Advances in Engineering Software 139 (2020) 102748

Contents lists available at ScienceDirect

Advances in Engineering Software

journal homepage: www.elsevier.com/locate/advengsoft

Research paper

AMPLE: A Material Point Learning Environment

William M. Coombs*, Charles E. Augarde

Department of Engineering, Durham University, UK

ARTICLE INFO

Keywords:
Material point method
Elasto-plasticity
Finite deformation mechanics
MATLAB
Researcher development

ABSTRACT

The Material Point Method (MPM) is a computational tool ideally suited to modelling solid mechanics problems involving large deformations where conventional mesh-based methods struggle. Explicit and implicit formulations are available, but for both the learning curve for understanding the method and arriving at a useful implementation is severe. Researchers must understand and implement finite element analysis, non-linear material behaviour, finite deformation mechanics and non-linear solution methods before they can even verify their formulations. This issue represents a significant barrier for post-doctoral researchers, graduate students and undergraduate students to start working with (and understanding) the method. This paper presents A Material Point Learning Environment (AMPLE) based around implicit variants of the method, with the aim of softening this steep learning curve via MATLAB-based, accessible and compact scripts. The code is freely available from github.com/wmcoombs/AMPLE.

1. Introduction

Mesh-based Lagrangian approaches such as the finite element method dominate the analysis of solid mechanics problems. However there are issues with these methods for problems involving very large deformation in that the discretisation used (i.e. the mesh) becomes distorted, leading to inaccurate results and in extreme cases, eventual breakdown of the numerics due to element inversion. Various attempts have been made to address this issue within mesh-based methods themselves, notably with Arbitrary Lagrangian Eulerian (ALE) methods [2] where a standard Lagrangian step determines material displacements and is followed by an Eulerian step to generate a new mesh for the deformed problem domain. Apart from the complexity of the approach, which is off-putting, the major error source in using ALE methods comes in remapping of variables between the two meshes [27]

The Material Point Method (MPM) is an alternative to pure Lagrangian approaches and is well suited to problems involving very large deformations. The method was developed in the 1990s by Sulsky et al. [31] as a solid mechanics extension to the FLuid Implicit Particle (FLIP) method [4] which itself was developed from the Particle-In-Cell (PIC) method [15]. The basic idea of the method is to discretise a problem domain with material points (sometimes called particles) which carry information (mass, volume, stress, state variables, etc.). The information is then mapped to the nodes of a regular background finite element grid where calculations are carried out. The results are mapped back to the material points and the mesh is then discarded. This cycle is then repeated, each step using a regular mesh to conduct the calculations thus avoiding any issues with mesh distortion, as translation is recorded only at the material points. The MPM has been used for a variety of solid mechanics problems such as fracture [16,23], contact [21], machining [13] and impact [34]. Recently the method has caught the particular interest of the geotechnical community for the modelling of landslides, where coupled displacement-pressure formulations of the MPM have been used to model saturated and unsaturated soils, examples include [29,35]. Outside solid mechanics it is interesting to note that the MPM has also been used to create animations for films [18,30].

In the standard MPM, material points contribute stiffness to the nodes of the grid in which they are currently located. When a material point translates from one grid cell to another (often termed “cell-crossing”) the jump in stiffness can cause problems such as oscillations in the stress resultants and consequential instability in the solution. For this reason, MPMs have been developed where each material point “owns” a finite subvolume of the problem domain, which in some methods can itself change shape. These so-called advanced domain methods include the Generalised Interpolation material point method (GIMPM), [1] and the more recent Convected Particle Domain Interpolation (CPDI) methods [25,26].

The material point method was initially developed in an explicit format for purely dynamic problems where material stiffness is ignored and a simple forward difference solution obtained for the particle velocities over a number of time steps. Providing the mass matrix is

* Corresponding author.
E-mail address: w.m.coombs@durham.ac.uk (W.M. Coombs).

https://doi.org/10.1016/j.advengsoft.2019.102748
Received 17 May 2019; Received in revised form 23 July 2019; Accepted 22 October 2019

diagonal, an explicit approach avoids the need to solve a linear system (as required by an implicit approach) however, there are well-known restrictions on the step size possible without loss of stability, and a lack of error control. The first implicit MPM appeared in [14], and a semi-implicit approach for dynamics was presented in [32]. In the latter the cost of forming the precise material tangent matrix was partially avoided with an approximation, and an iterative method was then used to solve the linear system of equations. A glance at citations for the landmark papers in explicit and implicit MPM (i.e. [14] and [31]) indicates a continuing preference for explicit MPM, however the advantages of an implicit approach when working in some areas of modelling, such as materials with sensitive material nonlinearity, has not been fully highlighted.

Guilkey and Weiss [14] highlight the clear links between implicit MPM and standard finite elements, and indeed, one way to summarise the implicit MPM is

> *a finite element method where the integration points (material points) are allowed to move independently of the mesh.*

The purpose of this paper is to introduce a Matlab-based "learning environment" for those wishing to explore the MPM, based on the authors' own experiences working with students and other researchers over the past five years.

1.1. Material point learning curve

Despite the clear links between the material point method and the finite element method, the learning curve for researchers to arrive at a *useful* material point method code is far more severe. The authors' interpretation of the learning curve for those attempting to understand, and implement, the material point method is shown in Fig. 1. The starting point for any researcher looking to use/investigate the material point method is to understand finite elements. However, even if a small strain assumption and linear material behaviour are adopted, the material point method is still a globally non-linear method; each time step will be linear but the overall response will be non-linear due to changes in stiffness as material points move through the background mesh. This means that, in order to obtain physically meaningful results, researchers are forced to adopt a geometrically non-linear analysis framework and understand both finite deformation mechanics and non-linear solution methods (implicit or explicit). In addition to this, the types of problems that are normally investigated using the material point method (geotechnical failures, impact, fragmentation, etc.) also require non-linear material (or constitutive) models to be incorporated into the code. Once all of these ingredients have been assembled one can undertake *useful* material point analysis and start to add additional features, such as advanced boundary conditions, etc.

1.2. Development principles

AMPLE has been developed as an environment through which researchers, especially PhD/MSc students, can understand the material point method. The development was guided by the following principles:

- **MATLAB-based** - as discussed in the previous section, the material point method is scientifically complex and if users/developers also have to understand thousands of lines of Fortran/C/C++ code the hurdle to its use may become insurmountable. AMPLE has been developed in MATLAB to remove, or at least significantly lessen, the syntax learning curve and allow researchers to concentrate on understanding the key elements of the material point method.
- **Compact** - AMPLE's implementation was inspired by Trefethen's [33] philosophy of ten digit algorithms

  > *"... a little gem of a program to compute something numerical: Ten digits, Five seconds, And just one page"*

  The basic idea is that if a program is compact enough to be viewed on a single page it allows the structure of the whole algorithm to be understood and visualised, making mistakes and misunderstandings far less likely. Each of AMPLE's scripts/functions have been written so that they are readable on a single A4 page (or computer screen), this allows users to understand the structure of each of the code segments.
- **Modular & expandable** - AMPLE has been written as a series of compact functions called by a core analysis script. For example, the link between the material points and the background mesh is contained within one function and all of the other functions remain unchanged if the background mesh is changed. The continuum mechanics formulation is contained in another function and the material model (stress-strain relationship) in another. Although the initial AMPLE release is focused on two-dimensional analysis, all of the continuum mechanics is implemented for three-dimensional analysis making it straightforward to reduce or increase the code to one and three-dimensions, respectively. This allows users/

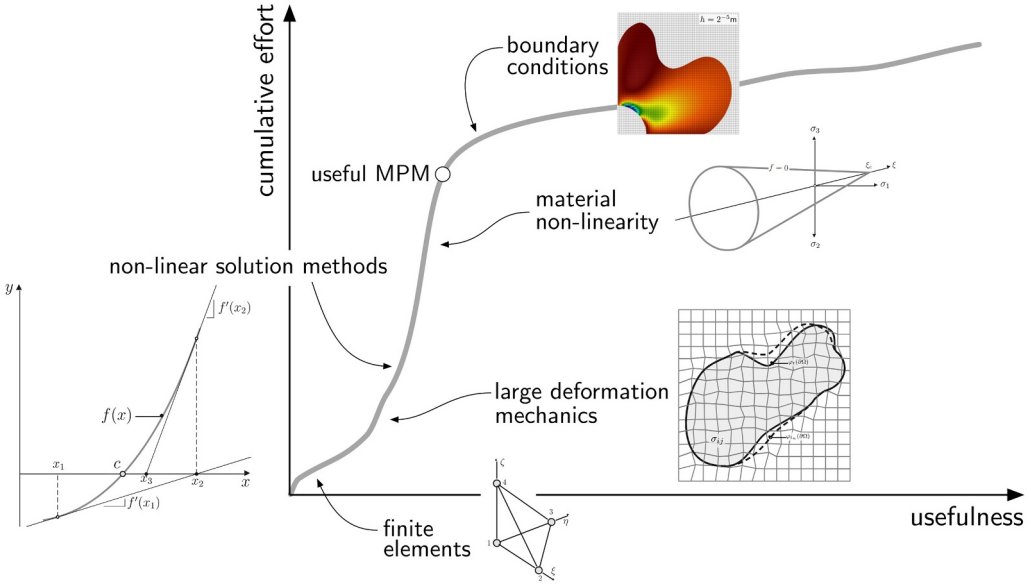

Fig. 1. Material point method learning curve.

developers to quickly understand the purpose of each of the functions and adapt/modify/replace functions as required.
- **Rigorous** - all aspects of AMPLE have been verified and are based on a rigorous updated Lagrangian continuum mechanics framework [9]. The implemented algorithms are based on published material [5,6,11], including convergence analysis to ensure that the fundamentals of the code are sound. This should provide confidence when using/expanding AMPLE.
- **Proof rather than performance** - it is well documented that MATLAB is not as computationally efficient as compiled languages (such as Fortran or C/C++). However, the focus of AMPLE is on proof of new concepts and ideas, not high performance computations. It provides a environment for researchers to understand the material point method and test out new ideas and explore the impact of these changes on the performance of the method.

1.3. Paper layout

The layout of the remainder of the paper is as follows: Section 2 details, as concisely as possible, the material point formulation implemented within AMPLE; Section 3 describes some of the implementation aspects associated with this formulation, such as the basis functions of the implemented MPMs, amongst other details; Section 4 details AMPLE's code structure including function formats and data structures; Section 5 provides three demonstration cases that explore key aspects of the AMPLE environment; finally, Section 6 briefly concludes the paper. A mix of index and matrix-vector notation is used for the continuum formulation and numerical implementation aspects of the paper, respectively.

2. Material point formulation

AMPLE is an implementation of a quasi-static implicit finite deformation elasto-plastic material point method based on an updated Lagrangian formulation[1] defined by the following weak statement of equilibrium

$$\int_{\varphi_t(\Omega)} (\sigma_{ij}(\nabla_x \eta)_{ij} - b_i \eta_i) \mathrm{d}v - \int_{\varphi_t(\partial\Omega)} (t_i \eta_i) \mathrm{d}s = 0. \tag{1}$$

φ_t is the motion of the material body with domain, Ω, which is subjected to tractions, t_i, on the boundary of the domain (with surface, s), $\partial\Omega$, and body forces, b_i, acting over the volume, v of the domain, which lead to a Cauchy stress field, σ_{ij}, through the body. The weak form is derived in the current frame assuming a field of admissible virtual displacements, η_i.

Note that AMPLE does not include tractions so the surface integral term in (1) is neglected in what follows. Unlike finite element methods, the imposition of tractions in the standard material point method is not trivial and requires a representation of the physical boundary to be constructed; see the work of Bing et al. [3] and Remmerswaal [24] for methods of imposing traction boundary conditions in the material point method.

