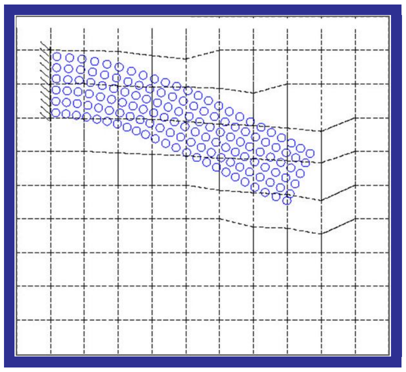
A Primer on the Material Point Method (MPM)

Rebecca Brannon

2013-09-05



ABSTRACT: This manuscript is a beginner's tutorial on the Material Point Method (MPM), which can be viewed as an extension of an updated-Lagrangian finite-element method (FEM) to better accommodate very complicated geometries (like intricate porous micro-structures), as well as extremely large deformations (as seen in penetration) without requiring any remapping or advection schemes for a material model's internal variables. The MPM further provides, at no additional cost, an automatic no-slip/no-stick material contact at locations not known in advance (as in compression of a porous microstructure or sloshing of a fluid). This primer includes reviews of traditional linear FEM in 1-D, transient nonlinear FEM in 1-D and 2-D. Following each review of traditional FEM, the revisions in the algorithm required to solve the same problem using MPM are explained.

Chapter 1

A Primer on the Material Point Method (MPM)

The goal of this tutorial on the Material Point Method (MPM) is to show you how to write your own MPM code, assuming that you already have a finite-element method (FEM) code available to modify (or at least you have sufficient FEM experience to write one as needed). This tutorial does not explain syntax of any particular MPM code, nor does it re-teach the finite-element method. As a tutorial, this manuscript naturally lacks detailed discourse on the rigorous mathematical foundations of the MPM, but citations are provided pointing to appropriate archival literature [**?**]. In a sequence of increasingly complicated example problems, a conventional FEM solution is summarized for each problem, thus forming a starting point from which revisions are then applied to convert an FEM code to an MPM code. Each of these case-study problems finishes with and algorithm and/or pseudo-code for computer implementation, along with at least one verification test problem.

1.1 The classic starting point: linear quasistatic 1-D bar problem

This section provides a "learn by example" introduction to the Material Point Method (MPM) by showing how it is used to solve the classical quasistatic uniaxial bar equation,

$$\begin{aligned} &\frac{\mathrm{d}}{\mathrm{d}x}\left[E(x)A(x)\frac{\mathrm{d}u(x)}{\mathrm{d}x}\right]+f(x)=0 && \text{on domain } \Omega \text{ defined by } 0<x<L \\ &\qquad\qquad\text{subject to Robin BCs:} && \alpha_0\mathcal{F}(0)+\beta_0 u(0)=\gamma_0 \\ &\qquad\qquad\qquad\qquad\qquad\text{and} && \alpha_L\mathcal{F}(L)+\beta_L u(L)=\gamma_L \end{aligned} \tag{1.1}$$

Here, $E(x)$ is the bar stiffness,[1] $A(x)$ is the cross-sectional area of the bar, $f(x)$ is the spatially varying distributed load on the bar[2], and $u(x)$ is the displacement. In the Robin boundary conditions, $u(0)$ and $u(L)$ are, respectively, the displacements at the left and right ends of the bar, while $\mathcal{F}(0)$ and $\mathcal{F}(L)$ are the forces at those boundaries. The Robin parameters α_0, β_0, γ_0, α_L, β_L, and γ_L, are user-prescribed constants.[3] The force $\mathcal{F}$ is defined to be positive when pointing to the right in the direction of increasing x. By definition, force equals stress times area, where stress is positive in tension. For a linear-elastic bar, stress σ is related to strain ε by Hooke's law, $\sigma = E\varepsilon$. For small strains (needed for geometric linearity), the strain is related to displacement by $\varepsilon = u'(x)$. Therefore, the boundary forces are related to the first derivative of the displacement by

$$\mathcal{F}(0) := -E(0)A(0)u'(0) \qquad \text{and} \qquad \mathcal{F}(L) := E(L)A(L)u'(L), \tag{1.2}$$

The negative in the first equation is present because a positive $u'(0)$ corresponds to tension at the left end of the bar, and hence a left-pointing force.

To work towards an MPM solution to this problem, this section is broken into four subsections. First, the governing equation is recast as an integral equation to lessen the differentiability requirements of approximate solutions. Next, the traditional FEM solution is provided, after which it is converted to an MPM formulation through only minor adjustments. Our discussion of this elementary problem is finished with a step-by-step MPM algorithm. Subsequent sections will slowly build up from this simple 1-D bar problem to allow transients (such as acoustic waves) in the linear-elastic case, material nonlinearity in the absence of transients, and then both material nonlinearity and transient waves. Complications of geometric nonlinearity (*i.e.,* large stretching of the material) are not discussed in the 1-D case studies; large-deformation is instead delayed until we generalize the small-strain concepts to 2-D and 3-D problems.

1.1.1 The weak form of the governing equation

The ODE in Eq. (1.1) is called the **strong form** of the governing equation, which requires the displacement function [4] $u(x)$ to have no jumps in slope (no cusps) in order to be twice differentiable. This section recasts the governing equation as an integral, which lessens the continuity requirements on solutions. To obtain the integral formulation, the strong form is first multiplied by an arbitrary weight function[5] $w(x)$, and then

[1]The modulus E is commonly thought to be equal to Young's modulus, which is true for uniaxial stress (where the lateral stress is zero), but it is actually a different modulus, called the constrained modulus, if the loading is uniaxial strain (where the lateral strain is zero)

[2]The distributed load has dimensions of force per unit length. Point loads are accommodated by allowing the distributed load to include Dirac delta contributions.

[3]The Robin boundary conditions provide a constraint between force and displacement of the form $\alpha\mathcal{F} + \beta u = \gamma$, which is convenient because this generalized boundary condition includes the most common boundary conditions as special cases. Specifically, prescribed displacement is specified by setting the $\alpha = 0$, $\beta = 1$, and γ equal to the prescribed displacement. Prescribed force is applied by setting $\alpha = 1$, $\beta = 0$, and γ equal to the precribed force. A spring BC is defined by setting $\alpha = 1$ and β equal to the negative of the spring constant.

[4]The displacement function is often called called the **trial function**, which is an abstract term that reminds us that this same ODE applies to many more physical applications than just motion of an elastic bar.

[5]In variational formulations, the function $w(x)$ is called the **test function**, and it is interpreted physically as a small perturbation of the displacement, often denoted $\delta u(x)$. Variational formulations of the finite-element method often state that the test function $w(x)$ is required to vanish on the parts of the boundary where the trial function u is specified, but this restriction is NOT actually needed unless you are seeking a formula for the so-called energy potential to be minimized. Not only does an energy potential not always exist (it does for this problem), but it isn't needed to obtain the finite-element solution. We therefore place no restrictions whatsoever on the test function $w(x)$ at places where displacement is prescribed.

integrated over the domain from $x = 0$ to $x = L$:

$$\int_0^L w(x)\frac{\mathrm{d}}{\mathrm{d}x}\left[E(x)A(x)\frac{\mathrm{d}u(x)}{\mathrm{d}x}\right]\mathrm{d}x + \int_0^L w(x)f(x)\mathrm{d}x = 0 \qquad \forall w(x) \tag{1.3}$$

The first term is integrated by parts[6] to give the so-called **weak form** of the governing equation:

$$E(x)A(x)w(x)u'(x)\big|_0^L - \int_0^L E(x)A(x)w'(x)u'(x)\mathrm{d}x + \int_0^L w(x)f(x)\mathrm{d}x = 0 \qquad \forall w(x) \tag{1.4}$$

Using the definition of boundary forces defined in Eq. (1.2), this may be written

WEAK FORM OF THE BAR PROBLEM

$$\boxed{\int_0^L E(x)A(x)w'(x)u'(x)\mathrm{d}x = w(0)\mathcal{F}(0) + w(L)\mathcal{F}(L) + \int_0^L w(x)f(x)\mathrm{d}x \qquad \forall w(x)} \tag{1.5}$$

This is called the "weak" form of the governing equation because the trial function, displacement $u(x)$, now has only one derivative on it. This will allow us to seek approximate solutions that are merely continuous rather than needing to find functions that have continuous slopes.

1.1.2 Traditional FEM formulation of the bar problem

A traditional FEM solution to any 1-D problem begins by introducing a set of nodes, which are points distributed along the length of the bar (assumed to be numbered sequentially from 1 to ν_n, where ν_n denotes the total number of nodes). The nodes are furthermore used to define elements on the finite-element mesh, as explained in any introductory FEM textbook. The approximate FEM solution for the displacment field is

$$\tilde{u}(x) := \sum_{j=1}^{\nu_\mathrm{n}} \tilde{u}_j N_j(x) \tag{1.6}$$

where $\{N_1(x), N_2(x), \ldots, N_{\nu_\mathrm{n}}\}$ are the nodal basis functions (such as the "tent" functions constructed from linear shape functions), and $\{\tilde{u}_1, \tilde{u}_1, \ldots, \tilde{u}_{\nu_\mathrm{n}}\}$ are the nodal displacements.[7]

The weight function in the weak formulation is similarly expanded as

$$w(x) := \sum_{i=1}^{\nu_\mathrm{n}} w_i N_i(x) \tag{1.7}$$

In the finite-element method, the nodal basis functions are required to satisfy the following properties:

Kronecker property: $$N_i(x_j) = \delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \tag{1.8a}$$

Partition of unity: $$\sum_{i=1}^{\nu_\mathrm{n}} N_i(x) = 1 \qquad \forall x \tag{1.8b}$$

Linear completeness: $$\sum_{i=1}^{\nu_\mathrm{n}} x_i N_i(x) = x \qquad \forall x \tag{1.8c}$$

FEM compact support: $$N_i(x) = 0 \qquad \forall x \text{ falling in an element not including node } i \tag{1.8d}$$

Weak-form integrability: $$N_i(x) \text{ must be continuous} \tag{1.8e}$$

[6]Namely, $\int_0^L u\frac{\mathrm{d}v}{\mathrm{d}x}dx = uv\big|_0^L - \int_0^L u\frac{\mathrm{d}v}{\mathrm{d}x}dx$. In this case the *entire* expression in the brackets, $\left[E(x)A(x)\frac{\mathrm{d}u(x)}{\mathrm{d}x}\right]$ is taken as the quantity "v", and $w(x)$ is taken as "u".

[7]The reader is presumed to be familiar enough with the finite-element method that these statements require no detailed explanations or clarification.

As explained in any good introductory FEM textbook, the last condition imposes the requirement implied from the weak formulation that we must seek solutions that are continuous.

Let $\hat{\boldsymbol{N}}(x)$ denote a $\nu_{\rm n} \times 1$ column array of the nodal basis functions, and let $\hat{\boldsymbol{G}}$ be the array of gradients of the these functions. Similarly, let $\hat{\boldsymbol{u}}$ and $\hat{\boldsymbol{w}}$ be $\nu_{\rm n}$ $\times 1$ column arrays of nodal displacements and weights:

$$\hat{\boldsymbol{N}}(x) := \begin{bmatrix} N_1(x) \\ N_2(x) \\ \vdots \\ N_{\nu_{\rm n}}(x) \end{bmatrix} \qquad \hat{\boldsymbol{G}}(x) := \begin{bmatrix} N'_1(x) \\ N'_2(x) \\ \vdots \\ N'_{\nu_{\rm n}}(x) \end{bmatrix} \qquad \hat{\boldsymbol{u}} := \begin{bmatrix} \tilde{u}_1 \\ \tilde{u}_2 \\ \vdots \\ \tilde{u}_{\nu_{\rm n}} \end{bmatrix} \qquad \hat{\boldsymbol{w}} := \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_{\nu_{\rm n}} \end{bmatrix} \tag{1.9}$$

Then Eqs. (1.6) and (1.7) may be written in matrix form as[8]

$$\tilde{u}(x) = \underbrace{\underbrace{\hat{\boldsymbol{N}}^T(x)}_{1\times\nu_{\rm n}} \underbrace{\hat{\boldsymbol{u}}}_{\nu_{\rm n}\times 1}}_{1\times 1} \qquad \text{and} \qquad w(x) = \underbrace{\underbrace{\hat{\boldsymbol{w}}^T}_{1\times\nu_{\rm n}} \underbrace{\hat{\boldsymbol{N}}(x)}_{\nu_{\rm n}\times 1}}_{1\times 1} \tag{1.10}$$

