Engineering with Computers manuscript No.
(will be inserted by the editor)

A Virtual Test Facility for the Efficient Simulation of Solid Material Response under Strong Shock and Detonation Wave Loading

Ralf Deiterding[1], Raul Radovitzky[2], Sean P. Mauch[1], Ludovic Noels[2⋆], Julian C. Cummings[1], Daniel I. Meiron[1]

[1] California Institute of Technology, 1200 East California Blvd., Pasadena, CA 91125
[2] Massachusetts Institute of Technology, 77 Massachusetts Ave., Cambridge, MA 02139

Received: date / Revised version: date

Abstract A Virtual Test Facility (VTF) for studying the three-dimensional dynamic response of solid materials subject to strong shock and detonation waves has been constructed as part of the Center for Simulating the Dynamic Response of Materials at the California Institute of Technology. The compressible fluid flow is simulated with a Cartesian finite volume method treating the solid as an embedded moving body, while a Lagrangian finite element scheme is employed to describe the material response to the hydrodynamic pressure loading on the solid. A temporal splitting method is applied to update the position and velocity of the boundary between time steps. The incorporation of Cartesian finite volume schemes into a structured dynamic mesh adaptation algorithm with hierarchical time step refinement allows for an efficient treatment of the issues arising from the possibly disparate fluid and solid time scales. The boundary is represented implicitly in the fluid solver with a level set function. An efficient algorithm used to transform a given triangulated surface provided by the Lagrangian finite element solver into a point set of signed distance values is described. The modification of this algorithm for patch-based mesh refinement methods is discussed. All algorithmic components have been parallelized for distributed memory machines; partitioning strategies (which are critical to efficient parallel performance) are sketched briefly. The dynamic deformation of a Tantalum cylinder due to the detonation of a high-explosive (HMX) interior to the cylinder, and the impact of an explosion-induced shock wave on a multi-material soft tissue body are presented as examples of our approach.

Send offprint requests to: R. Deiterding, ralf@cacr.caltech.edu

⋆ FNRS fellow (University of Liège, Belgium)

1 Introduction

The Virtual Test Facility (VTF) is a software environment for coupling solvers for compressible computational fluid dynamics (CFD) with solvers for computational solid dynamics (CSD). The CFD solvers facilitate the computation of flows with strong shocks as well as fluid mixing. The CSD solvers provide capabilities for simulation of dynamic response in solids such as large plastic deformations, fracture and fragmentation. In addition, the VTF can be used to simulate highly coupled fluid-structure interaction problems, such as the high rate deformation experienced by a metallic solid target forced by the loading originating from the detonation of energetic materials, or the rupture and fragmentation of brittle materials under shock wave impact. At present, all VTF solvers use time-explicit numerical methods that track the various wave phenomena responsible for mediating the dynamic response through the application of suitable numerical methods.

In order to implement the solid-fluid coupling in the VTF, we apply a loosely coupled, partitioned approach, with modular software components for each solver. The fluid-structure coupling technique operates as follows: one assumes disjoint fluid and solid domains and that the interaction takes place only at the fluid-solid interface. In this way, one can apply algorithms that are intrinsically suited for simulation of phenomena such as shock propagation, detonation or fluid mixing in the fluid solver, while applying algorithms similarly optimized for phenomena such as high-rate plastic deformation, fracture, etc. in the solid solver. For example, a Lagrangian representation is most suitable to account numerically for large solid deformations, contact and fracture, while the governing equations of compressible fluid motion are most effectively solved in an Eulerian frame of reference [1]. In the loose fluid-structure coupling adopted, the information exchange between the fluid and the solid solver is reduced to communicating the velocities and the geometry of the solid surface to the

Eulerian fluid, and communicating the hydrostatic pressure back to the Lagrangian solid as a force acting on its exterior [1,2,3,4,5,6,7]. This approach offers several advantages. Firstly, it allows for solver reuse (see [8] or [2] for details on the idea of modularization). Secondly, it becomes straightforward to take advantage of recent advances in multiscale constitutive modeling to describe the dynamic response of both the solid and fluid. Such modeling typically also employs a Lagrangian description for solids and an Eulerian description for the fluid.

A key issue that arises with the proposed approach is how to represent the evolving surface geometry on the Eulerian fluid mesh. The application of body-conforming meshes is extraordinarily cumbersome, because the fluid equations first need to be cast into a local arbitrary Lagrangian-Eulerian (ALE) frame of reference [9]. At each step, the mesh topology would have to be reconstructed and the solution re-interpolated. While this is possible (and successfully implemented in many present day codes), the issues of mesh tangling and the requirements of frequent re-meshing in the case of large deformations remain a challenge. The need to re-mesh is also an inherent bottleneck in massively parallel simulations [6].

An alternative to the use of body-aligned fluid grids is the use of Cartesian meshes with immersed or embedded irregular boundaries. Here, there are two basic approaches: "cut-cell" techniques that construct smaller cells by intersecting the Cartesian mesh exactly with the (triangulated) boundary and techniques that "diffuse" the boundary within one cell [10]. Cut-cell methods have the advantage that they can represent accurately the boundary flux and thus facilitate the implementation of discretely conservative fluid solvers. However, the proposed numerical circumventions of the severe time step restriction in time-explicit schemes [11,12], which can result from very small cells created by the boundary intersection, are logically quite complicated. Most approaches have not yet been extended successfully to three spatial dimensions even for pure fluid flow problems. In the VTF, we therefore employ a diffused boundary technique in which some interior cells are used directly to enforce the embedded boundary conditions in the vicinity of the solid surface [13,4]. This has been called the "ghost fluid" approach. One advantage of this approach is that the numerical stencil is not modified, thus ensuring optimal parallel scalability. We minimize conservation errors as well as possible numerical "staircase" artifacts resulting from the use of numerics to "capture" the boundary by using block-structured dynamic mesh adaptation to refine the Cartesian mesh along the boundary. As the solid deforms, the solid-fluid boundary is represented implicitly with a scalar level set variable that is updated on-the-fly using an efficient algorithm described in more detail below. An important additional advantage of this approach is the ability to cope with topological transitions such as fracture or penetration of the solid-fluid boundary.

The present paper details the implementation of the VTF approach and also describes its application to fluid-solid interaction problems wherein detonation and shock waves impinge on thick three-dimensional solid materials. An extension of the fluid-solid coupling algorithm adopted in this paper to thin, open structures has been presented in [15]. In Sec. 2, a Cartesian dynamically adaptive finite volume fluid solver for Euler equations with one-step chemistry is described. Section 4 describes a highly efficient algorithm to transform a triangulated surface mesh into a signed distance function on a hierarchical Cartesian mesh. Section 3 outlines the Lagrangian finite element solver for solid materials subjected to high-intensity shock loadings, and Sec. 6 provides two three-dimensional computational examples. In Sec. 6.1, we simulate the impact event of a strong hydrodynamic shock wave on a body comprised of soft-tissue; in Sec. 6.2 the propagation of a detonation wave in HMX through a plastic Tantalum cylinder is simulated. Both computations have been run on distributed memory machines. We also briefly comment on the overall computational efficiency of our approach.

2 Eulerian Fluid Dynamics

In this section, we are concerned with the construction of an Eulerian fluid solver framework suitable for efficient fluid-structure coupling. Although the presentation is tailored to the Euler equations with simple one-step reaction, the concepts are equally applicable to general conservation laws with arbitrary source terms. Within the Center for Simulating the Dynamic Response of Materials at the California Institute of Technology, the same framework is also used successfully with solvers for the compressible Favre-averaged Navier-Stokes equations with large-eddy turbulence model [16,17] and for detonation simulation in thermally perfect gas mixtures with detailed chemical kinetics [18,19,20,21,22,23,24].

2.1 Governing Equations

In order to model detonation waves in solid energetic materials we utilize the single-phase model proposed by Fickett and Davis [25], which has also been used by Clarke et al. [26] to evaluate numerical methods for detonation simulation. We assume a single chemical reaction $A \longrightarrow B$ that is modeled by a progress variable λ, which corresponds to the mass fraction ratio between the density of the product B and the total density ρ, i.e.

$\lambda = \rho_B/\rho$. The governing equations of the model read

$$\partial_t \rho + \nabla \cdot (\rho \mathbf{u}) = 0 \,, \quad (1)$$
$$\partial_t (\rho \mathbf{u}) + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u}) + \nabla p = 0 \,, \quad (2)$$
$$\partial_t (\rho E) + \nabla \cdot ((\rho E + p)\mathbf{u}) = 0 \,, \quad (3)$$
$$\partial_t \lambda + \mathbf{u} \cdot \nabla \lambda = \psi \,. \quad (4)$$

Here $\mathbf{u}$ is the velocity vector and E the specific total energy. The hydrostatic pressure p is given by

$$p = (\gamma - 1)(\rho E - \frac{1}{2}\rho \mathbf{u}^T \mathbf{u} + \rho \lambda q) \quad (5)$$

with γ denoting the ratio of specific heats and q the heat release due to the chemical reaction per unit mass. The reaction itself is modeled by the simple rate function

$$\psi = \frac{2}{T_R}(1 - \lambda)^{1/2} \,. \quad (6)$$

In (6), T_R denotes a typical time associated with the reaction, in which the depletion from A to B is complete. It is worth mentioning that the above model includes the Euler equations for a single polytropic gas as (1)-(3) and (5) for $\psi \equiv 0$ and $q \equiv 0$, which is the appropriate model for purely hydrodynamic shock wave propagation (cf. Sec. 6.1).

2.2 Cartesian Finite Volume Schemes with Embedded Boundaries

Following Clarke et al. [26], we apply the method of fractional steps to decouple the chemical reaction and hydrodynamic transport numerically. The *homogeneous* system of (1) to (4) and the scalar ordinary differential equation

$$\partial_t \lambda = \psi(\lambda)$$

are solved successively with the data of the preceding step as initial conditions. As the homogeneous system (1) to (4) is a hyperbolic conservation law that admits discontinuous solutions (cf. [26]), we use a time-explicit finite volume discretization that achieves a proper upwinding in all characteristic fields. The scheme is based on a straightforward generalization of the Roe scheme for the purely hydrodynamic Euler equations (1) to (3) and is extended to a multi-dimensional Cartesian scheme via the method of fractional steps (cf. [27]). To circumvent the intrinsic problem of unphysical total densities and internal energies near vacuum due to the Roe linearization (cf. [28]), the scheme has the possibility to switch to the simple, but extremely robust Harten-Lax-Van Leer (HLL) Riemann solver. The occurrence of the disastrous carbuncle phenomena [29], a multi-dimensional numerical cross-flow instability that affects every simulation of strong grid-aligned shocks or detonation waves, is prevented by introducing a small amount of additional numerical viscosity in a multi-dimensional way [30]. This hybrid Riemann solver is supplemented with the MUSCL-Hancock variable extrapolation technique of Van Leer [27] to achieve second-order accuracy in regions where the solution is smooth.

Geometrically complex moving boundaries are considered within the Cartesian method outlined above by utilizing some of the finite volume cells as ghost cells to enforce immersed boundary conditions [31,10]. An extension of this approach to the case of arbitrarily-thin open immersed boundaries has recently been proposed by Tam et al [15]. The ghost cell values are set immediately before the original numerical update to model moving embedded walls. The boundary geometry is mapped onto the Cartesian mesh by employing a scalar level set function ϕ that stores the signed distance to the boundary surface and allows the efficient evaluation of the boundary outer normal in every mesh point as $\mathbf{n} = -\nabla\phi/|\nabla\phi|$ [32]. In coupled Eulerian-Lagrangian simulations ϕ is updated on-the-fly by calling the closest-point-transform algorithm described in detail in Sec. 4. A cell is considered to be a valid fluid cell within the interior if the distance ϕ in the cell *midpoint* is positive, and is treated as exterior otherwise. The numerical stencil itself is not modified, which causes a slight diffusion of the boundary location throughout the method and results in an overall non-conservative scheme. We alleviate such errors and the unavoidable staircase approximation of the boundary with this approach effectively by using the dynamic mesh adaptation technique described in the next section to also refine the Cartesian mesh appropriately along the boundary.