2.1. Finite deformation mechanics

In finite deformation mechanics, the deformation gradient, F_{ij}, provides the fundamental link between the original and deformed configurations

$$F_{ij} = \frac{\partial x_i}{\partial X_j}, \tag{2}$$

where X_i are the original (reference) coordinates and $x_i = \varphi(X_i, t)$ are the updated coordinates in the current (deformed) body, where φ is the motion of the body. It is assumed that the deformation gradient can be multiplicatively decomposed into elastic and plastic components [19,20]

$$F_{ij} = F^{\mathrm{e}}_{ik} F^{\mathrm{p}}_{kj}, \tag{3}$$

where the superscripts e and p denote the elastic and plastic components. In this paper we adopt logarithmic strains and Kirchhoff stresses and combine these measures with an exponential map of the plastic flow rule to allow the use of conventional small-strain stress integration algorithms with a finite deformation framework. This is a powerful combination as it allows existing constitutive formulations to be used directly rather than reformulating them for the particular choice of stress and strain measures used in the large deformation mechanics [17,28]. Within this formulation, the elastic logarithmic strain is defined as

$$\varepsilon^{\mathrm{e}}_{ij} = \frac{1}{2} \ln(b^{\mathrm{e}}_{ij}), \qquad \text{where} \qquad b^{\mathrm{e}}_{ij} = F^{\mathrm{e}}_{ik} F^{\mathrm{e}}_{jk} \tag{4}$$

is the left elastic Cauchy-Green strain and the Kirchhoff stress, τ_{ij}, can be obtained using

$$\tau_{ij} = D^{\mathrm{e}}_{ijkl} \varepsilon^{\mathrm{e}}_{kl}, \tag{5}$$

where is the linear elastic stiffness matrix. The Cauchy stress can be obtained from the Kirchhoff stress through

$$\sigma_{ij} = \frac{1}{J} \tau_{ij}, \qquad \text{where} \qquad J = \det(F_{ij}) \tag{6}$$

is the volume ratio between the deformed and reference configurations. In order to advance the non-linear solution, the finite deformation equations are discretised in pseudo-time by imposing the deformation over a number of load (or pseudo-time) steps. This allows the current deformation gradient to be defined using

$$F_{ij} = \Delta F_{ik} (F_n)_{kj}, \tag{7}$$

where ΔF_{ij} is the increment in the deformation gradient between the previously converged state, denoted using a subscript n, and the current state. In order to obtain the updated Kirchhoff stress state for the current deformation gradient, a constitutive model requires an initial estimate (or trial) of the elastic strain (or stress) state. In this approach the trial elastic Cauchy-Green strain tensor is given by

$$(b^{\mathrm{e}}_t)_{ij} = \Delta F_{ik} (b^{\mathrm{e}}_n)_{kl} \Delta F_{jl}, \tag{8}$$

where the subscript t denotes a quantity defined in the trial state. The previous elastic Cauchy-Green strain tensor, $(b^{\mathrm{e}}_n)_{ij}$, can be obtained from the previous elastic strain state through

$$(b^{\mathrm{e}}_n)_{ij} = \exp(2(\varepsilon^{\mathrm{e}}_n)_{ij}) \tag{9}$$

and the trial elastic strain state follows as

$$(\varepsilon^{\mathrm{e}}_t)_{ij} = \frac{1}{2} \ln((b^{\mathrm{e}}_t)_{ij}). \tag{10}$$

The adopted constitutive algorithm can then be used to return the updated elastic strain, $\varepsilon^{\mathrm{e}}_{ij}$, and Kirchhoff stress, τ_{ij}, states.

2.2. Constitutive formulations & stress updating

AMPLE includes two different constitutive models: (i) Isotropic linear elasticity and (ii) isotropic linear elasticity with perfect plasticity and a von Mises yield surface with associated plastic flow. In the case of isotropic linear elasticity, the updated elastic strain, $\varepsilon^{\mathrm{e}}_{ij}$, is equal to the elastic trial strain, $(\varepsilon^{\mathrm{e}}_t)_{ij}$, and the Kirchhoff stress is given by (5).

The elasto-plastic von Mises model is integrated using an implicit elastic predictor, plastic corrector algorithm. The von Mises yield function can be defined as

[1] AMPLE adopts the same formulation as implemented in the generalised interpolation approach of Charlton et al. [5] but has been included here for completeness.

$$f = \frac{\rho}{\rho_y} - 1 = 0, \tag{11}$$

where $\rho = \sqrt{2J_2}$, $J_2 = s_{ij}s_{ji}/2$, $s_{ij} = \tau_{ij} - \tau_{kk}\delta_{ij}/3$ and ρ_y is the yield strength of the material. Within this algorithm the elastic trial strain, $(\varepsilon_t^e)_{ij}$, acts as the initial estimate for the updated elastic strain state. If the corresponding trial Kirchhoff stress is inside the von Mises yield envelope ($f < 0$), then the material is undergoing elastic behaviour and the updated elastic strain is equal to the trial state. If the trial stress state is outside of the yield envelope then it must be corrected back onto the yield surface. AMPLE uses a backward Euler procedure to perform this correction, details of which can be found in [6], amongst others.

2.3. Discrete material point formulation

The Galerkin form of the weak statement of equilibrium over each background grid cell, E, can be obtained from (1) as

$$\int_{\varphi_t(E)} [\nabla_x S_{vp}]^T \{\sigma\} \mathrm{d}v - \int_{\varphi_t(E)} [S_{vp}]^T \{b\} \mathrm{d}v = \{0\}, \tag{12}$$

where $[\nabla_x S_{vp}]$ is the tensorial form of the strain-displacement matrix containing derivatives of the shape functions with respect to the updated coordinates.[2] The first term in (12) is the internal force within an grid cell and the second term is the external force vector, in this case the body forces. AMPLE adopts a standard implicit Newton–Raphson (N-R) procedure to solve (12) for the unknown nodal displacements. This requires (12) to be linearised with respect to the nodal displacements to arrive at the stiffness of each cell in the background mesh

$$[k^E] = \int_{\varphi_t(E)} [\nabla_x S_{vp}]^T [a] [\nabla_x S_{vp}] \mathrm{d}v, \tag{13}$$

where $a_{ijkl} = (\partial\sigma_{ij}/\partial F_{km})F_{lm}$ is the spatial consistent tangent modulus for a point within the background grid cell (see Charlton et al. [5] for details).

In material point methods the physical domain is discretised by a number of material points. These points are used to numerically approximate the stiffness (13) of the grid cells in the background mesh, essentially replacing the conventional Gauss-Legendre points (or other integration method). The key difference between material point and finite element methods is that these integration points move relative to the background mesh rather than being directly coupled to the positions of the background grid nodes. The stiffness contribution of a single material point to the background mesh is

$$[k_p] = [\nabla_x S_{vp}]^T [a] [\nabla_x S_{vp}] V_p, \tag{14}$$

where V_p is the volume associated with the material point in the spatial (updated) frame

$$V_p = \det(\Delta F_{ij}) V_p^n = \det(F_{ij}) V_p^0, \tag{15}$$

where V_p^n and V_p^0 are the volumes associated with the material point in the previously converged state and the initial configuration, respectively. Comparing (14) with conventional finite element literature, the volume of the material point, V_p, replaces the Gauss–Legendre quadrature weight and the determinant of the Jacobian that maps between the local and global frames. The product of the determinant of the Jacobian and the Gauss point weight provides the physical volume associated with the Gauss point which is equivalent to V_p in the material point method. The internal force contribution of a single material point to the background mesh is

$$\{f^p\} = [\nabla_x S_{vp}]^T \{\sigma\} V_p. \tag{16}$$

Following the work of [5,10], the increment in the deformation gradient is obtained from

$$\Delta F_{ij} = \delta_{ij} + \frac{\partial \Delta u_i}{\partial \tilde{X}_j} = \delta_{ij} + \sum_{v=1}^{n} (\Delta u_v)_i \frac{\partial (S_{vp})}{\partial \tilde{X}_j}, \tag{17}$$

where Δu_i is the displacement increment within the current load step, $\tilde{X}_j$ are the coordinates at the start of the load step and n is the number of nodes that influence the material point. This allows the increment in the deformation gradient to be obtained from derivatives of the basis functions based on the coordinates of the nodes at the start of the load step. The spatial derivatives of the basis functions can subsequently be calculated using the method proposed by Charlton et al. [5], that is

$$\frac{\partial (S_{vp})}{\partial x_j} = \frac{\partial (S_{vp})}{\partial \tilde{X}_i} \frac{\partial \tilde{X}_i}{\partial x_j} = \frac{\partial (S_{vp})}{\partial \tilde{X}_i} (\Delta F_{ij})^{-1}. \tag{18}$$

It is essential that the spatial derivatives are used in the strain-displacement matrix, $[\nabla_x S_{vp}]$, to both ensure the correct order of convergence in the N-R process and convergence towards the correct solution based on the internal force contribution, (16), of each material point [5,10].

It is not necessary to map the basis functions between the start of the load step and the current configuration because it is assumed that, during a load step, the displacement of a material point, $(\Delta u_p)_i$, is linked to the nodal displacements through the basis functions, S_{vp}, that is

$$(\Delta u_p)_i = \sum_{v=1}^{n} S_{vp} (\Delta u_v)_i, \tag{19}$$

where n is the number of nodes that influence the material point and $(\Delta u_v)_i$ are the displacement of the nodes over the current load step. Therefore the basis functions evaluated at the start or the end of the load step are identical (see [9] for a detailed discussion on this point).

3. Implementation aspects

This section presents key information regarding the implementation of AMPLE. This includes the basis functions for both the standard and generalised interpolation material point methods, boundary conditions and material point position/domain updating. An outline of the computational procedure is also given at the end of the section.

3.1. Basis functions

The basis functions are one of the key areas where the material point method and the finite element method diverge. For the standard material point method the basis functions of a material point are simply the basis functions of the underlying finite element grid. In this case, one way to obtain the basis functions (and the spatial derivatives of these functions, if required) is to calculate the local position of the material point within its associated background grid cell and then evaluate the basis functions as in the finite element method. However, if the background grid comprises a regular grid with the cell boundaries aligned with the global coordinate system, it is straightforward to evaluate directly the basis functions of a material point. This is the approach taken in this paper as the initial release of AMPLE assumes that the background grid consists of regular two-dimensional bi-linear quadrilateral grid cells with their edges aligned with the global Cartesian coordinates. With this assumption, the basis functions for the both the standard and generalised interpolation material point methods can be expressed as the convolution of material point's domain with the basis functions of the underlying finite element grid, that is

$$S_{vp} = \frac{1}{V_p^n} \int_{\Omega_p} \chi_p N_v(\tilde{X}_p) dx, \tag{20}$$

where Ω_p is the influence domain associated with the material point, χ_p is the material point's characteristic function which defines the zone of influence (or domain) of a material point, N_v are the underlying shape

[2] $[\nabla_x S_{vp}]$ is essentially the same as the conventional strain-displacement $[B]$ matrix found in finite element literature and $[S_{vp}]$ is the equivalent of the shape function matrix, often denoted $[N]$.

functions of the finite element grid which are dependent of the position of the material point at the start of the load step, $\tilde{X}_p$. The basis function is only given in one-dimension; the extension to higher dimensions is obtained through the Cartesian product of the shape functions in each direction.[3] The gradient of the basis function can be expressed as

$$\nabla_{\tilde{X}} S_{vp} = \frac{1}{V_p^n} \int_{\Omega_p} \chi_p \nabla_{\tilde{X}} N_v(\tilde{X}_p) dx. \tag{21}$$

For multi-dimension problems the gradient of the basis functions are obtained from the product of the gradient in one direction with the shape function in the other direction, for example the derivative of the basis functions with respect to $\tilde{X}$ in a two dimensional problem is

$$\frac{\partial S_{vp}(\tilde{X}, \tilde{Y})}{\partial \tilde{X}} = \nabla_{\tilde{X}} S_{vp}(\tilde{X}) \times S_{vp}(\tilde{Y}). \tag{22}$$

The following sections provide the basis functions for the standard and generalised interpolation material point methods.