The superscript "T" denotes the transpose, which transforms a column array, reversing the matrix dimensions as indicated. The final result is a 1×1 matrix, which is simply a single number. Let's also define a "reaction array" $\hat{\boldsymbol{\mathfrak{F}}} = \hat{\boldsymbol{N}}(0)\mathcal{F}(0) + \hat{\boldsymbol{N}}(L)\mathcal{F}(L)$. When the Kronecker property, Eq. (1.8a), is imposed, this array is zero in all but the first and last componentsas follows:

$$\hat{\boldsymbol{\mathfrak{F}}} := \begin{bmatrix} \mathcal{F}(0) \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ \mathcal{F}(L) \end{bmatrix} \qquad \text{which allows writing} \qquad w(0)\mathcal{F}(0) + w(L)\mathcal{F}(L) = \hat{\boldsymbol{w}}^T \hat{\boldsymbol{\mathfrak{F}}} \tag{1.11}$$

Substituting the above two equations into Eq. (1.5) gives

$$\int_0^L E(x)A(x) \underbrace{\underbrace{\hat{\boldsymbol{w}}^T}_{1\times\nu_{\rm n}} \underbrace{\underbrace{\hat{\boldsymbol{G}}(x)}_{\nu_{\rm n}\times 1} \underbrace{\hat{\boldsymbol{G}}^T(x)}_{1\times\nu_{\rm n}}}_{\nu_{\rm n}\times\nu_{\rm n}} \underbrace{\hat{\boldsymbol{u}}}_{\nu_{\rm n}\times 1}}_{1\times 1} \mathrm{d}x = \underbrace{\underbrace{\hat{\boldsymbol{w}}^T}_{1\times\nu_{\rm n}} \underbrace{\hat{\boldsymbol{\mathfrak{F}}}}_{\nu_{\rm n}\times 1}}_{1\times 1} + \int_0^L \underbrace{\underbrace{\hat{\boldsymbol{w}}^T}_{1\times\nu_{\rm n}} \underbrace{\hat{\boldsymbol{N}}(x)}_{\nu_{\rm n}\times 1}}_{1\times 1} f(x)\mathrm{d}x \qquad \forall \hat{\boldsymbol{w}} \tag{1.12}$$

where (recall) the array $\hat{\boldsymbol{G}}(x)$ contains the shape function gradients. Since this must hold $\forall\hat{\boldsymbol{w}}$, the premultiplication by $\hat{\boldsymbol{w}}^T$ may be removed. Also, recognizing that $\hat{\boldsymbol{u}}$ does not vary with x, it may be moved outside of the first integral, giving

$$\hat{\hat{\boldsymbol{K}}}\hat{\boldsymbol{u}} = \hat{\boldsymbol{\mathfrak{F}}} + \hat{\boldsymbol{f}} \tag{1.13}$$

where

$$\boxed{\textbf{Global Stiffness:} \quad \underbrace{\hat{\hat{\boldsymbol{K}}}}_{\nu_{\rm n}\times\nu_{\rm n}} := \int_0^L E(x)A(x) \underbrace{\underbrace{\hat{\boldsymbol{G}}(x)}_{\nu_{\rm n}\times 1} \underbrace{\hat{\boldsymbol{G}}^T(x)}_{1\times\nu_{\rm n}}}_{\nu_{\rm n}\times\nu_{\rm n}} \mathrm{d}x} \tag{1.14}$$

[8]Commutativity of scalar multiplication allows us to put the nodal basis array either on the left (with a transpose) or on the right (without transpose) as done here. In anticipation of upcoming parts of the analysis, we find it convenient to have the displacement array placed on the right with the weight array on the left.

and

$$\boxed{\textbf{Body force array:}\quad \underbrace{\hat{\boldsymbol{f}}}_{\nu_\mathrm{n}\times 1} := \int_0^L f(x)\,\underbrace{\hat{\boldsymbol{N}}(x)}_{\nu_\mathrm{n}\times 1}\,\mathrm{d}x} \tag{1.15}$$

Counting unknowns and equations to confirm solvability

Equation (1.13) represents a set of ν_n equations involving $\nu_\mathrm{n}+2$ unknowns:

$$\textbf{A total of } \nu_\mathrm{n}+2 \textbf{ unknowns}: \qquad u_1, u_2, \ldots, u_{\nu_\mathrm{n}}, \quad \mathcal{F}(0), \mathcal{F}(L) \tag{1.16}$$

Not only are the nodal displacements unknown, so are the boundary forces, $\mathcal{F}(0)$ and $\mathcal{F}(L)$, within $\hat{\boldsymbol{\mathcal{F}}}$. To achieve a solvable system, Eq. (1.13) must be supplemented with the two boundary conditions in Eq. (1.1). Specifically, again using the Kronecker property to assert that $u(0) = u_1$ and $u(L) = u_{\nu_\mathrm{n}}$,

$$\begin{aligned} \alpha_0\mathcal{F}(0) + \beta_0 u_1 &= \gamma_0 \\ \alpha_L\mathcal{F}(L) + \beta_L u_{\nu_\mathrm{n}} &= \gamma_L \end{aligned} \tag{1.17}$$

Together, Eqs. (1.13) and (1.17) form a linear system of $\nu_\mathrm{n}+2$ equations solvable for the $\nu_\mathrm{n}+2$ unknowns! Specific methods of solution, covered in an elementary FEM course, are not discussed in this manuscript.

FEM integral evaluation

In the finite-element method, a significant computational expense goes into evaluating the integrals in Eqs. (1.14) and (1.15). Since two different integrals (stiffness and force) must be evaluated, let's discuss them generically by letting $\zeta(x)$ denote the integrand. Then the task is to evaluate

$$\int_\Omega \zeta(x)\mathrm{d}x \qquad \text{where} \quad \begin{cases} \text{For the stiffness integral:} & \zeta(x) = E(x)A(x)\hat{\boldsymbol{G}}(x)\hat{\boldsymbol{G}}^T(x) \\ \text{For the force integral:} & \zeta(x) = f(x)\hat{\boldsymbol{N}}(x) \end{cases} \tag{1.18}$$

Here, Ω represents the integration domain spanning from x=0 to x=L. In a conventional FEM code, each integral of this form is evaluated by breaking it into the sum of integrals over element domains:

$$\int_\Omega \zeta(x)\mathrm{d}x = \sum_{e=1}^{\nu_\mathrm{e}} \int_{\Omega_e} \zeta(x)\mathrm{d}x \tag{1.19}$$

where Ω_e represents the e^th element domain, and ν_e is the number of elements. Because finite elements fully tessellate the domain (i.e., they cover it without gaps or overlaps), this reformulation of the integral entails no error or loss of generality. Once the integral over the *entire* domain has been recast this way in terms of element integrals, a conventional FEM integrator will then employ Gauss integration to evaluate each element integral, thus ultimately giving

$$\int_\Omega \zeta(x)\mathrm{d}x = \sum_{e=1}^{\nu_\mathrm{e}} \int_{\Omega_e} \zeta(x)\mathrm{d}x \approx \sum_{e=1}^{\nu_\mathrm{e}} \sum_{g=1}^{\nu_\mathrm{g}(e)} \zeta(x_{ge})\omega_{ge}\mathrm{d}x \tag{1.20}$$

where $\nu_\mathrm{g}(e)$ denotes the number of gauss points used on the e^th element, x_{ge} is the location of the g^th Gauss point on the e^th element, and ω_{ge} is the associated Gauss weight factor. As seen, this method of evaluating the FEM integrals requires function evaluations at a finite number of locations (the Gauss points). Consequently, the FEM code user does not have to write computer software functions for the problem's material, structural, and loading functions.[9] Instead, the FEM code user only needs to supply arrays containing values of these functions at the Gauss points!

Once the FEM integrals have been evaluated, the remainder of the problem is to solve the linear system for the $\nu_\mathrm{n}+2$ unknowns, as previously described. We are now in a position to convert the formulation to become an MPM method, where the same FEM integrals must be evaluated except that problem data are no longer saved at Gauss points.

[9]Namely, the spatially varying stiffness $E(x)$, area $A(x)$, and body force $f(x)$.

1.1.3 Converting the 1-D linear static bar FEM code to an MPM code

Like an FEM formulation, the Material Point Method (MPM) solves the weak form of the governing equations on an overlaid "helper" grid. The grid has nodes and elements that play the same role in an MPM formulation as in an FEM formulation.[10] Accordingly, most of the equations in the preceding FEM section continue to apply for the MPM, with only the relatively minor revisions explained below. Recognizing this similarity of MPM to FEM can greatly facilitate revising an existing FEM code to include MPM as an option.

In the previously discussed traditional FEM formulation, all problem data[11] are specified at the Gauss points, which (by definition of what it means to be a Gauss point) must be placed at locations that have a specific connection to the element and node topology. For an MPM formulation, on the other hand, all problem data are saved at so-called material points,[12] which, as illustrated in Fig. 1.1, are not required to have any connection whatsoever to the topology of the overlaid mesh.

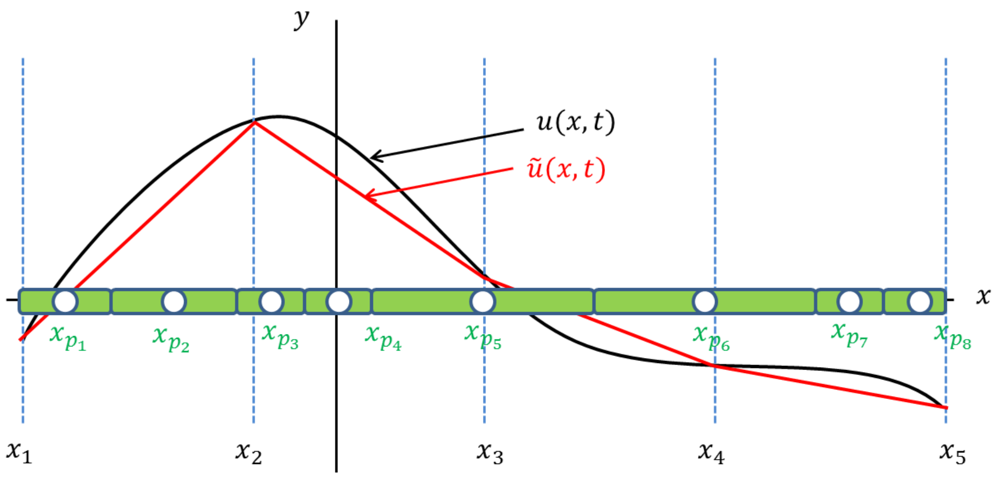


Figure 1.1: A depiction of MPM particles (hollow dots) and their associated physical domains (green pill boxes) on a 1-D domain. Blue dashed lines mark node locations for the topologically disconnected overlaid FEM-like mesh. The sizes of the MPM particle domains do not need to have any connection to the underlying topology of the grid. For clarity in this sketch, the location of a particle is shown with a double subscript (*e.g.,* x_{p_2}), but references to the p^{th} particle location will hereafter be written as x_p while the i^{th} grid node location is written x_i (*i.e.,* the subscript symbol, p or i, will indicate particle or node, respectively.

One difficulty in describing MPM formulations is the notational headaches of distinguishing between numerous countable entities. As in the FEM formulation, we still have a countable number of nodes on the grid. In the MPM, we also have a countable number of material points whose locations must be distinguished from grid node locations. Sometimes, we even need to the left and right endpoints of a particle's domain (ends of the pillbox shapes in the figure). To alleviate some notational clutter in MPM discussions, it is fairly common (*cf.,* [**?**]) for MPM formulations to adopt the convention that the letter of the alphabet used in indices (subscripts) defines the intended meaning as follows:

- Index i stands for a grid node, and $\sum_i$ refers to a sum over all grid nodes. For example, x_i refers to the location of the i^{th} grid node.
- Index p stands for an MPM particle, and $\sum_p$ refers to a sum over all MPM particles. For example, x_p refers to the location of the p^{th} particle. This convention supercedes the double subscripting notation used in Fig. 1.1.[13]

[10] In MPM formulations, an element is often alternatively referred to as a **cell**, but its meaning is still unchanged.
[11] namely, values of material, structural, and forcing functions: stiffness $E(x)$, area $A(x)$, and body force $f(x)$.
[12] called **markers** by some MPM researchers.
[13] The double-subscripting can always be reinstated (as needed) to refer to particular particles. For example, the location of the 29th particle would be denoted $x_{p_{29}}$ in order to distinguish it from the 29th grid node location, which would be denoted $x_{i_{29}}$

- Index e stands for a grid element (sometimes called a "cell" in MPM formulations).
- Index c stands for an MPM particle "corner." The name **corner** is inherited from terminology of 2-D formulations, where a particle corner is a vertex of a polygon-shaped domain associated with the particle. In 1-D, each particle has only two corners, located at $x_p \pm r_p$, where r_p is the **particle radius** (equal to half of the particle length).[14] Also, the double-indexed position, x_{cp} refers to the c^{th} corner of the p^{th} particle. It is important to write this with the c written before the p, as we intend to later assign a different meaning to x_{pc}.
- The MPM formulation, derived below, introduces a double-index quantity, ϕ_{ip}, equal to an average of the grid's i^{th} nodal basis function N_i over the p^{th} particle domain Ω_p.
- Similarly, the double-index quantity $\boldsymbol{G}_{ip}$ will be introduced below to refer to an average of the i^{th} nodal basis function's gradient $\boldsymbol{G}_i$ over the p^{th} particle domain Ω_p.

