Modeling Nanoindentation using the Material Point Method

Chad C. Hammerquist and John A. Nairn
Wood Science and Engineering, Oregon State University,
Corvallis, OR 97330, USA

1. Supplemental Material

The published paper associated with these notes concentrated on material science aspects of nanoindentation by using virtual, numerical experiments to investigate best approaches to extracting effective modulus from nanoindentation experiments. The simulations all used the material point (MPM) method and this work included two relatively minor enhancements to MPM specifically needed to optimize its use for nanoindentation simulations. Because those two MPM enhancements have not been previously published, this supplemental material is being provided to document them.

2. Contact Modeling

Contact detection in MPM is improved by including a calculation or particle edge displacements. In other words, two material domains are only modeled as in contact when $d_{i,b} - d_{i,a} \le 0$ where $d_{i,j}$ is the distance along the normal vector from the edge of material j to node i [1]. The challenge in implementing this criterion is that edge locations are typically not tracked in MPM. A method for that calculation is needed.

The calculation of $d_{i,j}$ is started by extrapolating material point positions to the grid using usual mass-weight MPM extrapolations :

$$\boldsymbol{x}_{i,j} = \frac{1}{m_{i,j}} \sum_{p\in j} m_p \boldsymbol{x}_p S_{ip} \qquad \text{where} \qquad m_{i,j} = \sum_{p\in j} m_p S_{ip} \tag{1}$$

Here m_p and $\boldsymbol{x}_p$ are particle mass and location S_{ip} is MPM shape function for particle p and node i, and the sums are for all material points of material j. An "apparent" distance from extrapolated particle position ($\boldsymbol{x}_{i,j}$) to node i (at $\boldsymbol{x}_i$) along normal vector $\hat{\boldsymbol{n}}_i$ for interface between two contacting materials is:

$$d^{(ext)}_{i,j} = (\boldsymbol{x}_{i,j} - \boldsymbol{x}_i)\cdot \hat{\boldsymbol{n}}_i \tag{2}$$

But this distance is not the "actual" distance needed for contact calculations. To determine the "actual" distance, the value of $d^{(ext)}_{i,j}$ needs to be corrected. The process is equivalent to finding the function $d_{i,j}(d^{(ext)}_{i,j})$ for actual distance as a function of $d^{(ext)}_{i,j}$.

The required calculations are done by using MPM shape functions to find $d^{(ext)}_{i,j}$ as function of actual distance and then inverting the results. Imagine a 1D grid with material a approaching node i from the left and material b approaching from the right (see Fig. S1). From Eqs. (1) and (2) with node i at the origin:

$$d^{(ext)}_{i,a} = \frac{\sum_{k=1}^{n} m_p S_{ip}\big(d_{i,a} - r_p(2k-1)\big)\big(d_{i,a} - r_p(2k-1)\big)}{\sum_{i=1}^{n} m_p S_{ip}\big(d_{i,a} - r_p(2k-1)\big)} \tag{3}$$

The similar expression for material b is

$$d^{(ext)}_{i,b} = \frac{\sum_{k=1}^{n} m_p S_{ip}\big(d_{i,b} + r_p(2k-1)\big)\big(d_{i,b} + r_p(2k-1)\big)}{\sum_{i=1}^{n} m_p S_{ip}\big(d_{i,b} + r_p(2k-1)\big)} \tag{4}$$

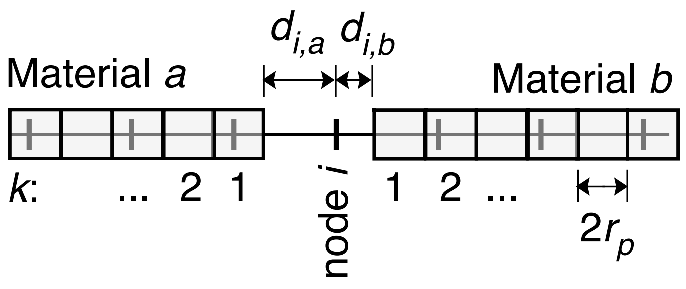

Figure S1: *A 1D MPM grid with materials a and b approaching node i from left and right, respectively. The indicated lengths $d_{i,a}$ and $d_{i,b}$ are actually distances from edges of materials a and b to node i. Each particle has radius r_p. Particles for each material type are numbered $k = 1, 2, \ldots$ starting from edge of the material near node i.*