3.1.1. Standard interpolation

The basis functions for the standard material point method are obtained by replacing the characteristic function, χ_p, with a Dirac delta function. With this substitution, (20) becomes

$$\begin{aligned} S_{vp} &= 1 + (\tilde{X}_p - \tilde{X}_v)/h \qquad -h < \tilde{X}_p - \tilde{X}_v \leq 0 \\ S_{vp} &= 1 - (\tilde{X}_p - \tilde{X}_v)/h \qquad 0 < \tilde{X}_p - \tilde{X}_v \leq h, \end{aligned} \tag{23}$$

where h is the size of the background grid (distance between the nodes in each direction) and $\tilde{X}_v$ is the position of the node (or vertex) associated with the basis function at the start of the load step. The gradients of the basis functions with respect to the material point position are

$$\begin{aligned} \nabla_{\tilde{X}} S_{vp} &= 1/h \qquad -h < \tilde{X}_p - \tilde{X}_v \leq 0 \\ \nabla_{\tilde{X}} S_{vp} &= -1/h \qquad 0 < \tilde{X}_p - \tilde{X}_v \leq h \end{aligned} \tag{24}$$

3.1.2. Generalised interpolation

The particular form of the generalised interpolation material point method adopted in this paper assumes a unity characteristic function (a hat function with the value of one inside the material point's domain and zero outside) of length $2l_p$ centred on $\tilde{X}_p$.[4] This characteristic yields the following basis functions

$$\begin{aligned} &\text{A:} \quad S_{vp} = \frac{(h + l_p + \tilde{X}_p - \tilde{X}_v)^2}{4hl_p} && -h - l_p < \tilde{X}_p - \tilde{X}_v \leq -h + l_p \\ &\text{B:} \quad S_{vp} = 1 + \frac{\tilde{X}_p - \tilde{X}_v}{h} && -h + l_p < \tilde{X}_p - \tilde{X}_v \leq -l_p \\ &\text{C:} \quad S_{vp} = 1 - \frac{(\tilde{X}_p - \tilde{X}_v)^2 + l_p^2}{2hl_p} && -l_p < \tilde{X}_p - \tilde{X}_v \leq l_p \\ &\text{D:} \quad S_{vp} = 1 - \frac{\tilde{X}_p - \tilde{X}_v}{h} && l_p < \tilde{X}_p - \tilde{X}_v \leq h - l_p \\ &\text{E:} \quad S_{vp} = \frac{(h + l_p - \tilde{X}_p + \tilde{X}_v)^2}{4hl_p} && h - l_p < \tilde{X}_p - \tilde{X}_v \leq h + l_p. \end{aligned} \tag{25}$$

These one-dimensional generalised interpolation basis functions are shown in Fig. 2 for node 2 where the A through E regions correspond to the five conditions in (25). In regions B and D, the generalised interpolation functions are the same as the conventional finite element functions (and the same at the standard material point method). This is because the material point's characteristic function lies entirely within the background grid cell. The basis functions in regions A, C and E (grey regions in the top figure) depart from the conventional finite element functions due to the material point domain overlapping multiple grid cells. The one-dimensional gradients of the basis functions with respect to the material point position at the start of the load step are

$$\begin{aligned} \nabla_{\tilde{X}} S_{vp} &= \tfrac{(h + l_p + \tilde{X}_p - \tilde{X}_v)}{2hl_p} && -h - l_p < \tilde{X}_p - \tilde{X}_v \leq -h + l_p \\ \nabla_{\tilde{X}} S_{vp} &= 1/h && -h + l_p < \tilde{X}_p - \tilde{X}_v \leq -l_p \\ \nabla_{\tilde{X}} S_{vp} &= -\tfrac{(\tilde{X}_p - \tilde{X}_v)}{hl_p} && -l_p < \tilde{X}_p - \tilde{X}_v \leq l_p \\ \nabla_{\tilde{X}} S_{vp} &= -1/h && l_p < \tilde{X}_p - \tilde{X}_v \leq h - l_p \\ \nabla_{\tilde{X}} S_{vp} &= -\tfrac{(h + l_p - \tilde{X}_p + \tilde{X}_v)}{2hl_p} && h - l_p < \tilde{X}_p - \tilde{X}_v \leq h + l_p. \end{aligned} \tag{26}$$

As with the basis functions in (25), when a material point's domain lies entirely within a background grid the gradient of the basis function equals that of the standard finite element function.

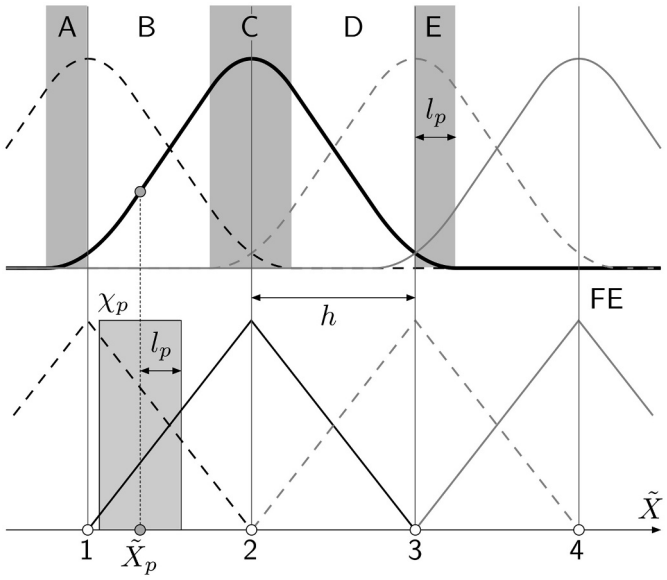

Fig. 2. Generalised interpolation basis functions (top) and standard (or standard material point) basis functions (bottom), where the numbers are associated with the grid nodes and the letters with the different conditions in (25). Reproduced with permission from Coombs et al. [10].

3.2. Boundary conditions

In this paper we ignore the external tractions as their general implementation within material point methods is complex. Dirichlet (essential/displacement) boundary conditions are imposed directly on the background mesh in the same way as the standard finite element method. This imposes the restriction that essential boundary conditions must be imposed along parts of the problem domain that are contiguous with background grid cell boundaries, as with the majority of MPMs presetned to date. Deviating from this requires special treatments such as those developed in [12]. The nodal body forces in (12) are approximated using

$$\{f_b\} = [S_{vp}]^T \{b\} V_p = [S_{vp}]^T \{g\} m_p, \tag{27}$$

where $\{b\} = \varrho\{g\}$ is the body force associated with the material point. For two dimensional analysis, $\{g\} = g\{0 \quad -1\}$, where ϱ is the material's density, g is gravitational acceleration and $m_p = \varrho V_p$ is the mass associated with the material point. Point forces, if required, are held and convected with the material points. They are mapped to the background grid using

$$\{f_P\} = [S_{vp}]^T \{f_p\}, \tag{28}$$

[3] In two dimensions the basis functions of a point are obtained via $S_{vp}(\tilde{X}, \tilde{Y}) = S_{vp}(\tilde{X}) \times S_{vp}(\tilde{Y})$, where $S_{vp}(\cdot)$ is obtained from (20).

[4] Generalised interpolation material point methods are restricted to regular domains (line in 1D, rectangle in 2D and cuboid in 3D) which allows the basis functions to be determined analytically for a regular background grid. Other domain-based material point methods, such as CPDI1 or CPDI2, relax the restriction on the shape of the material point's domain but approximate the convolution by sampling at the vertices of the material point's domain.

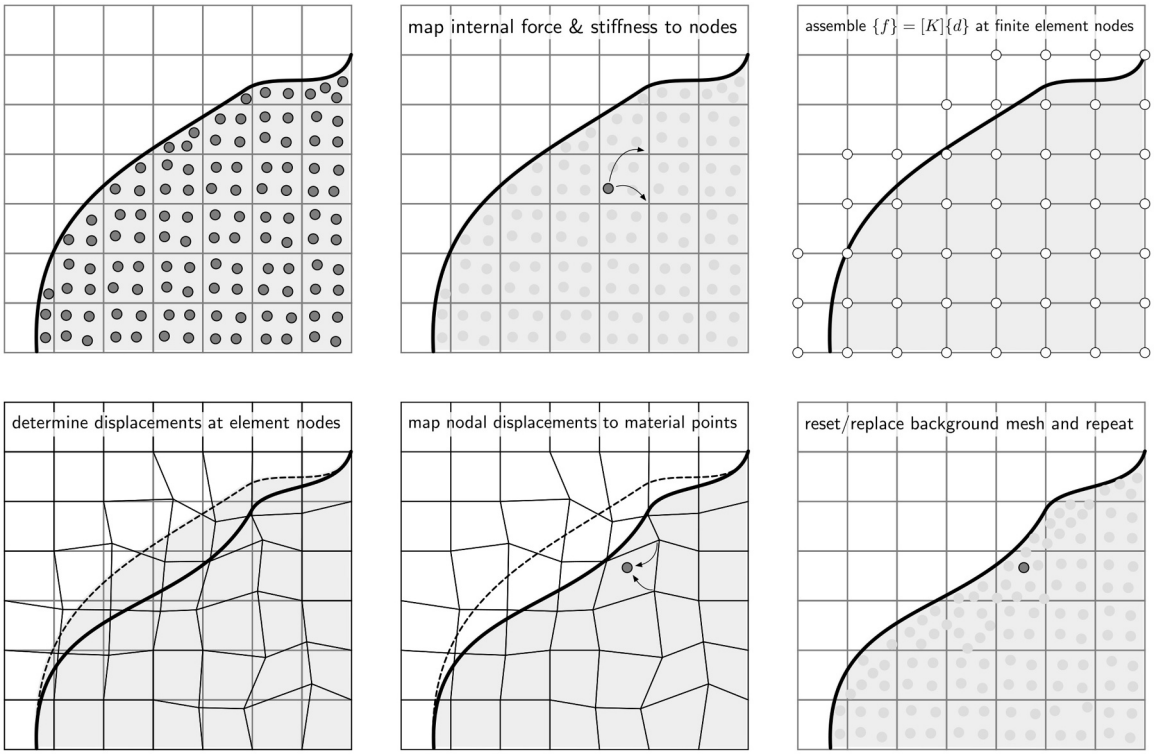

Fig. 3. AMPLE load step phases.

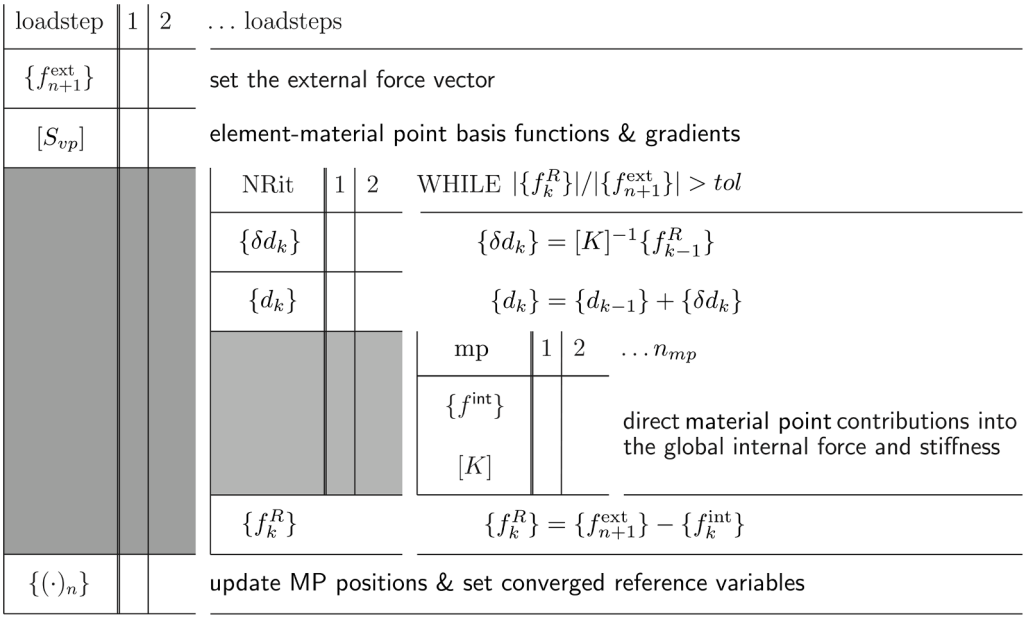

Fig. 4. AMPLE high level algorithm structure.

where $\{f_p\}$ is the point force associated with the material point and $\{f_P\}$ are the equivalent nodal values.