How is MPM different from FEM?

An existing FEM code for the linear bar problem can be revised to accommodate MPM as an option by going to any point in the FEM source code where spatial integrals are evaluated using the standard FEM-style integration given in Eq. (1.20), and there adding an "`if MPM`" branch to permit evaluation of the same integrals in a different way (using particle data as described below). After this MPM-integration branch, you would then return to the existing FEM coding to finish solving for nodal displacements on the grid as usual.[15] Once nodal displacements are found on the grid, they are simply mapped to the particles. Differences in the choices made to map from grid to particles are additional sources of variance among MPM methods used in the literature. You could, for example, map an updated displacement field on the grid directly to the particles using Eq. (1.6) evaluated at $x = x_p$. A more common (and more accurate in a least-squares sense, not to mention more efficient) mapping from grid to particles uses

$$\boxed{u_p = \sum_i \phi_{ip} u_i} \tag{1.21}$$

where (recall) ϕ_{ip} is the average of the i^{th} nodal basis function over the p^{th} particle. **SS: using the above equation is, I think, of paramount importance when using the CPDI method, discussed later in this tutorial. We should carefully watch out if Uintah mappings are doing this correctly in that case. I don't know if Uintah was written intelligently enough to have mapping from grid to particle be a callable function – if so, then there is less chance of having it screwed up somewhere.**

What definitive attributes must exist in order to declare a code to be a finite-element code? Remarkably, no publications (to our knowledge) provide a clear definition of the phrase **finite-element method**, so will make our own assertion here: Any code that describes fields using standard FEM nodal basis functions [*i.e.,* functions satisfying all of the conditions in Eq. (1.8)] is, by definition, a finite-element code. We assert that this definition is true even if the code employs non-standard methods (like MPM) to evaluate the FEM nodal integrals. In this sense, most MPM codes may be interpreted as finite-element methods. Later, we define a variant of MPM that actually redefines the nodal basis functions to adaptively match the underlying particle topology,[16] thus making such a method no longer properly a finite-element method.

[14]Some extensions to MPM might allow the particle location x_p to be located somewhere other than the particle centroid, but it is assumed that you can make appropriate revisions in that case.

[15]Actually, you might also need an "`if MPM`" branch in the construction of the boundary force array $\hat{\mathfrak{F}}$, depending on the MPM particle placements. When the physical boundary does not coincide with a grid boundary, as in Fig. 1.2, the construction of $\hat{\mathfrak{F}}$ is a bit more complicated. In that case, it might have nonzero components at more than the first and last locations in the array, but it will still depend on $\mathcal{F}(0)$ and $\mathcal{F}(L)$, thus making no change in the solvability of the system of $\nu_{\text{n}} + 2$ equations for $\nu_{\text{n}} + 2$ unknowns, and only minor changes in the solution procedure itself. For the simple 1-D small-displacement problems, we presume for simplicity that the boundary of the physical domain coincides with grid nodes, as in Fig. 1.1, hence making the boundary force array the same as it was in the previous classical FEM formulation. Evaluation of the boundary term for large-displacement problems is discussed later.

[16]This method, discussed later, stretches the nodal basis functions in regions having elongated MPM particles; it even effectively deletes some of the nodal basis functions by setting them to zero, but the key is that partition of unity is nevertheless preserved.

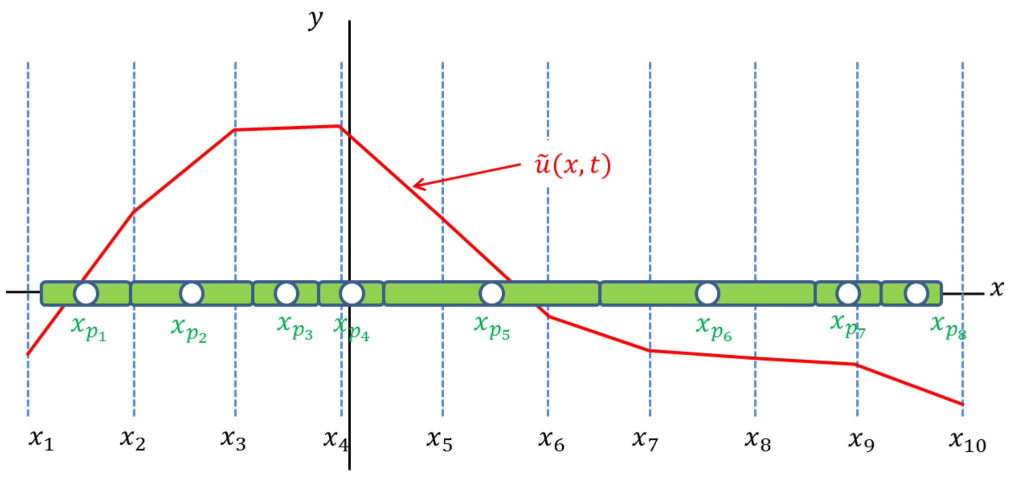


Figure 1.2: The same particle topology as in Fig. 1.1, but with a different overlaid grid. In this case, there is no grid node at the body's boundary. This additional complication with the Material Point Method (or, for that matter, any particle method that uses an overlaid grid) is avoidable for small-deformation problems where it is always possible to place overlay grid nodes at the boundary. Algorithmic adjustments for large displacements of particles (relative to the ovarlaid grid) are deferred until later.

MPM-style evaluation of FEM-style nodal integrals

All of the equations previously given for the standard FEM solution to the bar equation continue to apply, except that the FEM-style summation over *elements* in Eq. (1.19) is instead replaced with a summation over particle domains:

$$\int_{\Omega} \zeta(x)\mathrm{d}x \;=\; \sum_{p} \int_{\Omega_p} \zeta(x)\,\mathrm{d}x \tag{1.22}$$

where (recall) Ω refers to the entire domain from $x{=}0$ to $x{=}L$, and Ω_p refers to particle domains, shown as pillboxes in Fig. 1.1. If the particle domains form a proper tessellation (*i.e.,* , if they cover the domain without gaps or overlaps), then the above equation is true exactly without any loss in generality.

Let V_p denote the "volume" of the p^{th} particle (which is simply the particle length in the 1-D bar example). Then, again without loss in generality, the above equation may be written

$$\int_{\Omega} \zeta(x)\,\mathrm{d}x \;=\; \sum_{p} V_p \overline{\zeta_p} \tag{1.23}$$

where

$$\overline{\zeta_p} := \frac{1}{V_p} \int_{\Omega_p} \zeta(x)\mathrm{d}x \tag{1.24}$$

This shows that an MPM-style integration over Ω boils down to needing the average $\overline{\zeta_p}$ of the function $\zeta(x)$ over the p^{th} particle domain Ω_p. Variants of MPM are distinguished by how they compute such averages. As pointed out by Bardenhagen and Kober [?], with further generalization and clarification by Sadeghirad *et al.* [?], all commonly used MPM formulations can be unified as

GENERALIZED APPROXIMATE AVERAGE

$$\boxed{\frac{1}{V_p}\int_{\Omega_p} \zeta(x)\mathrm{d}x \approx \frac{1}{V_p^*}\int_{\Omega_p^*} \zeta_p^*(x)\omega^*(x)\mathrm{d}x} \qquad \text{where} \qquad V_p^* := \int_{\Omega_p^*} \omega^*(x)\mathrm{d}x \tag{1.25}$$

The function $\omega^*(x)$ denotes a weighting function, the selection of which (among other things) distinguishes variants of MPM formulations, as discussed below. The **weighted particle volume** V_p^* is generally different from the *actual* particle volume V_p.[17] The generalized approximate average in Eq. (1.25) permits the integration domain Ω_p^* to be selected differently from the actual particle domain Ω_p. Choosing to change the integration domain is not particularly useful in 1-D problems, but we will see that it is almost essential in 2-D and 3-D problems. How can we "get away with" changing the integration domain? The key is that we are *finding averages*, which makes this revision very reasonable as long as all domains over which the average is taken contain the material point.[18] If both the actual particle domain Ω_p and the alternative averaging domain Ω_p^* shrink to zero size with discretization refinement, then both integrals converge to each other as long as the integrand does not contain an essential singularity (which it won't for regular engineering problems). The potential for discrepancies on coarser meshes is highest when the integrand is discontinuous, as is the case in the stiffness integral when low order shape functions are used; this problem leads to the "cell-straddling error" discussed below. Aside from allowing a change in the integration domain, notice that the generalized average in Eq. (1.25) allowed a revised integrand $\zeta^*(x)$ to take the place of the actual integrand, $\zeta_p(x)$. To motivate (and justify) this aspect of the generalized average, we will now discuss specific choices for the weight function, and other approximations used in this unified particle averaging formula.

Early MPM formulations treated the body as if it were made from point masses[**?**]. Bardenhagen and Kober [**?**] pointed out that this somewhat unsavory assumption is equivalent to the following selections in the generalized approximate average of Eq. (1.25):

$$\Omega_p^* = \Omega_p \tag{1.26a}$$
$$\zeta_p^*(x) = \zeta(x) \tag{1.26b}$$
$$\omega^*(x) = \delta(x - x_p) \tag{1.26c}$$

where $\delta(x - x_p)$ is the Dirac delta function centered at x_p. By properties of the Dirac delta (noting that $x_p \in \Omega_p$), these choices therefore give the following approximation for the particle domain average:

The single-point average: $$\frac{1}{V_p}\int_{\Omega_p} \zeta(x)\mathrm{d}x \approx \zeta(x_p) \tag{1.27}$$

In other words, the choices in Eq. (1.26) are equivalent to taking the average of $\zeta(x)$ over the particle domain to be approximately the integrand evaluated at the particle![19] This is equivalent to single-point Gauss integration on the particle domain. The single-point average is also equivalent to the following alternative choices in the generalized average:

$$\Omega_p^* = \Omega_p \tag{1.28a}$$
$$\zeta_p^*(x) = \zeta(x_p) \qquad = \text{constant} \tag{1.28b}$$
$$\omega^*(x) = T_p^*(x) \tag{1.28c}$$

where $T_p^*(x)$ is the so-called **tophat** function defined by

$$T_p^*(x) := \begin{cases} 1 & \text{for } x \in \Omega_p^* \\ 0 & \text{for } x \notin \Omega_p^* \end{cases} \tag{1.29}$$

These choices, which correspond to replacing the spatially varying integrand $\zeta(x)$ by a function that is constant over the particle, give the same result as the Dirac choices in Eq. (1.26). The single-point average is

[17]ALERT! The actual volume V_p, not V_p^* must still be used in Eq. (1.23). **SS: This assertion is something that I believe needs to be checked in Uintah. Are they accidentally using the weighted particle volume where the actual volume is needed? If so, this mistake could conceivably cause a kinematic anomaly**

[18]An analogous assertion from everyday life would be that your average gasoline efficiency (in miles per gallon, MPG) this *year* is approximately equal to your average gasoline efficiency this *month*. Accuracy of such an approximation is good if your driving habits don't change too much over a year.

[19]This would be analogous to assuming that your average gasoline usage rate over a year is approximately equal to your average usage at this instant in time – it's a lousy approximation if you aren't running your engine continuously.

a very good approximation if the integrand $\zeta(x)$ does not vary significantly over Ω_p, but we now explain that this is not generally the case for most MPM formulations. To judge the accuracy of such an approximation, we must recall the specific forms of the $\zeta(x)$ functions actually needed in the nodal grid integrals of Eq. (1.18):

$$\text{For the stiffness integral, } \zeta(x) = E(x)A(x)\hat{\boldsymbol{G}}(x)\hat{\boldsymbol{G}}^T(x) \tag{1.30a}$$

$$\text{For the force integral, } \zeta(x) = f(x)\hat{\boldsymbol{N}}(x) \tag{1.30b}$$

Deciding if the single-point average in Eq. (1.27) will give acceptably accurate results requires judging if these functions can be assumed to be approximately constant over the particle domain, at least in the limit as the discretization is refined. These considerations are as follows:

FORCE INTEGRAL, Eq. (1.30a): If linear shape functions are used, the nodal basis functions $\hat{\boldsymbol{N}}$ will be piecewise linear "tent" functions. Then their gradients in $\hat{\boldsymbol{G}}$ will be discontinuous at grid cell boundaries. Accordingly, even though $E(x)$ and $A(x)$ might be reasonably approximated to be constant over a particle domain, such an approximation is unreasonable for $\hat{\boldsymbol{G}}$ if the particle domain is not contained entirely within a grid cell. For example, the single-point average is expected to be moderately accurate for the second particle in Fig. 1.1, because that particle is contained entirely within a grid cell. The error of the single-point average for the third particle in that figure is expected to be quite high because that particle straddles two grid cells, thus causing the integrand function $\zeta(x)$ to vary discontinously over the particle despite its small size. We call this a **cell-straddling error**.