For the system of equations (1)-(4), the boundary condition at a rigid wall moving with velocity $\mathbf{w}$ is $\mathbf{u}\cdot\mathbf{n} = \mathbf{w}\cdot\mathbf{n}$. Enforcing the latter with ghost cells, in which the discrete values are located in the cell centers, requires the mirroring of the primitive values ρ, $\mathbf{u}$, p, λ across the embedded boundary. The normal velocity in the ghost cells is set to $(2\mathbf{w}\cdot\mathbf{n} - \mathbf{u}\cdot\mathbf{n})\mathbf{n}$, while the mirrored tangential velocity remains unmodified. The construction of the velocity vector within the ghost cells therefore reads

$$\mathbf{u}' = (2\mathbf{w}\cdot\mathbf{n} - \mathbf{u}\cdot\mathbf{n})\mathbf{n} + (\mathbf{u}\cdot\mathbf{t})\mathbf{t} = 2\left((\mathbf{w} - \mathbf{u})\cdot\mathbf{n}\right)\mathbf{n} + \mathbf{u} \quad (7)$$

with $\mathbf{t}$ denoting the boundary tangent.

The utilization of mirrored ghost cell values in a ghost cell center $\mathbf{x}$ requires the calculation of spatially interpolated values at the point

$$\tilde{\mathbf{x}} = \mathbf{x} + 2\phi\mathbf{n} \quad (8)$$

from neighboring interior cells. For instance, in two spatial dimensions we employ a bilinear interpolation between (usually) four adjacent cell values, but directly near the boundary the number of cells contributing to the interpolation needs to be decreased (cf. Fig. 1). It has to be emphasized that for hyperbolic problems with discontinuities like detonation waves, special care must

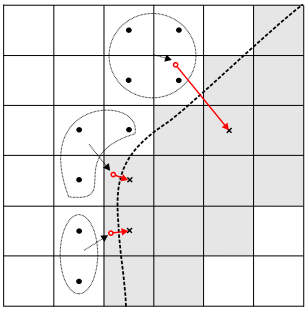

Figure 1 Interpolation from interior cells to construct mirrored values to be used within internal ghost cells (gray).

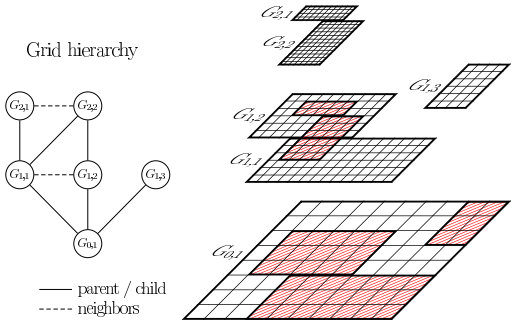

Figure 2 The AMR method creates a hierarchy of rectangular subgrids.

be taken throughout the extrapolation operation to preserve the monotonicity of the numerical solution. Figure 1 highlights the reduction of the interpolation stencil for some exemplary cases close to the embedded boundary. The interpolation locations according to (8) are indicated by the origins of the arrows normal to the complex boundary (dotted).

After the application of the numerical scheme, the cells that have been used to impose the internal boundary conditions are set to the entire state vector of the nearest cell in the interior. This operation achieves a constant value extrapolation and ensures proper values in case such a cell becomes a regular interior cell in the next step due to boundary movement. Note that the boundary velocity $\mathbf{w}$ gets automatically considered via operation (7) and the usual stability condition for time-explicit methods for system (1)-(4) also ensures that the embedded boundary propagates at most one cell further in every time step.

2.3 Structured Adaptive Mesh Refinement

In order to supply the required temporal and spatial resolution efficiently, we employ the structured adaptive mesh refinement (SAMR) method of Berger and Colella [33], which is tailored especially for hyperbolic conservation laws on logically rectilinear finite volume grids. Instead of replacing single cells by finer ones, as it is done in cell-oriented refinement techniques, the Berger-Collela SAMR method follows a patch-oriented approach. Cells being flagged by various error indicators (shaded in Fig. 2) are clustered with a special algorithm [14] into non-overlapping rectangular grids. Refinement grids are derived recursively from coarser ones and a hierarchy of successively embedded levels is thereby constructed (cf. Fig. 2). All mesh widths on level l are r_l-times finer than on level $l-1$, i.e. $\Delta t_l := \Delta t_{l-1}/r_l$ and $\Delta x_{k,l} := \Delta x_{k,l-1}/r_l$ with $r_l \geq 2$ for $l > 0$ and with $r_0 = 1$, and a time-explicit finite volume scheme will (in principle) remain stable on all levels of the hierarchy.

The numerical scheme is applied on level l by calling a single-grid routine in a loop over all subgrids. The subgrids get computationally decoupled by employing additional ghost cells around each computational grid. Three different types of ghost cells have to be considered: Cells outside of the root domain are used to implement physical boundary conditions. Ghost cells overlaid by a grid on level l have a unique interior cell analog and are set by copying the data value from the grid, where the interior cell is contained (synchronization). On the root level no further boundary conditions need to be considered, but for $l > 0$ internal boundaries can also occur. They are set by a conservative time-space interpolation from two previously calculated time steps of level $l-1$.

Besides a general tree data structure that stores the topology of the hierarchy (cf. Fig. 2), the SAMR method requires at most two regular arrays assigned to each subgrid. They contain the discrete vector of state for the actual and updated time step. The regularity of the data allows high performance on vector and super-scalar processors that allow cache optimizations. Small data arrays are effectively avoided by leaving coarse level data structures untouched when higher level grids are created. Values of cells covered by finer subgrids are subsequently overwritten by averaged fine grid values. This operation leads to a modification of the numerical stencil on the coarse mesh and requires a special flux correction in cells abutting a fine grid. The correction replaces the coarse grid flux along the fine grid boundary by a *sum* of fine grid fluxes and ensures the discrete conservation property of the hierarchical method (at least for purely Cartesian problems without embedded boundaries; see [33] or [23] for details).

2.4 Parallel Implementation

SAMR in the Virtual Test Facility at the California Institute of Technology is provided generically by the AMROC (Adaptive Mesh Refinement in Object-oriented C++) framework [34]. AMROC has been parallelized effectively for distributed memory machines [35] and can be used on all systems that provide the MPI library. The parallelization strategy is a rigorous domain decomposition approach that partitions the SAMR hierarchy from the root level on. The key idea is that all higher level

domains are required to follow this "floor plan". A careful analysis of the AMR algorithm uncovers that the only parallel operations under this paradigm are ghost cell synchronization, redistribution of the SAMR hierarchy and the application of the previously mentioned flux correction terms. Interpolation and averaging, and in particular the calculation of the flux corrections, remain strictly local [35]. In AMROC, a generalization of Hilbert's space-filling curve [36] is used to derive load-balanced root level distributions at run time. The entire SAMR hierarchy is considered by projecting the accumulated work from higher levels onto the root level cells. On top of the generic SAMR algorithm and its parallel hierarchical data structures in C++, a specific application is formulated with single-grid routines. A user must provide routines for the numerical scheme, for setting up boundary and initial conditions, and for the interpolation and averaging operations. The results throughout this paper have been produced with all these routines written in Fortran 77.

3 Lagrangian Formulation of Solid Dynamics

We adopt a conventional Lagrangian formulation [37] for describing the large, dynamic deformations of solid materials subject to high-intensity shock loadings. The formulation accounts for finite kinematics, inertia and general constitutive behavior, including strength.

We select the configuration $B_0 \subset \mathbb{R}^3$ of the body at time t_0 as the reference configuration. The coordinates $\mathbf{X}$ of points in B_0 are used to identify material particles throughout the motion. The motion of the body is described by the deformation mapping

$$\mathbf{x} = \boldsymbol{\varphi}(\mathbf{X}, t), \qquad \mathbf{X} \in B_0 \tag{9}$$

Thus, $\mathbf{x}$ is the position of material particle $\mathbf{X}$ at time t. We shall denote by B_t the deformed configuration of the body at time t. The material velocity and acceleration fields follow from (9) as $\dot{\boldsymbol{\varphi}}(\mathbf{X}, t)$ and $\ddot{\boldsymbol{\varphi}}(\mathbf{X}, t)$, $\mathbf{X} \in B_0$, respectively, where a superposed $\dot{()}$ denotes partial differentiation with respect to time at fixed $\mathbf{X}$. The local deformation of an infinitesimal material neighborhood is described by the deformation gradient

$$\mathbf{F} = \nabla_0 \boldsymbol{\varphi}(\mathbf{X}, t), \qquad \mathbf{X} \in B_0 \tag{10}$$

where ∇_0 denotes the material gradient of a function defined over B_0. Thus, the components of $\nabla_0 f$ are the partial derivatives of f with respect to $\mathbf{X}$. The scalar function

$$J = \det\big(\mathbf{F}(\mathbf{X}, t)\big) \tag{11}$$

is the Jacobian of the deformation, and measures the ratio of the deformed to undeformed volume of an infinitesimal material neighborhood.

The motion of the body is subject to conservation of mass, linear momentum and energy (cf. [37]). The local form of mass balance is

$$\dot{\rho_0} = 0 \qquad \text{in } B_0 \tag{12}$$

where $\rho_0(\mathbf{X})$ is the mass density over B_0. The local form of linear momentum balance is

$$\nabla_0 \cdot \mathbf{P} + \rho_0 \mathbf{B} = \rho_0 \ddot{\boldsymbol{\varphi}}, \qquad \text{in } B_0 \tag{13}$$

where $\mathbf{B}(\mathbf{X}, t)$ are the body forces per unit mass, and $\mathbf{P}(\mathbf{X}, t)$ is the first Piola-Kirchhoff stress tensor. The Cauchy stress tensor follows from $\mathbf{P}$ through the relation

$$\boldsymbol{\sigma} = J^{-1} \mathbf{P} \mathbf{F}^T \tag{14}$$

Conservation of angular momentum requires $\boldsymbol{\sigma}$ to be symmetric. The local form of energy balance is

$$\rho_0 \frac{\partial U}{\partial t} + \nabla_0 \cdot \mathbf{H} = \mathbf{P} : \nabla_0 \dot{\boldsymbol{\varphi}} + \rho_0 R \tag{15}$$

where U is the internal energy per unit mass, $\mathbf{H}$ is the heat flux per unit of undeformed area, and R is the heat source rate per unit mass.

For purposes of formulating boundary conditions, we partition the boundary ∂B_0 of B_0 into a Dirichlet or displacement boundary ∂B_{01} and a Neumann or traction boundary ∂B_{02}. The displacement boundary conditions then take the form:

$$\boldsymbol{\varphi} = \bar{\boldsymbol{\varphi}}, \qquad \text{on } \partial B_{01} \tag{16}$$

where $\bar{\boldsymbol{\varphi}}(\mathbf{X}, t)$ is the prescribed deformation mapping on ∂B_{01}. The traction boundary conditions take the form:

$$\mathbf{P} \cdot \mathbf{N} = \bar{\mathbf{T}}, \qquad \text{on } \partial B_{02} \tag{17}$$

where $\mathbf{N}$ is the unit outward normal to ∂B_{02} and $\bar{\mathbf{T}}(\mathbf{X}, t)$ are the prescribed tractions applied to ∂B_{02}. Finally, dynamic problems require initial conditions $\boldsymbol{\varphi}_0(\mathbf{X})$, $\dot{\boldsymbol{\varphi}}_0(\mathbf{X})$ and $U_0(\mathbf{X})$ to be specified over B_0.

A general constitutive theory of inelastic material behavior may be based on irreversible continuum thermodynamics (cf. [38,39,40] for more extensive accounts). In this context, viscosity may be modeled by assuming an additive decomposition

$$\mathbf{P} = \mathbf{P}^e + \mathbf{P}^v \tag{18}$$

of the first Piola-Kirchhoff stress tensor into an equilibrium part $\mathbf{P}^e$ and a viscous part $\mathbf{P}^v$. Additionally, we assume that the local thermodynamic state of the material is fully described by the local deformation $\mathbf{F}$, the local absolute temperature T, and a collection $\mathbf{q}$ of internal state variables. In particular, the free energy per unit undeformed volume is expressible as a function $A(\mathbf{F}, T, \mathbf{q})$. The equilibrium stresses then follow from the free energy in the form

$$\mathbf{P}^e = A_{,\mathbf{F}}\left(\mathbf{F}, T, \mathbf{q}\right) \tag{19}$$

In materials without strength, A depends on deformation only through the Jacobian J of the deformation and (19) reduces to

$$\mathbf{P}^e = -Jp\mathbf{F}^{-T} \tag{20}$$

where $p = -A_{,J}$ is the hydrostatic pressure. Here we adopt the usual fluids convention and regard compressive pressure as positive. In the presence of shocks propagating in the solid, the volumetric response is described by a suitable equation of state (EOS). The constitutive library in the VTF is endowed with well-established phenomenological equations of state for solids including the Mie-Gruneisen EOS, as well as others obtained from first-principles quantum mechanics calculations as part of the Center's efforts in multiscale modeling.