3.3. Non-linear solution procedure

The nodal displacements within a load step, $\{\Delta d\}$, can be obtained by iteratively updating the nodal displacements until (12) is satisfied within a given tolerance using

$$\{\Delta d_k\} = \sum_{m=1}^{k} \{\delta d_m\}, \qquad \text{where} \qquad \{\delta d_k\} = [K]^{-1}\{f^R_{k-1}\}, \tag{29}$$

k is the current iteration within the load step, $[K]$ is the global stiffness matrix and $\{\delta d_k\}$ is the iterative increment in the displacements from that iteration. The global stiffness matrix is obtained through assembling the individual material point contributions, that is

$$[K] = \underset{\forall p}{A}[k_p], \tag{30}$$

```
%AMPLE: A Material Point Learning Environment
%-------------------------------------------------------------------------
% Author: William Coombs
% Date:   29/01/2019
% Description:
% Large deformation elasto-plastic (EP) material point method (MPM) code
% based on an updated Lagrangian (UL) descripition of motion with a
% quadrilateral background mesh.
%
%-------------------------------------------------------------------------
% See also:
% SETUPGRID             - analysis specific information
% ELEMMPINFO            - material point-element information
% DETEXTFORCE           - external forces
% DETFDOFS              - mesh unknown degrees of freedom
% LINSOLVE              - linear solver
% DETMPS                - material point stiffness and internal force
% UPDATEMPS             - update material points
% POSTPRO               - post processing function including vtk output
%-------------------------------------------------------------------------
clear;
addpath('constitutive','functions','plotting','setup');
[lstps,g,mpData,mesh]=setupGrid;                                % setup infor
NRitMax = 10; tol = 1e-9;                                       % Newton Raphs
[nodes,nD] = size(mesh.coord);                                  % number of n
[nels,nen] = size(mesh.etpl);                                   % number of e
nDoF = nodes*nD;                                                % total numbe
nmp  = length(mpData);                                          % number of m
for lstp=1:lstps                                                % loadstep lo
  fprintf(1,'\n%s %4i %s %4i\n','loadstep ',lstp,' of ',lstps);  % text output
  [mesh,mpData] = elemMPinfo(mesh,mpData);                      % material poi
  fext = detExtForce(nodes,nD,g,mpData);                        % external fo
  fext = fext*lstp/lstps;                                       % current ext
  oobf = fext;                                                  % initial out
  fErr = 1;                                                     % initial err
  frct = zeros(nDoF,1);                                         % zero the re
  uvw  = zeros(nDoF,1);                                         % zero the dis
  fd   = detFDoFs(mesh);                                        % free degrees
  NRit = 0;                                                     % zero the it
  Kt   = 0;                                                     % zero global
  while (fErr > tol) && (NRit < NRitMax) || (NRit < 2)          % global equi
    [duvw,drct] = linSolve(mesh.bc,Kt,oobf,NRit,fd);            % linear solv
    uvw  = uvw+duvw;                                            % update disp
    frct = frct+drct;                                           % update react
    [fint,Kt,mpData] = detMPs(uvw,mpData);                      % global stiff
    oobf = (fext-fint+frct);                                    % out-of-bala
    fErr = norm(oobf)/norm(fext+frct+eps);                      % normalised
    NRit = NRit+1;                                              % increment t
    fprintf(1,'%s %2i %s %8.3e\n','  iteration ',NRit,' error ',fErr);  % text output
  end
  mpData = updateMPs(uvw,mpData);                               % upate materi
  run postPro;                                                  % Plotting an
end
```

Fig. 5. AMPLE main script: `ample.m`.

where A is the standard assembly operator acting over all of the material points in the problem and $[k_p]$ is the stiffness contribution of a single material point. $\{f^R_{k-1}\}$ is the residual out of balance force vector associated with the previous displacement value, i.e. the difference between the internal forces due to the stresses within the material and the applied boundary conditions, as given by (12). The global residual vector is assembled through

$$\{f^R_k\} = \underset{\forall p}{A}([\nabla_x S_{vp}]^T\{\sigma\}V_p) - \{f^{\text{ext}}\} \tag{31}$$

where the first term is the internal force vector and the external force vector, $\{f^{\text{ext}}\}$, which is constant over the load step, is given by

$$\{f^{\text{ext}}\} = \underset{\forall p}{A}([S_{vp}]^T\{b\}V^n_p + [S_{vp}]^T\{f_p\}) \tag{32}$$

3.4. Position and domain updating

At the end of each load step the material point positions, volumes and (if using the generalised interpolation material point method) domain half-lengths, l_p, should be updated. The updated positions of the material points at the end of the load step are given by

$$(x_p)_i = (\tilde{X}_p)_i + \underbrace{\sum_{v=1}^{n}(S_{vp})(\Delta u_i)_v}_{(\Delta u_p)_i}, \tag{33}$$

Table 1
AMPLE function descriptions.

Function	Description	Called by	Calls
setupGrid	analysis-specific information including the initial mesh and mpData structured arrays	ample	formCoord2D detMpPos shapefunc
elemMPinfo	basis functions and spatial derivatives at, cells and nodes associated with, and the number of stiffness matrix entries for, each material point	ample	MPMbasis elemForMP nodesForMP
detFDoFs	free degrees of freedom of the background mesh based on the grid cells that contain material points and the displacement boundary conditions	ample	–
detExtForce	nodal external force vector based on body forces and point loads at material points	ample	–
linSolve	linear solution of the global system of equations	ample	–
detMPs	internal force and stiffness contribution of all of the material points to the background mesh (contains the updated Lagrangian mechanics)	ample	Hooke3D VMconst formULstiff
updateMPs	function to update the positions and internal variables (Cauchy stress, elastic logarithmic strain, domain size, etc.) of the material points	ample	–
postPro	script to generate the VTK files for both the background mesh and the material points	ample	makeVtk makeVtkMP
MPMbasis	multi-dimensional basis functions and spatial derivatives	elemMPinfo	SvpMPM SvpGIMP
elemForMP	background grid cell(s) associated with the material point	elemMPinfo	–
nodesForMP	nodes linked with the material point	elemMPinfo	–
Hooke3D	linear elastic constitutive model	detMPs	–
VMconst	von Mises perfectly plastic constitutive model†	detMPs	–
formULstiff	formation of the spatial consistent tangent matrix†	detMPs	–
detMpPos	initial local grid cell positions of material points	setupGrid	–
formCoord2D	background mesh generation (regular quads)	setupGrid	–
shapefunc	finite element basis functions	setupGrid	–
SvpMPM	1D material point basis functions	MPMbasis	–
SvpGIMP	1D generalised interpolation basis functions	MPMbasis	–
makeVtk	background mesh VTK file generation	postPro	–
makeVtkMP	material point VTK file generation	postPro	–

†note that these functions contain sub functions within the same.m file

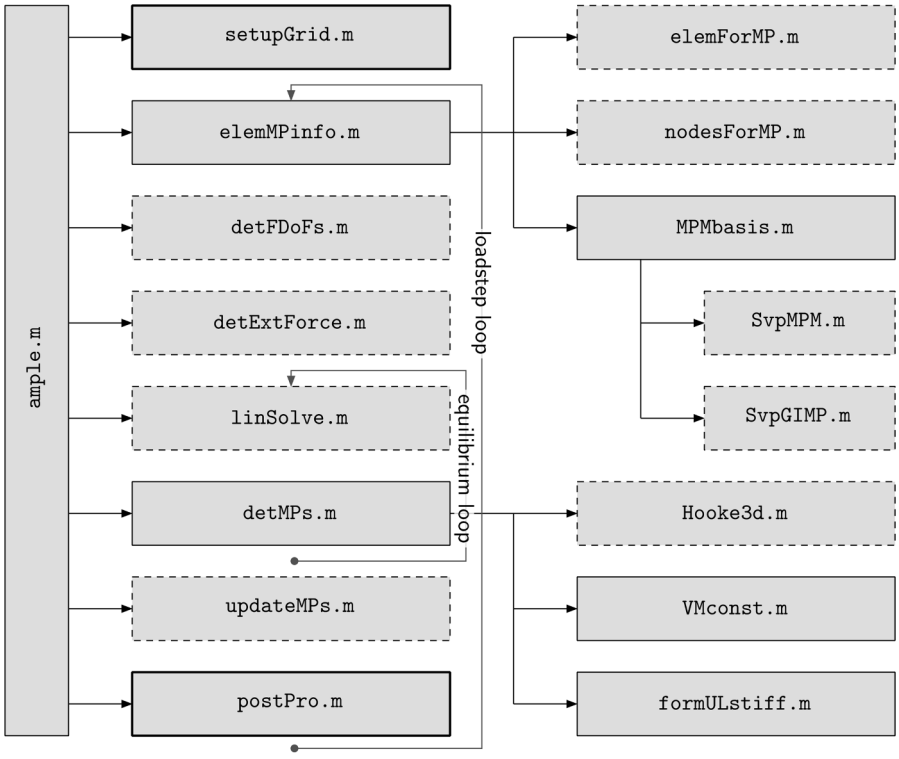

Fig. 6. AMPLE function dependencies: dashed and solid lines indicate functions with and without sub functions, respectively. VMconst.m and formULstiff.m contain sub functions within the same.m file.

where $(\Delta u_p)_i$ is the displacement of the material point over the load step. In the case of the standard material point method the volume at each material point is simply updated from the original volume using the determinant of the deformation gradient, (15). For generalised interpolation material point methods, one domain updating approach, proposed by Charlton et al. [5] and adopted in AMPLE, is to map the domain sizes according to the normal components of the material stretch tensor, that is

$$l_i^p = l_i^{p_0} U_{ii} \qquad \text{(no implied sum on } i\text{)}, \tag{34}$$

where $l_i^{p_0}$ are the original domain half lengths and $U_{ij} = \sqrt{F_{ki} F_{kj}}$.

Table 2
`ample` variable definitions (in order of appearance).

Variable	Symbol	Description	Dimensions
`lstps`	–	total number of load steps	`(1)`
`g`	g	gravitational acceleration	`(1)`
`mpData`	–	material point structured array (see Table 3)	–
`mesh`	–	mesh structured array (see Table 4)	–
`NRitMax`	–	maximum number of N-R iterations	`(1)`
`tol`	–	N-R tolerance	`(1)`
`nodes`	–	number of nodes in the background mesh	`(1)`
`nD`	–	number of dimensions	`(1)`
`nels`	–	number of elements in the background mesh	`(1)`
`nen`	–	number of element nodes	`(1)`
`nDoF`	–	number of degrees of freedom of the background mesh	`(1)`
`nmp`	–	number of material points	`(1)`
`lstp`	–	current loadstep number	`(1)`
`fext`	$\{f^{\text{ext}}\}$	external force vector	`(nDoF,1)`
`oobf`	$\{f^{R}\}$	out of balance force residual	`(nDoF,1)`
`fErr`	$\bar{f}^{R}$	normalised N-R error measure	`(1)`
`frct`	$\{f^{\text{react}}\}$	nodal reaction forces	`(nDoF,1)`
`uvw`	$\{\Delta d_k\}$	incremental nodal displacements in the load step	`(nDoF,1)`
`fd`	–	free degrees of freedom of the background mesh	`(*,1)`
`NRit`	k	current N-R iteration	`(1)`
`Kt`	$[K]$	global stiffness matrix (sparse)	`(nDoF,nDoF)`
`duvw`	$\{\delta d_k\}$	iterative nodal displacements	`(nDoF,1)`
`drct`	–	iterative nodal reaction forces	`(nDoF,1)`
`fint`	–	nodal internal forces	`(nDoF,1)`

* depends on the active grid cells and the boundary conditions

3.5. Computational procedure

The steps in the implemented algorithm are concisely summarised below. The applied body force and/or point forces are split into a number of load steps and for each of these steps the following process is adopted:

1. calculate the stiffness contribution, $[k^p]$, of all of the material points using (14) and assemble the individual contribution of each material point into the global stiffness matrix, $[K]$;
2. calculate the internal force contribution, $\{f^p\}$, of all of the material points using (16) and assemble the contributions into the global internal force vector, $\{f^R\}$, in (12);
3. solve for the nodal displacements within a load step, using the N-R process (29) until the out-of-balance force converges to a specified tolerance;
4. update the material point positions, volumes and domain lengths through interpolation from the incremental nodal displacements, deformation gradient and stretch tensor using (33), (15) and (34);
5. reset the background grid.