FORCE INTEGRAL, Eq. (1.30b): As long as the applied distributed load $f(x)$ is continuous, Eq. (1.8e) ensures that the integrand in Eq. (1.30b) is also continuous. Accordingly, it would be increasingly accurate to assume that $\zeta(x)$ in that equation to be approximately constant over a small particle domain. In other words, the primary source of error with the Dirac averaging scheme arises in the stiffness integral, not the distributed force integral.

Eliminating the cell-straddling error

The cell-straddling error may be dramatically reduced by making the following choices for the generalized average:

$$\Omega_p^* = \Omega_p \tag{1.31a}$$

$$\text{For the stiffness integral, } \zeta_p^*(x) = E(x_p)A(x_p)\hat{\boldsymbol{G}}(x)\hat{\boldsymbol{G}}^T(x) \tag{1.31b}$$

$$\text{For the force integral, } \zeta_p^*(x) = f(x_p)\hat{\boldsymbol{N}}(x) \tag{1.31c}$$

$$\omega^*(x) = T_p^*(x) \tag{1.31d}$$

where $T_p^*(x)$ is the tophat function defined previously. Note that the approximate $\zeta_p^*(x)$ functions are the same as the exact $\zeta(x)$ functions except that the *physical* field functions[20] are treated as constant over each particle domain, given by their values at the particle. Substituting Eq. (1.28) into Eq. (1.25), and then using Eq. (1.23) in Eqs. (1.14) and (1.15) gives

APPROXIMATE MPM PARTICLE STIFFNESS

$$\boxed{\hat{\hat{\boldsymbol{K}}} \approx \sum_p E(x_p)A(x_p) \int_{\Omega_p} \hat{\boldsymbol{G}}(x)\hat{\boldsymbol{G}}^T(x)\mathrm{d}x} \tag{1.32}$$

and

APPROXIMATE MPM FORCE

$$\boxed{\hat{\boldsymbol{f}} \approx \sum_p f(x_p) \int_{\Omega_p} \hat{\boldsymbol{N}}(x)\mathrm{d}x} \tag{1.33}$$

[20]namely, stiffness $E(x)$, area $A(x)$, and distributed load $f(x)$

In these forms, the remaining integrals involve only the nodal basis functions or their gradients, hence allowing these integrals to be evaluated exactly over each particle domain, thereby significantly increasing the accuracy of the result in comparison to the single-point method.

1.1.4 Exotic idea: make the integrals easier by changing the basis

Above, the FEM integrals reduced to needing to find exact solutions for

$$\int_{\Omega_p} \hat{\boldsymbol{G}}(x)\hat{\boldsymbol{G}}^T(x)\mathrm{d}x \qquad \text{and} \qquad \int_{\Omega_p} \hat{\boldsymbol{N}}(x)\mathrm{d}x \tag{1.34}$$

Exactly evaluating these integrals isn't horribly difficult in 1-D, but getting exact solutions to the analogous integrals is impractical in 2-D and 3-D, at least when all of the properties in Eq. (1.8) are assumed to hold. However, nothing in our preceding analysis required Eq. (1.8a) or Eq. (1.8d). Convergence is ensured by retaining the properties of partition of unity, linear completeness, and continuity, but we do not need the Kronecker property or FEM-style compactness. The alternative continuously adaptive grid basis in Fig. 1.3 is linearly complete, and it can be shown to significantly increase convergence rates [**?**]. Moreover, although this alternative basis might seem to be more complicated than ordinary tent functions when viewed from the global perspective, these CPDI functions are *much* simpler (straight lines) when viewed on any given particle domain. Thus, each integral over a particle is much simpler! Over any given particle the alternative basis, which we denote $N^*(x)$ is simply a linear interpolation of the regular tent-function basis $N(x)$ evaluated at the particle corners (*i.e.,* at the particle endpoints).[21] Thus, in a loop over particles, the integrals in Eq. (1.34) become trivial to evaluate! Not only is the computational effort reduced when using $N^*(x)$, these choices also automatically avoid the cell-straddling error, and they ensure that neighboring particles will interact with each other even when they are separated by an arbitrary number of grid cells, thus allowing coarse particle distributions in regions of little interest.

SS: It is VERY important for you to understand the meaning and algorithmic simplicity of the CPDI basis – make sure you understand it and can use it immediately in your programming exercises.

1.1.5 MPM revised boundary force array

Recall that $\hat{\boldsymbol{\mathfrak{F}}} = \hat{\boldsymbol{N}}(0)\mathcal{F}(0) + \hat{\boldsymbol{N}}(L)\mathcal{F}(L)$. For finite elements, the Kronecker property ensured that the boundary force array $\hat{\boldsymbol{\mathfrak{F}}}$ would have nonzero components only in the first and last components. In an MPM problem, the boundary force array will generally have a larger number of nonzero components. When the particle boundary happens to coincide with the grid boundary, then $\widehat{\boldsymbol{tractionForce}}$ will again be nonzero only in its first and last components. However, for the particle distribution shown in Fig. 1.2, where the outer particle boundary falls in the interior of a grid cell, both $N_1(0)$ and $N_2(0)$ are nonzero, thus implying that $\hat{\boldsymbol{\mathfrak{F}}}$ will be nonzero in its first two components. It will be similarly nonzero in its last two components. **SS: I suspect that Carlos did not fully appreciate this observation, so he might not have had appropriate nodal contributions near the boundary where he was having so much trouble. Uintah supports a force (pressure) boundary condition only by putting a point load directly on a boundary particle, but we proved that this approach gives large errors under large deformations. The weak formulation's boundary term must be evaluated accurately. I think this might be contributing to some of our problems.**

1.1.6 MPM Algorithm and data management for the 1-D bar

The following describes a data structure that I don't think is actually used in any MPM codes, but which I think ought to be considered in light of the fact that we want to move towards support of particles that stretch across multiple grid cells. This data scheme will, I think, work well with CPDI-style integral evaluations.

To revise an existing MPM linear bar code to accommodate an MPM option, you must create a new data array containing the locations and lengths of each material point. To fold the MPM option smoothly

[21] This interpolation of the regular FEM-style nodal basis generalizes seemlessly to 2-D and 3-D analysis.

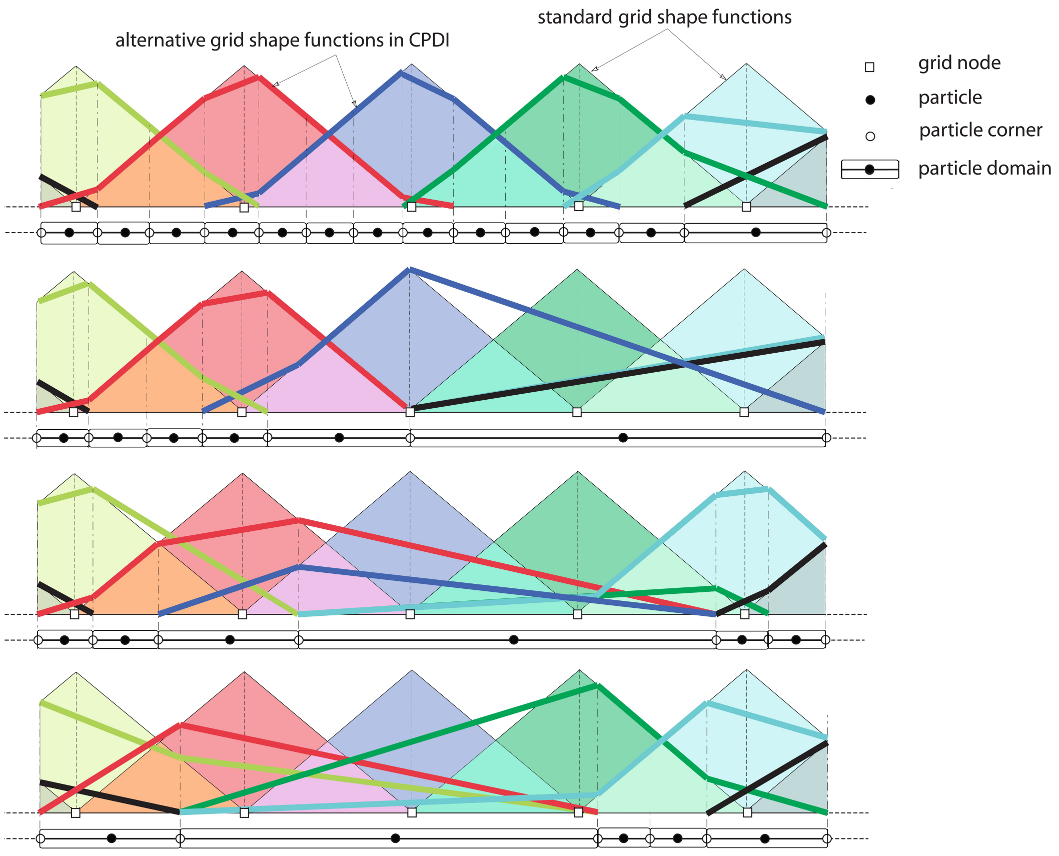

standard grid shape functions
alternative grid shape functions in CPDI
grid node
particle
particle corner
particle domain

Figure 1.3: Shaded triangles are the classical FEM “tent functions”, while the solid thick lines are alternative “convected particle domain interpolation (CPDI)” nodal basis functions, constructed to be piecewise-linear interpolations of the regular tent functions to the end points of the particle domains. That is, the CPDI basis functions exactly coincide with the regular tent functions at the particle corners, but the CPDI basis is a straight line within the interior of any particle, making integrals of the basis or its gradient trivial within particle domains.

into an existing FEM code, you should anticipate the possibility of material point domains stretching across arbitrarily large expanses of grid cells. Accordingly, you should bypass the standard FEM element loop, and instead perform a loop directly over particles. You will need to create a new ragged array, let's call it $\hat{\hat{\boldsymbol{Y}}}$, for which Y_{pn} contains the ID of the n^{th} node having a support domain intersecting the approximate particle domain Ω_p^*. If, for example, we choose $\Omega_p^* = \Omega_p$ and if we use standard linear shape functions, then the ragged array corresponding to Fig. 1.1 would be

$$\hat{\hat{\boldsymbol{Y}}} = \begin{bmatrix} 1 & 2 & \\ 1 & 2 & \\ 1 & 2 & 3 \\ 2 & 3 & \\ 2 & 3 & 4 \\ 3 & 4 & 5 \\ 4 & 5 & \\ 4 & 5 & \end{bmatrix} \tag{1.35}$$

Here, for example, the third row corresponds to the third particle, and that particle has a domain partly in the first element and partly in the second element. Since the nodes for these element are numbered 1, 2, and 3, those are the nodal values associated with (*i.e.,* affected by) this particle. The fourth particle, on the other hand resides entirely within the second element. Thus, since this element is bounded by nodes 2 and 3, these are the numbers that go into the forth row of the above matrix. Similarly, the list corresponding to Fig. 1.2 would be

$$\hat{\hat{\boldsymbol{Y}}} = \begin{bmatrix} 1 & 2 & & \\ 1 & 2 & 3 & 4 \\ 3 & 4 & & \\ 3 & 4 & 5 & \\ 4 & 5 & 6 & 7 \\ 6 & 7 & 8 & 9 \\ 8 & 9 & 10 & \\ 9 & 10 & & \end{bmatrix} \tag{1.36}$$

Need to finish this. Again, I'm not sure if I am describing the optimal data management scheme. We ought to think carefully about this, not allowing any existing MPM codes to prevent us from seeing the optimal structure, especially since we want particles to be able to stretch across multiple cells (which could cause significant parallelization problems if we aren't careful)

1.2 Dynamic linear bar (acoustic wave analysis)

For dynamic small-deformation acoustic wave motion in a linear-elastic bar, the governing equation in Eq. (1.1) is revised as follows:

$$\begin{aligned} &\frac{\partial}{\partial x}\left[E(x)A(x)\frac{\partial u(x,t)}{\partial x}\right] + f(x,t) = \rho(x)A(x)\ddot{u}(x,t) && \text{on domain } \Omega \text{ defined by } 0 < x < L \\ &\qquad\qquad \text{subject to Robin BCs:} && \alpha_0\mathcal{F}(0,t) + \beta_0 u(0,t) = \gamma_0 \\ & && \text{and} \quad \alpha_L\mathcal{F}(L,t) + \beta_L u(L,t) = \gamma_L \end{aligned} \tag{1.37}$$

where $\rho(x)$ is the spatially varying mass density of the material and $\ddot{u}(x,t)$ is the material acceleration: $\ddot{u}(x,t) := \partial^2 u/\partial t^2$, assuming small strains. Key differences between this problem and the previous (static) bar problem are

- There are now two independent variables: position x and time t. Accordingly, where the previous ODE had regular derivatives, this PDE has partial derivatives.
- The boundary conditions might vary with time (technically even the Robin parameters might vary with time).
- A complete problem description requires specification of initial conditions, giving the stress and displacement at time $t = 0$.

