Engineering with Computers (2006) 22:325–347
DOI 10.1007/s00366-006-0043-9

ORIGINAL ARTICLE

A virtual test facility for the efficient simulation of solid material response under strong shock and detonation wave loading

Ralf Deiterding · Raul Radovitzky · Sean P. Mauch · Ludovic Noels · Julian C. Cummings · Daniel I. Meiron

Received: 19 April 2005 / Accepted: 1 February 2006 / Published online: 18 October 2006

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 research program 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 and treating the solid as an embedded moving body, while a Lagrangian finite element scheme is employed to describe the structural response to the hydrodynamic pressure loading. A temporal splitting method is applied to update the position and velocity of the boundary between time steps. The boundary is represented implicitly in the fluid solver with a level set function that is constructed on-the-fly from the unstructured solid surface mesh. Block-structured mesh adaptation with time step refinement in the fluid allows for the efficient consideration of disparate fluid and solid time scales. We detail the design of the employed object-oriented mesh refinement framework AMROC and outline its effective extension for fluid–structure interaction problems. Further, we describe the parallelization of the most important algorithmic components for distributed memory machines and discuss the applied partitioning strategies. As computational examples for typical VTF applications, we present the dynamic deformation of a tantalum cylinder due to the detonation of an interior solid explosive and the impact of an explosion-induced shock wave on a multi-material soft tissue body.

Ludovic Noels is a postdoctoral scholar at the FNRS.

R. Deiterding (✉) · S. P. Mauch · J. C. Cummings · D. I. Meiron
California Institute of Technology,
1200 East California Blvd., Pasadena,
CA 91125, USA
e-mail: ralf@cacr.caltech.edu

R. Radovitzky · L. Noels
Massachusetts Institute of Technology,
77 Massachusetts Ave., Cambridge, MA 02139, USA

L. Noels
University of Liège, 1 Chemin des Chevreuils,
4000 Liege, 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 of metallic solid targets forced by the loading 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 shock-capturing schemes.

In order to implement fluid–structure coupling in the VTF, we apply a loosely coupled, partitioned approach. The 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 intrin-

sically suited for simulation of phenomena such as shock propagation, detonation or fluid mixing in the fluid solver, while algorithms similarly optimized for phenomena such as high-rate plastic deformation or fracture can be employed 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 is reduced to communicating the velocities and the geometry of the solid surface to the Eulerian fluid and imposing the hydrostatic pressure onto the Lagrangian solid as a force acting on its exterior [1–7]. 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 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 somewhat 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 application 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 [4, 13]. 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 at the embedded boundary by using block-structured dynamic adaptation of the fluid mesh. 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.

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 basic, non-adaptive fluid–solid coupling algorithm used herein to thin, open structures has recently been presented in [14]. In Sect. 2, a Cartesian dynamically adaptive finite volume fluid solver for Euler equations with one-step chemistry is described. We detail the design of the underlying mesh refinement framework and discuss its parallelization. Section 3 outlines the parallel Lagrangian finite element solver for solid materials subjected to high-intensity shock loadings. In Sect. 4, we describe a highly efficient algorithm to transform a triangulated surface mesh into a signed distance function on a hierarchical Cartesian mesh as a prerequisite for coupling. The fluid–structure interaction methodology, highlighting particular its incorporation into the adaptive fluid mesh refinement framework, and an efficient inter-solver communication library are detailed in Sect. 5. Section 6 provides two three-dimensional computational examples. In Sect. 6.1, we simulate the impact event of a strong hydrodynamic shock wave on a body comprised of soft-tissue; in Sect. 6.2 the propagation of a detonation wave in HMX through a plastic tantalum cylinder is described. Both computations were run on distributed memory machines and we comment briefly on the overall computational efficiency of the 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 pre-

sentation 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 a large-eddy turbulence model [15] and for detonation simulation in thermally perfect gas mixtures with detailed chemical kinetics [16–20].

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 [21], which has also been used by Clarke et al. [22] 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

$$\partial_t \rho + \nabla \cdot (\rho \mathbf{u}) = 0, \tag{1}$$

$$\partial_t (\rho \mathbf{u}) + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u}) + \nabla p = 0, \tag{2}$$

$$\partial_t (\rho E) + \nabla \cdot ((\rho E + p)\mathbf{u}) = 0, \tag{3}$$

$$\partial_t \lambda + \mathbf{u} \cdot \nabla \lambda = \psi. \tag{4}$$

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

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

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

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

In Eq. 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 Eqs. 1–3 and 5 for $\psi \equiv 0$ and $q \equiv 0$, which is the appropriate model for purely hydrodynamic shock wave propagation (cf. Sect. 6.1).

