Available online at www.sciencedirect.com
SCIENCE DIRECT®
Comput. Methods Appl. Mech. Engrg. 193 (2004) 1137–1170
Computer methods in applied mechanics and engineering
www.elsevier.com/locate/cma

Implicit dynamics in the material-point method

D. Sulsky [a,*], A. Kaul [b]

[a] Department of Mathematics and Statistics, University of New Mexico, Albuquerque, NM 87131, USA
[b] Los Alamos National Laboratory, P.O. Box MS B259, Los Alamos, NM 87545, USA

Received 14 March 2003; received in revised form 11 August 2003; accepted 4 December 2003

Abstract

A time-implicit discretization is derived and validated for the material-point method (MPM). The resulting nonlinear, discrete equations are solved using Newton's method combined with either the conjugate gradient method or the generalized minimum residual method. These Newton–Krylov solvers are implemented in a matrix-free fashion for numerical efficiency. A description of the algorithms and evaluation of their performance is presented. On all test problems, if the time step is chosen appropriately, the implicit solution technique is more efficient than an explicit method without loss of desired features in the solutions. In a dramatic example, time steps 10,000 times the explicit step size are possible for the large deformation compression of a cylindrical billet at 1.2% the computational cost.

Keywords: Implicit; Newton–Krylov; Conjugate-gradient; GMRES; Material-point method

1. Introduction

The material-point method (MPM) is a variant of the particle-in-cell methods originally developed at Los Alamos National Laboratory [1–3]. MPM adds the capability to model solid materials with strength and stiffness while retaining the capability to handle large deformations [4], and to model multiple materials such as fluid–structure interaction [5]. Previous work has employed explicit dynamics; however, there are manufacturing problems like metal rolling, upsetting, and machining, where the flow speed of the material through the process is much slower than the wave speed for elastic or plastic wave propagation. The explicit code operates with a time step restricted by the usual Courant–Friedrichs–Lewy (CFL) condition, and thus the code resolves all waves. Simulating many manufacturing and other problems with an implicit method can result in a considerable reduction in computational cost if one is interested only in the low-frequency, bulk motion, and not in the elastic wave motion. This paper is concerned with the implementation of implicit dynamics within the MPM framework, and the validation of the resulting implicit solver.

* Corresponding author. Fax: +1-505-277-5505.
E-mail addresses: sulsky@math.unm.edu (D. Sulsky), akaul@lanl.gov (A. Kaul).

0045-7825/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.cma.2003.12.011

A linear elastic constitutive model leads to a linear system of equations for the unknowns. However, constitutive models for solids undergoing large deformations are generally non-linear. Each non-linear iteration in an implicit solution entails the solution of a linear system containing a tangent matrix. The evaluation of this matrix and the solution of the associated linear system account for most of the cost of a non-linear iteration. Methods that remain stable for large time steps, that minimize the number of evaluations of the tangent matrix, and that minimize the number of iterations per time step, reduce these costs. To be efficient, the gain in the size of the time step, and therefore the reduction in the total number of steps must outweigh the extra cost per step of the implicit solver. We have chosen a matrix-free implementation of a Newton–Krylov solver that does not require actual formation of the tangent stiffness matrix to minimize the computational costs. The method can be made unconditionally stable, thus the time step is only restricted by the desired accuracy of the computed results. A similar approach is used by Cummins and Brackbill [6] for granular material with a slightly different time discretization. In contrast, Guilkey and Weiss [7] explicitly form the tangent matrix.

In the next section, the governing equations, in both continuum and discrete forms, are presented, including a discussion of the constitutive models for solids. An outline of the MPM method is also included, followed by a description of the implicit and explicit methods. A complete description of the implicit equations, both linear and non-linear, is provided in Section 2.4. Several methods for solving these equations are discussed, including conjugate gradients (CG) for the linear case, Newton–CG for the non-linear case, and Newton's method combined with the generalized minimum residual (GMRES) method, also for the non-linear case. The Newton–CG method is restricted to problems that have a symmetric discretization matrix, while Newton–GMRES can be applied more generally. Different formulations of MPM lead to either symmetric or non-symmetric discretizations.

Section 3 discusses test problems for various forms of the implicit solvers, with an analysis of the results. Tests have thus far been performed in two dimensions. We examine two linear problems, uniaxial extension of a bar, and the impact of two disks. The elastic bar is then carried into the plastic regime to test the non-linear solver. The simulations in Section 3.1 are based on the formulation of MPM presented in Section 2 which leads to a symmetric discretization matrix. The resulting discrete system of equations solved using Newton–CG. The implicit formulation is shown to be more efficient than an explicit method for the test problems without losing desired details in the solutions. However, the momentum formulation of MPM [8] is found to be more robust for large deformation problems. Section 3.2 discusses this formulation and the resulting non-symmetric discretization. Solutions are obtained for the same test problems using Newton–GMRES and the performance is comparable to the Newton–CG simulations. Finally, analysis is presented of an upsetting problem in which a cylinder is compressed to 40% of its original height. Convergent solutions for upsetting are obtained only with the Newton–GMRES formulation. In this example, it is possible to use time steps 10,000 times the step size for the explicit calculation with a cost of about 1.2% that of the explicit code.

2. Governing equations

The continuum equations for a solid, liquid or gas are described in Section 2.1. The governing equations for these media include the conservation of mass, conservation of momentum, conservation of energy, constitutive equations and kinematic constraints. In Section 2.2, the discrete conservation equations are developed. Section 2.3 provides a description of the MPM computational cycle. The implicit MPM equations and the methods used to solve them are discussed in Section 2.4. In these equations, vectors and tensors are denoted with a bold font; the order of the tensor is given in the accompanying text. The summation convention on repeated indices is also used.

2.1. Continuum equations

A solid or body of fluid is subject to the physical constraints of conservation of mass, linear momentum, angular momentum, and energy. The development of these equations can be found in many references, for example [9]. Suppose a solid or a body of fluid initially occupies a region $\Omega_3(0) \subset \mathbb{R}^3$ and a region $\Omega_3(t) \subset \mathbb{R}^3$ for time $t > 0$. For two-dimensional calculations, consider the Cartesian coordinate system with coordinates (x, y, z) where x and y are in the plane under consideration and z is out of the plane. $\Omega_3(t)$ can be decomposed into a two-dimensional, planar region $\Omega_2(t) \subset \mathbb{R}^2$ in the xy-plane with uniform thickness out of the plane. Points in $\Omega_2(t)$ can be specified by their (x, y) coordinates. If a material point is identified initially at the position $\boldsymbol{X} = (X, Y)$, then its current position $\boldsymbol{x} = (x, y)$ at time $t > 0$ is a function of $\boldsymbol{X}$ and t with $\boldsymbol{x}(\boldsymbol{X}, 0) = \boldsymbol{X}$. It is assumed that this function is invertible so that we may also consider $\boldsymbol{X}$ to be a function of $\boldsymbol{x}$ and t.

Let ρ be the mass density, $\boldsymbol{v}$ be the velocity, $\boldsymbol{\sigma}$ be the Cauchy stress tensor, $\boldsymbol{b}$ be the specific body force, $\boldsymbol{\epsilon}$ be the strain tensor and e be the internal energy per unit mass in the current configuration. Conservation of mass is described by the continuity equation

$$\frac{\mathrm{d}\rho}{\mathrm{d}t} + \rho \nabla \cdot \boldsymbol{v} = 0, \tag{2.1.1}$$

where $\mathrm{d}/\mathrm{d}t$ designates the material derivative. The Cauchy form of conservation of linear momentum is given by the equation

$$\rho \frac{\mathrm{d}\boldsymbol{v}}{\mathrm{d}t} = \nabla \cdot \boldsymbol{\sigma} + \rho \boldsymbol{b}. \tag{2.1.2}$$

Conservation of angular momentum results in a symmetric Cauchy stress tensor, $\boldsymbol{\sigma}$. If thermal conduction and heat generation are ignored, the energy equation is

$$\rho \frac{\mathrm{d}e}{\mathrm{d}t} = \boldsymbol{\sigma} : \frac{\mathrm{d}\boldsymbol{\epsilon}}{\mathrm{d}t}, \tag{2.1.3}$$

where $\boldsymbol{A} : \boldsymbol{B} = a_{ij} b_{ij}$ in Cartesian coordinates, for example. These equations will be solved in time for $\boldsymbol{x} \in \Omega_2(t)$.

For a well-posed problem, appropriate initial conditions and boundary conditions on the boundary of the domain, $\partial\Omega_2(t)$, need to be prescribed. Typically, only two types of boundary conditions are considered: displacement boundary conditions described by equations such as

$$\boldsymbol{u}(\boldsymbol{x}, t) = \boldsymbol{g}(t) \quad \text{on } \partial\Omega_2^u(t) \tag{2.1.4}$$

or prescribed traction boundary conditions described by equations such as

$$\boldsymbol{\sigma}(\boldsymbol{x}, t) \cdot \boldsymbol{n} = \boldsymbol{t}(t) \quad \text{on } \partial\Omega_2^t(t). \tag{2.1.5}$$

In these conditions, the boundary is divided into two disjoint sets, $\partial\Omega_2 = \partial\Omega_2^u(t) \cup \partial\Omega_2^t(t)$; $\boldsymbol{n}$ is the unit outward normal to the boundary; $\boldsymbol{g}(t)$ is the prescribed displacement and $\boldsymbol{t}(t)$ is the prescribed traction.

2.1.1. Constitutive equations

A constitutive equation relating strain or strain rate and stress is needed to complete the description of the material. A strain, $\boldsymbol{\epsilon}$, can be determined from the rate equation

$$\frac{\mathrm{d}\boldsymbol{\epsilon}}{\mathrm{d}t} = \frac{1}{2}(\nabla \boldsymbol{v} + \nabla \boldsymbol{v}^{\mathrm{T}}). \tag{2.1.6}$$

The stress will depend on the type of material. For example, the constitutive equation for a Newtonian fluid is

$$\boldsymbol{\sigma} = -p\boldsymbol{I} + \lambda(\nabla \cdot \boldsymbol{v})\boldsymbol{I} + \mu(\nabla \boldsymbol{v} + \nabla \boldsymbol{v}^{\mathrm{T}}), \tag{2.1.7}$$

where λ and μ are the viscosity coefficients of the fluid, p is the pressure and $\boldsymbol{I}$ is the second-order identity tensor.

Constitutive models for solids include isotropic and anisotropic elasticity models, various plasticity models, viscoelasticity, and a decohesion model [4,5,8,10,11]. In particular, processing of metals usually involves elastic–plastic materials. The rate form of the general linear elastic constitutive model can be written as

$$\frac{\mathrm{d}\boldsymbol{\sigma}}{\mathrm{d}t} = \boldsymbol{E}\frac{\mathrm{d}\boldsymbol{\epsilon}}{\mathrm{d}t}, \tag{2.1.8}$$

where $\boldsymbol{E}$ is the fourth-order elasticity tensor.

A general isotropic plasticity model with hardening has the form

$$\frac{\mathrm{d}\boldsymbol{\sigma}}{\mathrm{d}t} = \boldsymbol{E}\left(\frac{\mathrm{d}\boldsymbol{\epsilon}}{\mathrm{d}t} - \frac{\mathrm{d}\boldsymbol{\epsilon}^{\mathrm{pl}}}{\mathrm{d}t}\right), \tag{2.1.9}$$

$$\frac{\mathrm{d}\boldsymbol{\epsilon}^{\mathrm{pl}}}{\mathrm{d}t} = \frac{\mathrm{d}\gamma}{\mathrm{d}t}\boldsymbol{M}(\boldsymbol{\sigma},\boldsymbol{i}), \tag{2.1.10}$$

$$\frac{\mathrm{d}\boldsymbol{i}}{\mathrm{d}t} = \frac{\mathrm{d}\gamma}{\mathrm{d}t}\boldsymbol{h}(\boldsymbol{\sigma},\boldsymbol{i}), \tag{2.1.11}$$

$$0 = \frac{\mathrm{d}f}{\mathrm{d}t}(\boldsymbol{\sigma},\boldsymbol{i}). \tag{2.1.12}$$

The elastic strain, denoted by $\boldsymbol{\epsilon}^{\mathrm{el}}$, is the difference between the total strain and the plastic strain, $\boldsymbol{\epsilon}^{\mathrm{pl}}$. The plastic strain rate evolves using Eq. (2.1.10), where $\boldsymbol{M}$ is the plastic mode and γ is a non-negative parameter measuring the plastic strain path length. The unit tensor $\boldsymbol{M}$ gives the direction of plastic flow. The relationship between the vector $\boldsymbol{i}$ of internal variables describing the internal state of the material and the vector $\boldsymbol{h}$ specifying the hardening rules is given by Eq. (2.1.11). The function f is the yield function, which is defined by the following properties: if $f < 0$, then the effective stress is less than the current yield stress and elastic deformation is occurring; if $f = 0$, plastic deformation may be occurring and the effective stress is at the current yield stress; $f > 0$ is not allowed. After plasticity has occurred, Eq. (2.1.12) is equivalent to $f = 0$, which describes the yield surface [10].

Four classical models of isotropic plasticity are von Mises, Prager–Drucker, Tresca and Mohr–Coulomb. These models are distinguished by different choices for the yield function f. The simulations in the next section use a strain-hardening von Mises model with an associated flow rule, defined by

$$f(\boldsymbol{\sigma}, i) = 3J_2 - H^2(i), \tag{2.1.13}$$

$$J_2 = \frac{1}{2}\boldsymbol{\sigma}^{\mathrm{d}} : \boldsymbol{\sigma}^{\mathrm{d}}, \tag{2.1.14}$$

$$\boldsymbol{M} = \boldsymbol{N} = \frac{\boldsymbol{\sigma}^{\mathrm{d}}}{\left(\boldsymbol{\sigma}^{\mathrm{d}} : \boldsymbol{\sigma}^{\mathrm{d}}\right)^{1/2}}, \tag{2.1.15}$$

$$\frac{\mathrm{d}i}{\mathrm{d}t} = \sqrt{\frac{2}{3}}\frac{\mathrm{d}\gamma}{\mathrm{d}t} = \frac{\mathrm{d}\bar{\epsilon}^{\mathrm{pl}}}{\mathrm{d}t}, \tag{2.1.16}$$

$$\boldsymbol{\sigma}^{\mathrm{d}} = \boldsymbol{\sigma} - \frac{1}{3}\mathrm{tr}(\boldsymbol{\sigma})\boldsymbol{I}. \tag{2.1.17}$$

The second invariant of the plastic strain is given by

$$\bar{\epsilon}^{\mathrm{pl}} = \int (\mathrm{d}\boldsymbol{\epsilon}^{\mathrm{pl}} : \mathrm{d}\boldsymbol{\epsilon}^{\mathrm{pl}})^{1/2}\,\mathrm{d}t = \sqrt{\frac{2}{3}}\int \mathrm{d}\gamma = \sqrt{\frac{2}{3}}\gamma. \tag{2.1.18}$$

Eq. (2.1.13) defines the yield function, with the hardening function H. Eq. (2.1.15) gives the associated flow rule. From Eq. (2.1.16), H is a function of the second invariant of plastic strain. We use a bilinear form for H

$$H = \begin{cases} \tau_0 + c_1\bar{\epsilon}^{\mathrm{pl}} & 0 \leqslant \bar{\epsilon}^{\mathrm{pl}} \leqslant \bar{\epsilon}^{\mathrm{pl}}_{\mathrm{L}}, \\ \tau_{\mathrm{L}} - c_2(\bar{\epsilon}^{\mathrm{pl}} - \bar{\epsilon}^{\mathrm{pl}}_{\mathrm{L}}) & \bar{\epsilon}^{\mathrm{pl}} \geqslant \bar{\epsilon}^{\mathrm{pl}}_{\mathrm{L}}, \end{cases} \tag{2.1.19}$$

where τ_0 and τ_{L} correspond to the initial and limit strengths of the material. The parameters c_1 and c_2 govern the slopes of the two line segments. Positive values of c_1 and c_2 correspond to a linear hardening region followed by linear softening [10]. It is assumed that plastic strains are sufficiently small so that $H > 0$.