1.2.1 Conventional FEM is used to spatially discretize the acoustic wave problem

As was done in the static case, the governing equation is multiplied by an arbitrary weight (test) function and integrated by parts to reduce the spatial derivative on the displacement to obtain the weak form,

$$\begin{aligned} &-\int_0^L E(x)A(x)w'(x,t)u'(x,t)\mathrm{d}x + w(0,t)\mathcal{F}(0,t) + w(L,t)\mathcal{F}(L,t) + \int_0^L w(x,t)f(x,t)\mathrm{d}x \\ &= \int_0^L w(x,t)\rho(x)A(x)\ddot{u}(x,t)\mathrm{d}x \qquad\qquad \forall w(x,t) \end{aligned} \tag{1.38}$$

As in the static case, the approximations of Eq. (1.10) are again substituted into the weak form, except this time the nodal values are functions of time instead of being constants:

$$\tilde{u}(x) = \hat{\boldsymbol{N}}^T(x)\hat{\boldsymbol{u}}(t) \qquad \text{and} \qquad w(x) = \hat{\boldsymbol{w}}^T(t)\hat{\boldsymbol{N}}(x) \tag{1.39}$$

Substituting these into the weak form and asserting that the result must hold $\forall\hat{\boldsymbol{w}}$ gives

$$-\hat{\hat{\boldsymbol{K}}}\hat{\boldsymbol{u}} + \hat{\boldsymbol{\mathcal{F}}} + \hat{\boldsymbol{f}} = \hat{\hat{\boldsymbol{M}}}\hat{\ddot{\boldsymbol{u}}} \tag{1.40}$$

where, exactly as in the static case,

$$\hat{\hat{\boldsymbol{K}}} := -\int_0^L E(x)A(x)\hat{\boldsymbol{G}}(x)\hat{\boldsymbol{G}}^T(x)\mathrm{d}x \tag{1.41}$$

and, almost exactly as in the static case,

$$\hat{\boldsymbol{\mathcal{F}}}(t) = \hat{\boldsymbol{N}}(0)\mathcal{F}(0,t) + \hat{\boldsymbol{N}}(L)\mathcal{F}(L,t) \tag{1.42}$$

and

$$\hat{\boldsymbol{f}}(t) = \int_0^L f(x,t)\hat{\boldsymbol{N}}(x)\mathrm{d}\,x. \tag{1.43}$$

On the right-hand side of the spatially discretized equation, the inertial acceleration term introduces a new quantity, the so-called consistent mass matrix defined by

$$\boxed{\textbf{Consistent mass matrix:} \qquad \hat{\hat{\boldsymbol{M}}} := \int_0^L \hat{\boldsymbol{N}}(x)\hat{\boldsymbol{N}}^T(x)\rho(x)\mathrm{d}x} \tag{1.44}$$

In practice, the consistent mass matrix is often replaced with

$$\boxed{\textbf{Lumped mass matrix:} \qquad \hat{\hat{\boldsymbol{M}}} := \int_0^L \hat{\boldsymbol{N}}(x)\rho(x)\mathrm{d}x} \tag{1.45}$$

The transient spatially discretized Eq. (1.40) is a linear second-order system of ODEs in time. In practice, the single second-order ODE system is typically broken up into a larger system of single-order ODEs as follows:

$$-\hat{\hat{\boldsymbol{K}}}\hat{\boldsymbol{u}} + \hat{\boldsymbol{\mathfrak{F}}} + \hat{\boldsymbol{f}} = \hat{\hat{\boldsymbol{M}}}\hat{\boldsymbol{a}} \tag{1.46a}$$

$$\hat{\boldsymbol{a}} = \dot{\hat{\boldsymbol{v}}} \tag{1.46b}$$

$$\hat{\boldsymbol{v}} = \dot{\hat{\boldsymbol{u}}} \tag{1.46c}$$

For acoustic problems, this system is usually solved by simple explicit time integrator as follows: At the beginning of the first time step, the boundary forces are known as part of the initial conditions, and hence $\mathfrak{F}$ is known at the beginning of the time step. Likewise, the distributed load is given at the beginning of the timestep by $f(x,0)$, thus allowing $\hat{\boldsymbol{f}}$ to be evaluated at the beginning of the step. Likewise, the mass matrix may be evaluated at the beginning of the step, and the acceleration at the beginning of the step is then found from solving Eq. (1.46a) for $\hat{\boldsymbol{a}}$:

$$\hat{\boldsymbol{a}}^n = \hat{\hat{\boldsymbol{M}}}^{-1}\left(-\hat{\hat{\boldsymbol{K}}}\hat{\boldsymbol{u}}^n + \hat{\boldsymbol{\mathfrak{F}}}^n + \hat{\boldsymbol{f}}^n\right) \tag{1.47}$$

Here, the superscript n is used to indicate that this is evaluated at the beginning of the n^{th} timestep when $t = t^n$. When a lumped mass is used, the inverse of the mass matrix is trivial. With the acceleration known at the beginning of the step, the velocity is updated to the end of the step by using a first-order Taylor series in time from $t = t^n$ to $t = t^{n+1} = t^n + \Delta t$, where Δt is the timestep:

$$\hat{\boldsymbol{v}}^{n+1} = \hat{\boldsymbol{v}}^n + \hat{\boldsymbol{a}}^n\Delta t \tag{1.48}$$

The displacement is integrated using a second-order Taylor series in time:

$$\hat{\boldsymbol{u}}^{n+1} = \hat{\boldsymbol{u}}^n + \hat{\boldsymbol{v}}^n\Delta t + \frac{1}{2}\hat{\boldsymbol{a}}(\Delta t)^2 \tag{1.49}$$

These are the updates of NODAL quantities. Updates of particle quantities are done by mapping nodal acceleration and velocity from the grid to the particles, and then the second-order Taylor series integration is performed at the particle to update the particle position. **SS: Check carefully if this is what Uintah does. Check if this is what Ali described in our CPDI papers.**

1.2.2 Intrinsic no-interpenetration bonus attribute of MPM

I need to add a problem of two bars bumping into each other, showing that the MPM ensures that they will bounce away from each other without any additional coding of the algorithm – this behavior comes "for free!"

1.3 Adding material nonlinearity to the 1-D static bar

Let us now assume negligible accelerations so that we can focus on the complications that arise from material nonlinearity (*i.e.,* when Hooke's law no longer applies). The goal of this section is to show how to generalize the linear static bar problem to allow material nonlinearity. In particular, we want to treat the constitutive model as a 'black box' called by the host FEM or MPM framework.

There are two forms of nonlinearity in realistic mechanics problems:

- **Geometric nonlinearity** arises from large motions of the body so that now there is a significant difference between the initial and deformed locations of a point in the body. When large motions are possible, one must reconsider the problem functions, such as the distributed load $f(x)$ to decide if x represents the initial or deformed location. For now, we will ignore geometric nonlinearity by assuming that displacements are small enough that we don't need to worry if x is the initial or deformed position – the two are approximately equal to each other. Neglecting geometric nonlinearity also frees us from having to consider complications such as the meaning of strain[22]

- **Material nonlinearity** arises when the stress is not a linear function of strain, or possibly not even a function of strain at all! In this case, Hooke's law ($\sigma = E\varepsilon$) no longer applies, and we must go back to the strong form of the governing equations to remove all remnants of Hooke's law. In subsequent sections, we will assume that the material constituive model is a "black box" function $\sigma(\varepsilon)$ giving the stress as some unknown, generally nonlinear function of strain.

1.3.1 Traditional FEM formulation of the nonlinear static bar problem

As in the linear case, the approximate FEM solution for the displacement field is

$$\tilde{u}(x) := \sum_{j=1}^{\nu_\text{n}} \tilde{u}_j N_j(x) \tag{1.50}$$

and the weight function is still expanded as it was in the linear case,

$$w(x) := \sum_{i=1}^{\nu_\text{n}} w_i N_i(x) \tag{1.51}$$

In the nonlinear finite-element method, the nodal basis functions are still required to satisfy the properties listed in Eq. (1.8). The arrays $\hat{\boldsymbol{N}}(x)$, $\hat{\boldsymbol{G}}(x)$, $\hat{\boldsymbol{u}}$, $\hat{\boldsymbol{w}}$, and $\hat{\boldsymbol{\mathfrak{F}}}$ are defined the same as they were in the linear case. Accordingly, the array form of Eq. (1.63) becomes

$$\int_0^L \underbrace{\hat{\boldsymbol{w}}^T}_{1\times\nu_\text{n}} \underbrace{\hat{\boldsymbol{G}}(x)}_{\nu_\text{n}\times 1} A(x)\sigma(x)\mathrm{d}x = \underbrace{\underbrace{\hat{\boldsymbol{w}}^T}_{1\times\nu_\text{n}} \underbrace{\hat{\boldsymbol{\mathfrak{F}}}}_{\nu_\text{n}\times 1}}_{1\times 1} + \int_0^L \underbrace{\underbrace{\hat{\boldsymbol{w}}^T}_{1\times\nu_\text{n}} \underbrace{\hat{\boldsymbol{N}}(x)}_{\nu_\text{n}\times 1}}_{1\times 1} f(x)\mathrm{d}x \qquad \forall\hat{\boldsymbol{w}} \tag{1.52}$$

where (recall) $\hat{\boldsymbol{G}}(x)$ denotes the array of shape function gradients. Since this must hold $\forall\hat{\boldsymbol{w}}$, the premultiplication by $\hat{\boldsymbol{w}}^T$ may be removed, giving

$$\boxed{\hat{\boldsymbol{f}}^\text{int} + \hat{\boldsymbol{f}}^\text{ext} = \hat{\boldsymbol{0}}} \tag{1.53}$$

where

$$\boxed{\textbf{Internal force:} \quad \hat{\boldsymbol{f}}^\text{int} := -\int_0^L \hat{\boldsymbol{G}}(x)A(x)\sigma(x)\mathrm{d}x} \tag{1.54}$$

[22] Is it change in length divided by initial length or current length? This question is moot when there is negligible change in length.

and

$$\boxed{\textbf{External force:}\quad \hat{\boldsymbol{f}}^{\text{ext}} := \hat{\boldsymbol{\mathfrak{F}}} + \int_0^L \hat{\boldsymbol{N}}(x) f(x) \mathrm{d}x} \tag{1.55}$$

These equations state that the sum of all forces must be zero. In particular, the forces applied by external agents (the distributed body force and the boundary forces) must be balanced exactly by the body's internal resistance to those forces.

1.4 Nonlinear small-deformation static 1-D bar problem

For the 1-D bar problem with material nonlinearity, the governing equations are

$$\begin{aligned}
&\frac{\mathrm{d}}{\mathrm{d}x}\left[A(x)\sigma(x)\right] + f(x) = 0 && \text{on domain } \Omega \text{ defined by } 0 < x < L \\
&\sigma(x) = \xi(\varepsilon(x), x) && \text{subject to Robin BCs:} \\
&\varepsilon(x) = \frac{\mathrm{d}u(x)}{\mathrm{d}x} && \alpha_0 \mathcal{F}(0) + \beta_0 u(0) = \gamma_0 \\
& && \alpha_L \mathcal{F}(L) + \beta_L u(L) = \gamma_L
\end{aligned} \tag{1.56}$$

Here, $\xi(\varepsilon, x)$ is a "blackbox" nonlinear elastic constitutive model giving stress from strain ε. The explicit presence of x allows material heterogeneity (such as material properties varying from point to point along the bar). You, in your role as an MPM code developer, must treat the $\xi(\varepsilon, x)$ function Note that the linear bar problem in the previous section is recovered by choosing $\xi(\varepsilon, x) = E(x)\varepsilon$.

In the Robin boundary conditions,

$$\mathcal{F}(0) := -A(0)\sigma(0) \qquad \text{and} \qquad \mathcal{F}(L) := A(L)\sigma(0) \tag{1.57}$$

As in the previous linear bar analysis, these forces are defined to be positive when pointing to the right, in the direction of increasing x, while positive stresses correspond to tension (thus accounting for the negative in the first force definition, since a tensile force on the left end of the bar would point to the left). All other variables in these equations are defined as they were in Eq. (1.1). In what follows, we progress through the same sort of analysis steps as in the linear case, making appropriate adjustments for material nonlinearity.