These steps are displayed graphically in Fig. 3.

4. Code structure

AMPLE's high-level structure is shown in Fig. 4 and comprises three loops:

1. a loadstep `for` loop that first determines the external forces at the nodes for the current step and then calculates the nodal basis functions, S_{vp}, and spatial derivatives, $\nabla_{\check{X}} S_{vp}$, for each material point. The code then enters a `while` loop to find equilibrium between the internal and external forces. Once equilibrium has been obtained the material point positions and internal variables (deformation gradient, stress, elastic logarithmic strain, domain size, etc.) are updated.
2. a Newton–Raphson `while` loop to solve the global equilibrium equations on the background mesh. The loop first solves the non-linear system of equations to determine the displacement and reaction force increments at the nodes before moving onto the determination of the current internal force and stiffness via a material point `for` loop.
3. a material point `for` loop that determines the internal force and stiffness contribution of each of the material points to the background mesh.

AMPLE's main script, `ample.m`, is shown in its entirety in Fig. 5. The format of the code aligns with the algorithm shown in Fig. 4. Comments are shown in green and have been truncated on the right hand side for clarity of the executed code. Only two of the three loops described above can be seen in the main `ample.m` file: (1) A loadstep `for` loop spanning lines 9 through 33 and (2) a Newton–Raphson `while` loop over lines 21 through 30. The material point loop is contained within the `detMPs.m` function on line 25.

The main `ample.m` script calls eight functions which are described in Table 1. The purpose of each of these functions is explained below:

`setupGrid`: called on line 3 of `ample.m`, the function returns the analysis-specific information such as details of the background grid and any information held at material points (position, material properties, etc.). This information is held in two structured arrays, `mpData` and `mesh`, that are explained in Section 4.2. In addition to this the function returns the total number of load steps, `lstps`, and the gravitational acceleration, `g`, which are returned as variables independent of the material point and mesh data.

`elemMPinfo`: called on line 1 of `ample.m`, the function determines the parent background grid cell(s) of each material point and the nodes that the material point influences. Note that for the standard material point method each material point will only have one parent cell but in the generalised interpolation material point method the domain of the point can overlap multiple grid cells. The function also determines the basis functions at each of the material points and their spatial gradients based on the nodal positions at the start of the load step. The function calls three other functions: `elemForMP` that finds the background grid cell(s) associated with the material point, `nodesForMP` that determines a unique list of nodes based on the cell(s) associated with the material point and `MPMbasis` that determines the basis functions for the material point. This function depends on the type of material point method and the form of the background mesh.[5]

`detExtForce`: Called on line 12 of `ample.m`, the function determines the external force vector, `fext`, at the start of the load step based on the body (27) and point (28) forces at material points.

`detFDoFs`: Called on line 18 of `ample.m`, the function determines a list of the unknown displacement degrees of freedom of the background mesh, `fd`, based on the grid cells that *contain* material points and the displacement boundary conditions imposed on the mesh.

`linSolve`: Called on line 22 of `ample.m`, the function solves the linear system of equations to determine the iterative increment in the nodal displacements, $\{\delta d\}$ or `duvw`, via (29) based on the current out-of-balance force residual, `oobf`. The function also determines the increment in the reaction forces, `drct`, due to the constrained degrees of freedom on the background mesh.

[5] Note that `elemMPinfo` is the main function that needs to change if you wish to change the form of the background mesh and the material point method. The only other function that requires modification is `updateMPs` and this is only if the user wishes to implement one of the newer domain based material point methods, such as CPDI1 [25], CPDI2 [26], etc. If only changing the mesh, only `elemMPinfo` requires modification.

```
function [fext] = detExtForce(nodes,nD,g,mpData)

%Global external force determination
%-------------------------------------------------------------------------
% Author: William Coombs
% Date:   23/01/2019
% Description:
% Function to determine the external forces at nodes based on body forces
% and point forces at material points.
%
%-------------------------------------------------------------------------
% [fbdy,mpData] = DETEXTFORCE(coord,etpl,g,eIN,mpData)
%-------------------------------------------------------------------------
% Input(s):
% nodes  - number of nodes (total in mesh)
% nD     - number of dimensions
% g      - gravity
% mpData - material point structured array. Function requires:
%           mpM   : material point mass
%           nIN   : nodes linked to the material point
%           Svp   : basis functions for the material point
%           fp    : point forces at material points
%-------------------------------------------------------------------------
% Ouput(s);
% fext   - external force vector (nodes*nD,1)
%-------------------------------------------------------------------------
% See also:
%
%-------------------------------------------------------------------------
nmp  = size(mpData,2);                                          % number of mat
fext = zeros(nodes*nD,1);                                       % zero the exte
grav = zeros(nD,1); grav(nD) = -g;                              % gavity vecto
for mp = 1:nmp
   nIN = mpData(mp).nIN;                                        % nodes associ
   nn  = length(nIN);                                           % number of no
   ed  = reshape(ones(nD,1)*(nIN-1)*nD+(1:nD).'*ones(1,nn),1,nn*nD);  % node degrees
   Svp = mpData(mp).Svp;                                        % basis functio
   fext(ed) = fext(ed)+mpData(mp).mpM*reshape(grav*Svp,nn*nD,1)...  % global body
            + reshape(mpData(mp).fp*Svp,nn*nD,1);               % material poi
end
```

Fig. 7. example function: `detExtForce.m`.

`detMPs`: Called on line 25 of `ample.m`, the function determines the stiffness and internal force contribution of each material point to the background mesh and assembles these into a global stiffness matrix, `Kt`, and internal force vector, `fint`. This function contains the updated Lagrangian formulation and the constitutive model controlling the stress-strain behaviour of each material point. The function calls three other functions: `Hooke3D` a linear elastic constitutive model, `VMconst` a linear elastic perfectly plastic von Mises constitutive model and `formULstiff` a function to determine the consistent spatial tangent modulus, [a].

`updateMPs`: Called on line 31 of `ample.m`, the function updates the positions and volumes of the material points based on the incremental nodal displacements, `uvw`, and the current value of the deformation gradient at each material point. For generalised interpolation material point methods, the function also updates the domain lengths, l_p, via (34). This function depends on the type of material point method and will require modification if a user wishes to implement other domain-based methods.

`postPro`: Called on line 32 of `ample.m`, the function produces VTK output files that can be viewed when the analysis has finished via a suitable VTK file visualised (such as VisIt or Paraview, amongst others). The function calls two functions: `makeVtk` and `makeVtkMP` that generate VTK files for the background mesh and the material points, respectively.

The structure of these functions is shown diagrammatically in Fig. 6, where the solid and dashed lines around the function boxes indicate that the function does or does not call other functions, respectively. The black arrows indicate the function calls and the locations of the loops in the code are indicated by the grey lines. For example, `Hooke3d.m` is called by `detMPs.m`; and `linSolve.m` and `detMPs.m` are called within the equilibrium, or N-R, loop.

It is important to note that the majority of AMPLE's functions do not depend on: (i) The form of the background mesh, (ii) the type of material point methods adopted or (iii) the number of physical dimensions (1D, 2D plane strain or 3D). The exceptions to this are: `elemMPinfo.m` which depends on the background mesh type and the material point variant and `updateMPs.m` which depends on the material point variant as different updating procedures are required for different material point methods. Obviously `setupGrid.m` depends on all three of the above points as it contains the analysis-specific information that will change depending on the user's requirements.

The functions are organised into a number of folders depending on their purpose, specifically: `constitutive` contains the constitutive functions, `setup` contains the functions that are required to provide the analysis-specific information, `plotting` contains the post-processing files and `functions` contains the remaining `.m` files. AMPLE's file structure also contains `output` and `documentation` that contain the generated VTK files and AMPLE's html documentation.

Table 2 lists all of the variables used in the main `ample.m` script along with their mathematical symbol, size and description. The 24 variables are listed in terms of their order of appearance in the `ample.m` script. Note that `mesh` and `mpData` are structured arrays containing the mesh and material point data, respectively, and are

Table 3
`mpData` structured array: Field definitions, where `nD` is the number of dimensions.

Field	Variable	Description	Dimensions
`mpType`	–	**m**aterial **p**oint **type** (1 = MPM, 2 = GIMP)	`(1)`
`cmType`	–	**c**onstitutive **m**odel **type** (1 = linear elasticity, 2 = von Mises elasto-plasticity)	`(1)`
`mpC`	$\{x_p\}$	**m**aterial **p**oint **C**oordinate	`(1,nD)`
`vp`	V_p	current material point volume	`(1)`
`vp0`	V_p^0	original material point volume	`(1)`
`mpM`	m_p	**m**aterial **p**oint **M**ass	`(1)`
`nIN`	–	nodes linked to the material point	`(1,*)`
`eIN`	–	cell(s) associated with the material point	`(1,‡)`
`Svp`	$[S_{vp}]$	basis functions	`(1,*)`
`dSvp`	$[\nabla_{\tilde{x}} S_{vp}]$	derivative of the basis functions with respect to the coordinates at the start of the loadstep	`(nD,*)`
`Fn`	$[F_n]$	deformation gradient at the start of the loadstep	`(3,3)`
`F`	$[F]$	current deformation gradient	`(3,3)`
`sig`	$\{\sigma\}$	Cauchy stress	`(6,1)`
`epsEn`	$\{\varepsilon_n^e\}$	logarithmic elastic strain at the start of the loadstep	`(6,1)`
`epsE`	$\{\varepsilon^e\}$	current logarithmic elastic strain	`(6,1)`
`mCst`	E, ν, etc.	material constants (Young's modulus, etc.)	`(1,†)`
`fp`	$\{f_p\}$	point forces at material points	`(nD,1)`
`u`	$\{u_p\}$	total material point displacement	`(nD,1)`
`lp`	l_i^p	material point domain length (MPM, $l_i^p = 0$)	`(1,nD)`
`lp0`	l_i^{p0}	initial material point domain length (MPM, $l_i^{p0} = 0$)	`(1,nD)`
`nSMe`	–	**n**umber of **S**tiffness **M**atrix **e**ntries per material point	`(1)`

* depends on the number of nodes the material point influences (always `nen` for MPM) ‡ depends on the number of grid cells linked to the material point (always 1 for MPM) † depends on the adopted constitutive model (e.g. 2 for linear elasticity; E, ν)

Table 4
`mesh` structured array: Field definitions, where `nels` is the number of elements, `nen` the number of element nodes, `nodes` the number of nodes in the mesh and `nD` the number of dimensions.

Field	Variable	**Description**	**Dimensions**
`etpl`	–	**e**lement **topo**logy (one row per element)	`(nels,nen)`
`coord`	–	nodal **coord**inates	`(nodes,nD)`
`h`	h	background grid size in each direction	`(1,nD)`
`bc`	–	Dirichlet **b**oundary **c**onditions with the following format: [degree of freedom, displacement]	`(*,2)`
`eInA`	–	active **e**lements **In** the **A**nalysis (1 = active element containing MPs, 0 = inactive element)	`(nels,1)`

* depends on the problem analysed; number of displacement constraints.

described in detail in Section 4.2.

4.1. Function format

All of AMPLE's functions share a common file format. Fig. 7 shows an example AMPLE function, specifically the function to determine the external forces on the background mesh based on body and point forces at material points, `detExtForce.m`, which is called on line 12 of `ample.m`. The function starts with a common comment block that includes the following information: (i) brief description/title; (ii) author and date information with a more detailed description of the function's purpose; (iii) function call format; (iv) required input information; (v) the function's output; and (vi) a *see also* section detailing the functions that are called by the function (in the case of `detExtForce.m`, none).

All of AMPLE's functions support the MATLAB `help` command, for example typing `help detExtForce` will return the top comment block shown in Fig. 7.

4.2. Data structures

The majority of the analysis information required by AMPLE is stored in two structured arrays:

`mpData`: this structured array contains material point information, such as the point's position, deformation gradient, Cauchy stress, etc. The 21 fields within `mpData` are detailed in Table 3, along with their mathematical symbols and dimensions per material point. A number of the quantities depend on the number of physical dimensions, `nD`.
`mesh`: this structured array contains information about the background mesh, such as the positions of the nodes and the topology of each of the background mesh cells. The five fields of `mesh` are detailed in Table 4, where `nels`, `nodes` and `nen` are the number of background grid cells, the total number of nodes and the number of nodes per background grid cell, respectively.