2.2. Discrete equations

To create a numerical solver, the equations of a continuous medium, given in Eqs. (2.1.1)–(2.1.3), along with the constitutive models (2.1.7)–(2.1.12), must be converted to a discrete form. MPM uses two representations of the continuum, one based on a collection of material points and the other based on a computational grid. The material points are tracked through each computational step and they provide a Lagrangian description that is not subject to mesh entanglement. This feature permits constitutive equations with history-dependent variables to be applied at the material points without having to map the history parameters. A grid, which can be held fixed or be adaptive, is used to determine spatial gradients. Since the grid is used as an updated Lagrangian frame, the non-linear convection term associated with Eulerian formulations does not appear in the grid solution. Convection is modeled by moving the material points in the computed flow field and having the points carry properties. The discretization of the continuity equation, conservation of momentum, the energy equation and the constitutive equations, as used in MPM is discussed in turn.

2.2.1. Continuity equation

To define the material-point mass, first divide the initial configuration $\Omega_2(0)$ into N_p disjoint subdomains, Ω_p, whose union is the whole domain. Identify one material point, x_p, at the centroid of Ω_p. Let A_p be the area of Ω_p. Let the approximate density in the current configuration be given by

$$\rho(\boldsymbol{x}, t) = \sum_{i=1}^{N_p} m_p \delta(\boldsymbol{x} - \boldsymbol{x}_p). \tag{2.2.1}$$

The material points are given a mass m_p consistent with the initial density and geometry of the material, namely

$$m_p = \rho_p A_p. \tag{2.2.2}$$

Consistency with the continuity equation is obtained using constant material-point mass [4,10,12]. In practice, the subdomains, Ω_p, are formed in conjunction with the finite element mesh described below. Trial material points are placed evenly within the mesh and are kept if they fall within the body, and discarded otherwise.

2.2.2. Momentum equation

To obtain a discretization of the momentum equation, the differential equation is converted to a weak form and a finite-dimensional space is chosen in which to seek an approximate solution. The weak form is

derived by multiplying Eq. (2.1.2) by a test function $\boldsymbol{w}(\boldsymbol{x},t)$ and then integrating over the domain Ω_3. The test function is assumed to be zero over the part of the boundary where displacement is prescribed. For Cartesian coordinates, two-dimensional approximations are obtained by assuming that all functions are independent of z. Integrals over Ω_3 are reduced to integrals over Ω_2 by integrating out the z dependence. The result is an equation that holds per unit thickness in the z direction [4].

The computational domain is a region in $\mathbb{R}^2$ containing the material in Ω_2 and it is subdivided into N_e finite elements. The continuum equations are discretized on this underlying finite element mesh. For simplicity, the mesh is constructed from 4-node, isoparametric elements. Standard nodal basis functions, $N_i(\boldsymbol{x})$, associated with the spatial nodes, $\boldsymbol{x}_i$, $i = 1,\ldots,N_n$ are assembled from conventional finite element shape functions. Discrete equations are obtained by restricting the velocity, acceleration and test function to be in the span of the nodal basis functions. Therefore, $\boldsymbol{v}$ and $\mathrm{d}\boldsymbol{v}/\mathrm{d}t$ have the following representations:

$$\boldsymbol{v}(\boldsymbol{x},t) = \sum_{i=1}^{N_n} \boldsymbol{v}_i(t) N_i(\boldsymbol{x}), \tag{2.2.3}$$

$$\frac{\mathrm{d}\boldsymbol{v}}{\mathrm{d}t}(\boldsymbol{x},t) = \sum_{i=1}^{N_n} \frac{\mathrm{d}\boldsymbol{v}_i}{\mathrm{d}t}(t) N_i(\boldsymbol{x}). \tag{2.2.4}$$

Note that there is no derivative of the shape function in Eq. (2.2.4) since equations are solved using isoparametric elements in an updated Lagrangian frame. The test function $\boldsymbol{w}$ has a similar form, $\boldsymbol{w} = \sum_{i=1}^{N_n} \boldsymbol{w}_i(t) N_i(\boldsymbol{x})$.

The above equations are substituted into the weak form of the momentum equation. The components of $\boldsymbol{w}_i$ are arbitrary except where components of displacement are prescribed. Invoking the proper constraints on the displacement, the discretized momentum equation then becomes

$$\sum_{j=1}^{N_n} M_{ij}(t) \frac{\mathrm{d}\boldsymbol{v}_j}{\mathrm{d}t} = \boldsymbol{f}_i^{\mathrm{int}} + \boldsymbol{f}_i^{\mathrm{ext}}, \tag{2.2.5}$$

where the consistent nodal mass is calculated by

$$M_{ij}(t) = \sum_{p=1}^{N_p} m_p N_i(\boldsymbol{x}_p(t)) N_j(\boldsymbol{x}_p(t)), \tag{2.2.6}$$

the applied traction at a node is discretized as

$$\hat{\boldsymbol{t}}_i(t) = \int_{\partial\Omega_2^t} \boldsymbol{t}(\boldsymbol{x},t) N_i(\boldsymbol{x})\, \mathrm{d}S \tag{2.2.7}$$

and the nodal specific body force is calculated as

$$\boldsymbol{b}_i(t) = \sum_{p=1}^{N_p} m_p \boldsymbol{b}(\boldsymbol{x}_p(t),t) N_i(\boldsymbol{x}_p(t)). \tag{2.2.8}$$

For Cartesian coordinates, the components of the internal force at a node are $((f_x)_i^{\mathrm{int}}, (f_y)_i^{\mathrm{int}})$, where

$$(f_x)_i^{\mathrm{int}} = -\sum_{p=1}^{N_p} (m_p/\rho_p) \left\{ (\sigma_{xx})_p \frac{\partial N_i}{\partial x}(\boldsymbol{x}_p) + (\sigma_{xy})_p \frac{\partial N_i}{\partial y}(\boldsymbol{x}_p) \right\}, \tag{2.2.9}$$

$$(f_y)_i^{\mathrm{int}} = -\sum_{p=1}^{N_p} (m_p/\rho_p) \left\{ (\sigma_{xy})_p \frac{\partial N_i}{\partial x}(\boldsymbol{x}_p) + (\sigma_{yy})_p \frac{\partial N_i}{\partial y}(\boldsymbol{x}_p) \right\}. \tag{2.2.10}$$

The terms $(\sigma_{xx})_p$, $(\sigma_{xy})_p$ and $(\sigma_{yy})_p$ are the components of stress evaluated at the location of each material point. The external force vector is defined as

$$\boldsymbol{f}_i^{\text{ext}} = \boldsymbol{b}_i + \hat{\boldsymbol{t}}_i. \tag{2.2.11}$$

The diagonal lumped-mass matrix whose entries are the corresponding row sum of the consistent mass matrix defined in Eq. (2.2.6) is used in place of the consistent mass matrix, where the lumped-mass is

$$m_i(t) = \sum_{p=1}^{N_p} m_p N_i(\boldsymbol{x}_p(t)). \tag{2.2.12}$$

The discretized momentum equation with the lumped-mass matrix becomes

$$m_i \frac{\mathrm{d}\boldsymbol{v}_i}{\mathrm{d}t} = \boldsymbol{f}_i^{\text{int}} + \boldsymbol{f}_i^{\text{ext}}. \tag{2.2.13}$$

Note that this equation involves only quantities defined on the finite element grid.

The discrete equations must be solved at a discrete set of times, designated t^k, $k = 1, \ldots, K$. The discrete approximation of a material point or nodal quantity at time t^k is indicated by a superscript k on that quantity. The Lagrangian step of the calculation refers to the solution of the background mesh equations in an updated Lagrangian frame. The solution on the nodes of the mesh at the end of the Lagrangian step will be denoted with a superscript L.

In order to solve Eq. (2.2.13) for the nodal acceleration, $\mathrm{d}\boldsymbol{v}_i^L/\mathrm{d}t$, the forces at the nodes of the mesh must be calculated using Eqs. (2.2.9)–(2.2.11) and the nodal mass must be calculated using Eq. (2.2.12).

In order to use the acceleration, $\mathrm{d}\boldsymbol{v}_i^L/\mathrm{d}t$, in an explicit time integration of the nodal velocity, the velocity at the beginning of the step, $\boldsymbol{v}_i^k$, must be defined. Since a new finite element mesh is constructed at each time step, the data at time t^k for the solution over the time interval $[t^k, t^{k+1}]$ come from the material points rather than from the spatial mesh. At the beginning of the time step, a mass-weighted mapping of the velocity from the material points to the mesh is

$$m_i^k \boldsymbol{v}_i^k = \sum_{p=1}^{N_p} m_p \boldsymbol{v}_p^k N_i(\boldsymbol{x}_p^k). \tag{2.2.14}$$

Then, the nodal velocity at the end of the Lagrangian step is

$$\boldsymbol{v}_i^L = \boldsymbol{v}_i^k + \Delta t \frac{\mathrm{d}\boldsymbol{v}_i^L}{\mathrm{d}t}, \tag{2.2.15}$$

where Δt is the current time increment, $\Delta t = t^{k+1} - t^k$ [4,8,10].

2.2.3. Conservation of energy

If thermal conduction and heat generation are ignored, the internal energy per unit mass is given by Eq. (2.1.3). Since stress and strain are known at the material point locations, the internal energy can by integrated in time at these points

$$\rho_p(e_p^L - e_p^k) = \boldsymbol{\sigma}_p : (\boldsymbol{\epsilon}_p^L - \boldsymbol{\epsilon}_p^k). \tag{2.2.16}$$

2.2.4. Constitutive equations

The constitutive equations are evaluated for each material point to determine the strain and stress. For small deformations the rate equation for linear elasticity or isotropic plasticity is

$$\frac{\mathrm{d}\boldsymbol{\sigma}}{\mathrm{d}t} = \boldsymbol{T} : \frac{\mathrm{d}\boldsymbol{\epsilon}}{\mathrm{d}t}, \tag{2.2.17}$$

where $\boldsymbol{T}$ is the fourth-order tangent modulus. In Cartesian coordinates, if the nodal velocity is written as $\boldsymbol{v} = (v_x, v_y)$, the strain rate, given in Eq. (2.1.6), is discretized as [10]

$$(\Delta\epsilon_{xx})_p^k = \Delta t \sum_{i=1}^{N_n} (v_x)_i^k \frac{\partial N_i}{\partial x}(\boldsymbol{x}_p^k), \tag{2.2.18}$$

$$(\Delta\epsilon_{yy})_p^k = \Delta t \sum_{i=1}^{N_n} (v_y)_i^k \frac{\partial N_i}{\partial y}(\boldsymbol{x}_p^k), \tag{2.2.19}$$

$$(\Delta\epsilon_{xy})_p^k = (\Delta\epsilon_{yx})_p^k = \Delta t \sum_{i=1}^{N_n} \frac{1}{2}\left\{ (v_x)_i^k \frac{\partial N_i}{\partial y}(\boldsymbol{x}_p^k) + (v_y)_i^k \frac{\partial N_i}{\partial x}(\boldsymbol{x}_p^k) \right\}. \tag{2.2.20}$$

The stress update is written symbolically as

$$\boldsymbol{\sigma}_p^L = \boldsymbol{\sigma}_p^k + \Delta\boldsymbol{\sigma}_p^k = \boldsymbol{\sigma}_p^k + \boldsymbol{T} : \Delta\boldsymbol{\epsilon}_p^k. \tag{2.2.21}$$

For linear elasticity, the tangent modulus is the constant elasticity tensor. For elastoplasticity the tangent modulus is more complicated and is not formed explicitly in the numerical scheme. Instead, Eqs. (2.1.9)–(2.1.12) are discretized using backward Euler time differencing [10]. The resulting set of non-linear equations is solved under the assumption that the total strain increment is constant over a time step. The condition that f is zero at the end of the step is used to ensure that the stress state continues to lie on the yield surface for finite increments in strain as specified in Eq. (2.1.12). An iterative method is employed to solve the problem of finding a zero of f. For large deformations, the left-hand side of Eq. (2.2.17) is replaced by an objective rate (e.g. the Jaumann rate [10]).

2.3. MPM computational cycle—time discretization

A complete computational cycle of MPM consists of the following steps: an initialization phase where information is transferred from the material points to a grid, a Lagrangian phase where the equations of motion are solved in an updated Lagrangian frame on the grid, and a convective phase where the material points are updated and the grid is redefined. The original MPM code uses an explicit solver that is described first, followed by a description of the implicit solution.

2.3.1. Explicit formulation

During the initialization phase of MPM, information is gathered from the material points for the solution of the discrete momentum equation. The lumped mass matrix, Eq. (2.2.12), is formed at time t^k. The internal force is accumulated at each node using the stresses at the material points at time t^k in Eqs. (2.2.9) and (2.2.10), and the gradient of the nodal basis functions. Next, the nodal velocity is initialized. Since there are generally more material points than grid points, least squares is used to determine nodal velocities from the velocities at the material points weighted by the mass of the material point, Eq. (2.2.14).

During the Lagrangian phase of the computation, the discrete momentum equation, Eq. (2.2.13), is solved using the grid forces at time t^k. Processing in a Lagrangian frame means that the non-linear convective terms in purely Eulerian calculations do not appear. Once the nodal accelerations are computed, the velocities are updated at the nodes using Eq. (2.2.15). During this phase, the nodes are assumed to move according to this computed velocity

$$\boldsymbol{x}_i^L = \boldsymbol{x}_i^k + \Delta t \boldsymbol{v}_i^L. \tag{2.3.1}$$

During this Lagrangian step, each element is assumed to deform in the flow of material so that points in the interior of the element move in proportion to the motion of the nodes. Since nodal basis functions are

used to map the nodal velocity continuously to the interior of the element, the positions of the material points are updated by moving them in a single-valued, continuous velocity field

$$\boldsymbol{x}_p^L = \boldsymbol{x}_p^k + \Delta t \sum_{i=1}^{N_n} \boldsymbol{v}_i^L N_i(\boldsymbol{x}_p^k). \tag{2.3.2}$$

Similarly, the velocity of a material point is updated by mapping the nodal accelerations to the material point position

$$\boldsymbol{v}_p^L = \boldsymbol{v}_p^k + \Delta t \sum_{i=1}^{N_n} \frac{\mathrm{d}\boldsymbol{v}_i^L}{\mathrm{d}t} N_i(\boldsymbol{x}_p^k). \tag{2.3.3}$$

Because the velocity field is single-valued, interpenetration of material is precluded [2–4,8,10].

Strain, stress and history variables are maintained with each material point. Since each material point retains its material properties throughout the computation, applying constitutive equations at the material points allows easy evaluation and tracking of history-dependent variables. It also allows computations with multiple materials to be performed easily. To update stress, the Cartesian strain increments are evaluated at $\boldsymbol{x}_p$ as in Eqs. (2.2.18)–(2.2.20). The corresponding stress increment, $\Delta\boldsymbol{\sigma}_p$, is obtained from the strain increment using standard routines to evaluate the constitutive relations. Assuming small strain increments, the newly determined volumetric strain increment can be used to update the material point density using the continuity equation

$$\rho_p^L = \frac{\rho_p^k}{1 + \mathrm{tr}(\Delta\boldsymbol{\epsilon}_p)}. \tag{2.3.4}$$

Since the temperature is required in some constitutive equations, the temperature T_p is computed at $\boldsymbol{x}_p$ assuming that the energy dissipated due to plastic work goes to raise the temperature

$$\rho_p^L c(T_p^L - T_p^k) = \boldsymbol{\sigma}_p^L : \Delta\boldsymbol{\epsilon}_p^{\mathrm{pl}}, \tag{2.3.5}$$

where c is the specific heat at constant volume [10].

At this point in the computational cycle, the material points are updated with the new, time-advanced solution. During the convective phase, the material points are held fixed and the computational grid is redefined. The new grid can be chosen for computational convenience [2–4,8,10]. Since material points do not move during the convective phase, material point properties have the same value at the end of the convective phase as they had at the end of the Lagrangian phase, thus

$$\boldsymbol{x}_p^L = \boldsymbol{x}_p^{k+1}, \quad \boldsymbol{v}_p^L = \boldsymbol{v}_p^{k+1}, \quad \boldsymbol{\sigma}_p^L = \boldsymbol{\sigma}_p^{k+1}, \quad \boldsymbol{\epsilon}_p^L = \boldsymbol{\epsilon}_p^{k+1}, \quad \rho_p^L = \rho_p^{k+1}, \quad e_p^L = e_p^{k+1}, \quad T_p^L = T_p^{k+1}. \tag{2.3.6}$$

2.4. The implicit solution

The implicit formulation follows the basic structure of the explicit algorithm, with modifications to solve the grid equations implicitly during the Lagrangian phase. For the fully implicit formulation, the internal forces at the nodes given by Eqs. (2.2.9) and (2.2.10) are calculated at time t^L instead of time t^k. Eq. (2.2.21) updates the stress to obtain $\boldsymbol{\sigma}_p^L$. The discretized momentum equation, Eq. (2.2.13), has several terms that need to be evaluated at time t^L, so the equation becomes an implicit equation for the velocity.