Here r_p is particle radius, and $S_{ip}(x)$ is shape function for particle p on node i when midpoint of particle is at x and origin (or $x = 0$) is at node i. These calculations were done with GIMP shape functions that integrate grid shape functions over the current particle domain. For these undeformed particles, the explicit shape function for node i at $x = 0$ is [2]:

$$S_{ip}(x) = \begin{cases} \frac{2r_p(1-r_p)-x^2}{2r_p} & |x| < r_p \\ 1-|x| & r_p < |x| < 1-r_p \\ \frac{(1+r_p-|x|)^2}{4r_p} & 1-r_p < |x| < 1+r_p \\ 0 & |x| > 1+r_p \end{cases} \tag{5}$$

This evaluation of S_{ip} applies only to $r_p \le \Delta x/2$ where Δx is cell spacing or to MPM model with one or more particles per cell. Most MPM simulations use 2 particles per cell with $r_p = \Delta x/4$. The calculations using CPDI shape functions would not change much for these undeformed particles. The summations for $d_{i,j}^{(ext)}$ include all particles with non-zero shape functions. The resulting functions (inverted to plot actual distance as a function of extrapolated distance) are the dashed lines in Fig. S2. As expected, the extrapolated distance differs from the desired edge distance $d_{i,j}$. For example, $d_{i,a}^{(ext)}$ will always be negative as the edge of material a moves from node $i-1$ to node $i+1$, while actual $d_{i,a}$ will vary from $-\Delta x$ to $+\Delta x$. Similarly, the extrapolated distance for material b is always positive.

The imposition of contact detection in MPM calculations requires definition of a function $d_{i,j}(d_{i,j}^{(ext)})$ to calculate actual edge distance from extrapolated "apparent" distance. Reference [1] proposed a simple constant correction such that

$$d_{i,b} - d_{i,a} \approx (\boldsymbol{x}_{i,b} - \boldsymbol{x}_{i,b}) \cdot \hat{\boldsymbol{n}}_i - \delta_c \Delta x \tag{6}$$

where δ_c is a constant offset (in units of cell size) used to approximate actual separation. By analysis and test simulations, a value of $\delta_c = 0.8$ works well for MPM simulations with two particles per cell (or $r_p = \Delta x/4$). This approach is equivalent to assuming the distance mapping functions are:

$$d_{i,j}(d_{i,j}^{(ext)}) = \begin{cases} d_{i,a}^{(ext)} + 0.4\Delta x & \text{left or } j = a \\ d_{i,b}^{(ext)} - 0.4\Delta x & \text{right or } j = b \end{cases} \tag{7}$$

These "linear" functions are plotted in Fig. S2 and can be described as best fit to a line with slope equal to one or to $d_{i,j}(d_{i,j}^{(ext)}) = d_{i,a}^{(ext)} + \delta_c \Delta x/2$ where δ_c is the only fitting parameter. To improve the material

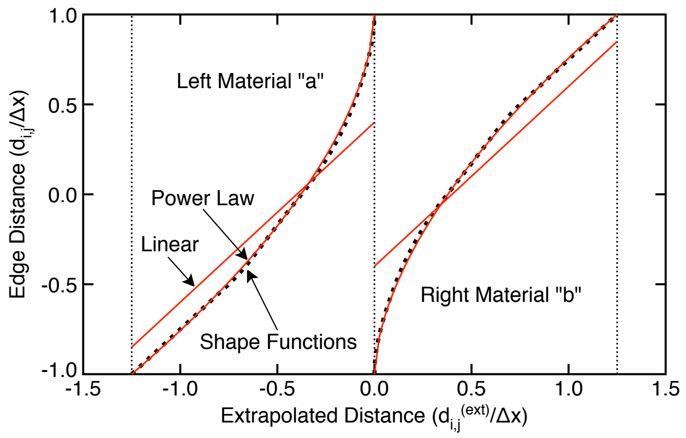

Figure S2: *Calculation of $d_{i,j}$ as a function of $d_{i,j}^{(ext)}$ for "left" and "right" materials approaching node i. The black dotted curves show explicit calculations using* `GIMP` *shape functions. The red solid lines show the linear functions (Eq. (7)) and the power-law functions (Eq. (8)) to approximate the shape function calculations.*

separation calculation, we replaced the constant correction with new mapping functions given by:

$$\frac{d_{i,j}(d_{i,j}^{(ext)})}{\Delta x} = \begin{cases} 1-2\left(\frac{-d_{i,a}^{(ext)}}{1.25\Delta x}\right)^{0.58} & \text{left or } j=a \\ -1+2\left(\frac{d_{i,b}^{(ext)}}{1.25\Delta x}\right)^{0.58} & \text{right or } j=b \end{cases} \tag{8}$$