2.2 Cartesian finite volume method with embedded boundaries

Following Clarke et al. [22], we apply the method of fractional steps to decouple the chemical reaction and hydrodynamic transport numerically. The *homogeneous* system of Eqs. 1–4 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 Eqs. 1–4 is a hyperbolic conservation law that admits discontinuous solutions (cf. [22]), 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 1–3 and is extended to a multi-dimensional Cartesian scheme via the method of fractional steps (cf. [23]). To circumvent the intrinsic problem of unphysical total densities and internal energies near vacuum due to the Roe linearization (cf. [24]), the scheme has the possibility to switch to the simple, but extremely robust Harten-Lax-Van Leer (HLL) Riemann solver [17–19]. The MUSCL-Hancock variable extrapolation technique of Van Leer [23] is employed 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 [10, 25]. 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 $\mathbf{n} = -\nabla\phi/|\nabla\phi|$ [26]. In coupled Eulerian–Lagrangian simulations, ϕ is updated on-the-fly by calling the closest-point-transform algorithm described in detail in Sect. 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 unavoidable staircase approximation of the boundary with this approach is alleviated 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 Eqs. 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 requires the mirroring of the primitive values ρ, $\mathbf{u}$, p, λ across the embedded boundary and the velocity modification

$\mathbf{u}' = 2((\mathbf{w} - \mathbf{u}) \cdot \mathbf{n})\mathbf{n} + \mathbf{u}$ within the ghost cells. We derive mirrored values in a ghost cell center $\mathbf{x}$ by calculating spatially interpolated values at the point $\mathbf{x} = \mathbf{x} + 2\phi\mathbf{n}$ from neighboring interior cells. For instance, in two spatial dimensions a bilinear interpolation between (usually) four adjacent cell values is employed, but directly near the boundary the number of cells contributing to the interpolation needs to be decreased 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 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 through the velocity modification in the internal ghost cells 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 after Berger and Colella [27], which is tailored especially for hyperbolic conservation laws on logically rectilinear finite volume grids. Instead of replacing single cells by finer ones, as 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 [28] 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 analogue 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 superscalar 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

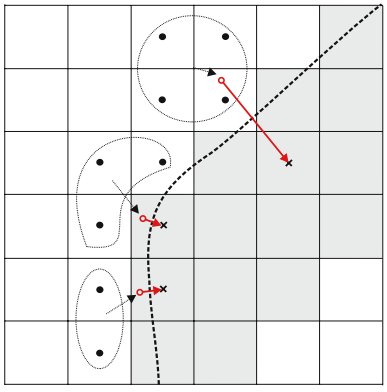

Fig. 1 Interpolation from interior cells to construct mirrored values to be used within internal ghost cells (*gray*)

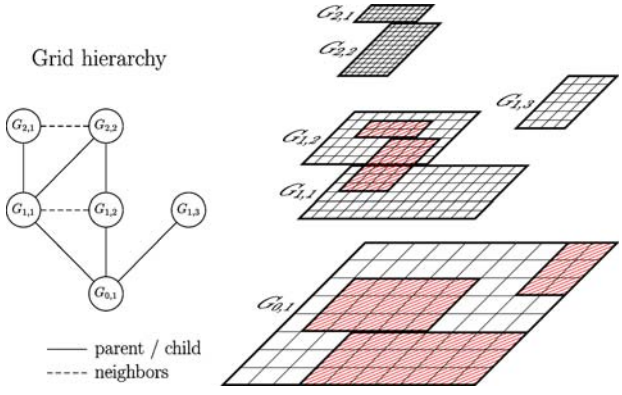

Fig. 2 The AMR method creates a hierarchy of rectangular subgrids

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 [27] or [19] for details).

2.4 Parallel implementation

Up to now, various reliable implementations of the SAMR method for single processor computers have been developed. Even the usage of parallel computers with shared memory is straightforward, because time-explicit methods allow the parallel calculation of the grid-wise numerical update [28]. But the question of an efficient parallelization strategy becomes more delicate for distributed memory architectures, because on such machines the costs for communication cannot be neglected. Due to the technical difficulties of implementing dynamical adaptive methods in distributed memory environments only a few parallelization strategies have been considered in practice to date (cf. [29, 30]).