MATLAB's structured arrays provide a convenient way to store the material point data as different material points will potentially influence different numbers of nodes and therefore will require different amounts of storage for, for example, the basis functions that influence the point and their spatial derivatives.[6]

It is worth highlighting two of the fields that provide different options within AMPLE. `mpType` controls the material point type used in the analysis: `mpType=1` results in a standard material point whereas when `mpType=2` the point uses generalised interpolation basis functions. `cmType` controls the constitutive model used by the material point: `cmType=1` specifies the isotropic linear elastic model contained within `Hooke3d.m`, whereas `cmType=2` calls the linear elastic-perfectly plastic von Mises constitutive model, `VMconst.m`. Both of these could be extended to include other material point variants, such as CPDI methods, and different material models.

5. Demonstration cases

The analyses presented in this section explore changing a number of AMPLE's features/options. The physical problems modelled and the features modelled are as follows:

1. Compaction of a two-dimensional column under self weight: changing numbers of material points, material point types and mesh density;
2. A large deformation elastic beam subject to an end load: visualisation of material point data; and
3. Elasto-plastic collapse of a two-dimensional body: changing the constitutive model.

In all cases the global tolerance on the normalised out of balance force in the Newton–Raphson process was set at 1×10^{-9}, where the normalised out of balance force was defined as

$$\bar{f}^R = \frac{||\{f^R\}||}{||\{f^{\text{ext}}\} + \{f^{\text{react}}\}||}. \tag{35}$$

$||(\cdot)||$ denotes the L2 norm of $(\cdot)$, $\{f^{\text{ext}}\}$ includes the external forces applied to the problem (body forces, tractions, point loads) and $\{f^{\text{react}}\}$ are the reaction forces due to the prescribed displacement boundary conditions.

[6] It is noted that structured arrays are not the most computationally efficient way to store data in MATLAB, however AMPLE's focus is on proof (and clarity) rather than performance.

```
E      = 1e4;   v = 0;            % Young's modulus, Poisson's ratio
mCst   = [E v];                   % material constants
g      = 10;                      % gravity
rho    = 80;                      % material density
lstps  = 40;                      % number of loadsteps
nelsx  = 1;                       % number of elements in the x direction
nelsy  = 2^6;                     % number of elements in the y direction
ly     = 50;  lx = ly/nelsy;      % domain dimensions
mp     = 2;                       % number of material points in each direction per element
mpType = 2;                       % material point type: 1 = MPM, 2 = GIMP
cmType = 1;                       % constitutive model: 1 = elastic, 2 = vM plasticity
```

Fig. 8. setup information for compaction under self weight.

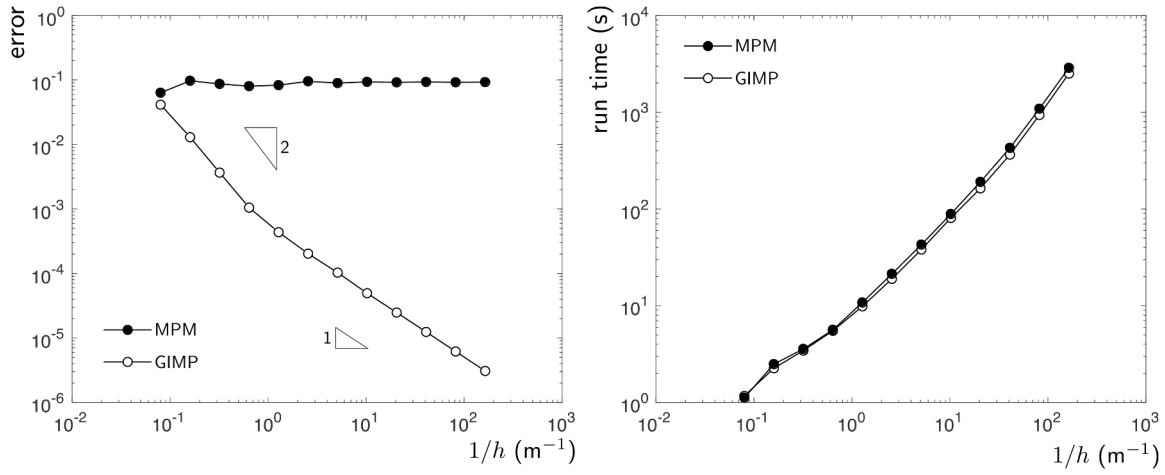

Fig. 9. compaction under self weight: convergence and run time.

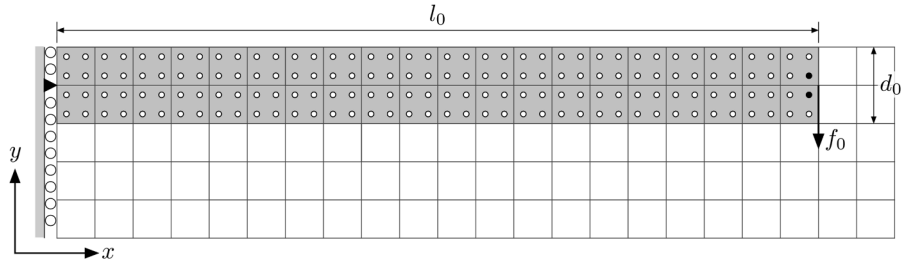

Fig. 10. elastic beam: initial discretisation and boundary conditions with $h = 0.5$m (only part of the background mesh is shown).

5.1. Convergence: Compaction under self weight

The first example is the one dimensional compaction of an elastic column with an initial height of $l_0 = 50$m under its own self weight. The material has a Young's modulus of 10kPa and a Poisson's ratio of zero. The background mesh is comprised of square background grid cells[7] with roller boundary conditions on the base and sides and the column is discretised by a 2×2 grid of equally spaced material points in each initially populated background grid cell. A body force of 800N/m^2 ($g = 10$m/s^2 and an initial density of $\varrho_0 = 80$kg/m^3) is applied over 40 equal loadsteps.

The basic setup information for this analysis, as contained on lines 57 through 67 of `setupGrid`, is shown in Fig. 8. The figure shows the setup information for an analysis with 64 grid cells in the y-vertical direction (`nelsy`) and a single grid cell in the x-direction (`nelsx`). The physical body is discretised by generalised interpolation material points, `mpType=2` (material point `Type`), with linear-elastic material behaviour, `cmType=1` (constitutive model `Type`).

The analytical solution for the normal stress in the y-direction for this problem is

$$\sigma^a_{yy} = \varrho_0\, g\,(l_0 - Y), \tag{36}$$

where Y is the original position of the point in the body and l_0 is the original height of the column. Fig. 9 shows the convergence of the standard and generalised interpolation material point methods with background mesh and material point refinement. The reported error is

$$\text{error} = \sum_{p=1}^{n_p} \frac{\|\sigma^p_{yy} - \sigma^a_{yy}(Y_p)\|\ V^0_p}{(g\varrho_0 l_0) V_0}, \tag{37}$$

where $V_0 = \sum V^0_p$ is the initial volume of the column and σ^p_{yy} is the vertical stress at each of the material points. For the convergence analysis, the number of background grid cells in the x-direction was kept constant at `nelsx=1` and the number of cells in the y-direction

[7] Note that the width of the problem is changed depending on the number of grid cells in the vertical direction to ensure that the cells remained square.

```
68  %% Mesh generation
69  [etpl,coord] = formCoord2D(nelsx,nelsy,lx,ly);        % background m
70  [~,nen]      = size(etpl);                            % number of el
71  [nodes,nD]   = size(coord);                           % number of no
72  h            = [lx ly]./[nelsx nelsy];                % element leng
73
74  %% Boundary conditions on background mesh
75  bc = zeros(nodes*nD,2);                               % generate emp
76  for node=1:nodes                                      % loop over no
77    if coord(node,1)==0                                 % roller (x=0)
78      bc(node*2-1,:)=[node*2-1 0];
79    end
80    if coord(node,1)==0 && coord(node,2)==(ly-d/2)      % mid-depth pin
81      bc(node*2  ,:)=[node*2   0];
82    end
83  end
84  bc = bc(bc(:,1)>0,:);                                 % remove empt
85
86  %% Mesh data structure generation
87  mesh.etpl  = etpl;                                    % element topo
88  mesh.coord = coord;                                   % nodal coordi
89  mesh.bc    = bc;                                      % boundary con
90  mesh.h     = h;                                       % mesh size
91
92  %% Material point generation
93  ngp    = mp^nD;                                       % number of ma
94  GpLoc  = detMpPos(mp,nD);                             % local MP loc
95  N      = shapefunc(nen,GpLoc,nD);                     % basis functi
96  [etplmp,coordmp] = formCoord2D(20*a,2*a,l,d);         % mesh for MP
97  coordmp(:,2)=coordmp(:,2)+(ly-d);                     % adjust MP lo
98  nelsmp = size(etplmp,1);                              % no. element
99  nmp    = ngp*nelsmp;                                  % total numbe
100
101 mpC=zeros(nmp,nD);                                    % zero MP coor
102 for nel=1:nelsmp
103   indx=(nel-1)*ngp+1:nel*ngp;                         % MP location
104   eC=coordmp(etplmp(nel,:),:);                        % element coor
105   mpPos=N*eC;                                         % global MP co
106   mpC(indx,:)=mpPos;                                  % store MP pos
107 end
108 lp = zeros(nmp,2);                                    % zero domain
109 lp(:,1) = h(1)/(2*mp);                                % domain half
110 lp(:,2) = h(2)/(2*mp);                                % domain half
111 vp      = 2^nD*lp(:,1).*lp(:,2);                      % volume assoc
```

Fig. 11. elastic beam: setup file segment (lines 68 to 111), where: `lx` and `ly` are the x and y lengths of the background grid, `d` and `l` are the depth and length of the beam and `a` is a scalar multiplier to control the number of background grid cells and material points. The other parameters are described in Fig. 8.

(`nelsy`) was varied between 2^2 and 2^{13}, that is between 4 and 8,192, in powers of 2 while maintaining the same material point/cell ratio[8] It is clear that the standard material point method (`mpType=1`) does not converge with uniform h refinement due to cell-crossing errors whereas the generalised interpolation method (`mpType=2`) converges at an approximately linear rate.

Fig. 9 also shows AMPLE's run time with progressive background mesh refinement[9] For most of the analyses, the run time scales approximately linearly with the number of material points ($\approx 4 \times 10^{-2}$ seconds per material point). However, when the number of grid cells in the y-direction exceeds 4,096 the time spent in the linear solver starts to dominate, with a corresponding increase in the gradient of the run time. Interestingly the generalised interpolation analyses typically have a slightly lower run time due to the Newton-Raphson process taking fewer iterations to find convergence.

[8] The number of material points in each grid cell was kept constant, therefore the total number of material points increased from 16, for the 2^2 mesh, to 32,768, for the 2^{13} mesh.

[9] All analyses were conducted using MATLAB R2017b within a macOS V.10.14.5 environment on a 2.3GHz Intel Core i7 with 16GB of RAM.

5.2. Visualisation: Large deformation elastic beam

The second demonstration problem is the large deformation bending of an elastic cantilever beam subjected to a point load at its free end using the generalised interpolation material point method (`mpType=2`). The beam is $l_0 = 10$m long and $d_0 = 1$m deep and the material has a Young's modulus of 12MPa and a Poisson's ratio of 0.2 (`cmType=1`). The $f_0 = 100$kN end point load is split between the two material points closest to the end of the beam either side of the neutral axis and applied over 50 equal load steps. The initial discretisation of the beam is shown in Fig. 10 with $h = 0.5$m and with 2^2 material points per initially populated background grid cells. The loaded material points are shown by the black-filled circles.