The strong form of the governing equation in Eq. (1.67) is multiplied by an arbitrary weight function $w(x)$, and then integrated over the domain from $x = 0$ to $x = L$:

$$\int_0^L \frac{\mathrm{d}}{\mathrm{d}x}\left[A(x)\sigma(x)\right] w(x) \mathrm{d}x + \int_0^L f(x) w(x) \mathrm{d}x = 0 \qquad \forall w(x) \tag{1.58}$$

The first term is integrated by parts to give the so-called **weak form** of the governing equation:

$$w(x)A(x)\sigma(x)\big|_0^L - \int_0^L w'(x)A(x)\sigma(x)\mathrm{d}x + \int_0^L w(x)f(x)\mathrm{d}x = 0 \qquad \forall w(x) \tag{1.59}$$

Using the definition of boundary forces defined in Eq. (1.2), this may be written

$$\int_0^L w'(x)A(x)\sigma(x)\mathrm{d}x = w(0)\mathcal{F}(0) + w(L)\mathcal{F}(L) + \int_0^L w(x)f(x)\mathrm{d}x = 0 \tag{1.60}$$

1.4.1 Traditional FEM formulation of the nonlinear bar problem

As in the linear case, the approximate FEM solution for the displacement field is

$$\tilde{u}(x) := \sum_{j=1}^{\nu_\mathrm{n}} \tilde{u}_j N_j(x) \tag{1.61}$$

and the weight function is still expanded as it was in the linear case,

$$w(x) := \sum_{i=1}^{\nu_\mathrm{n}} w_i N_i(x) \tag{1.62}$$

In the nonlinear finite-element method, the nodal basis functions are still required to satisfy the properties listed in Eq. (1.8). The arrays $\hat{\boldsymbol{N}}(x)$, $\hat{\boldsymbol{G}}(x)$, $\hat{\boldsymbol{u}}$, $\hat{\boldsymbol{w}}$, and $\hat{\boldsymbol{\mathfrak{F}}}$ are defined the same as they were in the linear case. Accordingly, the array form of Eq. (1.63) becomes

$$\int_0^L \underbrace{\hat{\boldsymbol{w}}^T}_{1\times\nu_\mathrm{n}} \underbrace{\hat{\boldsymbol{G}}(x)}_{\nu_\mathrm{n}\times 1} A(x)\sigma(x)\mathrm{d}x = \underbrace{\underbrace{\hat{\boldsymbol{w}}^T}_{1\times\nu_\mathrm{n}} \underbrace{\hat{\boldsymbol{\mathfrak{F}}}}_{\nu_\mathrm{n}\times 1}}_{1\times 1} + \int_0^L \underbrace{\underbrace{\hat{\boldsymbol{w}}^T}_{1\times\nu_\mathrm{n}} \underbrace{\hat{\boldsymbol{N}}(x)}_{\nu_\mathrm{n}\times 1}}_{1\times 1} f(x)\mathrm{d}x \qquad \forall \hat{\boldsymbol{w}} \tag{1.63}$$

where (recall) $\hat{\boldsymbol{G}}(x)$ denotes the array of shape function gradients. Since this must hold $\forall\hat{\boldsymbol{w}}$, the premultiplication by $\hat{\boldsymbol{w}}^T$ may be removed, giving

$$\boxed{\hat{\boldsymbol{f}}^{\mathrm{int}} + \hat{\boldsymbol{f}}^{\mathrm{ext}} = \hat{\boldsymbol{0}}} \tag{1.64}$$

where

$$\boxed{\textbf{Internal force:} \quad \hat{\boldsymbol{f}}^{\mathrm{int}} := -\int_0^L \hat{\boldsymbol{G}}(x)A(x)\sigma(x)\mathrm{d}x} \tag{1.65}$$

and

$$\boxed{\textbf{External force:} \quad \hat{\boldsymbol{f}}^{\mathrm{ext}} := \hat{\boldsymbol{\mathfrak{F}}} + \int_0^L \hat{\boldsymbol{N}}(x)f(x)\mathrm{d}x} \tag{1.66}$$

These equations state that the sum of all forces must be zero. In particular, the forces applied by external agents (the distributed body force and the boundary forces) must be balanced exactly by the body's internal resistance to those forces.

Counting unknowns and equations to confirm solvability

The internal force involves an unknown stress field, $\sigma(x)$. Therefore, all components of the internal force array are unknowns. These internal forces would become known if the values of their integrands were known at Gauss points. Thus, stresses at Gauss points become part of the list of unknowns. Stresses at Gauss points may be found from strain at Gauss points via the constitutive law, making strain at Gauss points added to the list of unknowns. Strain at Gauss points are defined to equal the displacement gradient at the Gauss points, which is found if we know the nodal displacements. Thus nodal displacements must be added to the list of unknowns. As was the case in the linear problem, the boundary forces, $\mathfrak{F}(0)$ and $\mathfrak{F}(L)$, are unknown.

Overall, the counting of equations and unknowns for the nonlinear bar problem is summarized as follows.

- NUMBER OF EQUATIONS: $2\nu_{\mathrm{n}} + 2 + 2\nu_{\mathrm{G}}$
  (a) ν_{n} linear equations from force balance: $\hat{\boldsymbol{f}}^{\mathrm{int}} + \hat{\boldsymbol{f}}^{\mathrm{ext}} = \hat{\mathbf{0}}$
  (b) ν_{n} linear equations from Gauss evaluation of the $\hat{\boldsymbol{f}}^{\mathrm{int}}$ integrals as linear combinations of the stresses at Gauss points.
  (c) 2 linear equations from boundary conditions: $\alpha_0\mathcal{F}(0) + \beta_0 u_1 = \gamma_0$ and $\alpha_L\mathcal{F}(L) + \beta_L u_{\nu_{\mathrm{n}}} = \gamma_L$
  (d) ν_{G} nonlinear equations from stresses at Gauss points determined from the strains at the Gauss points, via the the constitutive (material) model: $\sigma(x_g) = \xi(\varepsilon_g, x_g)$ where subscript g ranges over the number of Gauss points and x_g refers to the location of the g^{th} Gauss point.
  (e) ν_{G} linear equations from the definition of strain at a Gauss point: $\varepsilon(x_g) = u'(x_g) = \hat{\boldsymbol{u}}^T\hat{\boldsymbol{G}}(x_g)$
- NUMBER OF UNKNOWNS: $2\nu_{\mathrm{n}} + 2 + 2\nu_{\mathrm{G}}$
  (a) ν_{n} Internal forces: $f_1^{\mathrm{int}}, \ldots, f_{\nu_{\mathrm{n}}}^{\mathrm{int}}$
  (b) 2 Boundary forces: $\mathcal{F}(0), \mathcal{F}(L)$
  (c) ν_{G} stresses at Gauss points: $\sigma_1, \ldots, \sigma_{\nu_{\mathrm{G}}}$
  (d) ν_{G} strains at Gauss points: $\varepsilon_1, \ldots, \varepsilon_{\nu_{\mathrm{G}}}$
  (e) ν_{n} nodal displacements: $u_1, \ldots, u_{\nu_{\mathrm{n}}}$

The count of equations equals the count of unknowns, making a solution potentially possible. In this list, the only nonlinear set of equations comes from the constitutive model, but nonlinearity anywhere requires the governing system of equations to be solved iteratively, which is the subject of a numerical analysis class. Our goal of demonstrating solvability is accomplished.

1.5 Nonlinear small-deformation transient 1-D bar problem

The following is a cut-n-paste of the static nonlinear analysis. I need to revise it to include the inertial acceleration term, similar to what was done in the linear transient section earlier. SS: can you do the needed modifications here, using the linear transient section as a guide? For the 1-D bar problem with material nonlinearity, the governing equations are

$$\begin{aligned}
&\frac{\mathrm{d}}{\mathrm{d}x}\left[A(x)\sigma(x)\right]+f(x)=0 && \text{on domain } \Omega \text{ defined by } 0<x<L\\
&\sigma(x)=\xi(\varepsilon(x),x) && \text{subject to Robin BCs:}\\
&\varepsilon(x)=\frac{\mathrm{d}u(x)}{\mathrm{d}x} && \alpha_0\mathcal{F}(0)+\beta_0u(0)=\gamma_0\\
& && \alpha_L\mathcal{F}(L)+\beta_Lu(L)=\gamma_L
\end{aligned} \tag{1.67}$$

Here, $\xi(\varepsilon,x)$ is a "blackbox" nonlinear elastic constitutive model giving stress from strain ε. Note that the linear bar problem discussed in the previous section is recovered by choosing $\xi(\varepsilon,x)=E(x)\varepsilon$. In the Robin boundary conditions,

$$\mathcal{F}(0):=-A(0)\sigma(0) \qquad \text{and} \qquad \mathcal{F}(L):=A(L)\sigma(0) \tag{1.68}$$

As in the previous linear bar analysis, these forces are defined to be positive when pointing to the right, in the direction of increasing x, while positive stresses correspond to tension (thus accounting for the negative in the first force definition, since a tensile force on the left end of the bar would point to the left). All other variables in these equations are defined as they were in Eq. (1.1). In what follows, we progress through the same sort of analysis steps as in the linear case, making appropriate adjustments for material nonlinearity.

The strong form of the governing equation in Eq. (1.67) is multiplied by an arbitrary weight function $w(x)$, and then integrated over the domain from $x=0$ to $x=L$:

$$\int_0^L \frac{\mathrm{d}}{\mathrm{d}x}\left[A(x)\sigma(x)\right]w(x)\mathrm{d}x+\int_0^L f(x)w(x)\mathrm{d}x=0 \qquad \forall w(x) \tag{1.69}$$

The first term is integrated by parts to give the so-called **weak form** of the governing equation:

$$w(x)A(x)\sigma(x)\Big|_0^L-\int_0^L w'(x)A(x)\sigma(x)\mathrm{d}x+\int_0^L w(x)f(x)\mathrm{d}x=0 \qquad \forall w(x) \tag{1.70}$$

Using the definition of boundary forces defined in Eq. (1.2), this may be written

$$\int_0^L w'(x)A(x)\sigma(x)\mathrm{d}x=w(0)\mathcal{F}(0)+w(L)\mathcal{F}(L)+\int_0^L w(x)f(x)\mathrm{d}x=0 \tag{1.71}$$

1.5.1 Traditional FEM formulation for the transient nonlinear 1-D bar

1.5.2 Converting the 1-D transient nonlinear FEM solver to an MPM solver

1.5.3 MPM Algorithm for the 1-D transient nonlinear bar

Steve, I didn't get to this section yet, but the previous sections should address many of your questions. The basic algorithm for transient FEM is... Governing equation:

$$\boxed{\hat{\boldsymbol{f}}^{\text{int}}+\hat{\boldsymbol{f}}^{\text{ext}}=\hat{\hat{\boldsymbol{M}}}\hat{\boldsymbol{a}}} \tag{1.72}$$

where $\hat{\boldsymbol{a}}$ is the nodal acceleration, the internal and external force vectors are defined the same as in the previous section except, for transient dynamics, they are *known* at the beginning of the timestep from the initial conditions. The consistent mass matrix is

$$\hat{\hat{\boldsymbol{M}}} = \int_0^L \rho(x)\hat{\boldsymbol{N}}\hat{\boldsymbol{N}}^T \mathrm{d}x \qquad \text{That is, } M_{ij} = \int_0^L \rho(x) N_i(x) \hat{\boldsymbol{N}}_j(x) \mathrm{d}x \tag{1.73}$$

A lumped mass matrix is defined to be a diagonal mass matrix with the i^{th} diagonal component defined to be the i^{th} row sum, so that

$$m_i = \int_0^L \rho(x) N_i(x) \mathrm{d}x \tag{1.74}$$

When the mass matrix is lumped (and hence diagonal), its inverse is trivial. Thus, recalling that both internal and external nodal forces are known at the beginning of a time step, solving the force balance gives nodal accelerations $\hat{\boldsymbol{a}}^0$ at the 0^{th} time step.

In general, given $\hat{\boldsymbol{a}}^n$ at the beginning of a generic n^{th} time step, and given velocity $\hat{\boldsymbol{v}}^n$ at the beginning of the step, the velocity at the end of the step is, with explicit time integration,

$$\hat{\boldsymbol{v}}^{n+1} = \hat{\boldsymbol{v}}^n + \hat{\boldsymbol{a}}^n \Delta t \tag{1.75}$$

and position at the end of the step is, by Taylor series,

$$\hat{\boldsymbol{x}}^{n+1} = \hat{\boldsymbol{x}}^n + \hat{\boldsymbol{v}}^n \Delta t + \frac{1}{2}\hat{\boldsymbol{a}}^n \Delta t^2 \tag{1.76}$$

With this velocity and position (and hence displacement) updated to the end of the step, you can compute the velocity gradient (which is the strain rate in 1-D problems) or the displacement gradient (which is the strain in 1-D problems). One or both of these is typically required by the constitutive model to update the stress $\sigma(x,t)$ at a Gauss point (or at an MPM particle) to the end of the time step, thus setting all initial conditions to start the cycle over to update the state through time to the next step.