Parallel SAMR in the VTF at the California Institute of Technology is provided generically by the AMROC (Adaptive Mesh Refinement in Object-oriented C++) framework [31]. AMROC has been parallelized very effectively for distributed memory machines [32] 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 SAMR algorithm uncovers that the only parallel operations under this paradigm are ghost cell synchronization, redistribution of the hierarchical data 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 [19]. Currently, we employ a generalization of Hilbert's space-filling curve [33] to derive load-balanced root level distributions at runtime. The entire SAMR hierarchy is considered by projecting the accumulated work from higher levels onto the root level cells. Although rigorous domain decomposition does not lead to a perfect balance of workload on single levels, good scale-up is usually achieved for moderate CPU counts. Figure 3 shows a representative scalability test for a three-dimensional spherical shock wave problem for the Euler equations of a single non-reactive gas (Eqs. 1–3 and 5). The test was run on the ASC Linux cluster (ALC) at Lawrence Livermore National Laboratories that connects Pentium-4 2.4 GHz dual processor nodes with a Quadrics Interconnect. The base grid has 32^3 cells and two additional levels with refinement factors 2 and 4. The adaptive calculation uses approximately 7.0 M cells in each time step instead of 16.8 M cells in the uniform case. The calculation on 256 CPUs employs between 1,500 and 1,700 subgrids on each level. Displayed are the average costs for each root level time step, which involves two time steps on the middle level and eight on the highest. All components of the dynamically adaptive algorithm, especially regridding and parallel redistribution, are active so that realistic results are obtained. Although we utilize a finite volume scheme in Fortran 77 within a C++ framework with full compiler optimization, the fraction of the time spent in this Fortran routine is 90.5% on four and decreases to only 74.9% on 16 CPUs. Hence, Fig. 3 shows a reasonable scale-up for at least up to 64 CPUs. We are currently researching scalability strategies that will allow full scale-up to thousands of processors.

2.5 Design of the AMROC framework

A salient feature of AMROC compared to other available SAMR implementations, e.g. [29, 34], is the realization of object-oriented framework concepts in C++ on all levels of software design. This allows for effective code re-use in implementing parallel SAMR algorithms and extensive capability for customization through subclass derivation.

In block-structured dynamically adaptive codes, three abstraction levels can be identified. At the top level, a particular physical simulation problem is formulated by providing a numerical scheme, by setting

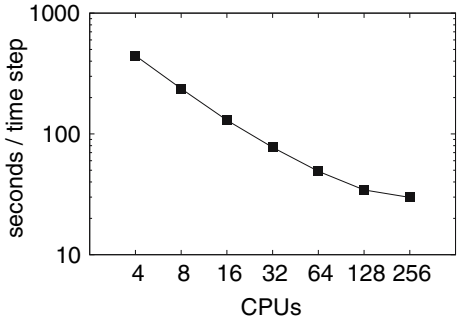

Fig. 3 Representative AMROC scale-up test for fixed problem size

boundary and initial conditions, and by specifying interpolation (prolongation) and averaging (restriction) methods for the inter-level transfer operations. Characteristic of block-structured methods is that at this level only single-patch routines need to be provided. In AMROC, SAMR implementation classes call the single-patch routines through abstract class interfaces. For a fully implemented SAMR algorithm, the system is used as an application framework invoked by a generic main program. Classes implementing SAMR algorithms and their auxiliary components operating on and manipulating complex hierarchical data make up the second level. A comparison of typical SAMR algorithms, e.g. the Berger–Colella technique for time-explicit finite volume schemes (cf. Sect. 2.3) with geometric adaptive multigrid methods for implicit discretizations, reveals that the SAMR auxiliary components show great similarity and can easily be reused. In AMROC, components such as the flagging of cells for refinement depending on various criteria, the clustering of flagged cells into rectangular regions, inter-level data transfer and flux correction (fixup) reside in clearly separated classes. This is highlighted in Fig. 4 which displays the most important AMROC classes and their relationships in unified modeling language (UML) notation [35] for the purely Cartesian case. The recursive Berger–Collela SAMR algorithm tailored for the hyperbolic problems of interest is realized here in the central class HypSAMRSolver; all others classes are generic, enabling the utilization of AMROC as a software framework for the efficient implementation of different SAMR algorithms typically implemented in new central SAMRSolver classes.

The intermediate AMROC design level naturally utilizes classes of the base level that provide hierarchical data structures. The base level is divided into elementary functionality for single grid patches and the implementation of various lists that store these patches hierarchically. A common design for the base level (see also [29]) involves a Box class to specify a single rectangular box in global integer index space. Methods for geometric operations on boxes like concatenation or intersection are available. A Patch class adds consecutive data storage to a Box. In AMROC, the geometrical description of all refinement areas is stored in hierarchical lists of Box objects inside a single GridHierarchy. The templatized class GridFunction creates Patch objects for various, possibly complex, data types following exactly the Box lists of GridHierarchy. As the refinement lists in GridHierarchy evolve and are dynamically distributed to an evolving set of processors, the Patch objects in