For linear elasticity, $\boldsymbol{f}_i^{\mathrm{int}}$ depends linearly on the velocity, $\boldsymbol{v}_i$, through Eqs. (2.2.18)–(2.2.21), with $\boldsymbol{T} = \boldsymbol{E}$ in Eq. (2.2.21). The momentum equation becomes a system of linear equations solved to determine $\boldsymbol{v}_i^L$. For plasticity, the system of equations is non-linear since $\boldsymbol{f}_i^{\mathrm{int}}$ depends non-linearly on the nodal velocity when the stress increment in Eq. (2.2.21) is computed from the discretized equations corresponding to Eqs. (2.2.18)–(2.2.21). The solution of both the linear and non-linear equations is detailed in this section.

One means of solving the system of equations is to use an iterative scheme which computes the solution to within a prescribed margin of error. The choice of method depends, in part, on the characteristics of the system of equations. There are two formulations of MPM available, the velocity formulation and the momentum formulation [8]. If the velocity formulation is used, the linear equations are symmetric. This is the method outlined above. If the momentum formulation is used, the linear equations are non-symmetric. This method is described in Section 3.2. Symmetric equations are solved using the CG method, and the GMRES method is used to solve the non-symmetric case.

As in the explicit formulation, the material points are updated at the end of the Lagrangian phase with the time-advanced solution. Again, the material points are held fixed and the computational grid is redefined during the convective phase. Since material points do not move during the convective phase, material-point properties have the same value at the end of the convective phase as they had at the end of the Lagrangian phase. Thus Eq. (2.3.6) also holds for the implicit solution.

2.4.1. The linear implicit equations

In Section 2, the constitutive equations for different materials are discussed. One of the simplest constitutive equations describes a linearly elastic material, which leads to the simplest system of equations for the discretized momentum equation, Eq. (2.2.13). This system of equations, which is linear in velocity, will be developed in this section.

The momentum equation can be written with a parameter, μ, which allows the user to adjust between explicit and implicit forces in the equation

$$m_i^k \frac{\boldsymbol{v}_i^{L,\mathrm{imp}} - \boldsymbol{v}_i^k}{\Delta t} = \mu \boldsymbol{f}_i^L + (1-\mu)\boldsymbol{f}_i^k. \tag{2.4.1}$$

Valid values are $0 \leqslant \mu \leqslant 1$, with $\mu = 0$ being fully explicit and $\mu = 1$ being fully implicit. When $0 \leqslant \mu < 1$ the method is conditionally stable; the usual CFL stability constraint applies to the time step if $\mu = 0$. It is worth mentioning that the CFL condition is based on the background mesh size, rather than the material-point spacing. When $\mu = 1$ the method is unconditionally stable [18]. The method is also dissipative for $\mu \neq 0$. In Eq. (2.4.1), the L and 'imp' superscript refers to the velocity at the end of the Lagrangian step calculated from the implicit forces. This is the unknown variable for which the iterative method solves. The external forces have been dropped from this equation to simplify the presentation, but they can be easily combined with the internal forces.

The nodal quantities in Eq. (2.4.1) with a superscript k are calculated at the beginning of the time step from the material point values. The implicit force on the right-hand side needs to be described in terms of the unknown velocity and known quantities in order to write the equation in terms of a matrix-vector multiplication. The forces at each node at the beginning of the time step, $\boldsymbol{f}_i^k$, are known; thus, the forces at the end of the Lagrangian step can be written as

$$\boldsymbol{f}_i^L = \boldsymbol{f}_i^k + \Delta \boldsymbol{f}_i, \tag{2.4.2}$$

where $\Delta \boldsymbol{f}_i$ is a quantity depending implicitly on the velocity. With this substitution, Eq. (2.4.1) becomes

$$m_i^k \boldsymbol{v}_i^{L,\mathrm{imp}} = m_i^k \boldsymbol{v}_i^k + \Delta t(\boldsymbol{f}_i^k + \mu \Delta \boldsymbol{f}_i). \tag{2.4.3}$$

The quantities needed to calculate the explicit momentum, the case where $\mu = 0$, at time L are known. The explicit momentum is designated with a $L, \exp$ superscript and is calculated from the equation

$$m_i^k \boldsymbol{v}_i^{L,\exp} = m_i^k \boldsymbol{v}_i^k + \Delta t \boldsymbol{f}_i^k. \tag{2.4.4}$$

Combining the last two equations and accumulating all unknown quantities on the left-hand side and all known quantities on the right-hand side results in the following expression:

$$m_i^k \boldsymbol{v}_i^{L,\mathrm{imp}} - \mu \Delta t \Delta \boldsymbol{f}_i = m_i^k \boldsymbol{v}_i^{L,\mathrm{exp}}. \tag{2.4.5}$$

It remains to show how $\Delta \boldsymbol{f}_i$ depends on the implicit velocity.

The internal forces at time step k are calculated from Eqs. (2.2.9) and (2.2.10), which can be written as

$$\boldsymbol{f}_i^k = -\sum_{p=1}^{N_p} (m_p/\rho_p) \frac{\partial N_i}{\partial \boldsymbol{x}}(\boldsymbol{x}_p^k)\boldsymbol{\sigma}_p^k, \tag{2.4.6}$$

while the forces at time step L are calculated from Eqs. (2.2.9) and (2.2.10) evaluated at t^L. Since the natural coordinates of a material point do not change during a Lagrangian step, these can be written as

$$\boldsymbol{f}_i^L = -\sum_{p=1}^{N_p} (m_p/\rho_p) \frac{\partial N_i}{\partial \boldsymbol{x}}(\boldsymbol{x}_p^k)\boldsymbol{\sigma}_p^L. \tag{2.4.7}$$

Define the quantity $\Delta \boldsymbol{\sigma}_p$ as

$$\Delta \boldsymbol{\sigma}_p = \boldsymbol{\sigma}_p^L - \boldsymbol{\sigma}_p^k. \tag{2.4.8}$$

Keeping the density fixed over the time step, the last three equations combine to become

$$\begin{aligned} \boldsymbol{f}_i^L &= -\sum_{p=1}^{N_p} (m_p/\rho_p) \frac{\partial N_i}{\partial \boldsymbol{x}}(\boldsymbol{x}_p^k)(\boldsymbol{\sigma}_p^k + \Delta \boldsymbol{\sigma}_p) \\ &= \boldsymbol{f}_i^k - \sum_{p=1}^{N_p} (m_p/\rho_p) \frac{\partial N_i}{\partial \boldsymbol{x}}(\boldsymbol{x}_p^k)\Delta \boldsymbol{\sigma}_p. \end{aligned} \tag{2.4.9}$$

Comparing this equation with (2.4.2) leads to a formula for $\Delta \boldsymbol{f}_i$ in terms of the stresses:

$$\Delta \boldsymbol{f}_i = -\sum_{p=1}^{N_p} (m_p/\rho_p) \frac{\partial N_i}{\partial \boldsymbol{x}}(\boldsymbol{x}_p^k)\Delta \boldsymbol{\sigma}_p. \tag{2.4.10}$$

In the case of a linearly elastic material, the stress increment is a linear function of the velocity, as seen in Eqs. (2.2.18)–(2.2.21). Thus, $\Delta \boldsymbol{f}_i$ is also a linear function of velocity and the system of equations (2.4.5) is a linear system.

Eq. (2.4.5) is the implicit equation to be solved for the unknown vector of nodal velocities $\boldsymbol{v}^{L,\mathrm{imp}}$. If N_n denotes the total number of nodes, then the vector $\boldsymbol{v}^{L,\mathrm{imp}}$ is a concatenation of all velocity components at all nodes, and nominally has length $N = 3N_n$ in three dimensions, $N = 2N_n$ in two dimensions, and $N = N_n$ in one dimension. The total number of unknowns is adjusted from the total number of degrees of freedom according to the boundary conditions. Nodes are excluded from the iterative process when displacement boundary conditions are applied. Eq. (2.4.5) is a system of the form $\boldsymbol{A}\boldsymbol{v} = \boldsymbol{b}$, where

$$\begin{aligned} \boldsymbol{A}\boldsymbol{v} &= \boldsymbol{M}\boldsymbol{v}^{L,\mathrm{imp}} - \mu \Delta t \Delta \boldsymbol{f}, \\ \boldsymbol{b} &= \boldsymbol{M}\boldsymbol{v}^{L,\mathrm{exp}}. \end{aligned} \tag{2.4.11}$$

In Eqs. (2.4.11), $\boldsymbol{M}$ denotes the diagonal lumped mass matrix. It should be noted that for the MPM equations of motion, the matrix $\boldsymbol{A}$ has the additional property of being a banded and sparse, as in standard finite element discretizations. In one-dimensional problems the matrix has three non-zero diagonals. In two-dimensional problems the matrix has nine non-zero diagonals, and in three-dimensional problems the matrix has 27 non-zero diagonals when tensor products of linear shape functions are used on quadrilateral elements or brick elements, respectively.

As the program begins processing for each time step, values have been mapped from the material points to the elements. Stress and strain are calculated on the material points and forces are computed on the elements. This provides values for $\boldsymbol{v}^k$, m^k, $m\boldsymbol{v}^k$, $\boldsymbol{\sigma}^k$, $\boldsymbol{f}^k$, ρ^k, ΔT^k, and $\boldsymbol{\epsilon}^k$. Only $\boldsymbol{v}$ is updated during the implicit step.

2.4.2. The non-linear implicit equations

As demonstrated in the previous section, the constitutive equation for a linearly elastic material leads to a linear system of equations when solving the conservation of momentum equation for the time-advanced velocity. For more general types of materials, the constitutive equations are given by Eqs. (2.1.9)–(2.1.12). Through these equations, the stress increment is a non-linear function of the velocity. Therefore, the force increment in Eq. (2.4.10) is also a non-linear function of the velocity. Since the rest of the previous analysis holds, the resulting system of equations for a general constitutive model has the same form as Eq. (2.4.5), except that $\Delta \boldsymbol{f}_i$ is now a non-linear function of the velocity $\boldsymbol{v}$.

For this system, a non-linear vector function $\boldsymbol{F}(\boldsymbol{v})$ is defined as follows:

$$\boldsymbol{F}(\boldsymbol{v}^{L,\mathrm{imp}}) = \boldsymbol{M}\boldsymbol{v}^{L,\mathrm{imp}} - \mu\Delta t\Delta \boldsymbol{f}\ (\boldsymbol{v}^{L,\mathrm{imp}}) - \boldsymbol{M}\boldsymbol{v}^{L,\mathrm{exp}}. \tag{2.4.12}$$

Solving the discretized momentum equation as given in Eq. (2.4.5) is then equivalent to solving the non-linear equation

$$\boldsymbol{F}(\boldsymbol{v}^{L,\mathrm{imp}}) = \boldsymbol{0}. \tag{2.4.13}$$

One standard method of solving a system of non-linear equations is Newton's method [13]. Since Newton's method is applied to a specific time step, the time superscript has been dropped from the variables in the following discussions.

In Newton's method, the function $\boldsymbol{F}(\boldsymbol{v})$ is expanded in a Taylor series around a given value of the unknown variable $\boldsymbol{v}$, say $\boldsymbol{v}_0$, and the resulting linear approximation is substituted for $\boldsymbol{F}(\boldsymbol{v})$ in Eq. (2.4.13) to obtain the equation

$$\boldsymbol{J}(\boldsymbol{v}_0)(\boldsymbol{v} - \boldsymbol{v}_0) = -\boldsymbol{F}(\boldsymbol{v}_0), \tag{2.4.14}$$

where $\boldsymbol{J}$ is the Jacobian of the function $\boldsymbol{F}$ evaluated at the given value of $\boldsymbol{v}$. By defining the quantity

$$\boldsymbol{s} = \boldsymbol{v} - \boldsymbol{v}_0, \tag{2.4.15}$$

Eq. (2.4.14) can be written as the linear system

$$\boldsymbol{J}(\boldsymbol{v}_0)\boldsymbol{s} = -\boldsymbol{F}(\boldsymbol{v}_0). \tag{2.4.16}$$

The solution to Eq. (2.4.16) is used to update the unknown variable $\boldsymbol{v}$. The algorithm for Newton's method is described next in Algorithm 1, where the subscript refers to the iteration number [13,14].

Algorithm 1

Step 1. Choose the initial value $\boldsymbol{v}_0$ and an appropriate error tolerance. Set $n = 0$.
Step 2. Evaluate $\boldsymbol{F}(\boldsymbol{v}_n)$. If $\|\boldsymbol{F}(\boldsymbol{v}_n)\|_2$ is less than the error tolerance, terminate the iteration. If not, continue.
Step 3. Compute $\boldsymbol{J}(\boldsymbol{v}_n)$.
Step 4. Solve $\boldsymbol{J}(\boldsymbol{v}_n)\boldsymbol{s} = -\boldsymbol{F}(\boldsymbol{v}_n)$ for $\boldsymbol{s}$.
Step 5. Update the solution by $\boldsymbol{v}_{n+1} = \boldsymbol{v}_n + \boldsymbol{s}$, set n to $n + 1$, and return to Step 2.

A direct method such as LU or QR factorization can be used to solve Step 4. In this case, Newton's method converges q-quadratically to the solution $\boldsymbol{v}^*$, that is

$$\|\boldsymbol{v}_{n+1} - \boldsymbol{v}^*\|_2 \leqslant K\|\boldsymbol{v}_n - \boldsymbol{v}^*\|_2^2 \tag{2.4.17}$$

for n sufficiently large and some $K > 0$ if the initial iterate is sufficiently close to the solution [13].

Direct computation of Step 3 and the solution of the equation in Step 4 by direct methods is quite costly for large systems of equations [13]. If a Krylov method is used to solve Step 4, only the matrix-vector product $\boldsymbol{J}(\boldsymbol{v})\boldsymbol{s}$ needs to be evaluated. Calculation of the matrix in Step 3 and factoring the matrix in Step 4 can be eliminated. This goal can be achieved using a difference approximation for the directional derivative of $\boldsymbol{F}(\boldsymbol{v})$ in the direction of the vector $\boldsymbol{s}$ to estimate the effect of the Jacobian. The directional derivative is calculated as

$$\boldsymbol{J}(\boldsymbol{v})\boldsymbol{s} \approx D_h\boldsymbol{F}(\boldsymbol{v},\boldsymbol{s}) = \begin{cases} \boldsymbol{0}, & \boldsymbol{s} = \boldsymbol{0}, \\ \|\boldsymbol{s}\|_2 \dfrac{\boldsymbol{F}(\boldsymbol{v} + h\|\boldsymbol{v}\|_2\boldsymbol{s}/\|\boldsymbol{s}\|_2) - \boldsymbol{F}(\boldsymbol{v})}{h\|\boldsymbol{v}\|_2}, & \boldsymbol{s} \neq \boldsymbol{0},\ \boldsymbol{v} \neq \boldsymbol{0}, \\ \|\boldsymbol{s}\|_2 \dfrac{\boldsymbol{F}(h\boldsymbol{s}/\|\boldsymbol{s}\|_2) - \boldsymbol{F}(\boldsymbol{0})}{h}, & \boldsymbol{s} \neq \boldsymbol{0},\ \boldsymbol{v} = \boldsymbol{0}. \end{cases} \tag{2.4.18}$$

For small h, Eq. (2.4.18) provides a reasonable approximation of $\boldsymbol{J}(\boldsymbol{v})\boldsymbol{s}$ with only one extra function evaluation since $\boldsymbol{F}(\boldsymbol{v})$ is already determined [13]. It should be noted that this forward difference approximation to the derivative is not a linear operator in $\boldsymbol{s}$ [13].