This elastic beam example differs from the previous demonstration case in that only part of the background grid contains the physical domain at the start of the analysis. Lines 68 through 111 of the setup file for this analysis are shown in Fig. 11. The figure shows the following steps:

`lines 68-72`: Generation of the background grid information - `etpl`, `coord` and `h`;
`lines 74-84`: Generation of the boundary conditions on the background grid, which are stored within `bc`. In this case, roller

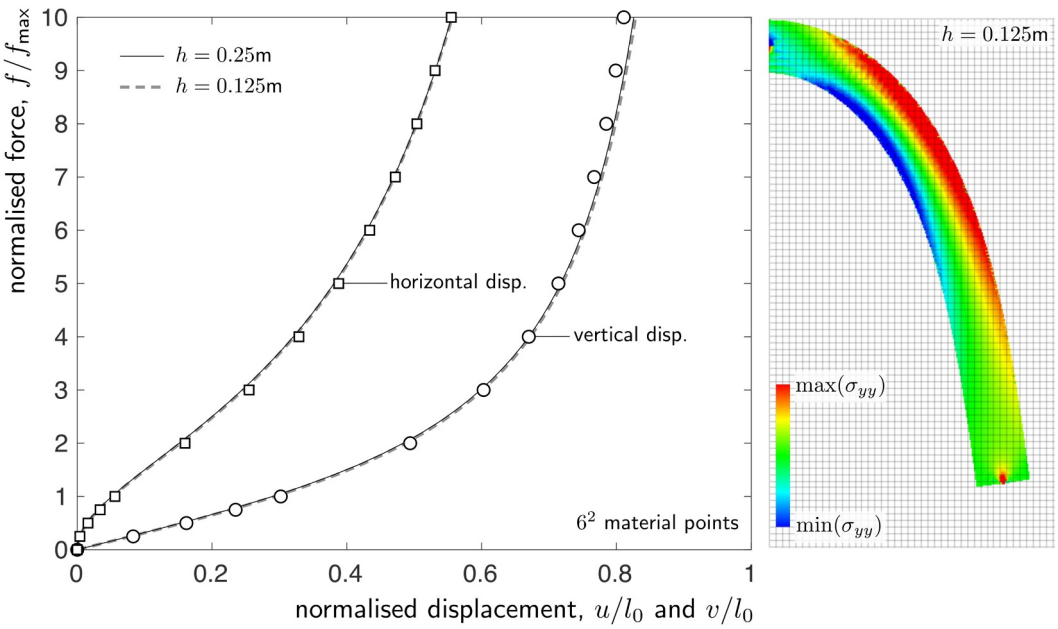

Fig. 12. elastic beam: normalised force versus displacement (where the squares and circles show the analytical solution of Molstad [22] for the horizontal and vertical displacements, respectively) and final deformed beam coloured according to $\sigma_{yy} \in [-500, 500]$kPa with $h = 0.125$m.

```
%Post processing script for the AMPLE code
%--------------------------------------------------------------------------
% Author: William Coombs
% Date:   29/01/2019
% Description:
% The script produces VTK output files based on the background mesh and
% material point data.
%
%--------------------------------------------------------------------------
% POSTPRO
%--------------------------------------------------------------------------
% See also:
% MAKEVTK           - VTK file for background mesh
% MAKEVTKMP         - VTK file for MP data
%--------------------------------------------------------------------------

01 mpDataName = sprintf('output/mpData_%i.vtk',lstp);                % MP output dat
02 sig = reshape([mpData.sig],6,nmp)';                               % all material
03 mpC = reshape([mpData.mpC],nD,nmp)';                              % all material
04 mpU = [mpData.u]';                                                % all material
05 makeVtkMP(mpC,sig,mpU,mpDataName);                                % generate mate
06 if lstp==1
07     makeVtk(mesh.coord,mesh.etpl,'output/mesh.vtk')               % generate mesh
08 end
```

Fig. 13. `postPro.m` script for VTK file generation (comments truncated for clarity).

boundary conditions are applied at $X = 0$ and the beam is fully-fixed at its mid-depth (`ly-d/2`);

`lines 86-90`: Storage of the background mesh information within the `mesh` data structure;

`lines 92-111`: Generation of the material point information which can be split down into a number of sub-steps:

- `lines 92-99`: Construction of a background grid for material point generation which, in this case, is different from the background grid used during the analysis as physical domain only covers part of the grid used for computation;
- `lines 101-107`: Determining the coordinates of each material point based on the background grid shape functions, `N`, and the coordinates of the nodes, `eC`, associated with the material point's parent grid cell; and
- `lines 108-11`: Calculation of the domain half lengths, `lp`, and material point volumes, `vp`, based on the size of the background grid, `h`, and the number of material points per grid cell in each direction, `mp`.

The material point information would then be stored in the `mpData` data structure but this has been omitted from the figure for sake of brevity.

The normalised global force versus displacement response for $h = 0.25$ m and $h = 0.125$ m are shown in Fig. 12. In both cases 6^2 generalised interpolation material points were included within each initially populated background grid cell, giving a total of 10,240 and 40,960 material points for the $h = 0.25$ m and $h = 0.125$ m analyses, respectively. The analytical solution, as detailed in the thesis of Molstad [22], is shown by the discrete points. The global response of both the $h = 0.25$ m and $h = 0.125$ m show good agreement with the analytical

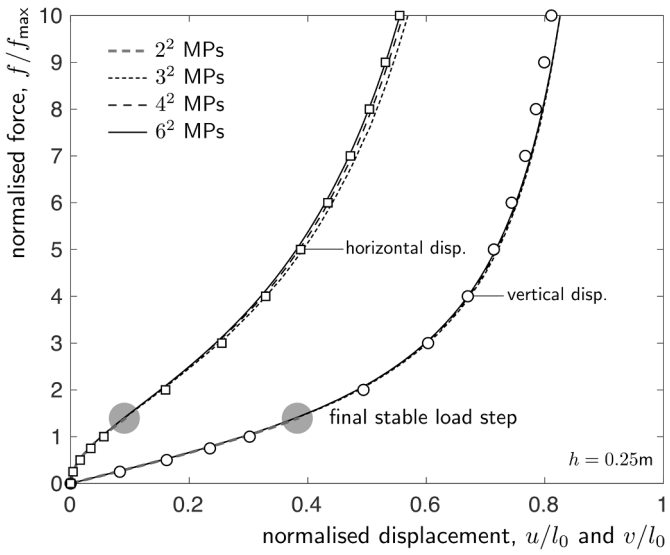

Fig. 14. elastic beam: normalised force versus displacement (where the white-filled squares and circles show the analytical solution of Molstad [22] for the horizontal and vertical displacements, respectively) for $h = 0.25$m and 4, 9, 16 and 36 material points per initially populated background grid cell. The grey-shaded circles highlight the final stable load step with 4 material points.

solution and there is very little difference in the force-displacement predictions of the two discretisations.

Fig. 12 also shows the deformed material point positions at the end of the $h = 0.125$m analysis that have been coloured according to the normal stress in the y-direction, σ_{yy}. The figure was produced from the VTK output files that are generated on line 32 of the main `ample.m` script (shown in Fig. 5) via `postPro.m` (shown in Fig. 13) and saved into the `output` folder.[10] The `postPro.m` script generates two different types of VTK files:

`mpData_#.xtk`: On line 5 where `#` is the current load step number (`lstp`). A VTK file based on the converged state at the end of each load step, including the following material point data: current position, total displacement between the current and initial positions and the Cauchy stress (all components). The script could be extended to include other data as appropriate, such as the volume at each material point, the level of straining, the deformation gradient, etc. AMPLE generates one `mpData_#.xtk` file per load step.

`mesh.vtk`: On line 7 which includes the background mesh information. AMPLE assumes that the background mesh is unchanged through the analysis and therefore only a single `mesh.vtk` file is generated and this happens on the first load step (`lstp=1`). If the mesh was different between loadsteps, a series of `mesh.vtk` files could be generated in a similar way to the material point data.

The number of material points per background cell is a point of debate in the material point method literature. In the authors' experience 2^6 material points per element for a two dimensional problem represents a reasonable balance between stability (in terms of conditioning of the global stiffness matrix), accuracy (in terms of reducing quadrature errors) and efficiency. In order to demonstrate this point, Fig. 14 presents the normalised global force versus displacement response for $h = 0.25$m with different numbers of material points per background grid cell, alongside the analytical solution shown by discrete white-filled circles and squares. The analysis with 4 material points per cell fails to converge in the 8th load step due to ill conditioning of the global stiffness matrix - the final stable load step for this analysis is highlighted by the grey-filled circles. The analysis with 3^2 and 4^2 material points per cell have very good agreement with the 6^2 analysis for the vertical displacement and converge to the 6^2 analysis for the horizontal displacement. The increase in accuracy is very minor between the analysis with 3^2 material points and that with 6^2; if efficiency is a user's primary concern then 3^2 material points per background grid cell is a reasonable choice.

5.3. Material model: Elasto-plastic collapse

The final demonstration example is the elasto-plastic collapse of a rectangular body of material subject to a gravitational body force using the generalised interpolation material point method (`mpType=2`). Due to symmetry only half of the body is modelled and the initial discretisation of the problem is shown in Fig. 15 with $l_0 = 8$ m, $h = 1$ m and 2^2 material points per initially filled background cell (the grey-shaded region). Roller boundary conditions are imposed on the base and the left hand edge of the background mesh, as indicated by the small circles in Fig. 15. These Dirichlet boundary conditions are imposed directly on the background grid by eliminating the degrees of freedom associated with these constraints from the global system of equations when solving the linear system during each N-R iteration. The degrees of freedom associated with the nodes that do not have a contribution from any material points are eliminated in the same way.

The material is modelled using a linear-elastic, perfectly-plastic von Mises constitutive formulation with the yield function defined in (11). In order for AMPLE to adopt this material behaviour it is necessary to set `cmType=2` for all of the material points in the analysis. For this analysis, the yield strength of the material is $\rho_y = 20$ kPa, the Young's modulus is 1MPa and Poisson's ratio is 0.3. These material properties are stored in `mCst` for each material point with the `mpData` structured array (see Table 3) in the following order $[E, \nu, \rho_y]$. A body force of 10 kN/m^3 (density of 1000 kg/m^3 and gravitational acceleration of 10 m/s^2) is applied over 40 equal loadsteps.

Fig. 16 shows the deformed material point positions at the end of the analysis for $h = 0.5$ m and $h = 0.25$ m. In both cases, 6^2 generalised interpolation material points (`cmType=2`) were included within each initially populated background grid cell; a total of 16,384 and 65,536 material points for the $h = 0.5$ m and $h = 0.25$ m analyses, respectively. The material points are coloured according to σ_{xx}, σ_{yy} and σ_{xy}, where the blue and red regions show areas of low and high stress, respectively. There is very little deformation for the first 20 loadsteps (up to a body force of 5 kN/m^3) but beyond this value the material rapidly collapses and achieves the final deformed state shown in Fig. 16.

Table 5 presents the horizontal extent and maximum height of the collapsed body at the end of the analysis for the generalised interpolation material point method with different numbers of material points and background mesh sizes. The results from an updated Lagrangian finite element analysis using bi-linear (q_1) and bi-quadratic (q_2) quadrilateral elements are shown for comparison[11]. The generalised interpolation material point method results are between the linear and quadratic finite element results which is consistent with the basis functions of the method being linear if the material point is contained within an element and quadratic if the material point contributes to multiple elements. In this case the analyses using 3^2 and 6^2 material points give similar results.

All of the examples presented in this paper have assumed quasi-static conditions - any inertia effects are ignored. However, in the case

[10] Although it is possible to plot material point data directly in MATLAB, AMPLE adopts VTK files as MATLAB struggles to plot (or rather render) large amounts of data in a computationally efficient manner. However, there is nothing stopping a user of AMPLE plotting material point data if required.

[11] Note that Charlton et al. [5] provides a more detailed comparison of the results from the generalised interpolation material point and finite element methods for this elasto-plastic collapse problem.