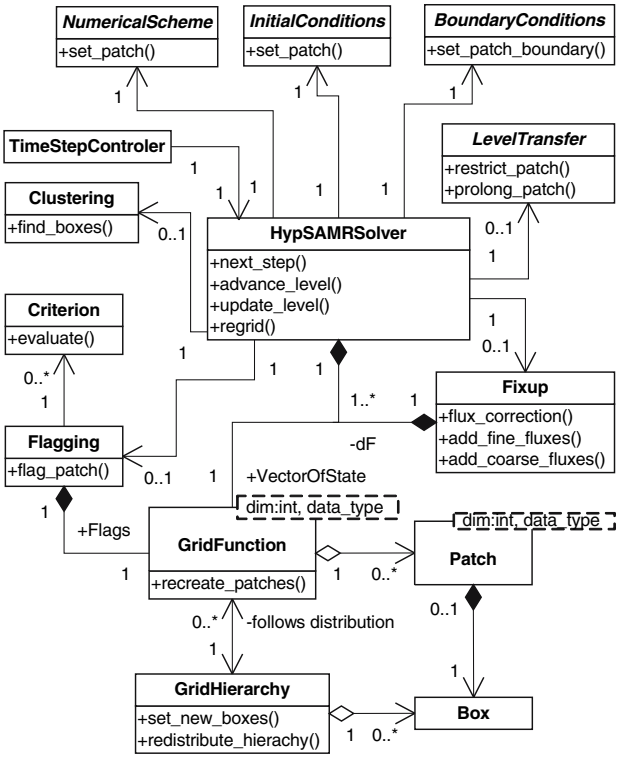

Fig. 4 UML class diagram for the most important AMROC components implementing Cartesian Berger-Collela-type SAMR

GridFunction are automatically re-created, including parallel redistribution and synchronization.

The design of the hierarchical data structures in AMROC follows the DAGH (Distributive Adaptive Grid Hierarchies) package by Parashar and Browne [30] that itself was intended as software framework for SAMR algorithms, but the enormous complexity in SAMR algorithms and their auxiliary components makes framework concepts at higher design levels (see above) more effective. As an illustration, Fig. 5 shows the most important classes that have been added to the originally Cartesian SAMR framework to implement arbitrary level-set-based embedded boundary methods like the one sketched in Sect. 2.2. An abstract class LevelSetEvaluation is provided to evaluate the scalar GridFunction ϕ patch-wise; EmbeddedBoundaryConditions allows the specification of the detailed boundary value modification. Multiple EmbeddedBoundaryMethods can also be considered and are incorporated with minimal implementation overhead into the existing algorithms of the SAMRSolver class for hyperbolic problems, HypSAMRSolver, through the derived class EBMHypSAMRSolver. The only operation that had to be extended was that of applying physical boundary conditions.

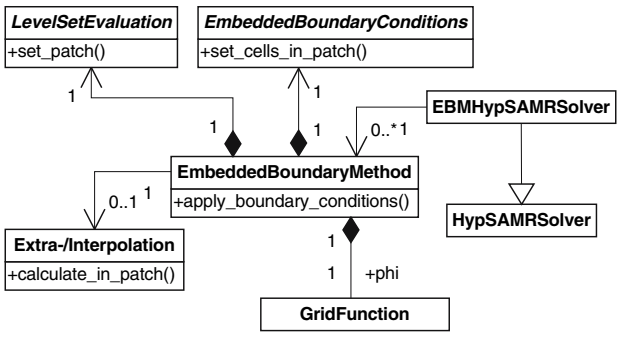

Fig. 5 Class structure extension of Fig. 4 for level-set-based embedded boundary methods

3 Lagrangian formulation of solid dynamics

We adopt a conventional Lagrangian formulation [36] 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.

3.1 Governing equations

We select the configuration $B_0 \subset \mathbb{R}^{\#}$ 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

$$\mathbf{x} = \varphi(\mathbf{X}, t) \tag{7}$$

(not to be confused with the levelset function). Thus, $\mathbf{x}$ is the position of material particle $\mathbf{X}$ at time t. The material velocity and acceleration fields follow from Eq. 7 as $\dot{\varphi}(\mathbf{X}, t)$ and $\ddot{\varphi}(\mathbf{X}, t)$, 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

$$\mathbf{F} = \nabla_0 \varphi(\mathbf{X}, t), \tag{8}$$

where ∇_0 denotes the material gradient of a function defined over B_0. The scalar function

$$J = \det\left(\mathbf{F}(\mathbf{X}, t)\right) \tag{9}$$

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 linear momentum (cf. [36]), which takes the form

$$\nabla_0 \cdot \mathbf{P} + \rho_0 \mathbf{B} = \rho_0 \ddot{\varphi}, \tag{10}$$

where $\mathbf{B}(\mathbf{X}, t)$ is the body force per unit mass, $\mathbf{P}(\mathbf{X},t)$ denotes the first Piola–Kirchhoff stress tensor and $\rho_0(\mathbf{X})$ the mass density over B_0. The symmetric Cauchy stress tensor follows from $\mathbf{P}$ through the relation

$$\sigma = J^{-1}\mathbf{P}\mathbf{F}^T. \tag{11}$$

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

$$\varphi = \bar{\varphi} \text{ on } \partial B_{01}, \tag{12}$$

$$\mathbf{P} \cdot \mathbf{N} = \bar{\mathbf{T}} \text{ on } \partial B_{02}, \tag{13}$$

where $\bar{\varphi}(\mathbf{X}, t)$ is the prescribed deformation mapping on ∂B_{01}, $\mathbf{N}$ 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 $\varphi_0(\mathbf{X})$ and $\dot{\varphi}_0(\mathbf{X})$ to be specified over B_0.