The viscous stresses are assumed to take the form

$$\mathbf{P}^v(\dot{\mathbf{F}}, \mathbf{F}) = J\sigma^v\mathbf{F}^{-T} \tag{21}$$

with

$$\sigma^v = 2\eta(\mathrm{sym}\dot{\mathbf{F}}\mathbf{F}^{-1})^{\mathrm{dev}} \tag{22}$$

where η is the viscosity coefficient, and sym and dev denote the symmetric and deviatoric components of a tensor, respectively.

In the presence of strong shocks propagating inside the solid material, a shock-capturing numerical scheme is necessary just as with the case of fluids. In the VTF, the artificial viscosity approach for Lagrangian analysis of shocks in solids with strength on unstructured, arbitrary-order tetrahedral meshes presented in [41] is adopted. The main purpose of artificial viscosity schemes is to enable the simulation of strong shocks of thickness so small that they cannot possibly be resolved by the grid. The addition of an artificial viscosity, as first proposed by von Neumann and Richtmyer [42], has the intended effect of spreading the shock front over several grid points without affecting key aspects of the shock dynamics such as the shock speed, and without the introduction of artifacts such as spurious oscillations in the shock profile. The formulation presented in [41] and adopted in this paper addresses some of the issues that are specific to solids undergoing large plastic deformations under shock loading: it is formulated within a general finite-deformation framework; it satisfies material-frame indifference exactly; it is applicable to high-order tetrahedral elements and unstructured meshes, which provides a means of avoiding locking problems associated with the plastic incompressibility constraint. For completeness, we summarize the key elements of the formulation.

We assume that the viscosity coefficient comprises two terms,

$$\eta_h = \eta + \Delta\eta \tag{23}$$

where η is the physical viscosity coefficient of the material and $\Delta\eta$ is the added artificial viscosity. The artificial viscosity coefficient $\Delta\eta$ at a given Gauss quadrature point of the mesh is assumed to be of the form

$$\Delta\eta = \begin{cases} \max\left(0, -\frac{3}{4}\ell\rho(c_1\Delta u - c_L a) - \eta\right) & \Delta u < 0 \\ 0 & \Delta u \geq 0 \end{cases} \tag{24}$$

where ℓ is a measure of the element size and Δu is a measure of the velocity jump across the element; c_1 and c_L are coefficients; a is a characteristic sound speed of the material; and $\rho = \rho_0/J$ is the mass density per unit deformed volume.

In a one-dimensional problem, the values of Δu and ℓ simply correspond to the jump in velocity across an element and the element size in the deformed configuration. In a multidimensional simulation, Δu and ℓ must be defined in a way that renders the artificial viscosity formulation material-frame indifferent. A simple way to satisfy this requirement is to express the value of Δu as a function of the Jacobian of the deformation or its time derivatives. This suggests writing

$$\Delta u = \ell\frac{\partial \log J}{\partial t} \tag{25}$$

where

$$\ell = (J\, d!|K|)^{1/d} \tag{26}$$

d is the dimension of space and $|K|$ is the volume of element K in its reference configuration. Eqs. (25) and (26) are evaluated at every Gauss quadrature point in the mesh. We further approximate the velocity jump as

$$\Delta u_{n+1} = \ell_{n+1}\frac{\log J_{n+1} - \log J_n}{\Delta t} \tag{27}$$

More general forms of the artificial viscosity can be formulated in terms of the invariants of the right Cauchy-Green tensor and their time derivatives.

A widely-used procedure for calibrating c_1 and c_L attributed to Christensen is discussed by Benson [43]. Briefly, the method is based on the fact that the Rankine-Hugoniot jump conditions define an implicit relation $D(\Delta u)$ between the shock speed and the jump in velocity. This relation is usually well approximated by a linear fit of the form

$$D = s_0 + s_1\Delta u \tag{28}$$

for a wide variety of materials and for each set of initial conditions, where s_0 and s_1 are adjustable parameters. By inserting this relation into the Rankine-Hugoniot linear momentum jump condition for a steady shock

$$\Delta p = \rho D\Delta u \tag{29}$$

a quadratic relation in Δu is obtained. In this relation, ρ represents the unshocked density. By regarding the artificial viscosity scheme as an approximate Riemann solver, the following relations for c_1 and c_L are obtained:

$$\begin{aligned} c_1 &\approx s_1 \\ c_L &\approx s_0/a \end{aligned} \tag{30}$$

Our experience suggests that these values tend to be overly diffusive and to perform better for strong shocks than for weak shocks. For weaker shocks, the coefficients (30) usually smear the shock over more elements than strictly necessary and exacerbate overheating effects. The value of c_1 may be expected to remain close to 1, since for strong shocks it follows directly from the Rankine-Hugoniot jump conditions that $\Delta p \sim \rho_1(\Delta u)^2$. The value of c_L is typically in the range of 0.1 to 1.

A complete characterization of the behavior of the solid requires, in addition to a volumetric equation of state, a description of its strength, including its elasticity, yield point, strain hardening, rate sensitivity and temperature dependence. In the constitutive framework described above, this is accomplished by the specification of the internal energy density A, plus suitable kinetic equations for the internal state variables $\mathbf{q}$ and the heat flux $\mathbf{H}$. For simplicity in the calculations presented subsequently, we assume adiabatic conditions; consequently, the heat flux and the heat sources are presumed negligible and drop out of the energy equation (15).

The VTF possesses a large set of constitutive models and algorithmic updates describing a wide range of material response, including elasticity, isotropic viscoplasticity, crystal plasticity elastic and plastic response of solid materials. Some of these models were developed as part of the Center's efforts on multiscale modeling of materials at high deformation rates, including variational updates for isotropic and crystal plasticity [40], a multiscale model of single-crystal b.c.c. Tantalum [44] and a polyconvex model for anisotropic cubic crystals [45]. The specific constitutive models employed in the calculations presented in this work are described in the respective subsections of Sec. 6.

The preceding continuum formulation may be rendered into a form suitable for computation by a combination of a time discretization of the momentum and constitutive equations and a finite-element discretization of the reference configuration of the solid. Some key aspects of the particular approach adopted here are summarized next for completeness and later referenced. More detailed accounts may be found in ([39,40]).

3.1 Temporal Discretization

Here and subsequently, we envision an incremental solution procedure aimed at sampling the solution at discrete times $t_0, t_1, \ldots, t_n$, where $t_{n+1} = t_n + \triangle t$. The mass conservation equation (12) is trivially satisfied keeping the value of ρ_0 constant at every sampling time instant. The linear-momentum balance equation (13) is discretized in time by employing the Newmark family of time-stepping algorithms:

$$\boldsymbol{\varphi}_{n+1} = \boldsymbol{\varphi}_n + \triangle t\, \dot{\boldsymbol{\varphi}}_n + \triangle t^2[(1/2-\beta)\ddot{\boldsymbol{\varphi}}_n + \beta\ddot{\boldsymbol{\varphi}}_{n+1}] \quad (31)$$

$$\dot{\boldsymbol{\varphi}}_{n+1} = \dot{\boldsymbol{\varphi}}_n + \triangle t\,[(1-\gamma)\ddot{\boldsymbol{\varphi}}_n + \gamma\ddot{\boldsymbol{\varphi}}_{n+1}] \quad (32)$$

$$\rho_0\ddot{\boldsymbol{\varphi}}_{n+1} - \nabla_0 \cdot \mathbf{P}_{n+1} = \rho_0 \mathbf{B}_{n+1} \quad (33)$$

where β and γ are Newmark's parameters, the subscript n refers to time t_n, and $\dot{\boldsymbol{\varphi}}_n$ and $\ddot{\boldsymbol{\varphi}}_n$ are the material velocity and acceleration fields. The performance of Newmark's algorithm, including its range of stability, has been extensively documented in the literature (e.g., [46, 47,48]). The particular case of $\beta = 0, \gamma = \frac{1}{2}$ is explicit and second-order accurate and corresponds to central differences.

The updated stresses $\mathbf{P}_{n+1}$ required for the computation of the internal forces in (33) follow from (18) as

$$\mathbf{P}_{n+1} = \mathbf{P}^e_{n+1} + \mathbf{P}^v_{n+1} \quad (34)$$

In calculations we adopt a staggered constitutive update procedure consisting of an evaluation of the equilibrium relations

$$\mathbf{P}^e_{n+1} = \mathbf{P}^e\left(\mathbf{F}_{n+1}, U_n\right) \quad (35)$$

at constant internal energy U_n, followed by an adiabatic update of the internal energy of the form

$$\rho_0 U_{n+1} = \rho_0 U_n + \mathbf{P}_{n+1} \cdot (\mathbf{F}_{n+1} - \mathbf{F}_n) \quad (36)$$

The viscous stresses $\mathbf{P}^v_{n+1}$ may be computed from Eqs. (21) and (22) by recourse to a difference approximation for $\dot{\mathbf{F}}$ [39]. For general materials, the formulation of incremental equilibrium relation such as Eq. (35) requires an algorithm for updating the internal variables. Variational forms of these update algorithms may be found elsewhere [39,40].

3.2 Spatial Discretization

The spatial finite-element discretization of the linear momentum balance equation adopted here is based on the weak form

$$\int_{B_0} \rho_0 \ddot{\boldsymbol{\varphi}}_{n+1} \cdot \mathbf{v}\, d\Omega + \int_{B_0} \mathbf{P}_{n+1} : \nabla_0 \mathbf{v}\, d\Omega = \int_{\partial B_{02}} \bar{\mathbf{T}} \cdot \mathbf{v}\, dS + \int_{B_0} \rho_0 \mathbf{B}_{n+1} \cdot \mathbf{v}\, d\Omega \qquad \forall \mathbf{v} \in V \quad (37)$$

where V is the space of admissible displacements, i.e., such incremental displacements, or alternatively velocities, that satisfy the essential boundary conditions (16) in the sense of traces. This weak statement is also known as the principle of virtual work. We consider finite-element interpolations of the form

$$\boldsymbol{\varphi}_h(\mathbf{X}) = \sum_{a=1}^{N} \mathbf{x}_a N_a(\mathbf{X}) \quad (38)$$

$$U_h(\mathbf{X}) = \sum_{e=1}^{E} \sum_{q=1}^{Q} U^e_q M^e_q(\mathbf{X}) \quad (39)$$

where $\boldsymbol{\varphi}_h$ is the deformation mapping interpolant, $\mathbf{U}_h$ is the internal energy interpolant, and N_a and M^e_q are the displacement and internal energy shape functions, respectively. The sum on a ranges over the N nodes in

the mesh, whereas the sum on e ranges over the E elements in the mesh, and the sum over q ranges over the Q quadrature points per element. The displacement shape functions N_a must be conforming. In calculations we employ standard quadratic, six-noded triangles or ten-noded tetrahedra (e.g., [49]). By contrast, because of the absence of heat conduction, the internal energy interpolation need only ensure that U_h be essentially bounded; consequently, the shape functions M_q^e can be chosen to be piecewise polynomials. In the calculations presented here, we take M_q^e to be constant over the Voronoi cell of the corresponding quadrature point, whence U_q^e becomes the value of the internal energy at the quadrature point. For general materials with strength, all remaining internal variables $\mathbf{q}$ are interpolated likewise.

3.3 Parallel Implementation

The parallel implementation of the solid solver is based on mesh partitioning using a heuristic graph partitioning algorithm as provided by the well-established software package METIS, [50]. An initial coarse mesh (typically of one hundred thousand elements) is constructed, partitioned and distributed among the solid processors participating in the simulation. Each processor adopts a single mesh partition for the remaining of the computation. The local mesh is recursively refined using a mesh subdivision algorithm for tetrahedral meshes proposed by Liu and Joe [51]. At each refinement iteration, each tetrahedron is subdivided into eight new ones. The chief advantage of this algorithm is that only two new classes of slightly lower-regularity tetrahedra are introduced in the refined mesh irrespective of the number of refinement iterations, thus enabling the creation of massive partition meshes in each processors in a scalable way. The conformity of the inter-processor meshes at partition boundaries is guaranteed by the initial conformity of the coarse mesh and the uniqueness of the refinement algorithms. In the case of subdivision, the inter-partition communication maps need to be reconstructed.