Substituting Eq. (2.4.18) into Step 4 leaves the following equation to be solved for $\boldsymbol{s}$:

$$D_h\boldsymbol{F}(\boldsymbol{v}_n,\boldsymbol{s}) = -\boldsymbol{F}(\boldsymbol{v}_n). \tag{2.4.19}$$

Using an iterative method to solve this equation inside a Newton step is generally referred to as an inexact Newton method. According to [13], if the initial iterate is sufficiently close to the solution, the inexact Newton method exhibits q-linear convergence locally, that is

$$\|\boldsymbol{v}_{n+1} - \boldsymbol{v}^*\|_2 \leqslant K\|\boldsymbol{v}_n - \boldsymbol{v}^*\|_2 \tag{2.4.20}$$

for n sufficiently large, where $0 < K < 1$ and $\boldsymbol{v}^*$ is the exact solution. In fact, for small enough tolerances on the linear solver the inexact Newton method has q-superlinear convergence, where

$$\|\boldsymbol{v}_{n+1} - \boldsymbol{v}^*\|_2 \leqslant K\|\boldsymbol{v}_n - \boldsymbol{v}^*\|_2^{\alpha} \tag{2.4.21}$$

for some $K > 0$ and $\alpha > 1$. If the linear solver is iterated to convergence the method has q-quadratic convergence as defined above.

Next, we cover the iterative methods for symmetric and non-symmetric systems used to solve the linear equations and Step 4 in Newton's method.

2.4.3. Conjugate gradient method

Both the CG and GMRES methods are Krylov space methods which can be used to solve linear systems of equations of the form $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, where $\boldsymbol{A}$ is $N \times N$. Let $\boldsymbol{x}_0$ be the initial iterate, $\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0$ be the initial residual, then the kth Krylov subspace is defined to be $\mathscr{K}_k = \mathrm{span}\{\boldsymbol{r}_0, \boldsymbol{A}\boldsymbol{r}_0, \ldots, \boldsymbol{A}^{k-1}\boldsymbol{r}_0\}$, for $k \geqslant 1$. Krylov space methods minimize some error measure over the space $\boldsymbol{x}_0 + \mathscr{K}_k$ in the kth iteration [13] A useful property of these methods is that only a routine providing the result of a matrix-vector product is required, the matrix $\boldsymbol{A}$ does not need to be formed or stored. For this reason these are often called matrix-free methods and they fit well with the choice of approximate Jacobian given in Eq. (2.4.18).

The CG method is a Krylov space method used to solve systems where $\boldsymbol{A}$ is symmetric positive definite. The kth iterate in the CG method minimizes the function

$$\Phi(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x} - \boldsymbol{x}^{\mathrm{T}}\boldsymbol{b} \tag{2.4.22}$$

over the space $\boldsymbol{x}_0 + \mathscr{K}_k$. For a symmetric matrix

$$\nabla\Phi(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} = -\boldsymbol{r}. \tag{2.4.23}$$

Thus, if $\boldsymbol{x}_k$ minimizes Eq. (2.4.22) on $\boldsymbol{x}_0 + \mathscr{K}_k$, the kth residual is orthogonal to all the vectors in $\mathscr{K}_k$ [13]. Eq. (2.4.23) implies that when $\boldsymbol{A}$ is symmetric the minimum of Φ occurs in $\mathbb{R}^N$ at the solution of $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$.

In the CG method, the new search direction $\boldsymbol{p}_{k+1}$ is required to be in $\mathscr{K}_{k+1}$ and to be A-orthogonal to all vectors in $\mathscr{K}_k$ [13]. Two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ are A-orthogonal if $\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{y} = 0$. This condition uniquely determines the search direction. Given this new search direction, the new iterate is

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_{k+1}\boldsymbol{p}_{k+1} \tag{2.4.24}$$

and α_{k+1} can be easily determined by minimizing $\Phi(\boldsymbol{x}_{k+1})$ with respect to α_{k+1}.

From this definition of the iterates, the kth residual vector is calculated to be

$$\boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \alpha_k \boldsymbol{A}\boldsymbol{p}_k. \tag{2.4.25}$$

It can be shown that $\boldsymbol{r}_i \in \mathscr{K}_k$ for $i \leqslant k-1$, therefore $\boldsymbol{r}_k \in \mathscr{K}_{k+1}$. Because $\boldsymbol{p}_i \in \mathscr{K}_k$ for $i \leqslant k$, $\boldsymbol{r}_k$ is orthogonal to all previous search directions and A-orthogonal to all of them except $\boldsymbol{p}_k$. The search direction $\boldsymbol{p}_{k+1}$ can thus be constructed from the residual and the previous search direction

$$\boldsymbol{p}_{k+1} = \boldsymbol{r}_k + \beta_{k+1}\boldsymbol{p}_k. \tag{2.4.26}$$

By the minimization property of CG, $\boldsymbol{r}_k$ is orthogonal to all the previous residuals and A-orthogonal to all previous residuals except $\boldsymbol{r}_{k-1}$, since $\boldsymbol{A}\mathscr{K}_{k-1} \subset \mathscr{K}_k$ [13]. Thus, β_{k+1} is easily calculated by enforcing the A-orthogonality condition with the last residual. For this reason, only the last residual needs to be retained for the next iteration.

During each iteration of the CG method, a new search direction is determined, the solution and the residual are updated, and a norm for the new residual is calculated. If this norm is small enough, the algorithm terminates. The first search direction is the initial residual of the linear system which is also a basis for the first Krylov subspace. For the linear equations described in Section 2.4, the CG iteration is applied to the system $\boldsymbol{A}\boldsymbol{v} = \boldsymbol{b}$ given by Eqs. (2.4.11) for a specific time. Dropping the time superscript and letting the k subscript refer to the CG iteration number, the method proceeds as given in Algorithm 2 [13,14].

Algorithm 2

Step 1. Choose an initial $\boldsymbol{v}_0$. Calculate $\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{v}_0$ and $\rho_0 = \|\boldsymbol{r}_0\|_2^2$. Set $k = 1$ and $\boldsymbol{p}_1 = \boldsymbol{r}_0$. Select an appropriate error tolerance.

Step 2. If the error tolerance is met by the residual, terminate the iteration. If not, continue with the following calculations:

(a) $\boldsymbol{w}_k = \boldsymbol{A}\boldsymbol{p}_k$,
(b) $\alpha_k = \rho_{k-1}/\boldsymbol{p}_k^{\mathrm{T}}\boldsymbol{w}_k$,
(c) $\boldsymbol{v}_k = \boldsymbol{v}_{k-1} + \alpha_k\boldsymbol{p}_k$,
(d) $\boldsymbol{r}_k = \boldsymbol{r}_{k-1} - \alpha_k\boldsymbol{w}_k$,
(e) $\rho_k = \|\boldsymbol{r}_k\|_2^2$,
(f) $\beta_k = \rho_k/\rho_{k-1}$,
(g) $\boldsymbol{p}_{k+1} = \boldsymbol{r}_k + \beta_k\boldsymbol{p}_k$,
(h) Replace k with $k+1$.

Repeat Step 2 as necessary until the error tolerance is met.

For the non-linear equations in Section 2.4, the CG iteration is applied to Eq. (2.4.19) for a specific non-linear iterate $\boldsymbol{v}_n$. With the k subscript referring to the CG iteration number, the method proceeds as in Algorithm 3.

Algorithm 3

Step 1. Set $\boldsymbol{s}_0 = \boldsymbol{0}$. Calculate $\boldsymbol{r}_0 = -\boldsymbol{F}(\boldsymbol{v}_n)$ and $\rho_0 = \|\boldsymbol{r}_0\|_2^2$. Set $k = 1$ and $\boldsymbol{p}_1 = \boldsymbol{r}_0/\|\boldsymbol{r}_0\|_2$. Select an appropriate error tolerance.

Step 2. If the error tolerance is met by the residual, terminate the iteration. If not, continue with the following calculations:

(a) $\boldsymbol{w}_k = D_h\boldsymbol{F}(\boldsymbol{v}_n, \boldsymbol{p}_k)$,
(b) $\alpha_k = \rho_{k-1}/\boldsymbol{p}_k^T\boldsymbol{w}_k$,
(c) $\boldsymbol{s}_k = \boldsymbol{s}_{k-1} + \alpha_k\boldsymbol{p}_k$,
(d) $\boldsymbol{r}_k = \boldsymbol{r}_{k-1} - \alpha_k\boldsymbol{w}_k$,
(e) $\rho_k = \|\boldsymbol{r}_k\|_2^2$,
(f) $\beta_k = \rho_k/\rho_{k-1}$,
(g) $\boldsymbol{p}_{k+1} = \boldsymbol{r}_k + \beta_k\boldsymbol{p}_k$,
(h) Replace k with $k + 1$.

Repeat Step 2 as necessary until the error tolerance is met.

The cost of each CG iteration is one matrix-vector product, two inner products of vectors and three operations of the form $a\boldsymbol{x} + \boldsymbol{y}$, where $\boldsymbol{x}$ and $\boldsymbol{y}$ are vectors and a is a scalar. Only four vectors are stored [13]. In exact arithmetic, CG will converge to the solution within N iterations, where N is the length of the unknown vector [13]. Actual performance of the method depends on the condition number of the matrix and clustering of the eigenvalues [13]. Preconditioning can improve the performance by clustering the eigenvalues near one. The CG iteration can be preconditioned at the cost of one solution of a system with the preconditioning matrix and one more inner product of two vectors [13].

2.4.4. Generalized minimum residual method

GMRES can be used to solve a system of linear equations when the system matrix is non-symmetric [15]. The kth iterate in the GMRES method is the solution to the least squares problem,

$$\min_{\boldsymbol{x}\in\boldsymbol{x}_0+\mathscr{K}_k} \|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}\|_2. \tag{2.4.27}$$

The underlying idea for GMRES is similar to that for CG, but the loss of symmetry means that, in principle, the entire basis for the Krylov space must be stored and the storage grows with the number of steps [13]. The least squares problem needs to be converted to a more computationally efficient form.

The following discussion is after Kelley [13] and Saad and Schultz [15]. Given an orthogonal projector $\boldsymbol{V}_k$ of $\mathbb{R}^{\mathbb{k}}$ onto $\mathscr{K}_k$, the goal of the least squares problem becomes finding $\boldsymbol{x}_k = \boldsymbol{x}_0 + \boldsymbol{V}_k\boldsymbol{y}$ where $\boldsymbol{y}$ is characterized by

$$\min_{\boldsymbol{y}\in\mathbb{R}^{\mathbb{k}}} \|\boldsymbol{r}_0 - \boldsymbol{A}\boldsymbol{V}_k\boldsymbol{y}\|_2. \tag{2.4.28}$$

To avoid multiplying $\boldsymbol{A}\boldsymbol{V}_k$ during each iteration in (2.4.28), Gram-Schmidt orthogonalization, starting from the initial residual $\boldsymbol{r}_0$, is used to form an orthonormal basis for $\mathscr{K}_k$ which is stored in the columns of $\boldsymbol{V}_k$. This step is called the Arnoldi process and produces orthogonal matrices such that $\boldsymbol{A}\boldsymbol{V}_k = \boldsymbol{V}_{k+1}\boldsymbol{H}_k$, where $\boldsymbol{H}_k$ is a $k + 1 \times k$ upper Hessenberg matrix. In practice, a modified Gram–Schmidt process is used, as it is mathematically equivalent to the Gram–Schmidt process but more likely to maintain numerical orthogonality of the basis [13]. Since $\boldsymbol{r}_0 = \beta\boldsymbol{V}_{k+1}\boldsymbol{e}_1$, where $\beta = \|\boldsymbol{r}_0\|_2$ and $\boldsymbol{e}_1 = (1, 0, \ldots, 0)^{\mathrm{T}}$, the least squares problem can now be rewritten to find $\boldsymbol{y}$ which satisfies

$$\min_{\boldsymbol{y}\in\mathbb{R}^{\mathbb{k}}} \|\beta\boldsymbol{e}_1 - \boldsymbol{H}_k\boldsymbol{y}\|_2. \tag{2.4.29}$$

To solve (2.4.29), Givens rotations are used to reduce the Hessenberg matrix to triangular form. Notice that the only non-zero entries in the first column of $\boldsymbol{H}_k$ are the components h_{11} and h_{21}. The appropriate Givens

rotation applied to the first two rows will annihilate h_{21} and change h_{11} and subsequent columns. The second column has non-zero entries only for h_{12} (perhaps), h_{22} and h_{23}. A Givens rotation applied to the second and third rows will annihilate h_{23} and not effect the first column. This process is continued across the matrix one column at a time. The result is a QR factorization of $\boldsymbol{H}_k$, $\boldsymbol{H}_k = \boldsymbol{Q}_k\boldsymbol{R}_k$, where $\boldsymbol{Q}_k$ is the product of the k Givens rotations, and $\boldsymbol{R}_k$ is a $k+1 \times k$ upper triangular matrix. It should be noted that the application of Givens rotations can be performed on each column as it is added to $\boldsymbol{H}$. Thus $\boldsymbol{Q}_k$ is never explicitly formed; only a record of the previous Givens rotations is kept, which requires storage of a $k \times 2$ vector.

Setting $\boldsymbol{g} = \beta\boldsymbol{Q}_k\boldsymbol{e}_1$, the least squares problem can be rewritten once more to find $\boldsymbol{y}$ which satisfies

$$\min_{\boldsymbol{y}\in\mathbb{R}^k} \|\boldsymbol{g} - \boldsymbol{R}_k\boldsymbol{y}\|_2. \tag{2.4.30}$$

Since the $(k+1)$st row of $\boldsymbol{R}_k$ consists entirely of zeros, a solution to (2.4.30) can now be computed by solving the triangular system. One nice property of GMRES is that the norm of the residual is $|g_{k+1}|$ which is available without actually computing the residual vector or the solution. The solution vector is only computed once after the last iteration. A summary of the GMRES algorithm, which is applied to the non-linear equations in Section 2.4, is given next. As with CG, the GMRES iteration is applied to Eq. (2.4.19) for a specific $\boldsymbol{v}_n$. With the k subscript referring to the GMRES iteration number, this method applied to Eq. (2.4.19) is given in Algorithm 4.

Algorithm 4

Step 1. Set $\boldsymbol{s}_0 = \boldsymbol{0}$. Calculate $\boldsymbol{r}_0 = -\boldsymbol{F}(\boldsymbol{v}_n)$ and $\rho_0 = \|\boldsymbol{r}_0\|_2^2$. Set $k = 1$, $\boldsymbol{p}_1 = \boldsymbol{r}_0/\|\boldsymbol{r}_0\|_2$ and $\boldsymbol{\beta} = \|\boldsymbol{r}_0\|_2\boldsymbol{e}_1$. Select an appropriate error tolerance.

Step 2. If the residual meets the error tolerance, terminate the algorithm. If not, continue.

(a) $\boldsymbol{p}_{k+1} = D_h\boldsymbol{F}(\boldsymbol{v}_n, \boldsymbol{p}_k)$,
(b) for $i = 1, 2, \ldots, k$ set $h_{k,i} = \boldsymbol{p}_{k+1}^T\boldsymbol{p}_i$, $\boldsymbol{p}_{k+1} = \boldsymbol{p}_{k+1} - h_{k,i}\boldsymbol{p}_i$,
(c) $h_{k+1,k} = \|\boldsymbol{p}_{k+1}\|_2$,
(d) $\boldsymbol{p}_{k+1} = \boldsymbol{p}_{k+1}/h_{k+1,k}$,
(e) apply all previous Givens rotations to $(h_{1,k}, \ldots, h_{k+1,k})^{\mathrm{T}}$,
(f) construct next Givens rotation on $h_{k,k}$ and $h_{k+1,k}$ to eliminate $h_{k+1,k}$,
(g) apply new Givens rotation to $\boldsymbol{\beta}$; β_{k+1} is the new norm estimate of the residual,
(h) Replace k with $k+1$.

Repeat Step 2 until the error tolerance is met.

Step 3. Solve the system $\boldsymbol{H}\boldsymbol{y} = \boldsymbol{\beta}$.

Step 4. Calculate the solution $\boldsymbol{s} = \boldsymbol{s}_0 + y_1\boldsymbol{p}_1 + \cdots + y_k\boldsymbol{p}_k$.

The kth GMRES iteration requires a matrix-vector product and k inner products of two vectors to obtain the new basis vector [13]. This process rapidly becomes expensive for a large system. GMRES can be stopped after a number of iterations and restarted using the last solution as the new initial guess. Restarting the algorithm limits the number of previous search vectors that must be stored, but can effect convergence [15]. However, in practice, it is just as effective for the non-linear problem to use the solution after a given number of steps to compute the next non-linear guess [16,17].

