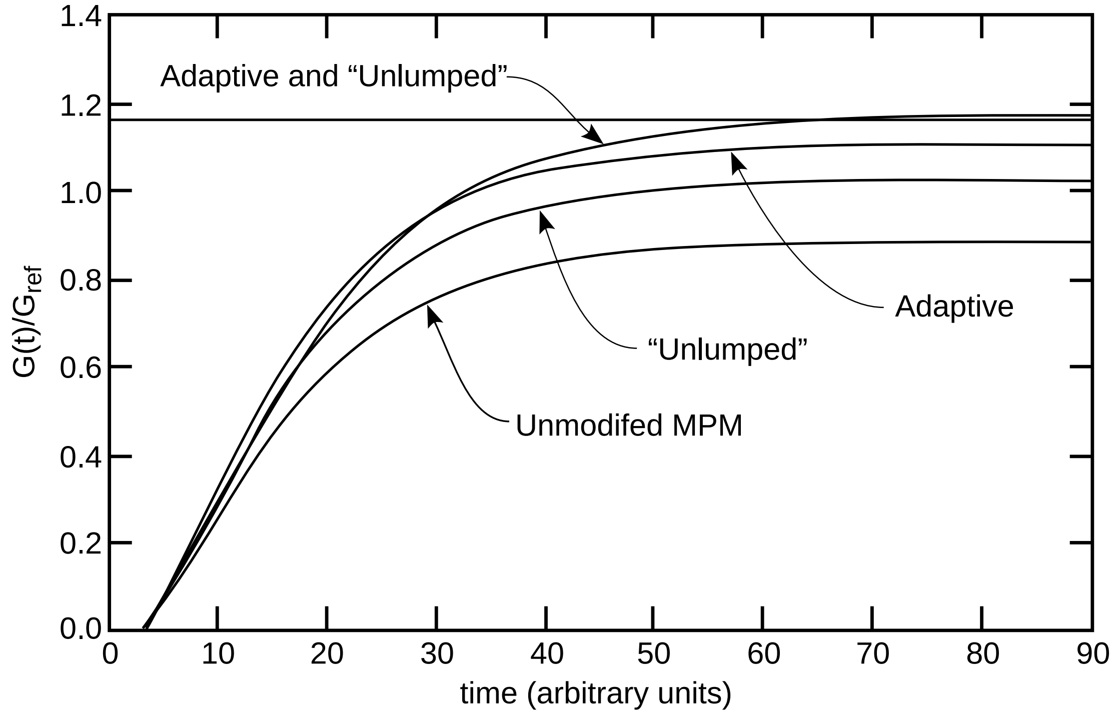

Fig. 10. Dynamic energy release rate calculated by four different methods. The “Unmodifed MPM” and “Unlumped” results used the regular mesh shown in Fig. 6. The two “Adaptive” results started with that mesh but included refinement in the crack tip region during the calculations. The horizontal line is the “exact” result from the beam-on-elastic-foundation model. All results are normalized to a reference energy release rate, G_{ref}, defined as the simple beam theory result for a double cantilever beam.

result can be calculated accurately using a beam-on-an-elastic-foundation model [11]:

$$G_I = \frac{12P^2(a+\chi D)^2}{EB^2D^3} \tag{33}$$

where a is crack length, D is beam depth, and B is beam thickness. The χ term is a factor that corrects simple beam theory to account for crack-root rotation effects on energy release rate [11]. $\chi = 0$ recovers simple beam theory while $\chi = 2/3$ gives a corrected beam theory that is essentially exact for isotropic materials and the geometry analyzed in Fig. 6. This “exact” result is plotted as the horizontal line in Fig. 10 for comparison to the MPM results. Finally all results are normalized to G_{ref} which is defined as the simple beam theory result or the result from (33) with $\chi = 0$.

The curve labeled “Unmodified MPM” gives the results for a regular grid and a “lumped” mass matrix; *i.e.*, the result of MPM methods currently documented in the literature and using the initial grid shown in Fig. 6. This result shows that standard MPM is not accurate for fracture calculations. The static $G(t)$ in this analysis differs from the exact result by 24%. Using either a full mass matrix (but still with a regular grid — see curve labeled “Unlumped”) or our adaptive elements (but still with a lumped mass matrix — see curved labeled “Adaptive”) improves the calculations. Mesh refinement is more effective in improving accuracy then is the full mass matrix. Mesh refinement reduces the error to 5%, while a full mass matrix only reduces the error to 14%. Finally, the combination of adaptive cells and a full mass matrix (see curve labeled ‘Adaptive and “Unlumped”’) leads to extremely accurate calculations (within 1% of the exact result).

These results show that MPM is capable of accurate and explicit fracture mechanics calculations. The most accurate calculations required use of both adaptive elements and the full mass matrix. The adaptive elements improved the accuracy of the calculations near the crack tip while the full mass matrix eliminates or minimizes energy dissipation effects. The use of a full mass matrix, however, might make MPM calculations for large systems with many time steps intractable. With a full mass matrix, a new matrix must be inverted at each time step. Three possible alternatives to the use of a full mass matrix are:

1. The results in Fig. 10 show that the use of adaptive elements is more effective in improving the accuracy of fracture calculations than is the use of a full mass matrix. In other words, it appears to be more important to refine the mesh near the crack tip than to eliminate energy dissipation. Perhaps refining even further, by using a lower value of R_{crit} will lead to sufficiently accurate calculations even with a lumped mass matrix.