WARNING: with this sort of explicit time integrator, the timestep must be set to a value about 1/10 as large as the amount of time it takes an acoustic wave to traverse a grid cell. For small-deformation linear elasticity, acoustic waves travel at a speed given by $\sqrt{E/\rho}$.

1.5.4 Intrinsic no-interpenetration bonus attribute of MPM

Need to add a section of two bars hitting each other

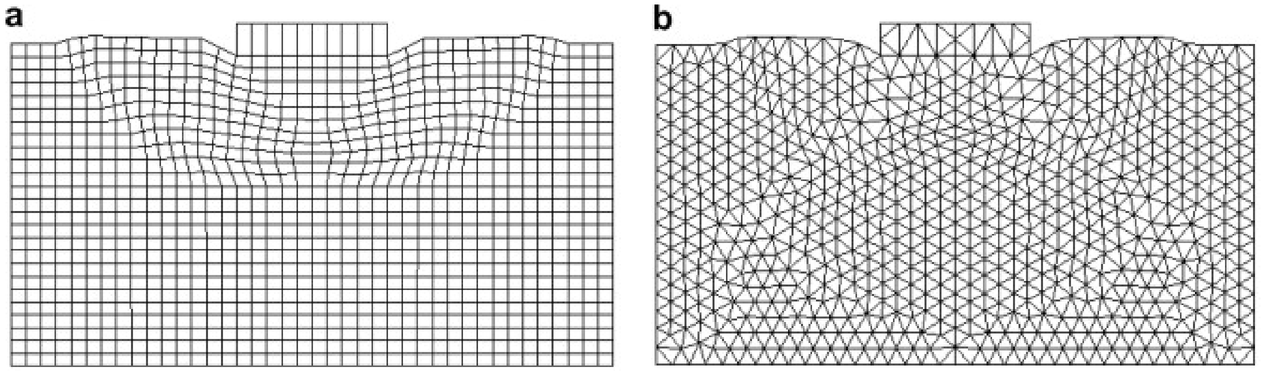


Figure 1.4: (a)quadrilateral mesh (b)triangle mesh. Image permission still needed from [**?**]

1.6 Nonlinear transient 2-D elasticity problems

1.6.1 Traditional FEM notation and formulation for the transient nonlinear 2-D elastica problem

As the reader is assumed to be familiar with the finite-element method (FEM), this section is primarily focused on defining our notational preferences.

1.6.2 2-D FEM grid, elements, nodes, and shape functions

An FEM solution to a differential equation begins by introducing a body-fitted grid that tessellates the spatial domain into **elements**.[23] In 2-D, some FEM codes break the body into quadrilateral elements, while other FEM codes use triangles (see Fig. 1.4). Higher-order methods use increasingly complicated shapes, but all choices have in common that the element shape is determined from special points on the element called **nodes**. The reader is presumed to be very comfortable working with nodes, elements, and **connectivity** (*i.e.,* the mapping of local node numbers on an element to/from global node numbers on the mesh).

The reader is further expected to know that, associated with the i^{th} node is a spatially varying the **nodal basis function** $N_i(\boldsymbol{x})$ (also called the **shape function**[24]), such that a discretized approximation of a field $f(\boldsymbol{x}, t)$, which varies with position $\boldsymbol{x}$ and time t, is[25]

$$\tilde{f}(\boldsymbol{x}, t) = \sum_{i=1}^{\nu_{\text{n}}} \tilde{f}_i(t) N_i(\boldsymbol{x}) \tag{1.77}$$

[23]A **tesselation** breaks up a physical domain, here called a **body**, into a set of subdomains without gaps or overlaps.

[24]Purists use the term **shape function** to refer to the function defined on an element, while **nodal basis function** is defined on the entire domain. Thus, for example, several different linear shape functions are used to construct the single classic "tent" nodal basis function, which certainly is not linear – it is piecewise linear! We will make no such distinction between *shape function* and *nodal basis function*, as the intended meaning is always clear from context.

[25]Here and throughout this primer, broadly applicable equations will be given in their general form for transient 3-D problems, where $\boldsymbol{x}$ denotes the position vector and t is time. The reader is expected to know how to revise these formulas for special case situations such as 1-D problems (where x is just a scalar) and/or for statics problems (where dependence on time is absent). Most of the upcoming examples will be limited to 1-D and 2-D problems for clarity.

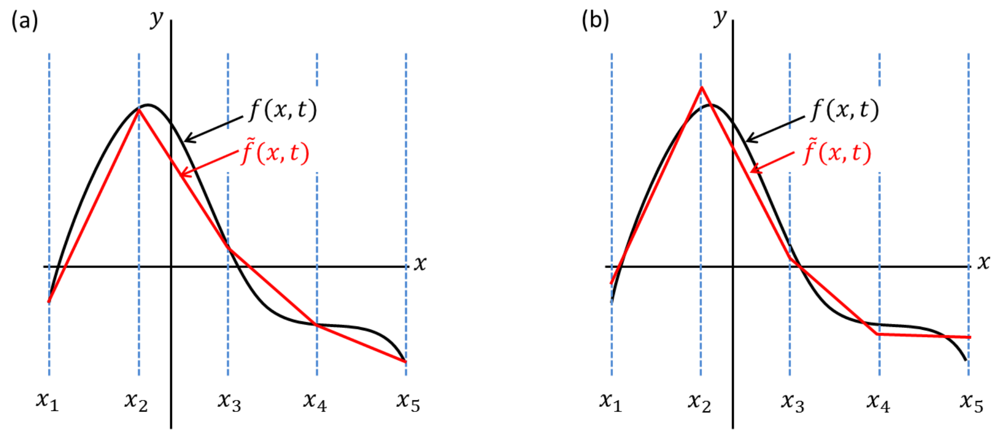


Figure 1.5: Exact function $f(\boldsymbol{x}, t)$ and two choices for a piecewise linear fit $\tilde{f}(\boldsymbol{x}, t)$, differing only by the choice of nodal values $\tilde{f}_i(t)$. (a) Best five-node piecewise-linear fit to a function when "best" corresponds to minimizing nodal errors; in this case, the nodal values exactly match the function values at the nodes: $\tilde{f}_i(t) = f(x_i, t)$. (b) Best piecewise-linear fit when the goal is instead to minimize overall mean square error; in this case $\tilde{f}_i(t) \neq f(x_i, t)$. In both graphs, dashed lines correspond to element boundaries. The piecewise linear fits (in red) use standard linear shape functions $N_i(x)$, which therefore correspond to "tent" nodal basis functions that are often used in FEM analyses.

where a tilde ($\sim$) is used to distinguish between an exact field f and its approximation $\tilde{f}$. The summation over i varies over the number of nodes ν_n, and $\tilde{f}_i(t)$ is the time-varying **nodal value** of the approximate field associated with the i^th node, having a position vector denoted $\boldsymbol{x}_i$.[26] The above expansion is called an **interpolation** if the shape functions satisfy

$$\boxed{\textbf{Kronecker property:} \qquad N_i(\boldsymbol{x}_j) = \delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}} \tag{1.78}$$

The Kronecker property ensures that $\tilde{f}_i(t) = \tilde{f}(\boldsymbol{x}_i, t)$, but not necessarily $\tilde{f}_i(t) = f(\boldsymbol{x}_i, t)$, where (recall) $f(\boldsymbol{x}, t)$ is the exact function and $\tilde{f}(\boldsymbol{x}, t)$ is its approximation. In fact, as illustrated in Fig. 1.5, the best fit might not be an interpolation to that function – it depends on how you define error!

Many FEM textbooks introduce a compact notation for the collection of shape functions and nodal values. Let $\hat{\boldsymbol{N}}(\boldsymbol{x})$ denote the column array containing the shape functions:

$$\hat{\boldsymbol{N}}(\boldsymbol{x}) := \begin{bmatrix} N_1(\boldsymbol{x}) \\ N_2(\boldsymbol{x}) \\ \vdots \\ N_{\nu_\text{n}}(\boldsymbol{x}) \end{bmatrix} \qquad \text{and therefore} \qquad \hat{\boldsymbol{N}}^T(\boldsymbol{x}) = [N_1(\boldsymbol{x}), N_2(\boldsymbol{x}), \ldots, N_{\nu_\text{n}}(\boldsymbol{x})] \, . \tag{1.79}$$

If we similarly define $\hat{\boldsymbol{f}} := [\tilde{f}_1, \ldots, \tilde{f}_{\nu_\text{n}}]^T$, the matrix versions of Eq. (1.77) are then

$$\tilde{f}(\boldsymbol{x}, t) = \hat{\boldsymbol{N}}^T(\boldsymbol{x})\hat{\boldsymbol{f}}(t) = \hat{\boldsymbol{f}}^T(t)\hat{\boldsymbol{N}}(\boldsymbol{x}) \, . \tag{1.80}$$

This notation will be adopted later when analyzing the weak form of the governing equations.

The Kronecker property is not required for a nodal basis function to provide an acceptable numerical approximation to a field. We will, however, require the nodal basis function to satisfy:

$$\boxed{\textbf{Partition of unity:} \qquad \sum_{i=1}^{\nu_\text{n}} N_i(\boldsymbol{x}) \equiv 1 \qquad \forall \boldsymbol{x}} \tag{1.81}$$

[26] which might or might not be equal to the actual field $f(\boldsymbol{x}, t)$ at that node.

When this property holds, $\tilde{f}(\boldsymbol{x},t)$ can be made to exactly represent a constant field by simply selecting each $\tilde{f}_i$ to equal that constant. The following *additional* property is needed to exactly represent a field that varies linearly in space:

$$\boxed{\textbf{Linearly complete:} \quad \sum_{i=1}^{\nu_n} \boldsymbol{x}_i N_i(\boldsymbol{x}) \equiv \boldsymbol{x} \qquad \forall \boldsymbol{x}} \tag{1.82}$$

What is the finite-element method, really?

In order to declare that a numerical solver is using the finite-element method, we assert that each nodal basis function must have all of the following properties:

i. The body must be tesselated into subdomains called elements. To be a tesselation, the elements must not overlap each other, and their union must represent an approximation of the body without gaps. Moreover, the elements must be constructed in a systematic way from discrete points on or inside the body, called nodes, such that each element is formed from a finite set of nodes. Generally one node is used to form multiple elements. Any two elements that share a node are said to be **neighboring elements**.

ii. A field $f(\boldsymbol{x},t)$ must be approximated by $\tilde{f}(\boldsymbol{x},t) := \sum_{1}^{\nu_n} \tilde{f}_i(t) N_i(\boldsymbol{x})$.

iii. Each $N_i(\boldsymbol{x})$ must be associated with nodes on tesselation domains (elements) and be nonzero only over the.

iv. Each $N_i(\boldsymbol{x})$ must satisfy the Kronecker property: $N_i(\boldsymbol{x}_j) = \delta_{ij}$

v. Each $N_i(\boldsymbol{x})$ must satisfy partition of unity: $\sum_{i=1}^{\nu_n} N_i(\boldsymbol{x}) \equiv 1 \qquad \forall \boldsymbol{x}$

vi. Each $N_i(\boldsymbol{x})$ must be linearly complete: $\sum_{i=1}^{\nu_n} \boldsymbol{x}_i N_i(\boldsymbol{x}) \equiv \boldsymbol{x} \qquad \forall \boldsymbol{x}$

FEM notation for the gradient of a scalar field

Let x, y, and z denote the **Cartesian components of the position vector** $\boldsymbol{x}$. Let $\boldsymbol{i}$, $\boldsymbol{j}$, and $\boldsymbol{k}$ be the associated **unit basis vectors**. Let $s(\boldsymbol{x},t)$ denote a scalar-valued function of position $\boldsymbol{x}$ and time t. The gradient of this field is a vector defined by

$$\frac{\partial s}{\partial \boldsymbol{x}} = \boldsymbol{\nabla} s = \frac{\partial s}{\partial x}\boldsymbol{i} + \frac{\partial s}{\partial y}\boldsymbol{j} + \frac{\partial s}{\partial z}\boldsymbol{k} \tag{1.83}$$

Applying this definition to the expansion in Eq. (1.77) gives

$$\frac{\partial \tilde{f}}{\partial \boldsymbol{x}} = \sum_{i=1}^{\nu_n} \tilde{f}_i(t)\boldsymbol{G}_i(\boldsymbol{x}) \qquad \text{where} \qquad \boldsymbol{G}_i(\boldsymbol{x}) := \frac{\partial N_i}{\partial \boldsymbol{x}} \tag{1.84}$$