3.2 Shock-capturing material model

Assuming the material behavior is reversible, the stress tensor can be computed from an internal energy function U as

$$\mathbf{P} = \frac{\partial U\left(\mathbf{F}^T\mathbf{F}\right)}{\partial \mathbf{F}}. \tag{14}$$

This model is well suited for low-intensity loading (see Sect. 6.1), but when the solid interacts with high-intensity fluid shocks, a strong stress wave propagates within the solid (cf. Sect. 6.2). The inelastic material behavior then has to be accounted for and a shock-capturing numerical scheme is also needed to accurately represent evolving stress fronts inside the solid. Here, we apply a Lagrangian artificial viscosity method for solids with strength presented in [37]. The addition of artificial viscosity has the intended effect of spreading fronts that are too steep to be resolved accurately over several grid points without altering their speed of propagation while avoiding the introduction of numerical artifacts such as spurious oscillations (cf. [38]).

A general constitutive theory of inelastic material behavior may be based on irreversible continuum thermodynamics (cf. [39–41] for more extensive accounts). In this context, viscosity may be modeled by assuming an additive decomposition

$$\mathbf{P} = \mathbf{P}^e + \mathbf{P}^v \tag{15}$$

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, we assume that the free energy per unit of undeformed volume can be expressed as a function $A(\mathbf{F}, T, \mathbf{q})$. As in the reversible case expressed in Eq. 14, the equilibrium stresses then follow from the free energy as

$$\mathbf{P}^e = A_{,\mathbf{F}}(\mathbf{F}, T, \mathbf{q}). \tag{16}$$

In materials without strength, A depends on the deformation only through the Jacobian J and the hydrostatic pressure p is computed by

$$p = -A_{,J}\,. \tag{17}$$

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.

In addition to a volumetric equation of state, a complete characterization of the behavior of the solid requires a description of its strength, including its elasticity, yield point, strain hardening, rate sensitivity and temperature dependence. This is accomplished by the specification of the internal energy density A, plus suitable kinetic equations for the internal state variables $\mathbf{q}$. Owing to the high strain rate under consideration, only adiabatic material models are considered. In addition, the material is assumed to obey the 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: $\mathbf{F} = \mathbf{F}^e\,\mathbf{F}^p$, where $\mathbf{F}^p$ is assumed to be volume-preserving. The equilibrium stress–strain relation 17 is now extended to include the elasticity of the material in shear, with the result [41, 42]

$$\sigma^e = -p\mathbf{I} + J^{-1}\mathbf{F}^e\left\{\mu(\log\sqrt{\mathbf{C}^e})^{\mathrm{dev}}\right\}\mathbf{F}^{eT}, \tag{18}$$

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 and μ is a shear modulus. The plastic deformation $\mathbf{F}^p$ is assumed to obey the Prandtl–Reuss flow rule

$$\dot{\mathbf{F}}^p\mathbf{F}^{p-1} = \dot{\epsilon}^p\frac{3\mathbf{s}^e}{2\bar{\sigma}}, \tag{19}$$

where $\mathbf{s}^e = \sigma^{e,\mathrm{dev}}$ is the stress deviator, $\bar{\sigma} = \sqrt{(3/2)s^e_{ij}s^e_{ij}}$ denotes the von Mises stress, and ε^p is the effective plastic strain. Relation 19 is solved by considering a hardening law

$$\bar{\sigma} = \bar{\sigma}(\epsilon^p, \dot{\epsilon}^p, J) \tag{20}$$

that characterizes the material behavior. The VTF possesses a large set of constitutive models and algorithmic updates that describe a wide range of material response, including variational updates for isotropic and crystal plasticity [41], a multiscale model of single-crystal b.c.c. tantalum [43] and a polyconvex model for anisotropic cubic crystals [44].

The viscous part of the stresses (Eq. 15) is assumed to take the form

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

where $\sigma^v = 2\eta_h(\mathrm{sym}\,\dot{\mathbf{F}}\mathbf{F}^{-1})^{\mathrm{dev}}$ and η_h denotes the viscosity coefficient. In the latter, sym and dev denote the symmetric and deviatoric components of a tensor, respectively. In order to construct an artificial-viscosity-based scheme, we assume that the viscosity coefficient is comprised of two terms, leading to

$$\eta_h = \eta + \Delta\eta, \tag{22}$$

where η is the physical viscosity coefficient of the material and $\Delta\,\eta$ the added artificial viscosity. We refer to [37, 38, 45] for details about the evaluation of $\Delta\,\eta$.