In the explicit time-integration scheme adopted to describe the fast dynamics processes of interest, the solid calculation proceeds by applying the initial conditions at the beginning of the time step and the traction boundary conditions produced by the fluid pressure, then by computing the predictor configuration of the solid and performing the necessary constitutive updates at each element quadrature point as part of the assembly of the global residual vector. At this point, a point-to-point communication step consisting of exchanging the incomplete residuals at partition boundary nodes is carried out after which the corrected nodal accelerations and velocities can be obtained locally by application of the unmodified corrector algorithmic step. It bears emphasis that the communications only involve partition boundary data and, thus, retain a surface to volume character which, in turn, results in excellent scalability properties. The approach described above has been extensively tested and successfully employed in a variety of problems related to the dynamic response of polycrystalline materials, [52,53,54,55,56,57], in which the computationally-intensive multiscale models of single-crystal plasticity and the need to obtain full-solutions in three dimensions can only be achieved by recourse to very large-scale simulation. In the problems studied in the references above, computations involving multi-million element meshes, the tracking of multi-billion internal variables and conducted on upward of one thousand processors are typical. These studies were possible owing to the massive computer resources provided by the ASC-DOE Supercomputers.

4 Closest-Point-Transform Algorithm

In section 2, we have introduced the concept of a signed distance function as a natural way to represent a complex embedded boundary on a Cartesian mesh. While signed distance functions are easily prescribed for single elementary geometric objects, their evaluation can be extremely cumbersome for complex shapes. In coupled Eulerian-Lagrangian simulations, this complex shape is defined by the boundary of the solid mesh. Since the solid mesh is tetrahedral (cf. Sec. 3), the interface is a triangular mesh. In the following, we outline the specific algorithm that we have developed to effectively transform the explicit description of a triangulated surface mesh into a signed distance function. The problem is equivalent to finding for every discrete point on the Cartesian SAMR grid the nearest or closest point on this surface mesh. The algorithm is therefore named the *closest point transform* (CPT). Without loss of generality, we assume a single uniform Cartesian grid in the following discussion.

4.1 Problem Description

Let $\phi(\mathbf{x})$, $\mathbf{x} \in \mathbb{R}^n$, be the distance from the point $\mathbf{x}$ to a manifold $\mathcal{I}$. If $\dim(\mathcal{I}) = n-1$ and the manifold is closed, (for example, curves in 2-D or surfaces in 3-D), then the distance may be signed. The orientation of the manifold determines the sign of the distance. We adopt the convention that the outward normal points in the positive direction. In order for the distance to be well defined, the manifold must be orientable and have a consistent orientation. A Klein bottle in 3-D, for example, is not orientable. Two concentric circles in 2-D have consistent orientations only if the normals of the inner circle point "inward" and the normals of the outer circle point "outward", or vice versa. Otherwise, the distance would be ill,defined in the region between the circles. For manifolds which are not closed, the signed distance is ill defined in any neighborhood of the boundary. However, the

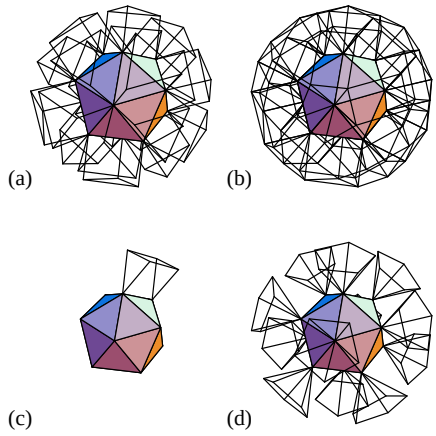

Figure 3 The characteristic polyhedra for faces, edges, and vertices.

distance is well defined in neighborhoods of the manifold which do not contain the boundary.

The signed distance ϕ to a surface $\mathcal{I}$ satisfies the eikonal equation

$$|\nabla\phi| = 1 \tag{40}$$

with boundary conditions $\phi\big|_{\mathcal{I}} = 0$ (see [58]). For most boundary conditions, a solution to (40) exists only in the weak sense. It is continuous, but only piecewise differentiable. The solution is non-differentiable where characteristics intersect. These are places that have multiple closest points to the manifold. At differentiable points on the manifold, the direction of the characteristics of (40) is given by the local normal on $\mathcal{I}$, i.e. $\nabla\phi/|\nabla\phi|$, and the characteristics are straight lines.

In computing the distance and closest point to a triangular mesh surface, one can consider each component of the mesh (face, edge or vertex) separately. For each entity, there are simple geometric formulas for computing the distance and closest point. For signed distance, one needs to use the surface normals to determine the sign of the distance. The surface normal along an edge is in the direction of the average of the incident face normals. The surface normal at a vertex is a weighted average of the incident face normals. The weighting is proportional to the angle in the face at that vertex.

For the fluid-solid coupling, we only need to determine the distance and closest point information in a narrow band around the interface. The information is only utilized in the ghost cells. Let δ be the Cartesian distance such that all ghost cells are within that distance of the interface surface. If the distance is computed up to δ, then one can flood fill the distance to determine which of the remaining grid points are inside or outside solid. (Flood filling means looping over the grid points while only keeping track of the sign of the distance.)

4.2 Iteration and Tree Data Structures

In the simplest algorithm for computing distance ϕ and closest point information C up to a distance δ, one loops over all components of the surface mesh $\mathcal{I}$ (faces, edges and vertices) and all Cartesian grid points. This straightforward algorithm reads:

```
simplest( φ, C, I, δ )
    for all i,j,k:
        φ[i,j,k] = ∞
    for all face in I:
        for all i,j,k:
            d = distance from grid point (i,j,k) to face
            if |d| ≤ δ and |d| < |φ[i,j,k]|:
                φ[i,j,k] = d
                C[i,j,k] = closest point on face
    for all edge in I:
        ...
    for all vertex in I:
        ...
return
```

The algorithm has computational complexity $\mathcal{O}(MG)$, where M is the number of components in the mesh and G is the number of grid points. Since time-explicit finite volume methods basically have complexity $\mathcal{O}(G)$ for a single time step, this naive algorithm is not suitable for computing the CPT during the course of a simulation.

An alternative could be to store the mesh in a data structure that supports minimum distance queries, like a bounding box tree [59]. However, the average expected complexity of a single distance or closest point computation in this approach would still be $\mathcal{O}(\log M)$. This implies an overall complexity of $\mathcal{O}(G \log M)$ for the CPT algorithm. By taking advantage of the fact that the grid points of the Cartesian mesh form a lattice, we have been able to develop a CPT algorithm tailored especially for our purposes with better computational complexity.

4.3 The CSC Algorithm

One can efficiently compute the distance and closest point on a grid by solving the eikonal equation with the method of characteristics and utilizing polyhedron scan conversion. This is called the characteristics/scan conversion (CSC) algorithm [60]. For a given grid point, the closest point on the triangular mesh lies on one of the primitives (faces, edges and vertices) that comprise the surface. The characteristics emanating from each of these primitives form polyhedral shapes, which we call *characteristic polyhedra*. A characteristic polyhedron contains all of the points which are possibly closest to its corresponding face, edge or vertex. We determine the grid points that lie within each of these polyhedra, then use simple geometric formulas to calculate distance and closest points for the primitive.

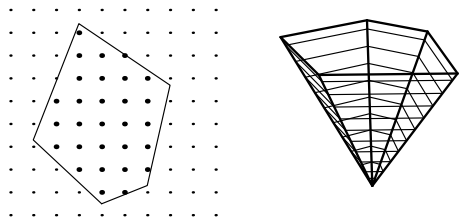

Figure 4 Scan conversion of a polygon in 2-D and slicing of a polyhedron to form polygons.

The closest points to a triangle face must lie within a triangular prism defined by the face and its normal. The prism contains the characteristic lines emanating from the face (see Figure 3a for the face polyhedra of an icosahedron). Each edge in the mesh is shared by two faces. The closest points to an edge must lie in a cylindrical wedge defined by the line segment and the normals to the two incident faces, which is depicted in Figure 3b. A single edge polyhedron is shown in Figure 3c. Each vertex in the mesh is shared by three or more faces. The closest points to a vertex must lie in a polygonal pyramid defined by the normals to the incident faces. The vertex polyhedra of an icosahedron are displayed in Figure 3d.

We can determine the grid points that lie inside a characteristic polyhedron with polyhedron scan conversion. The polyhedron is first sliced along each sheet of the grid lattice to produce polygons. Polygon scan conversion (or rasterization) is a standard technique in computer graphics for displaying filled polygons on raster displays [61,62]. It is a method for determining the pixels on the display which lie inside a polygon. Figure 4 depicts polygon scan conversion and slicing of a polyhedron. Utilizing the method of characteristics and scan conversion together, we formulate the algorithm for computing the CPT now as follows:

```
cpt( φ, C, I, δ )
   for all i,j,k:
      φ[i,j,k] = ∞
   for all face in I:
      p = polyhedron containing closest points to face
      grid_indices = scan_convert( p )
      // Loop over the scan converted points.
      for i,j,k in grid_indices:
         d = distance from grid point (i,j,k) to face
         if |d| ≤ δ and |d| < |φ[i,j,k]|:
            φ[i,j,k] = d
            C[i,j,k] = closest point on face
   for all edge in I:
      ...
   for all vertex in I:
      ...
return
```

4.4 Computational Complexity

Consider computing the closest point transform up to a distance of δ. If δ is small and the surface is smooth, the computational complexity of the algorithm is linear in both the size of the mesh and the number of grid points within δ of the surface. Thus, it has the optimal complexity.

Let the Cartesian grid have N points within a distance δ of the surface, and let ν be the ratio of the sum of the volumes of all the scan converted polyhedra divided by the volume of the domain within a distance δ of the surface. The ratio ν depends on the shape of the surface and the distance δ. If the surface is jagged and δ is relatively large, then ν will be large. If the surface is smooth and δ is relatively small, then ν will be close to unity. The total computational complexity of the algorithm is $\mathcal{O}(\nu N + M)$. The $\mathcal{O}(\nu N)$ term again comes from scan conversion and the closest point and distance computations for the grid points. The $\mathcal{O}(M)$ term represents the construction of the characteristic polyhedra. Since we expect both M and N to be small compared to the total number of grid points G, the CSC algorithm is suitable for computing the CPT during the course of a simulation.

The CSC algorithm stores the grids for distance and closest point and the mesh for which the CPT is computed. Beyond these data structures which define the problem, it does not require significant additional storage. The components of the mesh (i.e., the faces, edges and vertices) are dealt with one at a time. The memory required to scan convert a single polyhedron is insignificant compared to the memory needs of the grid and the mesh. Thus, the CSC algorithm essentially has the minimum storage requirements for the CPT problem.

4.5 Concurrency and SAMR

If the solid mesh (and hence the solid boundary) is distributed over multiple processes, the pieces must be assembled into a cohesive triangular mesh before computing the CPT. This is because one needs to know the incident faces of edges and vertices in order to compute the correct sign of the distance. We accomplish this by maintaining global identifiers for the nodes in the solid mesh.

In the course of a simulation with the SAMR framework sketched in Sec. 2.4, each fluid process performs the sequential CSC algorithm. As we use a rigorous domain decomposition to partition the SAMR hierarchy, only those components of the triangulated surface meshes that are within a distance δ of the local domain need to be considered for this computation. The necessary clipping operation is best performed *before* sending the distributed parts of the solid surface mesh to the receiving fluid processes. The details of our implementation are outlined in Sec. 5.2.

In order to make efficient use of the CPT algorithm within the SAMR method, we have organized our CPT implementation such that the algorithm can be called once for a multitude of subgrids that effectively lie within the same lattice. This arrangement ensures that each characteristic polyhedron is constructed and scan converted only once for each level in the SAMR hierarchy and guarantees computational performance that is basically independent of the number of subgrids.

5 Fluid-Structure Coupling

The explicit fluid and solid solvers are weakly coupled by applying appropriate boundary conditions at the fluid-solid interface $\mathcal{I}$ via a time-splitting technique described below. In the case of inviscid flows considered here, these boundary conditions correspond—in the Lagrangian notation of section 3, [15]—to the continuity of the normal component of the velocity field:

$$[\![\mathbf{u}\cdot\mathbf{n}]\!] = 0 \ \text{ on } \ \mathcal{I} \tag{41}$$

and the continuity of the normal component of the traction across the fluid-solid interface:

$$[\![\mathbf{t}\cdot\mathbf{n}]\!] = [\![\sigma_{ij}n_in_j]\!] = [\![\sigma_n]\!] = 0 \ \text{ on } \ \mathcal{I}, \tag{42}$$

which enforce conservation of mass and linear momentum, respectively. In the expressions above, $[\![.]\!]$ represents field jumps, $\mathbf{u}$ is the spatial velocity field which in the solid can be expressed in terms of the deformation mapping $\boldsymbol{\varphi}(\mathbf{X},t)$ as

$$\mathbf{u}^S(\mathbf{x},t) = \left(\dot{\boldsymbol{\varphi}}\circ\boldsymbol{\varphi}^{-1}\right)(\mathbf{x},t)\,, \tag{43}$$

$\mathbf{t}$ and $\mathbf{n}$ are the spatial surface traction and normal vectors, respectively and σ_{ij} are the components of the Cauchy stress tensor as defined in equation (14). The resulting boundary conditions at the fluid solid interface are simply

$$\left.\begin{aligned} u_n^S &= u_n^F \\ \sigma_{nn}^S &= p^F \end{aligned}\right|_{\mathcal{I}} \tag{44}$$

For simplicity and owing to the extremely-short time scales involved in the problems of interest, it is assumed that heat transfer across the fluid-solid interface is negligible and, thus, can be ignored.