Equivalently, using the "nabla" (del) notation,

$$\boldsymbol{\nabla}\tilde{f}(\boldsymbol{x},t) = \sum_{i=1}^{\nu_n} \tilde{f}_i(t)\boldsymbol{G}_i(\boldsymbol{x}) \qquad \text{where} \qquad \boldsymbol{G}_i(\boldsymbol{x}) := \boldsymbol{\nabla} N_i(\boldsymbol{x}) \tag{1.85}$$

Let us now introduce a matrix $\hat{\hat{\boldsymbol{G}}}$ whose i^{th} row contains the components of the gradient of the i^{th} shape function:

$$\hat{\hat{\boldsymbol{G}}} := \begin{bmatrix} \frac{\partial N_1}{\partial x} & \frac{\partial N_1}{\partial y} & \frac{\partial N_1}{\partial z} \\ \frac{\partial N_2}{\partial x} & \frac{\partial N_2}{\partial y} & \frac{\partial N_2}{\partial z} \\ & \vdots & \\ \frac{\partial N_{\nu_n}}{\partial x} & \frac{\partial N_{\nu_n}}{\partial y} & \frac{\partial N_{\nu_n}}{\partial z} \end{bmatrix} \tag{1.86}$$

Recalling that $\hat{\boldsymbol{f}}$ denotes the $\nu_\mathrm{n} \times 1$ array containing nodal values of the field, the matrix notation for Eq. (1.84) is

$$\underset{1\times 3}{\frac{\partial \tilde{f}}{\partial \boldsymbol{x}}} = \underset{1\times\nu_\mathrm{n}}{\hat{\boldsymbol{f}}^T(t)}\underset{\nu_\mathrm{n}\times 3}{\hat{\hat{\boldsymbol{G}}}(\boldsymbol{x})} \tag{1.87}$$

or, taking the transpose of both sides,

$$\underset{3\times 1}{\frac{\partial \tilde{f}}{\partial \boldsymbol{x}}} = \underset{3\times\nu_\mathrm{n}}{\hat{\hat{\boldsymbol{G}}}^T(\boldsymbol{x})}\underset{\nu_\mathrm{n}\times 1}{\hat{\boldsymbol{f}}(t)} \tag{1.88}$$

Here, the undersets show the dimensions of each matrix for clarity. Since $\boldsymbol{\nabla} f$ is a physical vector having three components, the above two equations clarify that these components may be arranged as either a 3×1 or 1×3 array.

FEM notation for vector fields and their gradients

Let $\boldsymbol{u} = u_x\boldsymbol{i} + u_y\boldsymbol{j} + u_z\boldsymbol{k}$ denote a generic vector (like displacement) in 3-D. Suppose that this generic vector is a field, $\boldsymbol{u}(\boldsymbol{x}, t)$, which is described approximately in terms of the basis functions by

$$\tilde{\boldsymbol{u}}(\boldsymbol{x}, t) \approx \sum_{i=1}^{\nu_\mathrm{n}} \tilde{\boldsymbol{u}}_i(t) N_i(\boldsymbol{x}) \tag{1.89}$$

We define the 3-D component array for the left-hand side of this equation vector using a "single hat" notation:

$$\hat{\boldsymbol{u}}(\boldsymbol{x}, t) := \begin{bmatrix} \tilde{u}_x(\boldsymbol{x}, t) \\ \tilde{u}_y(\boldsymbol{x}, t) \\ \tilde{u}_z(\boldsymbol{x}, t) \end{bmatrix} \tag{1.90}$$

For each node, there is an associated nodal vector $\tilde{\boldsymbol{u}}_i(t)$. This manuscript will use a DOUBLE-hat notation to refer to the $\nu_\mathrm{n} \times 3$ matrix whose i^th row contains the three components $\boldsymbol{u}_i$. That is,

$$\hat{\hat{\boldsymbol{u}}}(t) := \begin{bmatrix} \tilde{u}_{1x}(t) & \tilde{u}_{1y}(t) & \tilde{u}_{1z}(t) \\ \tilde{u}_{2x}(t) & \tilde{u}_{2y}(t) & \tilde{u}_{2z}(t) \\ & \vdots & \\ \tilde{u}_{\nu_\mathrm{n}x}(t) & \tilde{u}_{\nu_\mathrm{n}y}(t) & \tilde{u}_{\nu_\mathrm{n}z}(t) \end{bmatrix} \tag{1.91}$$

Thus, recalling that Eq. (1.79), the matrix form of Eq. (1.89) is

$$\hat{\boldsymbol{u}} = \hat{\hat{\boldsymbol{u}}}^T(t)\hat{\boldsymbol{N}}(\boldsymbol{x}) \tag{1.92}$$

As a special case, let $\hat{\boldsymbol{x}}$ denote the component array of the position vector $\boldsymbol{x}$, and let $\hat{\hat{\boldsymbol{x}}}$ denote the matrix of node locations as follows:

$$\hat{\boldsymbol{x}} := \begin{bmatrix} x \\ y \\ z \end{bmatrix} \qquad \text{and} \qquad \hat{\hat{\boldsymbol{x}}} := \begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ & \vdots & \\ x_{\nu_\mathrm{n}} & y_{\nu_\mathrm{n}} & z_{\nu_\mathrm{n}} \end{bmatrix} \tag{1.93}$$

Then the matrix form of Eq. (1.82) is

$$\boxed{\textbf{Linearly complete:} \quad \hat{\boldsymbol{x}} = \hat{\hat{\boldsymbol{x}}}^T\hat{\boldsymbol{N}}(\boldsymbol{x})} \tag{1.94}$$

The gradient of a generic vector field $\boldsymbol{u}(\boldsymbol{x},t)$ is a second-order tensor, denoted $\partial\boldsymbol{u}/\partial\boldsymbol{x}$, having a 3×3 component matrix given by

$$\left[\frac{\partial\boldsymbol{u}}{\partial\boldsymbol{x}}\right] := \begin{bmatrix} \frac{\partial u_x}{\partial x} & \frac{\partial u_x}{\partial y} & \frac{\partial u_x}{\partial z} \\ \frac{\partial u_y}{\partial x} & \frac{\partial u_x}{\partial y} & \frac{\partial u_y}{\partial z} \\ \frac{\partial u_z}{\partial x} & \frac{\partial u_z}{\partial y} & \frac{\partial u_z}{\partial z} \end{bmatrix} \tag{1.95}$$

Applying this definition, the gradient of the *approximate* vector field is evaluated by the matrix multiplication

$$\underset{3\times3}{\frac{\partial\tilde{\boldsymbol{u}}}{\partial\boldsymbol{x}}} = \underset{3\times\nu_{\mathrm{n}}}{\hat{\hat{\boldsymbol{u}}}^T(t)}\underset{\nu_{\mathrm{n}}\times3}{\hat{\hat{\boldsymbol{G}}}(\boldsymbol{x})} \tag{1.96}$$

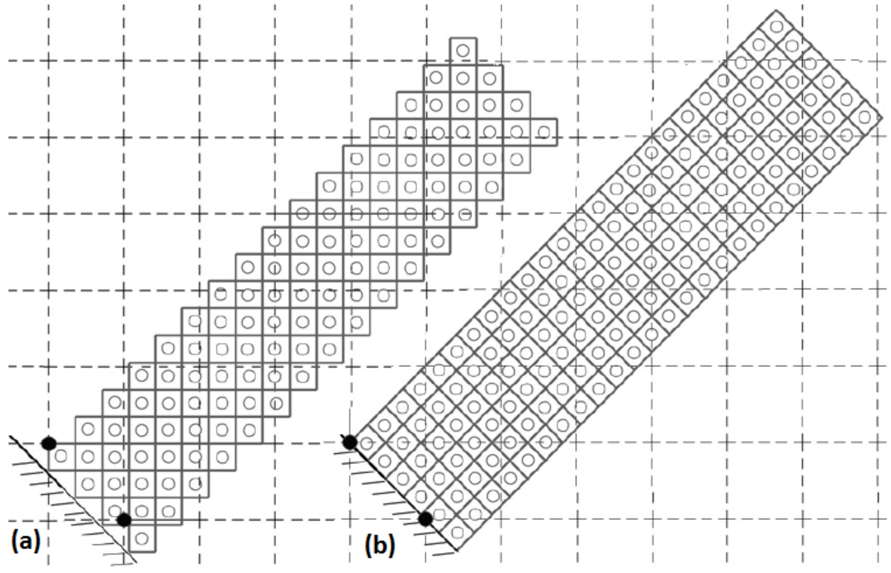


Figure 1.6: MPM particles (small hollow circles) distributed within an overlaid rectilinear grid (dashed lines). The rectangles around the particles represent the physical domain of material occupied by each particle. (a) A pixelated description of an angled beam has a jagged (stair-stepped) boundary, which will give errors in an MPM simulation that would be comparable to using FEM with the same jagged mesh. (b) So-called "conforming" particle domains, are shown here as tilted rectangles, and these might be more complicated convex polygons in more complicated geometries.

description of the field described by using nodal basis functions (also called element shape functions[27]) and nodal values mapping from discrete on the FEM grid to any

, more generally any convex polygon, to represent particle domains, which more accurately represents the boundary and thus gives lower errors on the same mesh resolution. In both (a) and (b), note that the particle density is generally

Because this tutorial treats MPM as an extension of the finite-element method, you should be able to convert an existing conventional FEM code to an MPM code without much difficulty.[28] Some reasons to add an MPM option to an FEM code are:

- Simulations can be run on complicated geometries, such as intricate porous structures, without the need for difficult material-based nodal connectivity tables that are used in conventional FEM to track material adjacent to any given element.

- The geometry for an MPM simulation can be initialized directly from voxel information from a computed tomography (CT) scan.[29] A voxel-based representation of a body treats the body as a collection of cuboids, somewhat like a Lego model of a structure. Such a representation is possible with both FEM and MPM (each having comparable errors caused by poor description of angled boundaries, such as those evident in Fig. 1.6.2), so the real advantage of MPM is that it does not require defining *material* connectivity information. Accordingly, an update to connectivity is not required to model subsequent material fracture.

- With MPM, the governing equations may be solved on a simple rectilinear grid, rather than requiring

[27]Purists prefer the phrase **element shape function** to refer to the function that applies on an element, while the nodal basis function applies over the entire domain. Thus, for example, a linear shape function has the equation of a straight line, but the nodal basis function is the assembly of shape functions into the form of a so-called "tent function," which is certainly not linear – it is piecewise linear.

[28]At Sandia National Laboratories, for example, a mature FEM code was converted to MPM in only one week, and simulations were run and visualized only a week after that.

[29]A voxel is the 3-D analog of a pixel in 2-D.

the body-fitted meshes used in traditional FEM.[30]

- FEM formulations save material constitutive data at Gauss points with displacement, velocity, and acceleration at nodes. Under conditions of extremely large deformations, an FEM formulation must remesh to remove excessive element distortion. This process of remeshing then must be followed by remapping of the material constitutive data to the new locations of Gauss points. The remapping phase (which is essentially an averaging process) results in so-called *advection* errors that can corrupt the physical meanings of material constitutive model data.[31] An MPM formulation saves all field variables (constitutive and kinematic) at material particles. These particles represent the body and flow through the overlaid grid, carrying all of their particle data with them. When an FEM code is revised to become an MPM code, the key algorithm adjustment is the insertion of a phase that maps data from particles to grid to solve the governing differential equations on the grid, followed by a mapping of the solution from the grid to particles to finish updating the state. Because MPM formulations save data at particles, and because those particles may flow an arbitrary distance through the overlaid mesh, the MPM can model very large deformations without any advection errors.

1.6.3 Converting the 2-D transient nonlinear FEM solver to an MPM solver

1.6.4 MPM Algorithm for the 2-D transient nonlinear bar

1.7 Intrinsic no-interpenetration bonus attribute of MPM

We need to add the classic colliding disk problem here.

[30]Optionally, a body-fitted mesh may still be used to define the body in MPM while simultaneously using a simple overlaid rectilinear grid for the field equations, and doing so can significantly improve MPM accuracy. Furthermore, a non-rectilinear mesh may be used with the MPM, but doing so entails additional computational cost.

[31]For example, mixing stress states in a plasticity model model might produce a new stress state violating the yield condition of the model. A fiber composite model that has the fiber direction (a unit vector) saved as a constitutive internal variable, then attempting to average these directions generally produces a vector that is no longer of unit length. While ad hoc workarounds can be usually found to assign physically admissible rezoned constitutive variables, those methods are rarely robust and certainly not defensible from a physical standpoint. The efforts to find workarounds for advection errors are so time consuming that it is far better to implement new methods, like MPM, that eliminate the need for advection corrections altogether.

1.8 Nonlinear transient 2-D mechanics

1.8.1 Traditional FEM formulation for the transient nonlinear 2-D mechanics

1.8.2 Converting the 2-D transient nonlinear FEM solver to an MPM solver

1.8.3 MPM Algorithm for 2-D transient nonlinear solid mechanics