3.3 Finite-element temporal integration

The preceding continuum formulation is rendered into a form suitable for computation by combining a time discretization of the momentum and constitutive equations with a finite element discretization for the reference configuration of the solid. More detailed accounts may be found in [40, 41]. We consider finite element interpolations of the form

$$\varphi(\mathbf{X}) = \sum_{a=1}^{N}\mathbf{x}_a N_a(\mathbf{X}), \tag{23}$$

where $\mathbf{x}_a$ is the current position at node a and N_a are the shape functions. The sum on a ranges over the N

nodes of the mesh. The displacement shape functions N_a must be conforming. In calculations, we employ standard quadratic, six-noded triangles or ten-noded tetrahedra (e.g. [46]). Utilizing these definitions, the weak formulation of the linear-momentum balance equation 10 is written as

$$\mathbf{f}_a^{\text{inert}} = \mathbf{f}_a^{\text{ext}} - \mathbf{f}_a^{\text{int}} \quad \text{with} \tag{24}$$

$$\mathbf{f}_a^{\text{inert}} = \int_{B_0} \rho_0 N_a N_b \mathrm{d}\Omega \ddot{\mathbf{x}}_b \simeq \mathbf{M}_{ab}\ddot{\mathbf{x}}_b, \tag{25}$$

$$\mathbf{f}_a^{\text{int}} = \int_{B_0} \mathbf{P} : \nabla_0 N_a \mathrm{d}\Omega \quad \text{and} \tag{26}$$

$$\mathbf{f}_a^{\text{ext}} = \int_{\partial B_{02}} \bar{\mathbf{T}} N_a \mathrm{d}S + \int_{B_0} \rho_0 \mathbf{B}_{n+1} N_a \mathrm{d}\Omega. \tag{27}$$

In these equations, $\mathbf{M}$ is the diagonal lumped mass matrix and $\mathbf{f}^{\text{inert}}$, $\mathbf{f}^{\text{int}}$ and $\mathbf{f}^{\text{ext}}$ are the inertial, internal and external forces, respectively.

Here and subsequently, we implement an incremental solution procedure aimed at sampling the solution at discrete times t_0, t_1,...,t_n, where $t_{n+1} = t_n + \triangle t$. The integration in time is performed with the Newmark family of time-stepping algorithms:

$$\mathbf{x}_a^{n+1} = \mathbf{x}_a^n + \triangle t \dot{\mathbf{x}}_a^n + \triangle t^2 \left[\left(\frac{1}{2} - \beta \right) \ddot{\mathbf{x}}_a^n + \beta \ddot{\mathbf{x}}_a^{n+1} \right], \tag{28}$$

$$\dot{\mathbf{x}}_a^{n+1} = \dot{\mathbf{x}}_a^n + \triangle t[(1-\gamma)\ddot{\mathbf{x}}_a^n + \gamma \ddot{\mathbf{x}}_a^{n+1}], \tag{29}$$

$$\ddot{\mathbf{x}}_a^{n+1} = \mathbf{M}_{ab}^{-1} \left[\mathbf{f}^{\text{ext}} - \mathbf{f}^{\text{int}} \right]_b^{n+1}, \tag{30}$$

where β and γ are Newmark parameters. The performance of the Newmark algorithm, including its range of stability, has been extensively documented in the literature (e.g. [47–49]). Our particular case of $\beta = 0, \gamma = \frac{1}{2}$ is explicit and second-order accurate and leads to a central difference scheme.

3.4 Parallel implementation

Our finite element solid solver "Adlib" is implemented in C following a modular concept that encapsulates similar functionality in libraries. The main libraries are

- The material library, which provides the material response to a given deformation history for a set of complex constitutive material models.
- The mesh manager, which is responsible for the construction of the elements, connectivity tables and for the attribution of the properties to the elements.
- The mechanics library, which manages the space and time integration of the finite element discretization.
- The partitioning library, which is responsible for distributing the mesh to the processors of a parallelized application.

In Adlib, different types of applications are realized by providing a specific main program that calls the required library functionality. Problem-specific front-end routines, e.g. for boundary conditions, are linked to enable detailed customization. Although more classical than the concepts discussed in Sect. 2.5, we achieve effective code re-use in the same application domain. For instance, the extensive VTF material library is shared among Adlib and the VTF thin-shell finite element solver with fracture capability (see [50] for a computational example).