The following simple temporal splitting scheme is adopted to accomplish the loose coupling between the fluid and solid solvers [2]:

$u_n^F := u_n^S(t)|_{\mathcal{I}}$
update_fluid(Δt)
$\sigma_{nn}^S := p^F(t+\Delta t)|_{\mathcal{I}}$
update_solid(Δt)
$t := t + \Delta t$

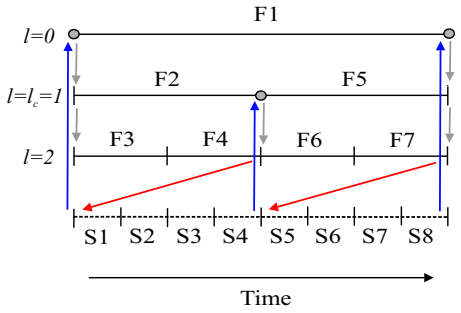

Figure 5 Data exchange between the recursive fluid SAMR solver and the linear solid solver throughout one root level time step. Red and blue arrows: flow of interface data from fluid to solid and vice-versa. Gray arrows: regridding of higher SAMR levels, base level (gray circles) stays fixed.

More general implicit and staggering schemes for coupled systems have been proposed and studied in detail in [63,64].

We have implemented this algorithm with an ad-hoc partitioning into dedicated fluid and solid processes that communicate to exchange the data along $\mathcal{I}$. In the following subsections we will outline some of the specifics of our approach that make the VTF a highly efficient framework for fluid-structure simulation on distributed memory machines.

5.1 Coupled Simulations with Eulerian SAMR

Unsteady compressible fluid flows typically show a wide range of temporal and spatial scales. While the correct numerical representation of supersonic shock and detonation waves usually requires very fine resolution only in a small band around the phenomenon of interest, a considerably coarser resolution is often sufficient in the majority of the fluid domain. This is in particular true in our case of strong pressure waves arising from the detonation of highly energetic materials; one is interested mainly in the stress waves produced by shock impact in the solid target materials and the resulting material response. Hence, incoming fluid waves and the near-body fluid-structure interaction have to be captured with high accuracy, but resolution can be reduced for outgoing fluid phenomena and in the far field. We achieve the required solution adaptation in the fluid by applying the dynamic mesh adaptation algorithm described in Sec. 2.3. The fluid-solid interface $\mathcal{I}$ is treated herein as a discontinuity with *a-priori* refinement at least up to the coupling level l_c. As the wave phenomena in solid materials are usually at least as fast as the waves in compressible fluids, the coupling level l_c will usually be the highest computationally permissible choice in order to ensure an accurate wave transmission. But special care is required to initiate the data exchange according to the above basic coupling method in a way that is compatible with the recursive SAMR algorithm.

The coupled SAMR method is implemented below in the routine advance_level() that calls itself recursively with the current level as argument l:

```
advance_level( l )
  repeat r_l times
    if time to regrid
      regrid( l )
    cpt( φ^l, C^l, I, δ_l )
    update_fluid_level( Q^l, φ^l, C^l, u^S|_I, Δt_l )
    if level l+1 exists
      advance_level(l+1)
      Correct Q^l(t+Δt_l) with Q^{l+1}(t+Δt_l)
    if l = l_c
      send_interface_data( p^F(t+Δt_l)|_I )
      if t+Δt_l < t_0+Δt_0
        receive_interface_data( I, u^S|_I )
    t := t+Δt_l
return
```

The algorithm calls the routine cpt() from Sec. 4.3 to evaluate the signed distance ϕ and the closest point information C for the actual level l based on the currently available interface $\mathcal{I}$. Together with the recent solid velocity on the interface $\mathbf{u}^S|_{\mathcal{I}}$, the discrete vector of state in the fluid $\mathbf{Q}$ is updated for the entire level with the numerical scheme outlined in Sec. 2.2. The SAMR method then proceeds as usual recursively to higher levels and utilizes the (more accurate) data from the next higher level to correct the values of the current level in cells overlaid by refinement. If level l is the coupling level l_c, we use the updated fluid data to evaluate the pressure values to be sent to the solid and to receive an updated interface mesh and velocities $\mathbf{u}^S|_{\mathcal{I}}$. The recursive order of the SAMR algorithm automatically ensures that updated interface mesh information is available for later time steps on coarser levels and to adjust the grids on level l_c dynamically before the current mesh (i.e., the level set information derived from it) is actually used to again advance level l_c. In order to achieve a proper matching of communication operations, we start the cycle by posting a receive-message in the routine fluid_step(), which does one fluid time step on level 0, before entering into the SAMR recursion. The routine fluid_step() below highlights a straightforward automatic time step adjustment for the SAMR method coupled to a solid solver.

```
fluid_step( )
  Δτ_F := min_{l=0,···,l_max} (R_l · stable_fluid_timestep(l), Δτ_S)
  Δt_l := Δτ_F/R_l for l = 0, ··· , L
  receive_interface_data( I, u^S|_I )
  advance_level( 0 )
return
```

During one root level time step at level 0 the time steps on all levels remain fixed and are calculated in advance by employing the refinement factor with respect to the root level $R_l = \prod_{\iota=0}^{l} r_\iota$ (cf. Sec. 2.3). The root level time step $\Delta\tau_F$ by itself is taken to be the minimum of the stable time step estimations from all levels and a corresponding time step $\Delta\tau_S$ in the solid. We define $\Delta\tau_S$ as a multiple of the stable time step estimation in the solid solver with respect to the communication frequency R_{l_c} in one fluid root level step and an additional factor K that allows sub-iterations in the solid solver in case of considerably smaller solid time steps. The solid update algorithm used to advance the solid by one fluid root level step $\Delta\tau_F$ is given below.

```
solid_step( )
  Δτ_S := min(K · R_{l_c} · stable_solid_timestep(), Δτ_F)
  repeat R_{l_c} times
    t_end := t + Δτ_S/R_{l_c}, Δt := Δτ_S/(K R_{l_c})
    while t < t_end
      send_interface_data( I(t), u^S|_I(t) )
      receive_interface_data( p^F|_I )
      update_solid( p^F|_I, Δt )
      t := t + Δt
      Δt := min(stable_solid_timestep(), t_end − t)
return
```

The data exchange between solid_step() and fluid_step(), within advance_level(), is visualized in Fig. 5 for an exemplary SAMR hierarchy with two additional levels with $r_1 = r_2 = 2$ and $K = 4$ sub-iterations in solid_step(). Like in the simulations in the Secs. 6.1 and 6.2, the coupling level $l_c = 1$ is not the maximum level of refinement. Figure 5 visualizes the recursion in the SAMR method by numbering the fluid update steps (F) according to the order determined by advance_level(). The order of the solid update steps (S) on the other hand is strictly linear. The flow of coupling information between solid_step() and advance_level() is visualized by the red and blue arrows. The red arrows correspond to the sending of the interface pressure values $p^F|_{\mathcal{I}}$ from fluid to solid at the end of advance_level(l_c). The blue arrows represent the sending of the interface mesh $\mathcal{I}$ and its nodal velocities $\mathbf{u}^S|_{\mathcal{I}}$ after at least K solid steps. Note that the receive_interface_data() call for the latter operation is placed into fluid_step() and advance_level() such that the updated mesh information can be employed to adjust the adaptive refinement in regrid() before it is actually used in an update_fluid_level() operation. The modification of refinement meshes is indicated in Fig. 5 by the gray arrows; the initiating base level that remains fixed throughout the regridding operation is indicated by the gray circles.

5.2 Efficient Inter-solver Communication

Critical for the performance of the coupled algorithms are the inter-solver communication routines send_interface_data() and receive_interface_data(). In order to ensure good communication performance, we have implemented a dedicated asynchronous communication library

that sets up detailed point-to-point communication patterns between the fluid and solid processes and avoids assembling global data structures.

The domain decomposition of the solid mesh across the solid processes also partitions the triangular surface mesh (cf. 3.3). Consider a single fluid process whose grids lie in a domain Ω. Suppose the closest point transform will be computed to a distance of δ. Then the fluid process needs only those portions of the interface that are within δ of Ω. If the fluid process receives only the relevant portions of the interface, it can assemble them into a local triangular mesh that is sufficient for computing the CPT and setting the boundary conditions in the ghost cells.

We determine the point-to-point communication pattern with bounding box information. Each solid process constructs a Cartesian bounding box around its portion of the interface; each fluid process makes a Cartesian bounding box around its domain and enlarges it by δ. These bounding boxes are gathered to the root fluid and solid processes. The root processes then exchange their sets of bounding boxes and broadcast the set to either the fluid or solid processes. Now each fluid process has all of the solid bounding boxes and vice versa. Each process intersects its own bounding box with the received set of bounding boxes to set up communication data structures that consider only those portions of the surface mesh and the data defined on it that are relevant to the local process. The communication between solid and fluid is non-blocking to enable overlapping synchronization and computation.

It is worth mentioning that the efficiency of the above point-to-point communication scheme necessarily relies on the fact that the fluid and solid meshes by themselves are reasonably partitioned. One could easily construct pathological cases where each is partitioned into long, thin pieces and each fluid process needs to communicate with each solid process. However, with the locality-preserving partitioning strategies employed in AMROC (generalized space-filling curve) and the parallel solid solver (graph-partitioning provided by Metis), this never occurs in practice.

6 Examples of Application

In this section, we present two examples of application of the computational framework described above. The first example corresponds to the simulation of the effects of a blast wave on the human body. The second example is a simulation of a detonation wave confined in a detonation tube and impacting a Tantalum target. These applications illustrate the robustness and versatility of the coupled computational approach as well as the computational performance on parallel machines of moderate size.

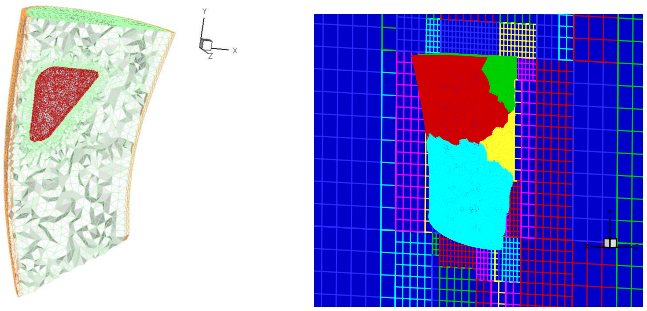

Figure 6 Mesh of the liver inside the torso (left). Domain decomposition of the solid mesh and fluid mesh adaptation at the boundary (right).

6.1 Shock Wave Impact on Soft-tissue Body

As a first example, we consider a three-dimensional fluid-structure interaction problem with $\psi \equiv 0$, $q \equiv 0$, which requires only the equations (1)-(3) and (5) to simulate the purely-hydrodynamic fluid flow. The problem is the dynamic interaction of a spherical blast pressure wave with a very simplified human body. Human injuries caused by the nearby explosion of a small amount of highly energetic material are divided into primary and secondary types: primary injuries are due to the blast wave, while secondary injuries are due to shrapnel. Primary injuries occur within a close distance of 1 to 2 m and lead to strong impulses of 1 to 2 ms duration. Under such conditions, conventional ballistic protective armor has proved to be effective for mitigating the causes leading to secondary injuries. Unfortunately, armored vests do not protect against the shock wave and can even enhance harmful blast effects. In the following, we study the stress concentrations in the human liver resulting from a blast event. An idealized geometric model of the liver is embedded in a homogeneous model torso of soft material (see Fig. 6). The liver is assumed softer and denser than the torso. Table 1) enumerates the elastic material properties adopted in the calculation.