3. Results

In this section, we solve various problems using the unconditionally stable, fully implicit method ($\mu = 1$). In [18], a conjugate gradient solver is used to perform preliminary tests involving only linear equations.

Although the linear solver is useful for problems involving linearly elastic materials, a practical numerical method must be capable of solving non-linear problems. Due to the success of the CG method in the linear case [18], the non-linear case is first examined using the Newton–CG method outlined in Algorithms 1 and 3 in Section 2.4. The non-linear system is given by Eq. (2.4.13). The directional derivative given in Eq. (2.4.18) is used to approximate the Jacobian of the non-linear function.

3.1. The symmetric non-linear two-dimensional solver

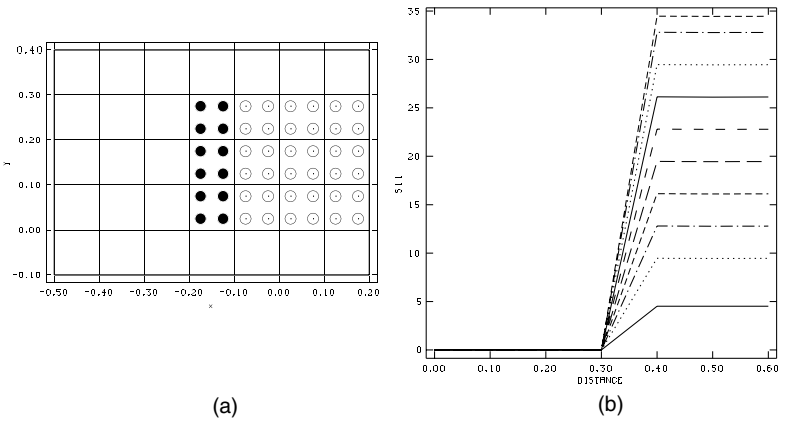
Before Newton–Krylov methods can be invoked, appropriate values for the parameters controlling the Newton convergence, the linear convergence, and the step in the directional derivative, must be chosen. In later sections, these three parameters are referred to as the Newton parameter, the linear parameter and the directional derivative parameter, respectively. Detailed tests are not presented for the Newton–CG method; however, there is a discussion of these choices for the Newton–GMRES method. According to [13], the parameter in the directional derivative should be chosen to be approximately the square root of numerical roundoff error. In double precision, this would be approximately 10^{-7}. However, in many cases, setting the parameter an order of magnitude or two larger, results in more robust iterations. This idea is developed further below.

3.1.1. Linear two-dimensional simulations

Two simulations of linearly elastic materials are examined to test the non-linear solver. The first linear, isotropic, elastic simulation involves tension in a bar. A rigid body attached to the left end of the bar moves with a constant velocity in the negative x-direction to provide the loading. A no-slip boundary condition is applied on the right boundary. Fig. 1a shows the initial configuration of the material points and the computational grid for such a bar. For this quasi-static loading, it is expected that the stress will be uniform along the bar at all times and, assuming uniaxial stress in the x-direction, the σ_{xx} component of the stress should equal

$$\sigma_{xx}(t) = Y\frac{v_r(t-t_0)}{L}, \tag{3.1.1}$$

where Y is the Young's modulus of the material, v_r is the velocity of the rigid body, t_0 is the initial time and L is the length of the bar.



(a) (b)

Fig. 1. The bar under tension. Initial configuration of the bar is shown in (a), stress in the middle of the bar as a function of distance along the bar for the implicit case is shown in (b) for 10 different times.

For the simulation, the domain of computation is 0.7 by 0.5. Eight nodes are used in the x-direction and six nodes in the y-direction, for a total of 48 nodes, with a resulting square mesh with side length $\Delta x = \Delta y = 0.1$. The bar is 0.3 by 0.3, with four material points per element. The rigid body is 0.1 by 0.3, with four material points per element. The bar has a Young's modulus of 1000.0, a Poisson's ratio of 0.3 and a density of 10.0. The initial velocity in the x-direction of the rigid body is -0.001. The problem is run to a maximum time of $t = 10.$, at which time the bar should be strained 3.45%. Fig. 1b shows the stress in the middle of the bar as a function of distance along the bar. Values of stress are graphed on the background computational mesh; therefore, the stress is constant on the right side of the plot where mesh points overlap with the bar. The stress drops to zero over one mesh width at the left end of the bar, and is zero at mesh points having no overlap with the bar. As time increases, the length of the bar increases and the stress level within the bar increases, as expected. At each time, the stress within the bar is constant and at the correct level based on the analytical solution. The figure shows the data for an implicit calculation with an initial time step equal to that of the explicit calculation, increasing at a rate of 20% per step up to a time step 100 times that of the explicit calculation. The stresses shown in the figure correspond to the times $t = 1.13$, 2.36, 3.34, 4.34, 5.34, 6.34, 7.34, 8.34, 9.34, 10.34. The expected final stress value is 34.5, which is also the predicted final stress shown in the figure.

For the implicit calculations done with a time step equal to that of the explicit calculation or 10 times that of the explicit calculation, stress as a function of distance along the bar looks the same as Fig. 1b. Table 1 provides data comparing the cost of this simulation for various cases. The table includes the time step size, the number of steps, the number of Newton iterations, the number of linear CG iterations, the number of operations required to reach the final time, and a comparison of the cost. To determine the cost ratio, the operation count for each case is divided by the operation count of the explicit case. For these simulations, the Newton parameter is 10^{-5}, the linear parameter is 10^{-3}, and the directional derivative parameter is 10^{-5}. In the cases marked with a (1), the initial time step is that of the explicit case with the time step increasing up to the given time step, at a rate of 20% per time step; in the cases marked with a (2), the initial time step is 10 times the time step of the explicit case with the time step increasing up to the given value, at a rate of 20% per time step; and the case marked with a (3) has an initial time step 100 times the time step of the explicit case.

Comparing the explicit and implicit solvers using the same time step, the implicit solver is more costly by approximately 70%. Using a time step 10 times as large, the cost is reduced to approximately 34% of the explicit solver, while using a time step 100 times the explicit time step reduces the cost to approximately 10% of the explicit solver. The non-linear solver is slightly more expensive than the linear solver for this problem [18]. The elastic bar in tension is a good example of the possible benefits of using an implicit method. When

Table 1
Cost analysis of bar in tension with Newton–CG solver

	Time step	Steps	Newton iterations	CG iterations	Flops	Cost ratio
Explicit	0.0048	2108			40876272	1.0000
Implicit	0.0050[a]	2041	87	433	70693392	1.7294
	0.0500[a]	214	66	423	14075784	0.3444
	0.0500[b]	205	65	424	13800768	0.3376
	0.5000[a]	42	30	193	4698696	0.1149
	0.5000[b]	30	29	192	4300248	0.1052
	0.5000[c]	22	23	156	3409176	0.0834

[a] The time step is increased initially from the explicit value to this value in 20% increments per step.
[b] The time step is increased initially from 10 times the explicit value to this value in 20% increments.
[c] The time step is increased initially from 100 times the explicit value to this value in 20% increments.

the problem involves small velocities and low frequencies, using an implicit method with larger time steps can result in considerable savings in computational cost over an explicit method.

The second linear elastic simulation involves the collision of two elastic disks with diameter 0.4. The disks start in the lower left corner and upper right corner of the computational domain with initial velocities $(0.001, 0.001)$ and $(-0.001, -0.001)$, respectively. The computational domain is a square of side length 1.0 with 21 nodes per side, resulting in a square mesh with side length $\Delta x = \Delta y = 0.05$. Four material points are used per element. The disks have a Young's modulus of 1000.0, a Poisson's ratio of 0.3 and a density of 1000.0. The simulation is run to a final time of $t = 150.0$, under the assumption of plane strain.

Fig. 2a shows the disks at time $t = 25.0$, when they have already traveled some distance through the grid. It should be noted that there is no error associated with uniform translation of an object through the mesh. Fig. 2b shows the impact of the disks at time $t = 75.0$. Because of the low impact speed, the distortion is small. The impact is handled without resorting to any special contact algorithm and is the result of moving the points in a single-valued velocity field. Fig. 2c and d show the disks rebounding and translating in the opposite direction after impact has occurred. The trajectories of the disks is not effected by the implicit solver and the figure is unchanged for the various simulations of this problem discussed below.

Fig. 3 shows the kinetic, strain and total energy as a function of time. Fig. 3a shows the results for the explicit case, Fig. 3b the results for the implicit case with time step equal to that of the explicit case, Fig. 3c the results for the implicit case with time step 10 times that of the explicit case, and Fig. 3d the results for the implicit case with time step 100 times that of the explicit case. In all four frames, initially, all of the energy is kinetic (dashed line).

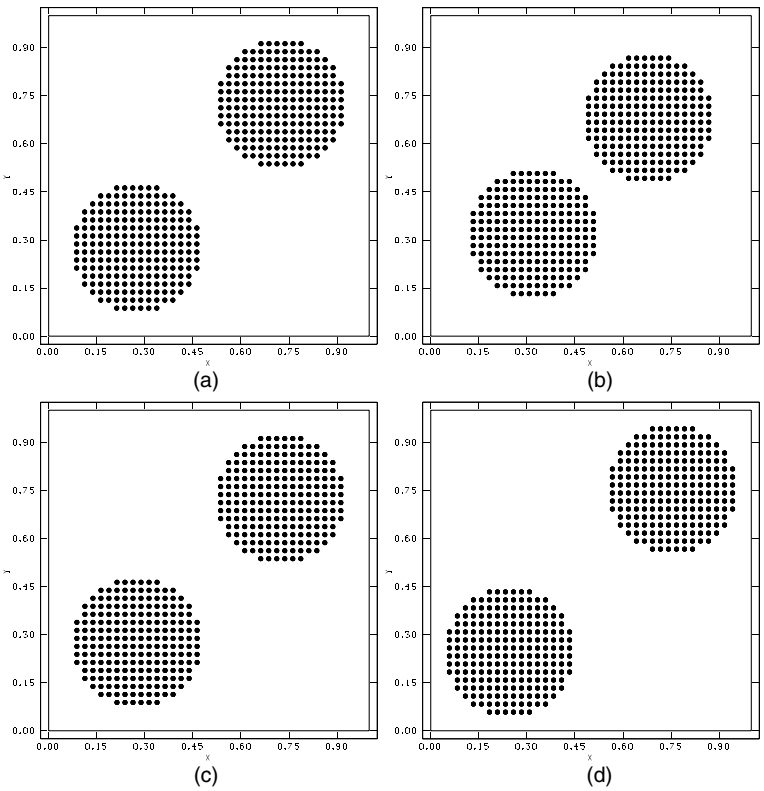


Fig. 2. Collision of elastic discs. Translation before collision is shown in (a), collision is shown in (b), rebounding and translation in the opposite direction is shown in (c) and (d).

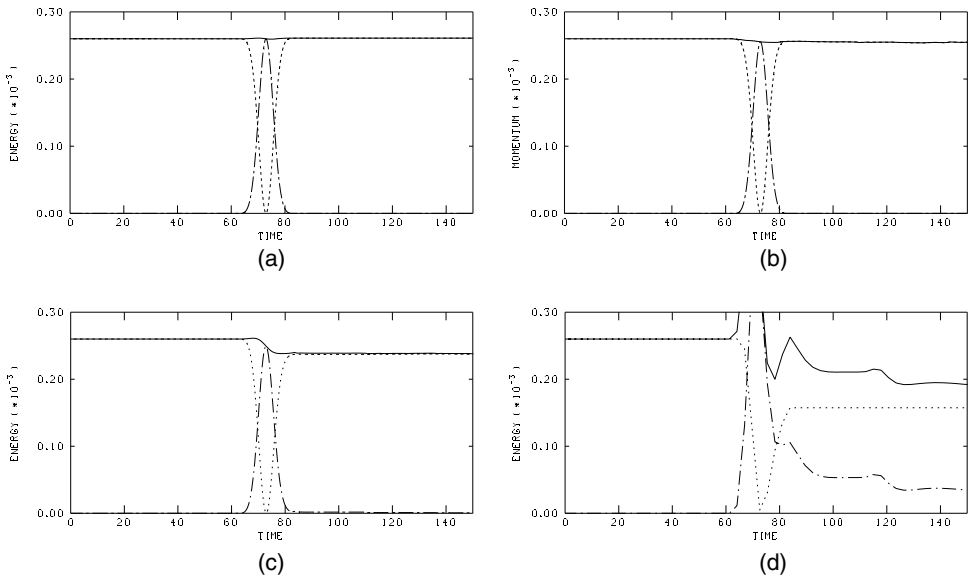


Fig. 3. Energy of elastic disc collision. The explicit results are shown in (a) and the implicit results are shown in (b), (c), and (d) for increasing time step sizes.

In the cases shown in Fig. 3a–c, the kinetic energy decreases during impact and is then mostly recovered after the disks separate. The strain energy (broken line) begins to accumulate upon impact of the disks, reaches its maximum value at the point of maximum deformation during impact, then decreases to a small amount associated with the free vibration of the disks after separation. The total energy (solid line) decreases with time, indicating numerical energy dissipation. It is apparent from the plots that a larger time step results in more energy dissipation during the impact. Only six steps are taken during the time of contact in the case shown in Fig. 3d. It appears from the figure that the frequencies associated with impact are under-resolved by this time step. The contact time is related to the length of time that it takes for a wave to traverse the body. In this example the bulk wave speed is about 0.9 and the diameter of a disk is 0.4, giving a characteristic collision time of about $T_c = 0.4$. Since T_c is about 20 times the explicit time step, this is a bound on the implicit time step imposed by accurate resolution of the collision.

Table 2 provides data comparing the cost of various cases of the colliding-disk simulation. For these simulations, the Newton parameter is 10^{-5}, the linear parameter is 10^{-3}, and the directional derivative parameter is 10^{-5}. By comparison, the implicit solver is more than three times as expensive as the explicit method. The implicit solver does not become competitive until the time step is 100 times larger than that of the explicit case. The non-linear implicit solver appears to be a reasonable choice for this problem, given that enough steps are taken during impact to resolve the resulting frequencies. The non-linear solver is again more computationally expensive than the linear solver [18]. The fact that the non-linear solver is more expensive than the linear solver on linear problems is not surprising. However, problems of interest are typically non-linear in nature, so a general simulation program needs to be able to deal with non-linear problems. The next example is a simple non-linear simulation.

3.1.2. Non-linear CG simulations

For this simulation, the same bar is used as described in the first example above, except that it is given an elastic–plastic constitutive equation. The bar has a Young's modulus of 1000.0, a Poisson's ratio of 0.3 and a density of 10.0. The initial velocity of the rigid body is -0.001 in the x-direction. The simulation is run to a

Table 2
Cost analysis of elastic disks with Newton–CG solver

	Time step	Steps	Newton iterations	CG iterations	Flops	Cost ratio
Explicit	0.02438	6152			1.32E+09	1.0000
Implicit	0.02438	6151	428	12524	4.28E+09	3.2347
	0.2438[a]	625	359	21158	3.81E+09	2.8845
	0.2438[b]	617	359	21110	3.8E+09	2.8763
	2.438[a]	83	58	3934	6.96E+08	0.5265
	2.438[b]	71	58	4005	7.04E+08	0.5324
	2.438[c]	63	62	4486	7.82E+08	0.5917

[a] The time step is increased initially from the explicit value to this value in 20% increments per step.
[b] The time step is increased initially from 10 times the explicit value to this value in 20% increments.
[c] The time step is increased initially from 100 times the explicit value to this value in 20% increments.

maximum time of $t = 24$, at which time the bar should exhibit a total strain of 8%. Von Mises plasticity is used with a linear hardening function with an initial yield stress of 6.0 and a slope of 5000.

The results of a simulation are shown in Fig. 4, where the stress in the middle of the bar is shown as a function of distance along the bar, with the plots corresponding to the times $t = 2, 5, 7, 10, 12, 14, 17, 19, 22, 24$. The stress in the bar is increasing with time and constant along the bar at each time, as expected for a bar under uniaxial stress. The expected final stress is 68.5, which agrees with the stress shown in Fig. 4.