These two mapping functions are compared to shape function calculations in Fig. S2. Although the "linear“ correction generally works well, it was easy to implement non-linear corrections instead. For nanoindentation simulations in this paper, all of which depend strongly on contact mechanics, we used the more accurate power law functions. This approach gave small, but noticeable, improvements in the results.

Note that the above mapping functions depend on particle r_p or different linear offsets (δ_c) and different non-linear functions would be needed for MPM simulations using different sizes of particles. We used the above approach that found δ_c when $r_p = \Delta x/4$ to investigate how it changes with differently-sized particles. For one particle per cell or $r_p = \Delta x/2$, $\delta_c = 1.07$ is the recommend offset. For two or more particles per cell $r_p < \Delta x/4$, the recommended δ_c depends only weakly on particle size decreasing from $\delta_c = 0.8$ for two particles per cell to $\delta_c = 0.72$ for six particle per cell. Because of these findings, an MPM simulation that varies size of particles with all $r_p < \Delta x/4$ could likely detect contact well with a single value for δ_c. We did not investigate changes in the non-linear functions needed for more advanced contact calculations at different particle sizes.

2.1. MPM Tartan Grid

For nano-indentation simulations, we used a grid scheme termed a "tartan" grid as illustrated in Fig. S3. In a tartan grid, one or more "regions of interest" are modeled with a high-resolution, regular grid with equally-sized elements. For the nanoindentation problem, the one region of interest under the indenter was modeled with 100 nm cells determined above as needed for convergence. Outside regions of interest, the grid cell sizes were allowed to increase, thus forming a tartan-like pattern.

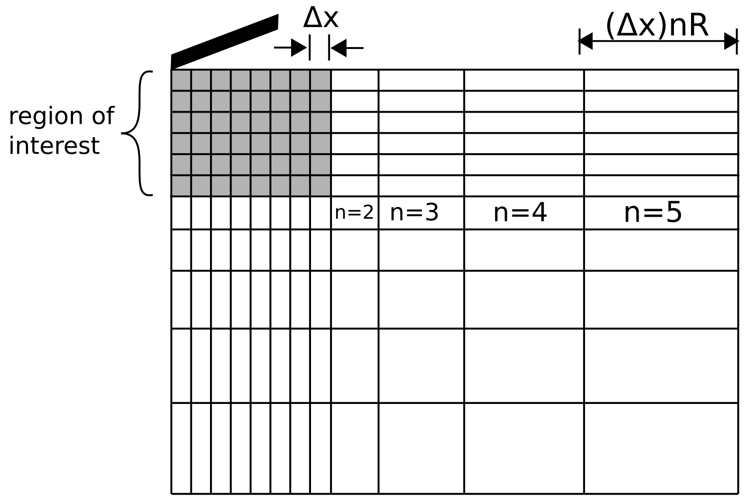

Figure S3: *An MPM tartan grid with orthogonal background cells for nanoindentation simulations. The "region of interest" has a regular grid with equal size elements. The grid cell size increases linearly with distance from the region of interest.*

Note that a tartan grid maintains orthogonal grid lines. This type of grid greatly simplifies implementation, especially when using CPDI shape functions [3] (see text of paper). To simplify contact calculations, which all occur in the region of interest, the cells within the region of interest are all the same size. Outside the region of interest, the cell size increases as a function of distance from the region of interest. We implemented two cell scaling methods. The first is a linear scaling where

$$\Delta x_n = nR\Delta x \tag{9}$$

where n is the number of cells away from the region of interest, R is a specified ratio, and Δx is the constant cell size in the region of interest (see Fig. S3). Alternatively, the cell size can increase geometrically such that

$$\Delta x_n = R^n \Delta x \tag{10}$$

where R is now ratio of cell size for element n to size of the previous element. For this simulations, the linear method worked well and converged simulations used $R = 2$ and $\Delta x = 100$ nm.