We assume an explosion of 0.5 kg TNT in air at a distance of 1.5 m from the body. The ambient fluid pressure is $p_a = 100\,\mathrm{KPa}$ and the temperature is $T_a = 293\,\mathrm{K}$. The ideal gas relation $p = \rho RT$ then yields an ambient density of $\rho_a = 1.212\,\mathrm{kg/m^3}$, and the ratio of specific heats is set to the constant value $\gamma = 1.4$. The energy release from the TNT explosion is $E_i = 2260\,\mathrm{kJ/kg}$, which, for simplicity, we model as uniformly distributed over a small sphere of radius 5 cm with its gas initially at rest.

Table 1 Material properties of soft tissues.

	Liver	Torso
Young modulus	5 MPa	100 MPa
Poisson coefficient	0.3	0.3
Density	$1500\,\mathrm{kg/m^3}$	$950\,\mathrm{kg/m^3}$

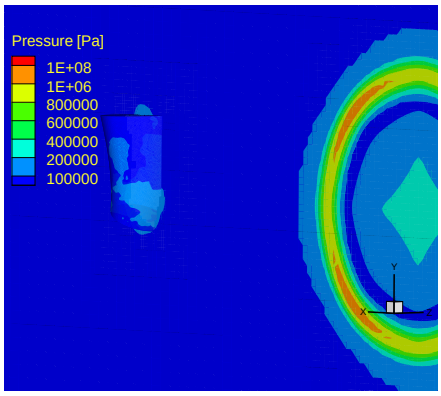

Figure 7 Formation and propagation of the blast wave (time=0.31 ms).

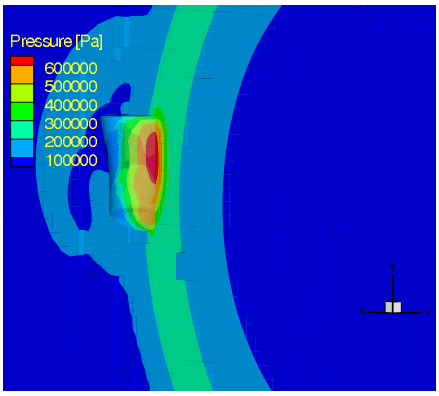

Figure 9 Interaction of the shock wave with the body (time=1.55 ms).

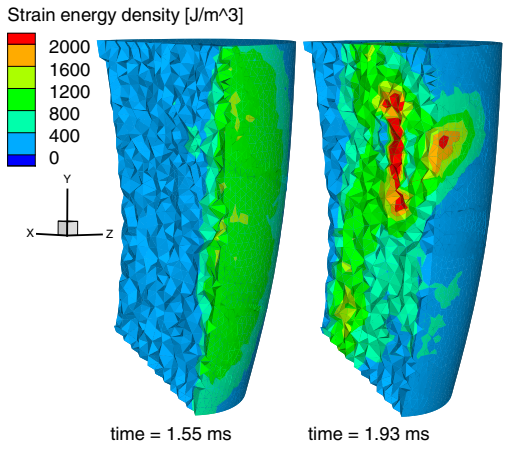

Figure 8 Stress waves in the torso after the impact.

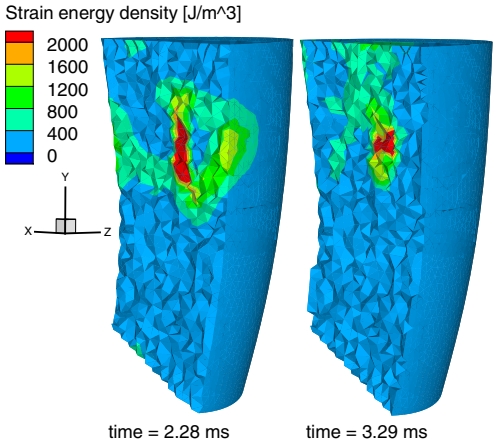

Figure 10 Stress wave reflection in the liver.

The initial temperature in this sphere is assumed to be $T_i = 1465\,\mathrm{K}$. Using (5) and the ideal gas relation together we evaluate pressure and density within the small sphere to be $p_i \approx 1700\,\mathrm{KPa}$ and $\rho_i \approx 4122\,\mathrm{kg/m^3}$. These initial conditions result in the formation of a blast wave in the fluid (cf. Fig. 7).

The solid mesh used for this simulation has 19562 nodes, while the SAMR mesh in the fluid has $100 \times 100 \times 100$ cells at the root level and employs two additional levels, both refined by a factor of 2 (cf. Fig. 6). The fluid domain is $5\,\mathrm{m} \times 5\,\mathrm{m} \times 5\,\mathrm{m}$. Scaled gradients of pressure and density are used as refinement criteria to resolve the incoming pressure wave accurately. The coupling level is set to $l_c = 1$, and we use $K = 20$ sub-iterations in the solid solver. The distance δ within which the exact signed distance information around the interface mesh is evaluated by the CPT algorithm is set to three times the diagonal of a finite volume cell.[1] This setting is sufficient to allow the construction of two internal ghost cells according to Sec. 2.2. With a target CFL condition of 0.3 in the fluid and 1.0 in the solid, we calculate 420 fluid root level steps to reach a final time of 6 ms, which involves 16800 update steps in the solid solver. The simulation is completed in about 10 hours on 10 dual-processor 2 GHz G5 nodes of a Linux Beowulf cluster connected with a Myrinet switch. In this computation, 14 and 6 processes were used for the fluid and solid solvers, respectively. The size of the fluid SAMR mesh increases during the simulation from approximately $1,045,000$ cells initially to about $2,026,000$ cells, when the pressure wave impacts the torso.

A non-dimensional analysis due to Taylor for a localized explosion gives the following relations for the peak overpressure Δp_s and its arrival time t at a distance d from the center of the explosion, [65]:

$$\Delta p_s = K_1 \frac{E_i}{d^3}, \quad t = K_2 \sqrt{\frac{d^5 \rho_a}{E_i}} \tag{45}$$

In these relations, K_1 and K_2 are non-dimensional constants and ρ_a is the ambient density. For air, Taylor has measured $K_1 = 0.155$ and $K_2 = 0.926$. More accurate

[1] This choice makes δ dependent on the mesh width on each SAMR level.

values have been given by Brode who conducted extensive numerical investigations and summarized the results in the following formulae [66]:

$$\begin{aligned} \Delta p_s[bar] &= \frac{6.7}{z^3} + 1 \text{ bar} \\ &\text{if } \Delta p_s > 10 p_a \\ \Delta p_s[bar] &= \frac{0.975}{z} + \frac{1.455}{z^2} + \frac{5.85}{z^3} - 0.019 \text{ bar} \\ &\text{if } 0.1 p_a < \Delta p_s < 10 p_a \end{aligned} \tag{46}$$

where pressures are in *bars* and $z = \frac{d}{W^{\frac{1}{3}}}$ expressed in $\mathrm{m\,kg^{-1/3}}$ is the distance scaled with the charge mass W expressed in kilograms of equivalent TNT. In our simulation a value $W = 0.5$ is adopted. Figures 7-10 show the simulation results. The shock wave reaches the torso at $t = 1.55\,\mathrm{ms}$ (see Fig. 9). This value is in reasonable agreement with the prediction from (45) considering the fact that (45) is derived for an ideal spherical shock wave emanating from a point source. At a distance $d = 1.5\,\mathrm{m}$, the overpressure peak according to (45) is $\Delta p_s \approx 104\,\mathrm{KPa}$. Our computation gives a value $\Delta p_s \approx 175\,\mathrm{KPa}$, which is close to Brode's approximation (46) $\Delta p_s \approx 177\,\mathrm{KPa}$ at $z = 1.89$. It leads to an overall pressure peak of $p_s = p_a + \Delta p_s \approx 275\,\mathrm{KPa}$. An estimate of the reflected overpressure Δp_r can be calculated from the standard Rankine-Hugoniot relations for the Euler equations (cf. [27]). For the normal reflection of a planar shock on a fixed wall we obtain

$$\begin{aligned} \Delta p_r &= 2\Delta p_s + [\gamma + 1]\,\frac{1}{2}\rho_s u_s^2\,, \\ u_s &= \Delta p_s c_a \sqrt{\frac{2}{\gamma p_a}}\,\frac{1}{\sqrt{[\gamma+1]\,\Delta p_s + 2\gamma p_a}}\,, \end{aligned}$$

where u_s denotes the fluid velocity behind the shock and c_a the ambient sound speed. For $\gamma = 1.4$ these relations lead to the simple form

$$\Delta p_r \simeq 2\Delta p_s \frac{7p_a + 4\Delta p_s}{7p_a + \Delta p_s} \in [2\Delta p_s; 8\Delta p_s] \tag{47}$$

which yields $\Delta p_r \approx 560\,\mathrm{KPa}$ for $\Delta p_s \approx 1.75\,\mathrm{KPa}$. The absolute reflected pressures obtained in the simulation are shown in Fig. 9. The peak values are in the order of 600 KPa, which is lower than the value $p_r = p_a + \Delta p_r \approx 660\,\mathrm{KPa}$ obtained from equation (47), as expected.

Finally, we discuss briefly the effects of the blast wave on the body. One of the main mechanisms of internal injury due to blast is related to the impedance mismatch between air and fluid filled organs in the human body. This significantly affects the propagation of stress waves transmitted by the blast and causes stress concentrations and localized deformations at high rates which are responsible for tissue failure and injury. The mitigating effect of the fluid-solid interaction which reduces the amount of impulse transmitted to the body is not well understood, especially when strong compressibility effects are important, as is the case in air blasts. Fig. 8-10 show different snapshots of the transmitted stress waves propagating in the torso and their interactions with the liver as measured by the elastic strain energy density. Despite the significant idealizations of this simulation, it is clear that this approach provides a viable strategy for exploring material systems for blast-injury mitigation.

6.2 HMX Detonation in a Tantalum Cylinder

The second and final simulation we want to discuss is the propagation of a detonation wave in in a high-energy explosive material confined in a thick-walled solid cylinder closed at one end. The properties of the explosive correspond to HMX ($C_4H_8N_8O_8$) and those of the cylinder material to Tantalum. The volumetric response of tantalum is modeled by recourse to Vinet's equation of state as fitted to first-principle calculations by Cohen et al [67]. The internal energy per unit mass is

$$\begin{aligned} U(T,v) = &-2.218 \times 10^{11} + \\ &e^{-4.68\left(\frac{v}{v_0}\right)^{\frac{1}{3}}} \left(1.813 \times 10^9 - 2.306 \times 10^9 \left(\frac{v}{v_0}\right)^{\frac{1}{3}}\right) + \\ &3RT - \sum_{i=2}^{3}\sum_{j=1}^{4}(i-1)A_{ij}T^i \left(\frac{v}{v_0}\right)^{j-1} \end{aligned} \tag{48}$$

where T is the absolute temperature, v the specific volume per unit mass, $v_0 = 5.959\,m^3Kg^{-1}$ is the specific volume at zero pressure and temperature, and A_{ij} is a matrix of constants. The entropy $S(T,v)$ per unit mass is given by

$$S(T,v) = 3R + 3R\log(T) - \sum_{i=1}^{3}\sum_{j=1}^{4} iA_{ij}T^{i-1}\,(v/v_0)^{j-1} \tag{49}$$

where $R = 1.5045\,J/(Kg\,K)$ is the gas constant for Tantalum. The matrix A_{ij} is identified as (SI units)

$$A = \begin{bmatrix} 9.212 \times 10^2 & -2.007 \times 10^2 & 7.640 & -3.574 \times 10 \\ 1.243 \times 10^{-2} & -3.896 \times 10^{-2} & 2.519 \times 10^{-2} & -9.233 \times 10^{-4} \\ -6.506 \times 10^{-7} & 1.505 \times 10^{-7} & -7.261 \times 10^{-7} & -2.320 \times 10^{-7} \end{bmatrix}$$

Eqs. (48) and (49) completely define the thermodynamic behavior of the material. In addition, the material is assumed to obey J_2-flow theory of plasticity. We adopt a standard formulation of finite-deformation plasticity based on a multiplicative decomposition of the deformation gradient into elastic and plastic components:

$$\mathbf{F} = \mathbf{F}^e\mathbf{F}^p \tag{50}$$

where $\mathbf{F}^p$ is assumed to be volume-preserving. The equilibrium stress-strain relation (20) is now extended to include the elasticity of the material in shear, with the result [68, 40]

$$\boldsymbol{\sigma}^e = -p\mathbf{I} + J^{-1}\mathbf{F}^e\left\{\mu(\log\sqrt{\mathbf{C}^e})^{\mathrm{dev}}\right\}\mathbf{F}^{eT} \tag{51}$$

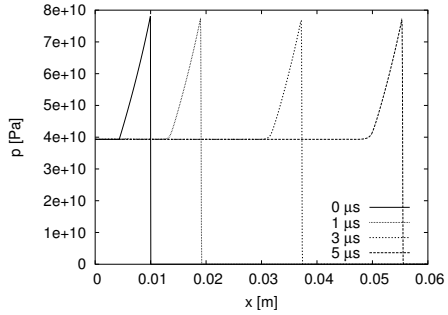

Figure 11 Pressure distributions of the detonation wave in HMX in the inner detonation chamber from a purely hydrodynamic one-dimensional simulation.

where $\mathbf{C}^e = \mathbf{F}^{eT}\mathbf{F}^e$ is the elastic Cauchy-Green deformation tensor, $\log\sqrt{\mathbf{C}^e}$ is the logarithmic elastic strain, μ is a shear modulus, and the pressure p follows from the equation of state defined by (48) and (49). The numerical advantages of using logarithmic elastic strains in conjunction with J_2-flow theory of plasticity have been documented by Cuitiño and Ortiz [68] and by Ortiz and Stainier [40]. Explicit formulae for the calculation of the exponential and logarithmic mappings, and the calculation of their first and second linearizations, have been given by Ortiz *et al.* [69]. For small incremental elastic strains, as may be expected to occur in explicit dynamics, the exponential and logarithmic mappings may be computed simply by recourse to a Taylor expansion [69].