Table 3 contains data analyzing the cost of this problem and is presented in a format similar to previous tables. For these simulations, the Newton parameter is 10^{-6}, the linear parameter is 10^{-4}, and the directional derivative parameter is 10^{-5}. Comparing the explicit and implicit solvers using the same time step size, the implicit solver is more costly by 86%. By using a time step 10 times as large the cost has been reduced to approximately 45% of the explicit solver, while using a time step 100 times the explicit time step reduces the cost to approximately 18% of the explicit solver.

All of the cases solved with the implicit method, except the last one, produce the same results as the explicit method. The exception occurs when a time step 100 times that of the explicit method is used for the

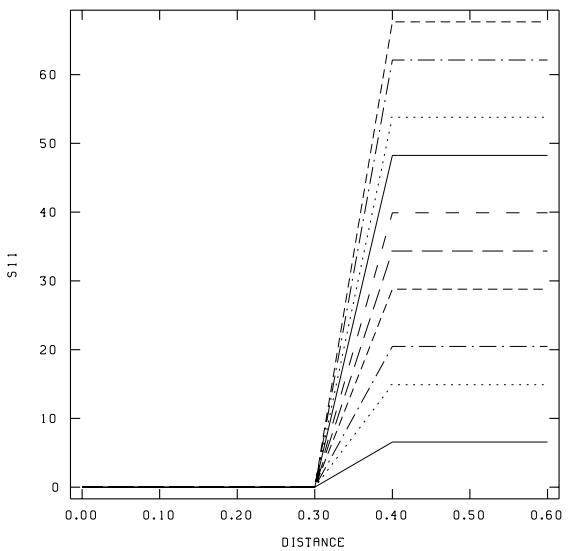


Fig. 4. The elastic–plastic bar under tension. Stress in the middle of the bar as a function of distance along the bar at various times.

Table 3
Cost analysis of the plastic bar under tension

	Time step	Steps	Newton iterations	CG iterations	Flops	Cost ratio
Explicit	0.0047[a]	5024			1.23E+08	1.0000
Implicit	0.0050[a]	4802	221	1381	2.29E+08	1.8600
	0.0500[a]	490	127	1450	5.22E+07	0.4241
	0.05000[b]	481	136	1563	5.42E+07	0.4405
	0.5000[a]	70	80	915	2.27E+07	0.1843
	0.5000[b]	58	71	841	2.10E+07	0.1704
	0.5000[c]	49	72	922	2.27E+07	0.1844

[a] The time step is increased initially from the explicit value to this value in 20% increments per step.
[b] The time step is increased initially from 10 times the explicit value to this value in 20% increments.
[c] The time step is increased initially from 100 times the explicit value to this value in 20% increments.

entire calculation. Some fluctuation in the stress along the bar is exhibited at early steps. The initial load is discontinuous and depends on the time step. Increasing from a smaller time step to a larger one makes the discontinuity smaller and more easily handled by the implicit method. Since essentially the same results are obtained with the explicit and implicit formulations, the implicit solver with a larger time step is the more efficient choice.

There are other variations of this problem that can be modeled, and these variations have been checked to be sure that they are attainable with the code. For example, the simulation above stops before any material points have moved from their original elements. Running the simulation to a final time of $t = 100$ results in 33% strain and two columns of the material points moving into new elements. The Newton–CG method successfully completed this simulation in all of the cases described above. The results for the explicit and implicit cases are the same. MPM can continue this simulation to the point where the columns of material points are far enough apart for an empty element to develop in the middle of the bar. At that point, the material points are no longer able to communicate with each other due to the compact support of the basis functions. Either more material points must be used initially, or some remeshing of the points [19] is required for such extreme deformations.

A second variation of the problem is uniaxial compression of the bar. The Newton–CG method easily compresses the bar into one column of elements, using both explicit and implicit solvers. The material points compress uniformly throughout the bar. A third variation is tension in the bar under the assumption of plane stress. With this assumption, the simulation is successful up to a strain of 75%. Due to the size of the bar, necking becomes the predominant feature at this point. A fourth variation is compression of the bar under the assumption of plane stress. Newton–CG also succeeds in this simulation until the bar is in one column of elements. For all of these cases, the initial speed in the x-direction of the rigid body is 0.001, in the appropriate direction for tension or compression.

3.2. The non-symmetric non-linear solver

The success of the Newton–CG method makes it a likely candidate for a general-purpose solver; however, our attempts to use Newton–CG to model upsetting, machining and rolling are unsuccessful. Upon further investigation, the explicit solver also does not work. The cause of the problem is found to be the formulation used, rather than the implicit solver. Many problems can be solved successfully with the velocity formulation; however, more extreme deformations can be problematical, as discussed in [8]. The approach taken in [8] is called the momentum formulation of MPM, and is presented in this section. The formulation produces a discrete system of equations with a non-symmetric matrix, which naturally leads to the application of the Newton–GMRES method. After summarizing the changes to the MPM, the

parameter choices inherent in the Newton–Krylov methods are discussed, along with the dimensionless form of the non-linear functions.

3.2.1. Momentum formulation of MPM

The difficultly with the algorithm described above and used in the previous solvers occurs when a single material point crosses into an element and is close to the element boundary. As discussed in [8], when this event occurs, the value of a nodal basis function identified with the node at the opposite side of the element may be small. The internal force vector, however, involves the gradient of the basis function, which does not approach zero for points near the element boundary. Solving Eq. (2.2.5) or (2.2.13) for the accelerations requires division of the forces, which depends on the gradient of the basis function, by the nodal mass, which depends on the basis function. As a result, accelerations computed at nodes along the outer edge of the material can occasionally be unphysical, and, as a result, material points can separate from the main body. A change in the order of arithmetic operations will eliminate this problem, as shown in [8].

The operations can be reordered to work with momentum instead of velocity as much as possible, thus avoiding divisions by nodal mass. A formulation using the consistent mass matrix produces an algorithm that is mathematically identical to the algorithm in Section 2; however, the numerical properties are improved. The consistent mass matrix formulation is given in [8]. We choose to use the lumped mass formulation to save the cost of inverting the consistent mass matrix, but a small amount of numerical dissipation ($O(\Delta t^2)$) is introduced [8]. The resulting system of equations is no longer symmetric since the unknown in the discrete equations is now the momentum rather than the velocity. To write the equations in momentum form, define the nodal momentum at time step k

$$\boldsymbol{P}_i^k = m_i^k \boldsymbol{v}_i^k \tag{3.2.1}$$

and the momentum for a material point

$$\boldsymbol{P}_p^k = m_p \boldsymbol{v}_p^k. \tag{3.2.2}$$

Eq. (2.2.14) can then be replaced by

$$\boldsymbol{P}_i^k = \sum_{p=1}^{N_p} \boldsymbol{P}_p^k N_i(\boldsymbol{x}_p^k), \tag{3.2.3}$$

and solving the momentum equation, Eq. (2.4.1), provides the momentum update

$$\boldsymbol{P}_i^L = \boldsymbol{P}_i^k + \Delta t(\mu \boldsymbol{f}_i^{L,\mathrm{int}} + (1-\mu)\boldsymbol{f}_i^{k,\mathrm{int}}), \tag{3.2.4}$$

where the internal force is calculated from the material points by Eqs. (2.2.9) and (2.2.10), as before. Note that no division by nodal masses is required. The material points are updated by

$$\boldsymbol{x}_p^L = \boldsymbol{x}_p^k + \Delta t \sum_{i=1}^{N_n} \boldsymbol{P}_i^L N_i(\boldsymbol{x}_p^k)/m_i^k \tag{3.2.5}$$

and

$$\boldsymbol{v}_p^L = \boldsymbol{v}_p^k + \Delta t \sum_{i=1}^{N_n} (\mu \boldsymbol{f}_i^{L,\mathrm{int}} + (1-\mu)\boldsymbol{f}_i^{k,\mathrm{int}}) N_i(\boldsymbol{x}_p^k)/m_i^k. \tag{3.2.6}$$

The multiplication of momenta and internal forces by the nodal basis function prior to division by the mass in Eqs. (3.2.5) and (3.2.6), has the effect of balancing the numerator and denominator in the case of small nodal masses. Balancing these terms makes the numerical implementation of MPM more robust.

Velocity gradients are still needed to compute the strain increment. The Lagrangian grid velocity is obtained from the updated material-point velocity (3.2.6) by solving the same least squares problem for $\boldsymbol{v}_i^L$ as is done at the beginning of the time step for $\boldsymbol{v}_i^k$

$$m_i^k \boldsymbol{v}_i^L = \sum_{p=1}^{N_p} m_p \boldsymbol{v}_p^L N_i(\boldsymbol{x}_p^k). \tag{3.2.7}$$

Using the lumped mass matrix in this formulation has the effect of smoothing the accelerations used to update these nodal velocities [8].

3.2.2. The parameters

The iterative process depends upon several parameters. These include the parameters specifying when to stop the non-linear and linear iterations, and the parameter governing the step size in the directional derivative. Besides governing the individual processes, these parameters interact in a complicated fashion. For example, it is usually inefficient to iterate the linear solution to a small residual since the linear solution is only an approximation to the non-linear solution. If the non-linear function has a large gradient at the current iteration, the linear approximation is effective for only a small distance [16,17].

While looking at various cases involving refinement of the mesh, it becomes apparent that the non-linear function given in Eq. (2.4.12) is dependent on simulation specific dimensions which makes it difficult to choose simulation parameters that are problem independent. The terms in the function have dimensions of mass times velocity, or equivalently, force times time. The scale factor, $Y\Delta x\Delta y\Delta t$, where Y is the largest of the Young's moduli, is used to make the non-linear function dimensionless. The Newton–GMRES solver uses this dimensionless form of the function.

As discussed in Section 2.4, if the linear system is iterated to the limit of machine precision, the Newton convergence rate is quadratic. This property is tested to begin the search for appropriate parameters. The elastic–plastic bar is used for this test, running to a final time of $t = 100$. The linear parameter is set to $\gamma = 10^{-12}$ and the non-linear parameter at $\varepsilon = 10^{-4}$. The time step is equal to that of the explicit case. The directional derivative parameter is varied by factors of 10 between $h = 10^{-2}$ and $h = 10^{-7}$. Comparing the results for these six cases, five steps were found to require two or more non-linear iterations to converge in all of the cases. Fig. 5 shows the non-linear residuals for these five steps as a function of iteration number. The dotted line is a line of quadratic convergence, so it is expected that the residuals will converge at least this fast at the end of the non-linear iteration. The result for $h = 10^{-7}$ is shown in Fig. 5a, for $h = 10^{-6}$ in Fig. 5b, for $h = 10^{-5}$ in Fig. 5c, for $h = 10^{-4}$ in Fig. 5d, for $h = 10^{-3}$ in Fig. 5e and for $h = 10^{-2}$ in Fig. 5f. It can be seen that the last four cases converge quadratically, while the case in Fig. 5b is converging quadratically for all except one step that is almost quadratic. The case in Fig. 5a has several steps that do not attain the quadratic convergence. The resulting stress/strain curves were the same for all cases except $h = 10^{-2}$. From Fig. 5, h is narrowed to the range 10^{-5} to 10^{-3}.

Figs. 6–8 show the effect of changing the value of the linear parameter γ has on the operation count. The plots in Fig. 6 correspond to a time step size equal to the explicit time step size, in Fig. 7 to a time step size 10 times as large as the explicit one, and in Fig. 8 to a time step size 100 times as large. The first plot marked (a) in each figure corresponds to $h = 10^{-3}$, the second plot marked (b) corresponds to $h = 10^{-4}$, and in (c) $h = 10^{-5}$. The number of operations is plotted as a function of the non-linear parameter ε. Each curve corresponds to a different γ, as labeled. Fig. 6a is missing several data points, as these cases did not run to completion. Apparently $h = 10^{-3}$ is too large, at least in some cases. Comparing values of γ, the trend is similar in all of the plots. A smaller γ results in a higher cost. Looking at the effect of ε, in all cases the cost of attaining $\varepsilon = 10^{-4}$ is basically the same as that of $\varepsilon = 10^{-1}$. Using the larger time steps, ε can be pushed to 10^{-6} at less than twice the cost of 10^{-4}. This difference is slightly more with the smallest time step, but that time step would not be used in practice since it is more costly than the explicit method.

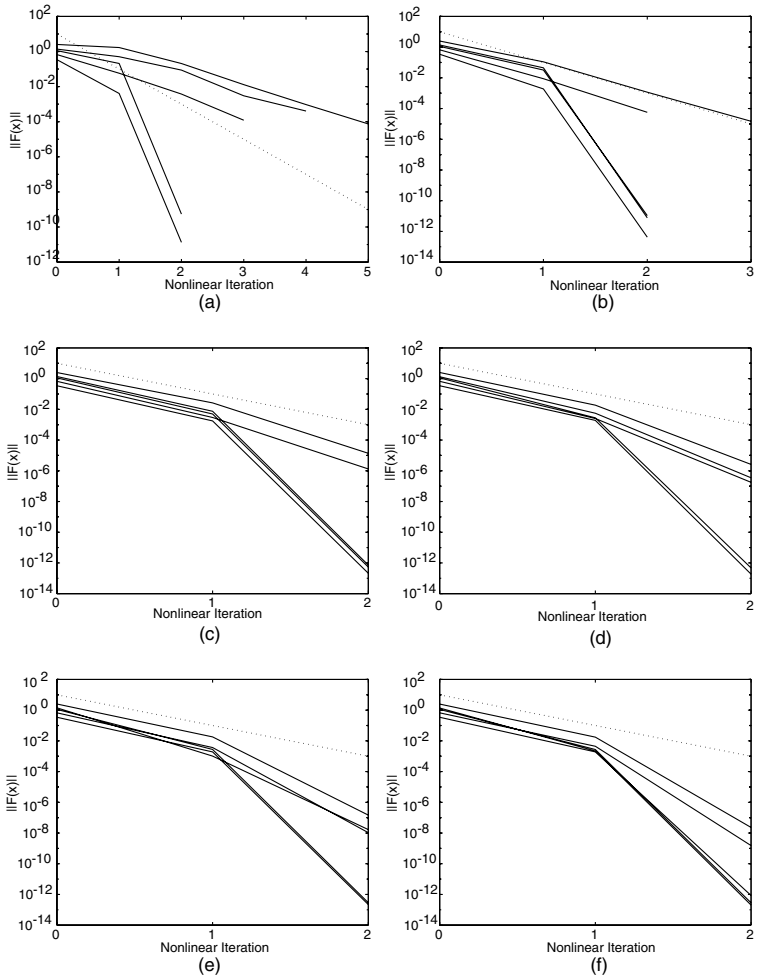


Fig. 5. Non-linear residuals for selected steps in the non-linear iteration as a function of iteration number.

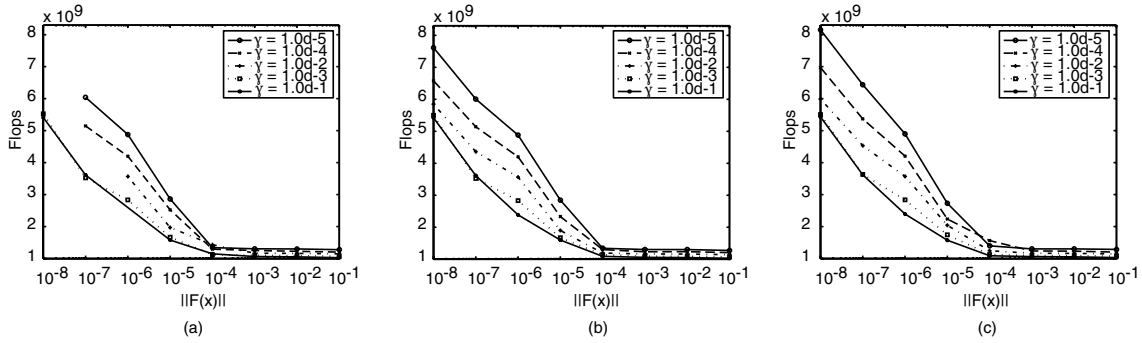


Fig. 6. Effect of the linear parameter γ on the operations count for: (a) $h = 10^{-3}$, (b) $h = 10^{-4}$ and (c) $h = 10^{-5}$ and a time step equal to the explicit case.