In the following, we detail the Adlib parallelization library. The parallelization strategy employed in Adlib is straightforward domain decomposition. The mesh distribution is based on heuristic graph partitioning as provided by the well-established software package METIS [51]. Optionally, the local mesh can be recursively refined using a mesh subdivision algorithm for tetrahedral meshes proposed by Liu and Joe [52]. In every refinement iteration, each tetrahedron is subdivided into eight new ones. For illustration purposes, this algorithm is applied to the mesh for a beam with about 10,000 elements shown in Fig. 6 that is distributed to 16 processors. On each processor, the mesh is subdivided with one level of refinement leading to the configuration displayed in Fig. 7. The main advantage of this algorithm is that only two new classes of slightly less regular tetrahedra are introduced, independent of the number of refinement iterations, thus enabling the creation of distributed meshes in a scalable way. The uniqueness of the algorithm ensures conforming elements in every level of subdivision. Note that the application of the subdivision procedure mandates the reconstruction of the interprocessor communication maps.

For parallel fluid–structure interaction simulations, all the above libraries are linked to the coupled code. A solid update step begins as usual by applying boundary conditions (here tractions produced by the hydrodynamic pressure), and proceeds 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 operation is necessary to exchange the incomplete residuals at partition boundary nodes. After this step, the corrected nodal accelerations and velocities can be obtained locally by

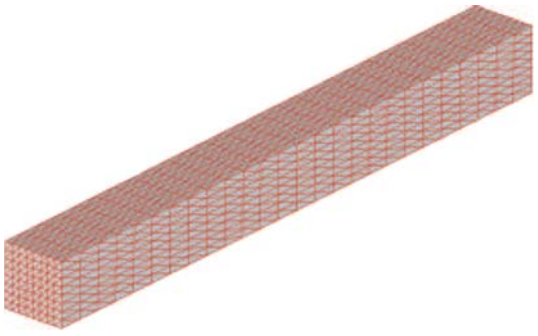

Fig. 6 Initial tetrahedral mesh for a beam (≈10,000 elements)

Fig. 7 Domain decomposition (indicated by *color*) to 16 processors with subdivision of one level for the mesh shown in Fig. 6

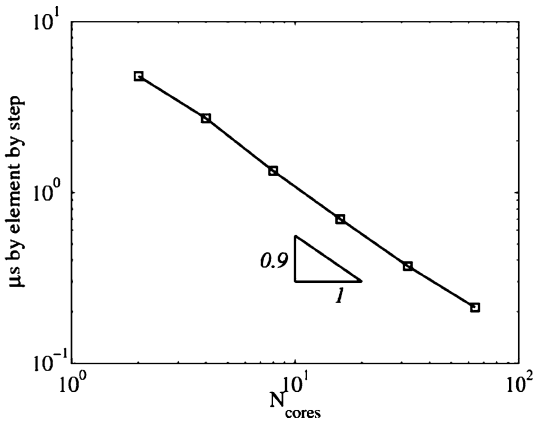

Fig. 8 Scalability test for the solid solver

applying the unmodified corrector step. As the communication operations involve only partition boundary data, and retain a surface to volume character, the solid update algorithm shows excellent scalability properties.

In order to illustrate this fact, a study of strong scaling has been performed for a tensile test utilizing the mesh depicted in Fig. 6 (one level of subdivision). The test was run on a cluster of AMD Opteron-2.2 GHz quad-core nodes connected with an Infiniband network. The average compute time per element per step is shown in Fig. 8 and we measure a constant parallel efficiency of approximately 0.9. The efficiency of the method has also been confirmed in several large-scale computations related to the dynamic response of polycrystalline materials [53–58] that have only been made possible by employing the massively parallel systems provided by the DoE ASC program.

4 Closest-point-transform algorithm

In Sect. 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. Sect. 3), the interface is a triangle 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, $\phi(\mathbf{x}), \mathbf{x} \in \mathbb{R}^{\varkappa}$, 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 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

$$|\nabla\phi| = 1 \tag{31}$$

with boundary conditions $\phi\big|_{\mathcal{I}} = 0$ (see [59]). For most boundary conditions, a solution to Eq. 31 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 Eq. 31 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 triangle 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 as 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 the solid.[1]

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 [60]. 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 [61]. For a given grid point, the closest point on the triangle 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.

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

[1] Flood filling means looping over the grid points while only keeping track of the sign of the distance.