The dependence of the elastic constants of Tantalum on pressure and temperature has been computed by Cohen [67] from first principles. It is found that the temperature dependence of the elastic constants at fixed volume is small, which suggests that the effect of temperature is ostensibly confined to thermal expansion. Therefore, to a first approximation we neglect the explicit dependence of the elastic constants on temperature at fixed volume and express c_{11}, c_{12} and c_{44} as a function of J only. We also neglect the anisotropy of the crystal and assume isotropic elastic behavior in terms of a shear modulus given by the Voigt average:

$$\mu = \frac{3}{5}\,C_{44} - \frac{1}{5}\,(2\,C_{12} - C_{11}) = \frac{3}{5}\,C_{44} + \frac{2}{5}\,C_s \qquad (52)$$

Table 2 Material parameters corresponding to the plastic response of Tantalum (SI units).

ϵ_0^p	5×10^{-4}
$\dot{\epsilon}_0^p$	3×10^{-3}
m	12.5
n	5
σ_y	5×10^8
T_{ref}	293
T_{melt}	1343
α	1
β	1

The plastic deformation $\mathbf{F}^p$ is assumed to obey the Prandtl-Reuss flow rule

$$\dot{\mathbf{F}}^p\mathbf{F}^{p-1} = \dot{\epsilon^p}\,\frac{3\mathbf{s}^e}{2\bar{\sigma}} \qquad (53)$$

where $\mathbf{s}^e = \boldsymbol{\sigma}^{e,\mathrm{dev}}$ is the stress deviator, $\bar{\sigma} = \sqrt{(3/2)s^e_{ij}s^e_{ij}}$ is the Mises stress, and ϵ^p is the effective plastic strain. A variational formulation of J_2-flow theory in finite deformations has been given by Ortiz and Stainier [40]. We additionally assume power-law rate-sensitivity, hardening and Steinberg-Guinan [70] pressure dependence of the form

$$\frac{\dot{\epsilon^p}}{\dot{\epsilon}_0^p} = \left(\frac{\bar{\sigma}}{g(\epsilon^p, T, J)}\right)^m - 1 \qquad (54)$$

$$\bar{g} = \sigma_y\left[1 - \left(\frac{T - T_{ref}}{T_{melt} - T_{ref}}\right)^{\alpha}\right]\left(1 + \frac{\epsilon^p}{\epsilon_0^p}\right)^{\frac{1}{n}} \qquad (55)$$

$$g(\epsilon^p, T, J) = \frac{\mu(J)}{\mu_0}\,\bar{g} \qquad (56)$$

Here, ϵ_0^p is a reference plastic strain, $\dot{\epsilon}_0^p$ is a reference plastic strain rate, m is the rate sensitivity exponent, n is the hardening exponent, g is the flow stress, σ_y is the initial yield stress, T_{ref} is a reference temperature, T_{melt} is the melting temperature, α is the thermal softening exponent, and μ_0 is the shear modulus at zero pressure. The contribution of the plastic dissipation to the energy equation, Eq. (15), is regarded as an additional heat source of strength

$$\rho_0 R = \beta\dot{W}^p \qquad (57)$$

where $\dot{W}^p$ is the plastic power per unit undeformed volume and β is the fraction of the plastic work converted into heat. The coefficient β is known to be a function of deformation and temperature. However, for simplicity we treat β as constant. The material parameters used in calculations are collected in Table 2.

The fluid part of this simulation uses the entire set of equations (1)-(5) and in particular the reaction term (6). The cylinder has length 0.10 m and an outer radius of 0.0185 m. An inner detonation chamber filled with HMX with radius 0.0085 m and depth 0.055 m opens at the left end of the cylinder. For the fluid initial conditions at $t = 0$, we assume a fully developed steady detonation wave with its front located at $x = 0.01$ m. The detonation is propagating in the positive direction, which is enforced by the prescription of constant inflow boundary conditions at the open left end (cf. Fig. 11). No deformations are allowed in the entire solid for $x < 0.01$ m to model a fully rigid material downstream of the initial wave. Further, no deformations are possible on the outer hull of the Ta cylinder for $0.01\,\mathrm{m} \le \mathrm{x} \le 0.03\,\mathrm{m}$.

According to Mader [71] unreacted HMX has a density of $\rho_0 = 1900\,\mathrm{kg/m^3}$, and we assume atmospheric pressure $p_0 = 100\,\mathrm{KPa}$ in the unreacted material. The

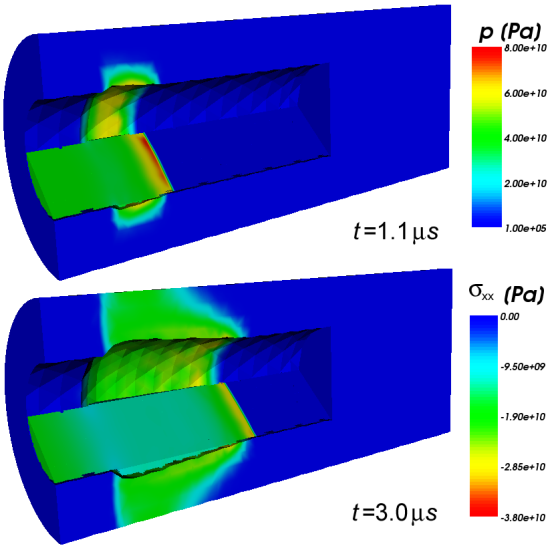

Figure 12 Initiation of stress waves in the solid and compression of the wall material next to the detonation chamber due to the detonation passage.

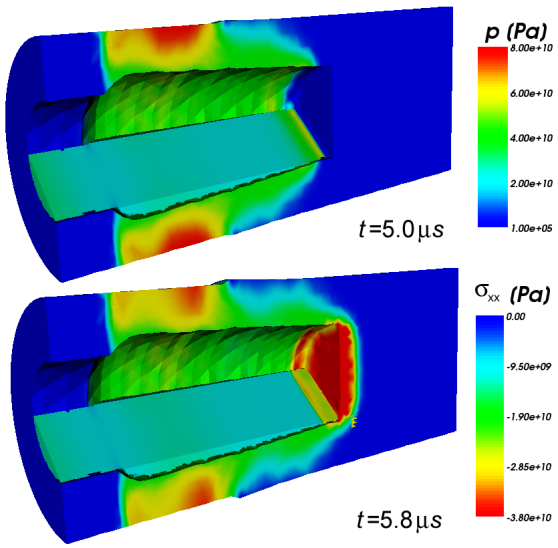

Figure 13 Strong material compression in constrained and outward movement of unconstrained walls and the strong compression in axial direction due to the impact event.

detonation velocity for a freely propagating Chapman-Jouguet detonation (cf. [25]) in HMX is experimentally known to be approximately 9100 m/s and the entire hydrodynamic flow can be described with reasonable accuracy with a constant adiabatic exponent of $\gamma = 3$ [71]. The rate factor T_R is unknown; we therefore set it to $T_R = 1\,\mu$s, which is a reasonable value for most solid explosives [25].

The above values specify the process of steady one-dimensional detonation propagation completely. A detonation wave consists of a leading hydrodynamic shock wave followed by a region of decaying continuous detonation toward chemical equilibrium. The simplified Chapman-Jouguet theory can be used to evaluate the energy release of our configuration to be $q = 5176\,\mathrm{kJ/kg}$ and to predict the hydrodynamic values in the equilibrium state $p_{\mathrm{CJ}} \approx 39.3\,\mathrm{GPa}$ and $\rho_{\mathrm{CJ}} \approx 2533\,\mathrm{kg/m^3}$. The steady internal structure can be calculated with the theory of Zeldovich, Neumann, and Döring (ZND), which constructs an analytic solution of Eqs. (1) to (6). Detailed derivations of the ZND solution can be found in the book by Ficket and Davis [25] or, for instance, in [35]. According to the ZND solution the peak values at the head of the detonation are $p_{\mathrm{vN}} \approx 78.7\,\mathrm{GPa}$ and $\rho_{\mathrm{vN}} \approx 3800\,\mathrm{kg/m^3}$. We use the analytic ZND solution as our hydrodynamic initial conditions. Fig. 11 displays the initial pressure distribution and its steady propagation in a one-dimensional simulation on a uniform mesh with 960 finite volume cells. At considerably coarser resolutions, the reaction front is not resolved with sufficient accuracy, resulting in an incorrect speed of propagation and a significant underestimation of the peak value p_{vN}. However, this high resolution is necessary only inside the reaction zone, which is made possible by the effective application of dynamic mesh adaptation.

We therefore simulate the three-dimensional fluid problem in the detonation chamber with a SAMR base grid of $60 \times 60 \times 120$ cells and use two additional levels of refinement with factors $r_1 = 2$, $r_2 = 4$. While the solid boundary is adequately refined at the coupling level $l_c = 1$, level 2 is necessary to capture the detonation wave accurately (adaptation criteria are scaled gradients in pressure and mass fraction λ). Its effective resolution corresponds to the uncoupled one-dimensional simulation shown in Fig. 11. To allow for large deformations of the cylinder walls the fluid domain spans $0.03\,\mathrm{m} \times 0.03\,\mathrm{m} \times 0.06\,\mathrm{m}$, but only the flow in the inner detonation chamber is simulated. Zero pressure values are exported to all interface mesh points at the outer hull within the fluid domain. The simulation shown in Figs. 12-14 uses a solid mesh of 56,080 elements. With a target CFL condition of 0.6 in the fluid and 0.2 in the solid, we simulate the entire detonation process and a small portion of the purely hydrodynamic shock wave reflection at the closed end of the tube propagating backwards through the fully reacted material. The final time is set to $5.8\,\mu$s, and it takes about 400 fluid root level base steps to reach it. $K = 4$ sub-iterations are used in the solid, which corresponds to approximately 3200 solid update steps. The distance parameter δ is chosen as in Sec. 6.1.

Throughout the simulation the SAMR mesh increases from an initial size of approximately 706 k cells on level 1 and 6.5 M on level 2 to about 930 k and 10.0 M, respectively. The number of grids on both levels varies between 400 and 1000. Compared with a uniform fluid mesh of $480 \times 480 \times 960 \simeq 221$ M cells that would otherwise be

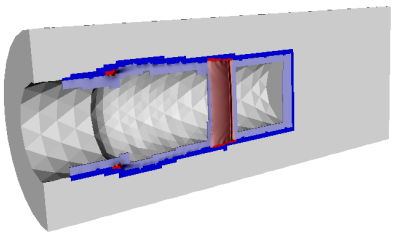

Figure 14 Schlieren plot of the density on regions covered by SAMR level 1 (blue) and 2 (red) inside the deforming cylinder for $t = 3.0\,\mu$s.

necessary to capture the detonation with similar accuracy, mesh adaptivity clearly provides enormous savings. Fig. 14 shows the highly-localized fluid mesh adaptation in the midplane for the time snapshot shown in Fig. 12.

The simulation ran on 4 nodes of a Pentium-4 2.4 GHz dual-processor system connected with Quadrics interconnect for about 63 h real time. Six processes were dedicated to the adaptive fluid simulation, while two were used for the smaller solid problem. The signed distance calculation with the CPT algorithm takes only 0.8 % of the computational costs on the fluid nodes, which impressively confirms the practical applicability of the idea of implicit geometry representation for evolving surface meshes of moderate size.