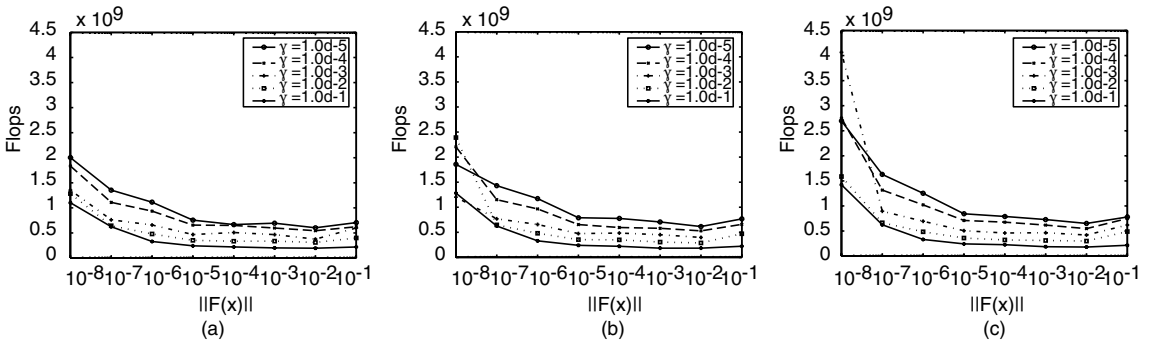


Fig. 7. Effect of the linear parameter γ on the operations count for: (a) $h = 10^{-3}$, (b) $h = 10^{-4}$ and (c) $h = 10^{-5}$ and a time step 10 times that of the explicit case.

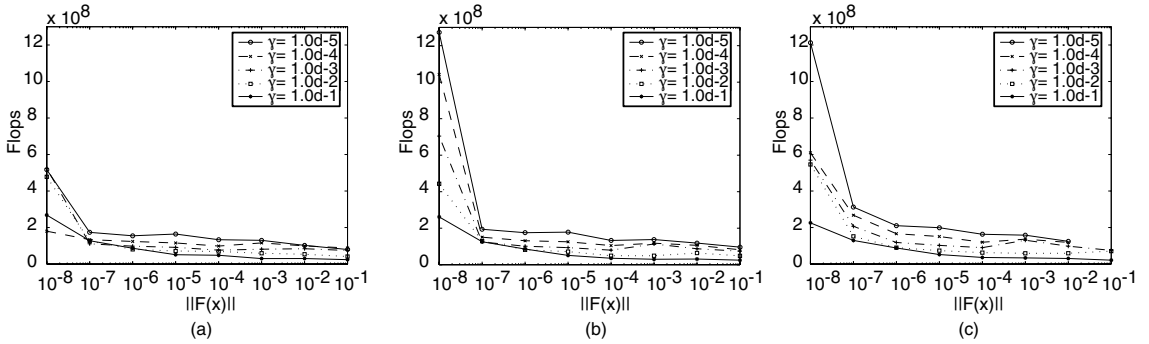


Fig. 8. Effect of the linear parameter γ on the operations count for: (a) $h = 10^{-3}$, (b) $h = 10^{-4}$ and (c) $h = 10^{-5}$ and a time step 100 times that of the explicit case.

Interpreting the results of these calculations, the value of h must be small enough to give a reasonable approximation to the derivative, while being large enough to maintain the quadratic convergence. Thus, reasonable values appear to be between 10^{-5} and 10^{-4}. Since ε controls the number of digits of accuracy in the solution, it is usually set somewhere between 10^{-7} and 10^{-4}, depending on the accuracy needed. Using a larger γ appears to help reduce the cost, so it is set somewhere between 10^{-4} and 10^{-2}.

3.2.3. Line search method

It is possible to begin with an initial iterate in Newton's method that is too far from the root for the local convergence theory to apply, even though the value being calculated by the linear iteration is in the right direction. However, a large step, even in this direction, results in the size of the non-linear residual growing for that iteration, rather than decreasing. A simple means of correcting this problem is to reduce the size of the step until the size of the non-linear residual is decreased. This is referred to as a line search method, since the search is made along the line segment connecting the current point and the newly calculated point [13]. This is also sometimes called backtracking.

The need for a line search can arise during an MPM calculation when a large number of material points are close to some element boundary at the same time. Information about a time step in the elastic–plastic bar simulation where such a problem occurs is shown in Fig. 9. Fig. 9a shows the number of Newton iterations used at each time step. Only two of the time steps require more than 10 Newton iterations to

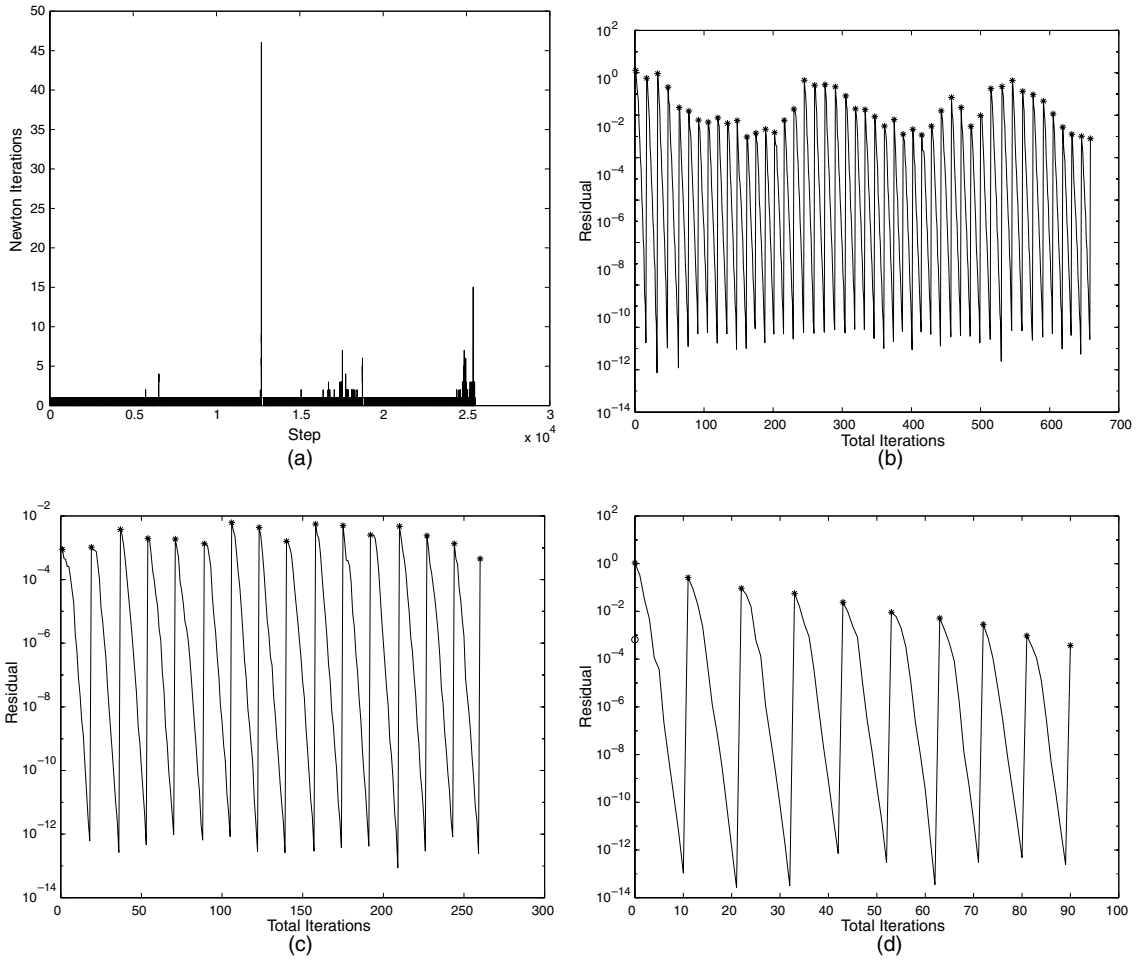


Fig. 9. Effect of line search on non-linear residuals. The number of non-linear iterations as a function of time step is shown in (a), non-linear and linear residuals for steps 12694 and 25402 without a line search are shown in (b) and (c), respectively, and with a line search in (d).

converge. Step 12694, which requires 46 Newton iterations, corresponds to the time when the first row of the bar crosses into a new element. Step 25402 requires 15 Newton iterations and corresponds to the time when the last row of the rigid body moves into a new element. Plots of the non-linear and linear residuals are shown in Fig. 9b and c for these two steps, respectively. The asterisk designates a non-linear residual, followed by several decreasing linear residuals, then a new non-linear residual. The iteration count is the total of the non-linear and linear iterations. The non-linear residuals do not decrease for some iterations. Fig. 9d shows the results for the same two steps after introduction of the line search. Step 12694 is plotted with the line and asterisks, while step 25402 is shown with the circle. Step 12694 now requires only 10 non-linear iterations, while step 25402 requires no non-linear iterations.

If there are a large number of material points and elements in a simulation, and if the deformations are large, material points are forced to cross several element boundaries during a simulation. The forces are discontinuous at these instances. The line search seems to eliminate convergence problems in these cases. The next section presents simulations using this technique and demonstrating its efficacy.

3.2.4. Non-linear GMRES simulations

For comparison, the first two simulations discussed in this section are the elastic bar under tension and the elastic–plastic bar described previously. The GMRES basis is restricted to the first 20 search directions.

3.2.5. The elastic bar

The Newton–GMRES method produces the same results presented in Fig. 1, so the figures are not repeated. In addition, compared to the linear and the Newton–CG methods, the fluctuations in the early stresses along the bar seen with the large time step, do not appear. For this set of simulations, the Newton parameter is 10^{-5}, the linear parameter is 10^{-3}, and the directional derivative parameter is 10^{-5}. No line searches are required in these simulations.

Table 4 provides data comparing the cost for various cases of this simulation. The data in the table are the time step size, the number of steps, the number of Newton iterations, the number of linear GMRES iterations, the number of operations required to reach the final time, and a comparison of the cost. To determine the cost ratio, the operation count for each case is divided by the operation count of the explicit case. Comparing the explicit and implicit solvers using the same time step, the implicit solver costs approximately twice as much. Using a time step 10 times as large the cost is reduced to approximately 35% of the explicit solver, while using a time step 100 times as the explicit time step reduces the cost to less than 10% of the explicit solver. The results for the middle time step compare well with those from the Newton–CG solver. The largest time step is 15–25% less expensive with the Newton–GMRES solver, while the smallest time step is about 15% more expensive.

3.2.6. The elastic–plastic bar

For this set of simulations, the Newton parameter is 10^{-6}, the linear parameter is 10^{-4}, and the directional derivative parameter is 10^{-5}. The Newton–GMRES method produces the same results as in Fig. 4, except that the fluctuation in the early stresses along the bar, seen using the large time step in the Newton–CG method, does not appear. No line searches are needed for this problem.

Table 5 contains data analyzing the cost of this problem. Comparing the explicit and implicit solvers using the same time step, the implicit solver is almost twice as costly. Using a time step 10 times as large the cost is reduced to approximately 56% of the explicit solver, while using a time step 100 times the explicit time step reduces the cost to approximately 9% of the explicit solver. The smallest time step is about 10% more expensive, while the middle time step is 35–40% more expensive than the Newton–CG solver. The largest time step is 40–48% less expensive with the Newton–GMRES solver. Note that the explicit case is about 5% more expensive with the new momentum formulation. Newton–GMRES also succeeds in completing all of the variations described in Section 3.1.

Table 4
Cost analysis of elastic bar with Newton–GMRES solver

	Time step	Steps	Newton iterations	GMRES iterations	Flops	Cost ratio
Explicit	0.0048	2108			40876272	1.0000
Implicit	0.0050[a]	2041	126	243	80917776	1.9796
	0.0500[a]	214	89	177	14819064	0.3625
	0.0500[b]	205	83	166	14038752	0.3434
	0.5000[a]	42	30	60	3946824	0.0966
	0.5000[b]	30	26	52	3195624	0.0782
	0.5000[c]	22	21	42	2499048	0.0611

[a] The time step is increased initially from the explicit value to this value in 20% increments per step.
[b] The time step is increased initially from 10 times the explicit value to this value in 20% increments.
[c] The time step is increased initially from 100 times the explicit value to this value in 20% increments.

Table 5
Cost analysis of elastic–plastic bar with Newton–GMRES solver

	Time step	Steps	Newton iterations	GMRES iterations	Flops	Cost ratio
Explicit	0.0047	5024			1.29E+08	1.0000
Implicit	0.0050[a]	4802	278	688	2.52E+08	1.9495
	0.0500[a]	490	211	1001	7.28E+07	0.5636
	0.0500[b]	481	207	1014	7.32E+07	0.5668
	0.5000[a]	70	65	214	1.37E+07	0.1059
	0.5000[b]	58	57	197	1.25E+07	0.0971
	0.5000[c]	49	48	186	1.17E+07	0.0909

[a] The time step is increased initially from the explicit value to this value in 20% increments per step.
[b] The time step is increased initially from 10 times the explicit value to this value in 20% increments.
[c] The time step is increased initially from 100 times the explicit value to this value in 20% increments.

3.2.7. Upsetting

The upsetting simulation models the compression of a cylindrical billet between two platens. This process is used in manufacturing as a means to pre-form workpieces prior to applying another operation such as rolling or extrusion. Since a detailed comparison of the explicit MPM results to results obtained by researchers using other methods is given in [10], only a comparison of the explicit and implicit MPM results is provided here.

The problem consists of a steel cylinder 20 mm in diameter and 30 mm in height. Fig. 10a shows the representation of this cylinder using material points. Only half the cylinder is modeled due to the symmetry about the midline, $z = 0$. The platen is modeled with material points that move as a rigid body. The platen starts above the workpiece and moves down at a constant velocity of 1 m/s, compressing the billet. There is no slip allowed between the platen and the cylindrical workpiece. The steel has a Young's modulus of 200 GPa, Poisson's ratio of 0.3, density of 7800 kg/m^3, and an initial yield strength of 0.70 GPa with a strain hardening slope of 0.30 GPa. A 24×18 mesh with a spacing of 1 mm in both directions is used. The Newton parameter is 10^{-4}, the linear parameter is 10^{-2}, and the directional derivative parameter is 10^{-5} for this set of simulations.

The upsetting problem requires the use of the line search in the implicit solver. A simulation using a time step 10 times the explicit time step requires 58 line searches. Only one backstep is taken on each iteration that is effected. Of the 9503 time steps, a backstep is required for at least one iteration in 48 of the steps. When using a time step 100 times the explicit time step, the line search is invoked a total of 206 times. Of the 953 time steps taken, 33 steps and a total of 126 iterations are impacted. When using a time step 1000 times the explicit time step, a total of 59 line searches are employed. Of the 98 time steps taken, 14 steps and 33 total iterations are treated. For the time step 10,000 times the explicit time step, the line search is used a total of 16 times. Of the 16 time steps taken, 1 time step is treated on 8 of its iterations.

Table 6 contains the computational cost data for the upsetting problem. Because the platen starts away from the body, increasing the time step has no effect except when an extremely large time step is used. Thus, for the data shown in Table 6 all steps are the time step size given in the table, except for the last case. When $\Delta t = 0.001$ s. the Lagrangian time step condition becomes the limiting factor. The Lagrangian condition restricts the time step to one that prevents the mesh from folding; that is, the determinant of the deformation gradient must remain positive. Attempting simulations with even larger time steps does not reduce the number of steps further since this Lagrangian time step restriction remains in effect. The implicit method using the explicit time step costs three times as much as the explicit case. Using an implicit time step 10 times larger requires about 17% more computations. Using a time step 100 times as large reduces the cost to less than 30% of the explicit cost, while a time step 1000 times as large reduces the cost to less than 6% of the explicit cost. A time step approximately 10,000 times as large reduces the cost to a mere 1.2% of the explicit cost.