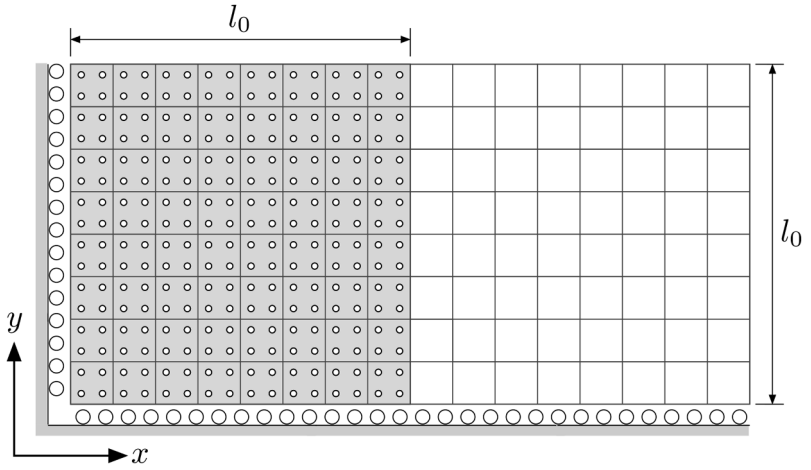

Fig. 15. Elasto-plastic collapse: Initial discretisation and boundary conditions with $h = 1$ m.

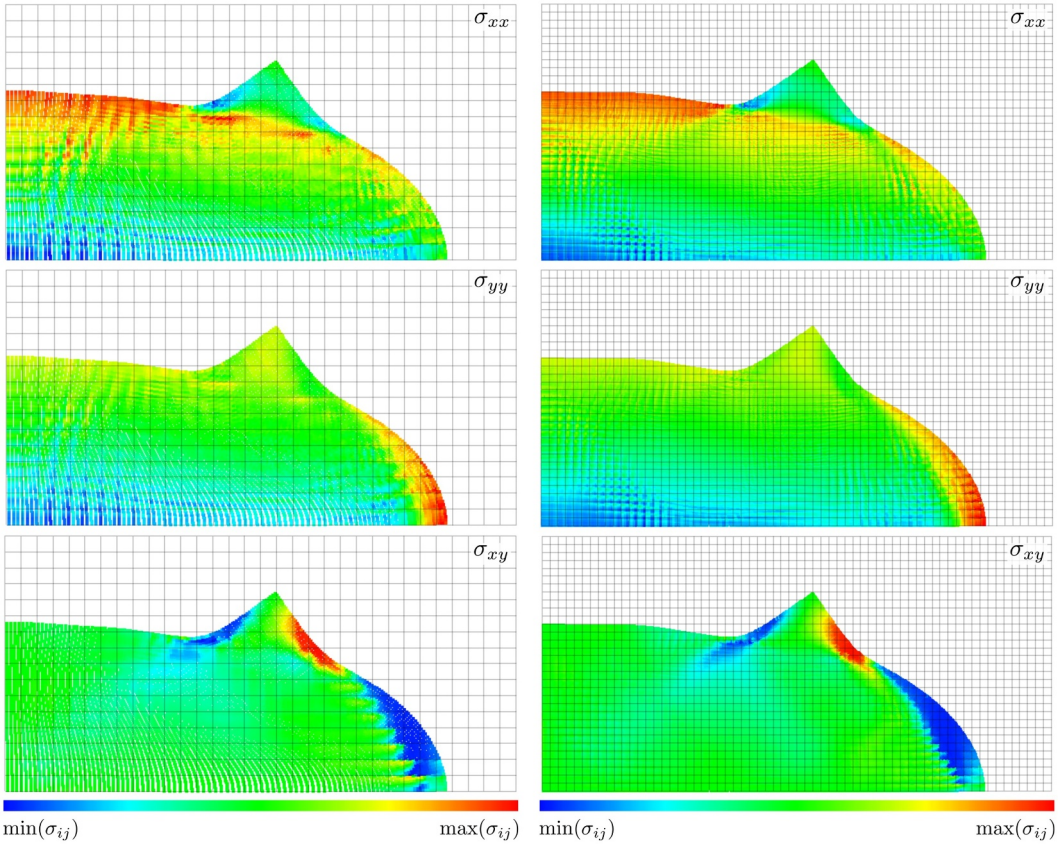

Fig. 16. Elasto-plastic collapse: Final deformed configuration with $h = 0.5$ m (left) and $h = 0.25$ m (right) with 6^2 material points per grid cell coloured according to σ_{xx}, σ_{yy} and σ_{xy}.

Table 5
Elasto-plastic collapse: Horizontal and vertical extents at the end of the analysis, where nPs denotes the number of integration or material points in each element/cell. For the finite element analyses the q_1 and q_2 denote bi-linear and bi-quadratic quadrilateral elements, respectively.

		Horizontal extent			Maximum height		
Method	nPs	$h = 1.0$	$h = 0.5$	$h = 0.25$	$h = 1.0$	$h = 0.5$	$h = 0.25$
GIMPM	3^2	13.781	13.841	13.896	6.194	6.215	6.190
GIMPM	6^2	13.773	13.832	13.882	6.256	6.240	6.213
FEM, q_1	2^2	13.271	13.626	13.814	6.581	6.365	6.254
FEM, q_2	3^2	13.855	13.930	13.914	6.230	6.178	6.153

of elasto-plastic collapse it could be argued that dynamic effects would impact on the physical process. We acknowledge this point, however the focus of AMPLE is on quasi-static analysis. An interesting future extension to this framework would be to dynamic problems via an implicit or explicit time stepping procedure.

6. Conclusion

The MPM is a relatively new approach for the modelling of large deformation solid mechanics problems and is attracting interest from a diverse range of applications. However, those interested in exploring its capabilities have to navigate their way through a number of challenging

aspects in order to understand the method and then have to turn the understanding into code. AMPLE is software for quasi-static implicit MPMs and has been developed to take some of the pain out of this exploration process. It provides a robust framework in which to explore MPMs and to develop new features appropriate for specific applications. AMPLE also provides a useful teaching tool for the MPM suitable for graduate students and post doctoral researchers wishing to consider using MPMs.

The code is available here [7] and more information is available on the AMPLE web pages [8].

Acknowledgements

The development of AMPLE was supported by the Engineering and Physical Sciences Research Council [grant numbers EP/N006054/1, EP/M017494/1 and EP/M000397/1]. The authors would also like to acknowledge the input of Yun Bing, Tim J. Charlton (supported by the Engineering and Phsyical Sciences Research Council [grant number EP/K502832/1]), Michael Cortis, Yousef Ghaffari Motlagh and Lei Wang who have all worked on the material point method at Durham University and contributed to the development of AMPLE in some way. All data created during this research are openly available at http://doi.org/10.15128/r2f1881k91g.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at 10.1016/j.advengsoft.2019.102748.

References

[1] Bardenhagen SG, Kober EM. The generalized interpolation material point method. Comput Model Eng Sci 2004;5(6):477–95.
[2] Belytschko T, Liu W, Moran B. Nonlinear finite elements for continua and structures. John Wiley & Sons; 2000.
[3] Bing Y, Cortis M, Charlton T, Coombs W, Augarde C. B-spline based boundary conditions in the material point method. Comput Struct 2019;212:257–74.
[4] Brackbill JU, Ruppel HM. FLIP - A Method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions. J Comput Phys 1986;65:314–43.
[5] Charlton TJ, Coombs WM, Augarde CE. iGIMP: an implicit generalised interpolation material point method for large deformations. Comput Struct 2017;190:108–25.
[6] Coombs WM. Finite deformation of particulate geomaterials: frictional and anisotropic critical state elasto-plasticity. Durham University; 2011.
[7] Coombs W.M. AMPLE: a material point learning environment (GitHub code). 2019a. Accessed 22/07/2019; https://github.com/wmcoombs/AMPLE.
[8] Coombs W.M.. AMPLE: a material point learning environment (project webpages). 2019b. Accessed 22/07/2019; https://wmcoombs.github.io/.
[9] Coombs WM, Augarde CE, Brennan AJ, Brown MJ, Charlton TJ, Ghaffari Motlagh Y, et al. On Lagrangian mechanics and the material point method for large deformation elasto-plasticity. Comput Methods Appl Mech Eng 2020;358:112622.
[10] Coombs WM, Charlton TJ, Cortis M, Augarde CE. Overcoming volumetric locking in material point methods. Compt Methods Appl Mech Eng 2018;333:1–21.
[11] Coombs WM, Crouch RS, Augarde CE. 70-line 3D finite deformation elastoplastic finite-element code. In: Benz T, Nordalm S, editors. 7th European Conference on Numerical Methods in Geoetchnical Engineering (NUMGE). Trondheim, Norway. 2010. p. 151–6.
[12] Cortis M, Coombs WM, Augarde CE, Brown MJ, Brennan A, Robinson S. Imposition of essential boundary conditions in the material point method. Int J Numer Methods Eng 2017;113(1):130–52.
[13] Gu XY, Dong CY, Li JL, Liu ZY, Xu JY. MPM Simulations of high-speed and ultra high-speed machining of titanium alloy (ti-6A1-4V) based on fracture energy approach. Eng Anal Bound Elements 2015;59:129–43.
[14] Guilkey JE, Weiss JA. Implicit time integration for the material point method: quantitative and algorithmic comparisons with the finite element method. Int J Numer Methods Eng 2003;57(9):1323–38.
[15] Harlow F. The particle-in-cell computing method for fluid dynamics. Methods Comput Phys 1964;3(319–343).
[16] Kakouris EG, Triantafyllou SP. Phase field material point method for brittle fracture. Int J Numer MethodsEng 2017;112(12):1750–76.
[17] Kim D-N, Montáns F, Bathe K. Insight into a model for large strain anisotropic elasto-plasticity. Comput Mech 2009;44(5):651–68.
[18] Klár G, Gast T, Pradhana A, Fu C, Schroeder C, Jiang C, et al. Drucker-Prager elastoplasticity for sand animation. ACM Trans Graph 2016;35(4). 103:1–103:12.
[19] Lee EH. Elastic-plastic deformation at finite strains. J Appl Mech 1969;36:1–6.
[20] Lee EH, Lu DT. Finite-strain elastic-plastic theory with application to plane-wave analysis. J Appl Phys 1967;38:19–27.
[21] Ma J, Wang D, Randolph MF. A new contact algorithm in the material point method for geotechnical simulations. Int J Numer Anal Methods Geomech 2014;38(11):1197–210.
[22] Molstad T. Finite deformation analysis using the finite element method. University of British Columbia; 1977.
[23] Nairn JA. Material point method calculations with explicit cracks. Comput Model Eng Sci 2003;4(6):649–63.
[24] Remmerswaal G. Development and implementation of moving boundary conditions in the material point method. TU Delft; 2017.
[25] Sadeghirad A, Brannon RM, Burghardt J. A convected particle domain interpolation technique to extend applicability of the material point method for problems involving massive deformations. Int J Numer Methods Eng 2011;86(12):1435–56.
[26] Sadeghirad A, Brannon RM, Guilkey JE. Second-order convected particle domain interpolation (CPDI2) with enrichment for weak discontinuities at material interfaces. Int J Numer Methods Eng 2013;95(11):928–52.
[27] Sheng D, Nazem M, Carter JP. Some computational aspects for solving deep penetration problems in geomechanics. Comput Mech 2009;44(4):549–61.
[28] Simo J. Algorithms for static and dynamic multiplicative plasticity that preserve the classical return mapping schemes of the infinitesimal theory. Comput Methods Appl Mechanics Eng 1992;99:61–112.
[29] Soga K, Alonso E, Yerro A, Kumar K, Bandara S. Trends in large-deformation analysis of landslide mass movements with particular emphasis on the material point method. Géotechnique 2016;66(3):248–73.
[30] Stomakhin A, Schroeder C, Chai L, Teran J, Selle A. A material point method for snow simulation. ACM Trans Graph 2013;32(4). 102:1–102:10.
[31] Sulsky D, Chen Z, Schreyer HL. A particle method for history-dependent materials. Comput Methods Appl Mech Eng 1994;118(1):179–96.
[32] Sulsky D, Kaul A. Implicit dynamics in the material-point method. Comput Methods Appl Mech Eng 2004;193:1137–70.
[33] Trefethen L. Ten digit algorithms. Tech. Rep.. Numerical Analysis Report 05/13, Oxford University; 2005.
[34] Ye Z, Zhang X, Zheng G, Jia G. A material point method model and ballistic limit equation for hyper velocity impact of multi-layer fabric coated aluminum plate. Int J Mech Mater Des 2018;14(4):511–26.
[35] Yerro A, Alonso EE, Pinyol NM. The material point method for unsaturated soils. Géotechnique 2015;65(3):201–17.