A tartan grid was particularly effective or nanoindentation simulations. Virtually the complex stress states occur under the indenter and that region is an area of interest with a regular grid. The stresses outside that are vary more slowly and are easily handled with large particles. Figure S4 shows compression force *vs.* time for the same problem done with a regular grid (black line) and a tartan grid (red line). The results are essentially superposable.

A tartan grid can have general applicability in MPM codes (although may not always be as effective as it is for nanoindentation simulations). Furthermore, a more complete implementation of tartan grid methods would required some additional details that were not needed for nanoindentation simulations:

1. Shape functions: what ever shape functions are used, they have to be capable of accounting for variable element sizes around any given node and variable size particles. In particular, using of GIMP shape functions would require re-evaluation of many standard functions used in current MPM codes. Use of CPDI shapes functions (as done here) needs no changes except for code to be able to find corners of particles within a grid with variably-sized particles and cells.

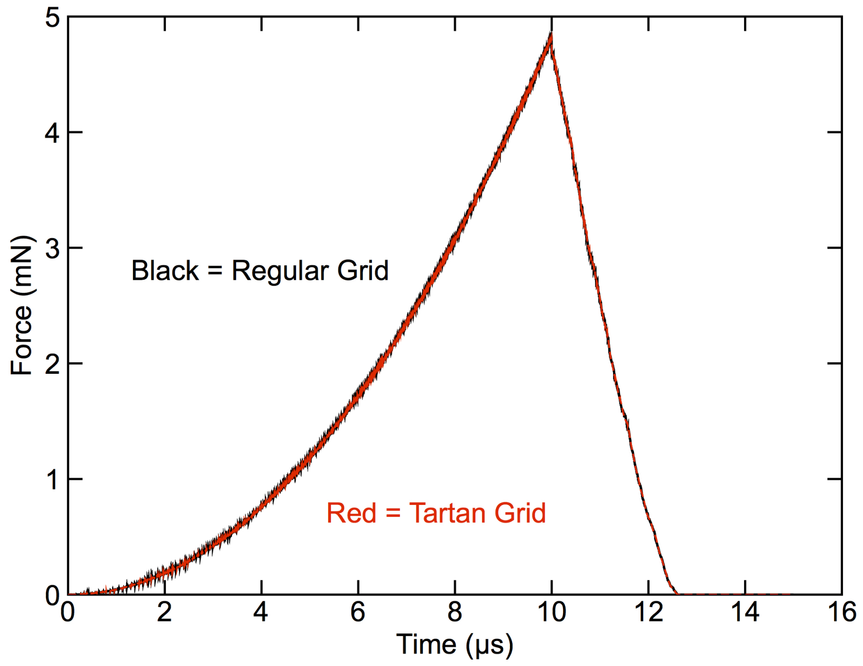

Figure S4: *Nanoindentation force as a function of time during loading and unload as calculated using a regular grid (black line) and a tartan grid (red line).*

2. Contact calculations: contact calculations depend on particle size and as mentioned above, contact detection depends on particle size as well. If contact occurs outside areas of interest, the use of tartan grids needs customization to account for variable element and particle sizes. For nanoindentation simulations, all contact was within the regular grid, area of interest.

3. Particle integrations: Sometimes MPM calculations look at integrations over particles such as average particle stress or integration inside a J-integral contour to account for dynamic stress states [4]. These integrations need to account for variable particle sizes. For example, the average stress should be found from a volume-weighted average stress.

References

1. J. A. Nairn: Modeling imperfect interfaces in the material point method using multimaterial methods. *Comput. Model. Eng. Sci*. **1**, 1–15 (2013).

2. S. G. Bardenhagen and E. M. Kober: The generalized interpolation material point method. *Computer Modeling in Engineering & Sciences*. **5**, 477–496 (2004).

3. A. Sadeghirad, R. M. Brannon, and J. Burghardt: A convected particle domain interpolation technique to extend applicability of the material point method for problems involving massive deformations. *International Journal for Numerical Methods in Engineering*. **86**, 1435–1456 (2011).

4. Y. Guo and J. A. Nairn: Calculation of j-integral and stress intensity factors using the material point method. *Computer Modeling in Engineering & Sciences*. **6**, 295–308 (2004).