Snapshots of the simulation displaying a cut through the hydrodynamic pressure distribution and the normal stress in the axial direction are shown in Figs. 12 and 13. These figures show several salient features of this coupled problem: the superseismic loading of the lower-impedance cylinder walls leading to an inclined shock front in the solid (Fig. 12), the ensuing large deformations of the cylinder wall (Fig. 13 and lower graphic of Fig. 12), the reflection of the shocks in the solid at the constrained outer cylinder wall (Fig. 13), and the transmission of a high-intensity shock into the solid target (lower graphic of Fig. 13). In the last shown time step, the HMX is fully depleted and a non-reactive, purely hyrdodynamic, shock wave, caused by the reflection of the detonation wave at the target, can be seen to propagate upstream in the fluid.

7 Conclusions

A loosely coupled fluid-structure interaction method for the time-accurate simulation of solid materials responding dynamically to strong shock and detonation waves arising from the detonation of highly energetic materials has been presented. The approach utilizes a Lagrangian finite element solver for large deformations and a Cartesian dynamically adaptive finite volume solver, with the additional capability to deal with moving embedded boundaries via the ghost-fluid method. Both solvers have been parallelized for distributed memory machines via domain decomposition, and an effective corresponding inter-solver communication module has been outlined. An algorithm has been presented that transforms the Lagrangian surface mesh very efficiently into a signed distance function on the Cartesian mesh. The application of our approach to two distinct fluid-solid interaction problems has also been described.

We close by noting that the combined approach can lead to high efficiencies in the solution of coupled fluid-solid interaction problems. For instance the second example, in which a detonation wave in solid highly energetic material impinges on a dynamically deforming Tantalum cylinder, demonstrates the enormous savings in computational costs that can be obtained through structured dynamic mesh adaptation in the fluid for the considered problem class. This calculation required only 504 h CPU, whereas a simulation with an equivalent fluid unigrid mesh can be expected to be in the range of 10^5 h CPU.

Future development efforts will focus on the implementation of dynamic adaptation and mesh smoothing techniques for the solid solver, as well as investigations of the integration of implicit solvers for both the fluid and solid into the VTF.

References

1. M. Aivazis, W. A. Goddard, D. I. Meiron, M. Ortiz, J. Pool, J. Shepherd (2000). Comput. Science & Engineering **2(2)** 42–53.
2. J. C. Cummings, M. Aivazis, R. Samtaney, R. Radovitzky, S. P. Mauch, D. I. Meiron, J. Supercomput. **23** (2002) 39–50.
3. S. Mauch, D. Meiron, R. Radovitzky, and R. Samtaney, In K.J. Bathe, editor, *2nd M.I.T. Conference on Computational Fluid and Solid Mechanics*, Cambridge, MA, June 17-20 (2003).
4. M. Arienti, P. Hung, E. Morano, J. E. Shepherd. J. Comput. Phys. **185** (2003) 213–251.
5. F. Cirak and R. Radovitzky. In *Proceedings of the IUTAM Symposium on Integrated Modeling of Fully Coupled Fluid-Structure Interactions Using Analysis, Computations and Experiments*, New Brunswick, NJ, June 1-6 2003. International Union of Theoretical and Applied Mechanics.
6. R. Löhner, J. D. Baum, C. Charman, D. Pelessone, in *Lecture Notes in Computer Science 2565* (Springer, Berlin, 2003) 3–23.
7. F. Cirak and R. Radovitzky., Computers and Structures **83** (2005) 491–498.
8. R. Löhner, J. Cebral, C. Yang, J. D. Baum, E. Mestreau, C. Charman, D. Pelessone, Comput. Science & Engineering **6(3)** (2004) 27–37.
9. Howard H. Hu, N. A. Patankar, M. Y. Zhu, J. Comput. Phys. **169(2)** (2001) 427–462.
10. R. Mittal and G. Iaccarino, Annu. Rev. Fluid Mech. **37** (2005) 239–261.
11. J. J. Quirk, Computers Fluids **23** (1994) 125–142.
12. M. J. Berger and C. Helzel, in *Proc. 3rd Int. Symp. Finite Volumes for Complex Applications* (Porquerolles, 2002).

13. R. P. Fedkiw, J. Comput. Phys. **175** (2002) 200–224.
14. J. Bell, M. Berger, J. Saltzman, and M. Welcome, SIAM J. Sci. Comp. **15(1)** (1994) 127–138.
15. D. Tam, R. Radovitzky, R. Samtaney, Int. J. Numer. Meth. Engineering (2005), **in press**.
16. C. Pantano, R. Deiterding, D .J. Hill, D. I. Pullin. Submitted to J. Comp. Physics, Elsevier.
17. C. Pantano, R. Deiterding, D .J. Hill, D. I. Pullin. in Proc. Cyprus Int. Symp. on Complex Effects in Large Eddy Simulations, (Univ. Cyprus, Nicosia, 2005).
18. R. Deiterding, in *Lecture Notes in Computer Science 3726*, (Springer, Berlin, 2005) 916–927.
19. R. Deiterding, G. Bader, in G. Warnecke (ed.), *Analysis and Numerics of Conservation Laws*, (Springer, Berlin, 2005) 69–91.
20. R. Deiterding. in *Proc. 10th Int. Conf. on Hyperbolic Problems: Theory, Numerics, Applications*, (Yokohama Publishers, 2005), in press.
21. R. Deiterding, in K. Kremer, V. Macho, (eds.), *Forschung und wissenschaftliches Rechnen : Beiträge zum Heinz-Billing-Preis 2003*, (Ges. für Wiss. Datenverarbeitung, Göttingen, 2004) 63–77.
22. R. Deiterding, in *Proc. Fall Meeting Western States Section of the Combustion Institute*, (The Combustion Institute, Los Angeles, 2003).
23. R. Deiterding, *Parallel adaptive simulation of multi-dimensional detonation structures* (PhD thesis, Brandenburgische Technische Universität Cottbus, 2003).
24. R. Deiterding, in A. Handlikova et al. (eds.), *Proc. ALGORITMY 2002, 16th Conf. Scientific Computing*, (Dep. of Math. and Descriptive Geometry, Slovak Univ. Techn., Bratislava) 94–101.
25. W. Fickett, W. C. Davis, *Detonation*, (University of California Press, Berkeley and Los Angeles, 1979).
26. J. F. Clarke, S. Karni, J. .J. Quirk, P. L. Roe, L. G. Simmonds, E. .F. Toro, J. Comput. Phys. **106** (1993) 215–233.
27. E. F. Toro, *Riemann solvers and numerical methods for fluid dynamics* (Springer, Berlin, Heidelberg, 1999).
28. B. Einfeldt, C. D. Munz, P. L. Roe, and B. Sjögreen, J. Comput. Phys. **92** (1991) 273–295.
29. J. J. Quirk. Int. J. Numer. Meth. Fluids **18** (1994) 555–574.
30. R. Sanders, E. Morano, M.-C. Druguett, J. Comput. Phys. **145** (1998) 511–537.
31. R. P. Fedkiw, T. Aslam, B. Merriman, and S. Osher, J. Comput. Phys. **152** (1999) 457–492.
32. S. Osher and R. Fedkiw, *Level set methods and dynamic implicit surfaces*, (Springer, New York, 2003).
33. M. Berger and P. Colella, J. Comput. Phys. **82** (1988) 64–84.
34. R. Deiterding, AMROC - Blockstructured Adaptive Mesh Refinement in Object-oriented C++, *http://amroc.sourceforge.net* (2005).
35. R. Deiterding, in *Lecture Notes in Computational Science and Engineering 41* (Springer, New York, 2005) 361–372.
36. M. Parashar and J. C. Browne, in *Proc. of the 29th Annual Hawaii Int. Conf. on System Sciences* (1996).
37. J. E. Marsden, T. J. R. Hughes, *Mathematical foundations of elasticity*, (Dover Publications, 1993).
38. J. Lubliner, Int. J. Non-Linear Mechanics, **7** (1972) 237–254.
39. R. Radovitzky and M. Ortiz, Computer Methods in Applied Mechanics and Engineering, **172** (1999) 203–240.
40. M. Ortiz and L. Stainier. Computer Methods in Applied Mechanics and Engineering, **171** (1999) 419–444.
41. A. Lew, R. Radovitzky, and M. Ortiz, J. Comput-Aided Mater. Des. **8** (2002) 213–231.
42. J. Von Neumann, R. D. Richtmyer, J. Appl. Physics, **21** (1950) 232–237.
43. D. J. Benson, Computer Methods in Applied Mechanics and Engineering **99** (1992) 235–394.
44. L. Stainier, A. M. Cuitino, M. Ortiz, J. of the Mechanics and Physics of Solids **50** (2002) 1511-1545.
45. N. Kambouchev, J. Fernandez, R. Radovitzky, *http://arxiv.org/abs/cond-mat/0505178* (2005).
46. T. Belytschko, in T. Belytschko, T. J. R. Hughes (eds.), *Computational Methods for Transient Analysis*, (North-Holland, 1983) 1–65.
47. T. J. R. Hughes, in T. Belytschko, T. J. R. Hughes (eds.), *Computational Methods for Transient Analysis*, (North-Holland, 1983) 67–155 .
48. Kane C., Marsden J.E., Ortiz M., and West M., Int. J. Numer. Meth. Engineering **49** (2000) 1295–1325.
49. P. G. Ciarlet. Numerical analysis of the finite element method (Les Presses de L'Universite de Montreal, Quebec, 1976).
50. George Karypis, `http://www-users.cs.umn.edu/karypis/metis/` (2001).
51. A. Liu, B. Joe, Math. Comput. **65** (1996) 1183-1200.
52. L. Stainier, R. Radovitzky, and M. Ortiz. In E. Croitoro, editor, *Proceedings of the 1st Canadian Conference on Nonlinear Solid Mechanics*, pages 203–213, Victoria, BC, June 1999.
53. R. Radovitzky and A. Cuitiño, In *Proceedings of the 44th AIAA/ASME/ASCE/AHS Structures, Structural Dynamics, and Materials Conference*, Norfolk, VA, April 2003. American Institute of Aeronautics and Astronautics.
54. Z. Zhao, R. Radovitzky, S. Kuchnicki, and A. Cuitino. In *Proceedings of the International Symposium on Plasticity and its Current Applications*, Quebec City, Canada, July 2003.
55. Alberto Cuitiño Raúl Radovitzky, editor, Journal of Modelling and Simulation in Materials Science and Engineering (2004).
56. Z. Zhao, R. Radovitzky, and A. Cuitino, Acta Materialia, **52** (2004) 5791–5804.
57. S. N. Kuchnicki, A. M. Cuitino, and R. A. Radovitzky, Submitted to International Journal of Plasticity (2005).
58. J. A. Sethian, *Level Set Methods and Fast Marching Methods*, (Cambridge University Press, Cambridge, United Kingdom, 1999).
59. D. E. Johnson, and E. Cohen, *Proc. IEEE Intl. Conf. Robotics & Automation*, (Leuven, 1998) 3678–3684.
60. S. P. Mauch, *Efficient Algorithms for Solving Static Hamilton-Jacobi Equations* (PhD thesis, California Institute of Technology, 2003).
61. J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes, *Computer Graphics: Principles and Practice*, (Addison-Wesley, Reading, Massachusetts, 1996)
62. A. Watt, *3D Computer Graphics*, (Addison-Wesley, Reading, Massachusetts, 1993)

63. K. C. Park, C. A. Felippa, and J. A. Deruntz. In T. Belytschko and T. L. Geers, editors, *Computational Methods for Fluid-Structure Interaction Problems*, pages 94–124, New York, 1997.
64. Q. Zhang and T. Hisada. International Journal for Numerical Methods in Engineering **60** (2004) 2013–2029.
65. G. I. Taylor, Proc. of Royal Society **Series A** (1950) 201–1065.
66. H. L. Brode, J. Appl. Phys. **26(6)** (1955) 766–775.
67. R. E. Cohen, O. Gülseren, Physical Review B, **63** (2001) 3363.
68. A. M. Cuitiño and M. Ortiz, Engineering Computations, **9** (1991) 437–451.
69. M. Ortiz, R. A. Radovitzky, and E. A. Repetto, Int. J. Numer. Meth. Engineering, **52** (2001) 1431–1441.
70. D. J. Steinberg, S. G. Cochran, M. W. Guinan, J. Appl. Physics, **51** (1980) 1498–1504.
71. C. L. Mader, *Numerical modeling of detonations*, (University of California Press, Berkeley and Los Angeles, 1979).
72. G. H. Miller and P. Colella, J. Comput. Phys. **183** (2002) 26–82.