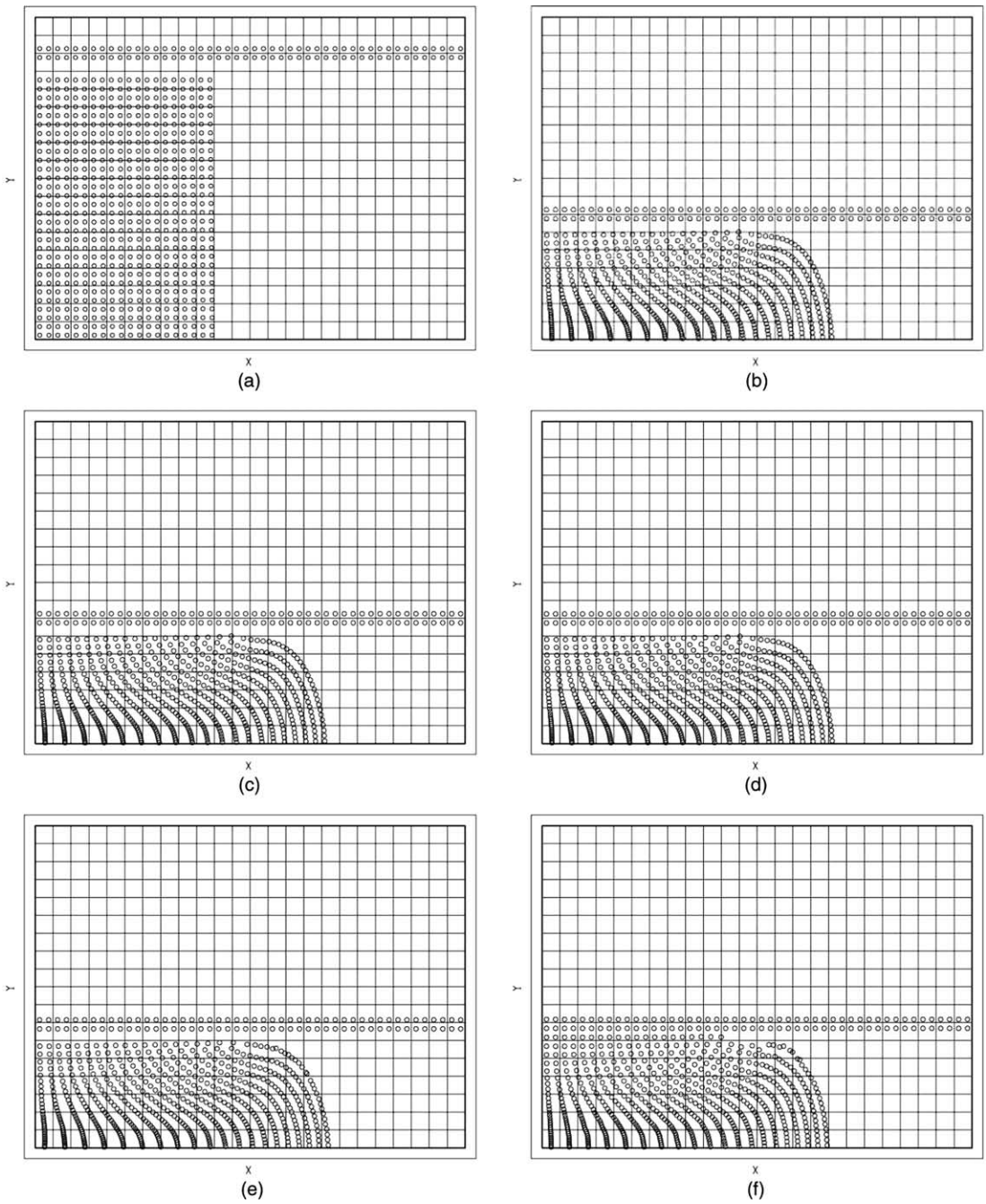


Fig. 10. The upsetting problem. The initial configuration is shown in (a) and results after 60% upsetting in (b)–(f) for implicit cases with increasing time step size.

The final configuration of material points is shown after 60% upsetting (reduction to 40% of original height) in Fig. 10b–f for the various cases. These plots correspond to the time steps given in Table 6, i.e. Fig.

Table 6
Cost analysis of upsetting

	Time step	Steps	Newton iterations	GMRES iterations	Flops	Cost ratio
Explicit	9.50E−08	99388			4.23E+10	1.0000
Implicit	1.00E−07	95003	51629	128738	1.28E+11	3.0120
	1.00E−06	9503	9952	102068	4.96E+10	1.1707
	1.00E−05	953	1659	26982	1.27E+10	0.2998
	1.00E−04	98	298	5344	2.49E+09	0.0587
	1.00E−03[a]	16	58	1040	4.88E+08	0.0115

[a] The time step is sometimes limited by the Lagrangian time step restriction.

10b is the explicit case, etc. The explicit solver predicts a final radius of approximately 16.5 mm in Fig. 10b. Fig. 10c and d agree with this result. The result in Fig. 10e is at a slightly later time, so the slightly larger radius is to be expected. However, the material points at the edge of the material are not as regularly spaced. In Fig. 10f, the material at the edge is starting to ruffle, indicating that the accuracy of the result is beginning to degrade with this large step size.

The fidelity of the solution can also be judged by examining quantities such as equivalent stress, plastic strain, or temperature rise. Contour plots in all cases are similar to the explicit solution [10]. For example, Fig. 11a shows a contour plot of the equivalent plastic strain when the step size is 1000 times the explicit step size. The maximum equivalent plastic strain occurs in the upper right side of the workpiece where there appears to be a singularity forming as the material folds against the platen when it is compressed. There is no indication in these plots that the accuracy might be deteriorating for the largest time step. Fig. 12a shows load–deflection curves for each case. These curves essentially coincide for step sizes from the explicit up to 1000 times the explicit case and predict a final load of about 1050 kN. The curve representing the predicted load when the time step is 10,000 times the explicit time step is distinct, indicating that accuracy has been lost with this large step size. The larger jumps in the load-deflection curves have been identified with material folding against the platen. Smaller amplitude oscillations in these curves are due to the coarseness of the spatial mesh, finite step size, and the simple contact algorithm between the rigid and deformable bodies [10].

The spatial mesh is refined by halving the size in each coordinate direction. Fig. 13 shows configurations of the material points on the original mesh and for two such refinements. The time step is kept fixed at 1000 times the explicit time step for the coarse mesh, so that on the medium and fine meshes we are actually running at 4000 and 16,000 times the explicit time step for that mesh size; respectively. Since the time step is fixed, all cases take 98 steps. The medium mesh requires about 1.5 times the number of Newton iterations as the coarse mesh, and the fine mesh requires 22 times the number of Newton iterations.

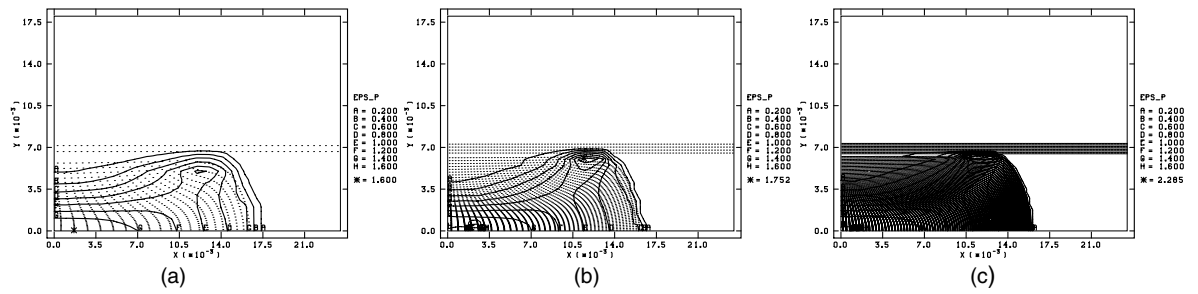


Fig. 11. Contour plots of equivalent plastic strain using a time step of 10^{-4} s, which is 1000 times the explicit time step, with a square spatial mesh size of: (a) 1 mm, (b) 0.5 mm and (c) 0.25 mm on a side.

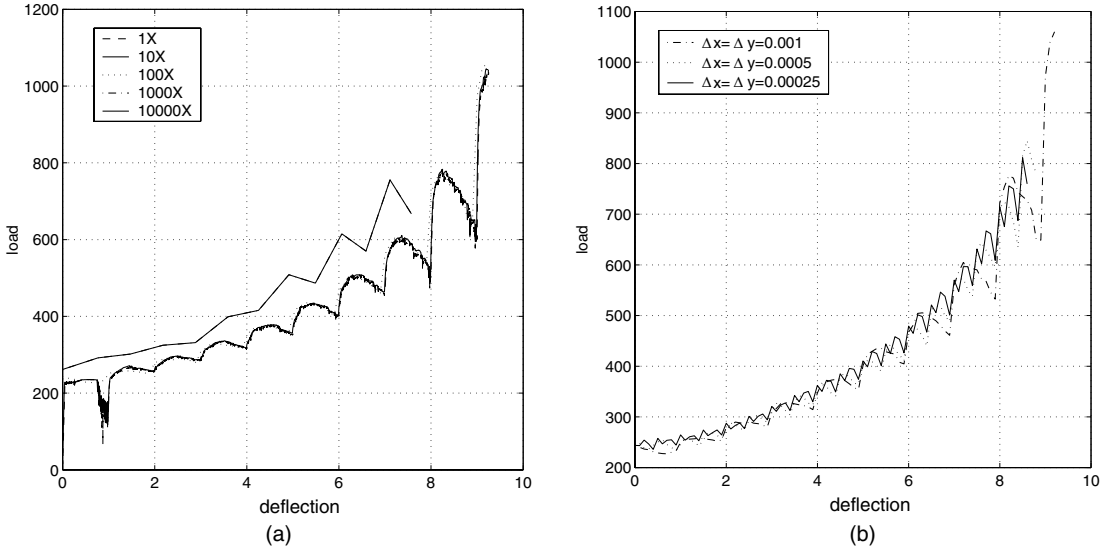


Fig. 12. Load on the rigid platen (kN) as a function of deflection (mm). In (a) the spatial mesh is kept fixed and the time step is varied and in (b) the time step is fixed at 10^{-4} s, which is 1000 times the explicit step size on the coarse mesh, and the spatial mesh is refined.

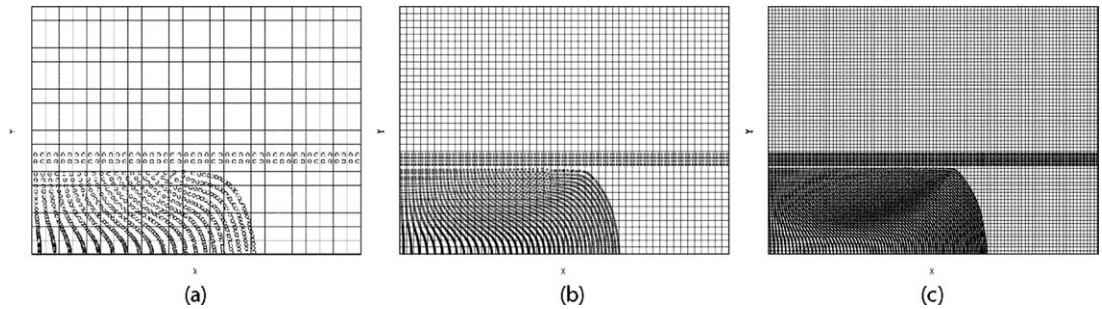


Fig. 13. The upsetting problem. The results after 60% upsetting in (a)–(c) for implicit cases with increasingly refined spatial meshes. The time step is kept fixed at 10^{-4} s, which is 1000 times the explicit time step for the coarse mesh.

Load–deflection curves are given in Fig. 12(b) for the three meshes. The figure indicates convergence with mesh refinement to a predicted final load of 800 kN. Fig. 11 compares contours of equivalent plastic strain on each of these spatial meshes. Except in the singular region where the maximum grows, the contour levels are all quite similar. Both the load-deflection curves and the contours of equivalent plastic strain agree with results obtained with the explicit code and by other researchers [10].

4. Summary

An implicit solver is derived for implicit dynamics in the MPM, based on a matrix-free implementation of Newton–Krylov methods. The velocity formulation of MPM leads to a symmetric system of equations

solved by the Newton–CG method. Two simple elastic problems are presented that demonstrate the abilities of this method. The case with a bar under tension shows the significant improvement in computational cost that can be expected from an implicit solver for a quasi-static problem. The case with the colliding elastic disks demonstrates that the implicit solver is capable of generating reasonable answers for problems where the solution has somewhat higher frequency content, although there is energy dissipation with the larger time steps and not all features of the solution are accurately reproduced. The implementation of the non-linear Newton–CG method is also capable of simulating the elastic–plastic bar under tension, with significant savings in computational cost over the explicit method. However, the Newton–CG method is unable to complete the upsetting simulation due to the MPM formulation used in the solver.

A description of the momentum formulation in the implicit MPM is presented, with an explanation of the improved numerical qualities. A search for the values assigned to the parameters controlling non-linear iterations, linear iterations and directional derivative step size results in a dimensionless non-linear function and a nominal range of values for each of the parameters. Use of a line search method to control global convergence of the non-linear iterations is discussed and demonstrated. The resulting Newton–GMRES method is capable of completing the same simulations as Newton–CG. Furthermore, the small fluctuations seen at large time steps in the previous method are eliminated by Newton–GMRES. This method is used to simulate the metal processing technique of upsetting, with significant savings in computational cost over the explicit case. The success of this simulation demonstrates the promise of the implicit solver for realistic simulations of metal forming processes.

References

[1] F.H. Harlow, The particle-in-cell computing method for fluid dynamics, Methods Comput. Phys. 3 (1964) 319–343.
[2] J.U. Brackbill, H.M. Ruppel, FLIP: a method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions, J. Comput. Phys. 65 (1986) 314–343.
[3] J.U. Brackbill, D.B. Kothe, H.M. Ruppel, FLIP: A low-dissipation, particle-in-cell method for fluid flow, Comput. Phys. Commun. 48 (1998) 25–38.
[4] D. Sulsky, Z. Chen, H.L. Schreyer, A particle method for history-dependent materials, Comput. Meths. Appl. Mech. Engrg. 118 (1994) 179–196.
[5] A.R. York, D. Sulsky, H.L. Schreyer, Fluid-membrane interaction based on the material-point method, Int. J. Numer. Methods Engrg. 48 (2000) 901–924.
[6] S.J. Cummins, J.U. Brackbill, An implicit particle-in-cell method for granular materials, J. Comput. Phys. 180 (2002) 506–548.
[7] J.E. Guilkey, J.A. Weiss, An implicit time integration strategy for use with the material point method, in: Proceedings from the First MIT Conference on Computational Fluid and Solid Mechanics, June 12–15, 2001.
[8] D. Sulsky, S. Zhou, H.L. Schreyer, Applications of a particle-in-cell method to solid mechanics, Comput. Phys. Commun. 87 (1995) 236–252.
[9] C.C. Lin, L.A. Segel, Mathematics Applied to Deterministic Problems in the Natural Sciences, SIAM, Philadelphia, 1988.
[10] D. Sulsky, H.L. Schreyer, Axisymmetric form of the material point method with applications to upsetting and Taylor impact problems, Comput. Meths. Appl. Mech. Engrg. 139 (1996) 409–429.
[11] H.L. Schreyer, D. Sulsky, S. Zhou, Modeling delamination with as a strong discontinuity with the material point method, Comput. Methods Appl. Mech. Engrg. 191 (2002) 2463–2481.
[12] D. Burgess, D. Sulsky, J.U. Brackbill, Mass matrix formulation of the FLIP particle-in-cell method, J. Comput. Phys. 103 (1992) 1–15.
[13] C.T. Kelly, Iterative Methods for Linear and Nonlinear Equations, SIAM, Philadelphia, 1995.
[14] R. Barret, M. Berry, T.F. Chan, J. Demmel, R. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, H. van der Vorst, Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, SIAM, Philadelphia, 1994.
[15] Y. Saad, M.H. Schultz, GMRES: A generalized minimum residual algorithm for solving nonsymmetric linear systems, SIAM J. Sci. Stat. Comput. 7 (1986) 856–869.
[16] D.A. Knoll, P.R. McHugh, Enhanced nonlinear iterative techniques applied to a nonequilibrium plasma flow, SIAM J. Sci. Comput. 19 (1998) 291–301.
[17] D.A. Knoll, G. Lapenta, J.U. Brackbill, A multilevel iterative field solver for implicit, kinetic, plasma simulation, J. Comput. Phys. 149 (1999) 377–388.

[18] A. Kaul, Implicit formulation of the material point method with application to metals processing, Ph.D. Dissertation, Department of Mathematics and Statistics, University of New Mexico, December, 2000.
[19] G. Lapenta, J. Brackbill, Control of the number of particles in fluid and MHD particle-in-cell methods, Comput. Phys. Commun. 87 (1995) 139–254.