2. It might be possible to partially unlump the mass matrix. The mass matrix is a banded, symmetric matrix. If the nodes in the background grid are numbered efficiently, as is commonly done in finite element meshes, the bandwidth of the mass matrix can be minimized. A potential algorithm is then to replace the lumping equation in (9) by a new result that lumps the mass matrix away from the crack tip, but does not lump it the refined elements near the crack tip. The resulting mass matrix would be block diagonal with non-diagonal blocks only in regions of high stress gradients such as near crack tips. Inversion of the partially unlumped mass matrix would only require inversion of the blocks near the crack tip and would likely be much more efficient than inversion of the full mass matrix for the entire structure.

3. Most MPM calculations use 4-noded, linear elements in the background grid with four material points per element. In finite element calculations of DCB specimens, linear elements require high refinement for accurate energy release rate results. In contrast, 8-noded, quadratic elements can achieve accurate energy release rates with far fewer total nodes than when using linear elements. There are two consequences to this observation. First, it might be possible to use quadratic elements in MPM for improved calculations on DCB specimens. Second, the great preference for quadratic elements may be an artifact of the bending geometry which has quadratic displacements. Perhaps the DCB specimen is particularly difficult for linear MPM elements, while many other types of crack geometries will have accurate results even when using a lumped mass matrix.

These demonstration calculations considered a particularly simple problem for which a crack is at a plane of symmetry and the crack propagation direction was between elements in the background calculational grid. Two new challenges arise when one considers propagation of internal cracks in arbitrary directions. First, conventional MPM does not handle internal cracks. The background calculational grid enforces continuity in deformations and velocity, but internal cracks require representations of discontinuous deformation and velocity fields. One solution would be to introduce cracks into the background grid. This method would be difficult to implement in general and would likely lead to MPM calculations that have little or no benefit of analogous finite element methods. An alternative solution is to modify the mass matrix calculation such that particles on one side of a crack do not interact with particles on the other side of the crack unless the crack surfaces contact each other. Although there are finite element methods for dealing with discontinuities embedded in an element [12, 13], our preliminary results with MPM suggest that inclusion of explicit cracks in MPM will require different methods. Possible methods for inclusion of explicit, internal cracks in MPM will be the subject of a future paper.

The second challenge is to allow cracks to propagate in any direction without regard to the grid lines in the background grid. This modification presents no difficulty. All information about stress and displacement are carried along with the particles. For the calculations given here, it was convenient to solve the crack closure problem along the grid lines of the background grid. It should be straight forward to extrapolate the stresses and displacements along any arbitrary direction and therefore evaluate the crack-closure integral for arbitrary crack growth directions. The potential ability of MPM to handle crack growth in arbitrary directions without constraint to the background grid and to adaptively refine as a crack approaches or coarsen after the crack passes suggests that MPM can be developed into a valuable simulation tool for crack propagation simulations.

Acknowledgment

This work was supported by the University of Utah Center for the Simulation of Accidental Fires and

Explosions (C-SAFE), funded by the Department of Energy, Lawrence Livermore National Laboratory, under Subcontract B341493.

References

[1] D. Sulsky, Z. Chen, and H. L. Schreyer, A Particle Method for History-Dependent Materials, *Comput. Methods Appl. Mech. Engrg.* **118** (1994) 179–186.

[2] D. Sulsky, S.-J. Zhou, and H. L. Schreyer, Application of a Particle-in-Cell Method to Solid Mechanics, *Comput. Phys. Commun.* **87** (1995) 236–252.

[3] D. Sulsky and H. K. Schreyer, Axisymnmetric Form of the Material Point Method with Applications to Upsetting and Taylor Impact Problems, *Comput. Methods. Appl. Mech. Engrg.* **139** (1996) 409–429.

[4] S. Zhou, The Numerical Prediction of Material Failure Based on the Material Point Method, Ph.D. Thesis, Department of Mechanical Engineering, University of Mexico, 1998.

[5] J. A. Nairn, Matrix Microcracking in Composites. *Polymer Matrix Composites*, eds., R. Talreja and J.-A. E. Manson, Vol. 2 of Comprehensive Composite Materials, Elsevier Science, 2000, 403–432.

[6] D. L. Flaggs and M. H. Kural, Experimental Determination of the In Situ Transverse Lamina Strength in Graphite/Epoxy Lamaintes, *J. Comp. Mat.* **16** (1982) 103–115.

[7] D. Burgess, D. Sulsky, and J. U. Brackbill, Mass Matrix Formulation of the FLIP Particle–in–Cell Method, *J. Comp. Phys.* **103** (1992) 1–15.

[8] G. R. Irwin, Fracture, *Handbuch der Physik* **6** (1958) 551–590.

[9] E. F. Rybicki and M. F. Kanninen, A Finite Element Calculation of Stress Intensity Factors By a Modified Crack Closure Integral, *Eng. Fract. Mech.* **9** (1977) 931–938.

[10] D. Sulsky and J. U. Brackbill, A Numerical Method for Suspension Flow, *J. Comput. Phys.* **96** (1991) 339–368.

[11] M. F. Kanninen, An Augmented Double Cantilever Beam Model for Studying Crack Propagation and Arrest, *Int. J. Fract.* **9** (1973) 83–92.

[12] G. N. Wells and L. J. Sluys, Application of Embedded Discontinuities for Softening Solids, *Eng. Fract. Mech.* **65** (2000) 263–281.

[13] G. N. Wells and L. J. Sluys, Three-Dimensional Embedded Discontinuity Model for Brittle Fracture, *Int. J. Solids and Structures* **38** (2001) 897–913.