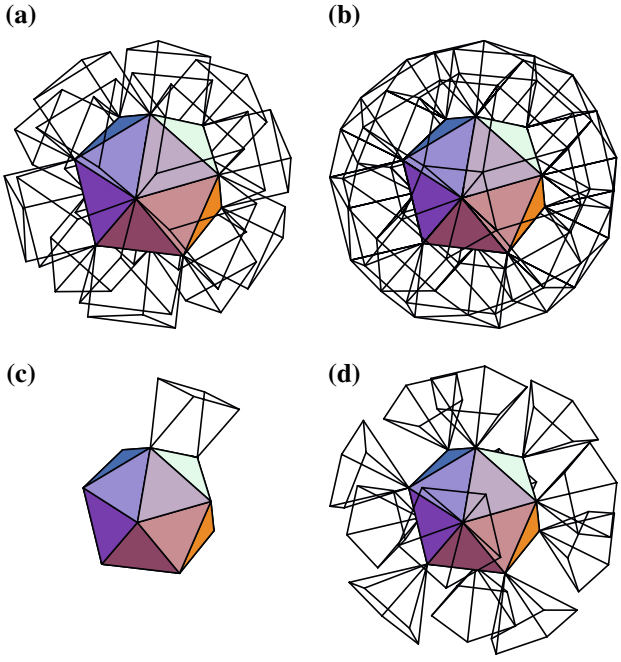

Fig. 9 The characteristic polyhedra for faces, edges, and vertices

from the face (see Fig. 9a 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 Fig. 9b. A single edge polyhedron is shown in Fig. 9c. 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 Fig. 9d.

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 [62, 63]. It is a method for determining the pixels on the display which lie inside a polygon. Figure 10 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:

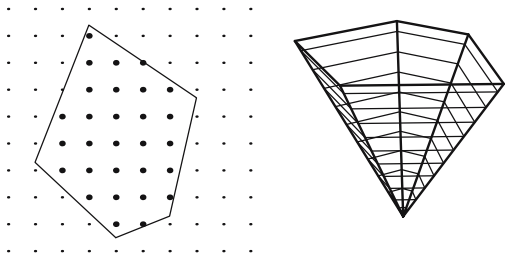

Fig. 10 Scan conversion of a polygon in 2-D and slicing of a polyhedron to form polygons

```
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 CPT 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 v 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 v depends on the shape of the surface and the distance δ. If the surface is jagged and δ is relatively large, then v will be large. If the surface is smooth and δ is relatively small, then v will be close to unity. The total computational complexity of the algorithm is $\mathcal{O}(vN + M)$. The $\mathcal{O}(vN)$ 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 triangle 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 Sect. 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 Sect. 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 Sect. 3, [14]—to the continuity of the normal component of the velocity field:

$$[\![\mathbf{u}\cdot\mathbf{n}]\!] = 0 \ \text{ on } \ \mathcal{I} \tag{32}$$

and the continuity of the normal component of the traction across the fluid–solid interface, i.e.

$$[\![\mathbf{t}\cdot\mathbf{n}]\!] = \left[\!\left[\sigma_{ij}n_i n_j\right]\!\right] = [\![\sigma_n]\!] = 0 \ \text{ on } \ \mathcal{I}. \tag{33}$$

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

$$\mathbf{u}^S(\mathbf{x},t) = \left(\dot{\varphi}\circ\varphi^{-1}\right)(\mathbf{x},t), \tag{34}$$

where $\mathbf{t}$ 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 Eq. 11. The resulting boundary conditions at the fluid solid interface are simply

$$u_n^S \;=\; u_n^F\sigma_{nn}^S \;=\; p^F\big|_{\mathcal{I}} \tag{35}$$

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 fluid and solid solver [2][2]:

$$\begin{aligned}
&u_n^F := u_n^S(t)|_{\mathcal{I}}\\
&\mathsf{update_fluid}(\,\Delta t\,)\\
&\sigma_{nn}^S := p^F(t+\Delta t)|_{\mathcal{I}}\\
&\mathsf{update_solid}(\,\Delta t\,)\\
&t := t+\Delta t
\end{aligned}$$

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

[2] More general implicit and staggering schemes for coupled systems have been proposed and studied in detail in [64, 65].

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 Sect. 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 Sect. 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 Sect. 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 in cells of the current level 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 $R_l = \prod_{i=0}^{l} r_l$ (cf. Sect. 2.3). The root level time step $\Delta\tau_F$ 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. 11 for an exemplary SAMR hierarchy with two additional levels with $r_1 = r_2 = 2$ and $K = 4$ sub-iterations in solid_step(). As in the simulations in the Sects. 6.1 and 6.2, the coupling level $l_c = 1$ is not the maximal level of refinement. Figure 11 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 ad-

vance_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. 11 by the gray arrows; the initiating base level that remains fixed throughout the regridding operation is indicated by gray circles.

The incorporation of the algorithms described above into the AMROC framework is relatively straightforward. Utilizing the design for general embedded boundary methods sketched in Sect. 2.5, we have implemented fluid_step() and the fluid–structure coupled version of advance_level() in a class CoupledHypSAMRSolver derived from EBMHypSAMRSolver (cf. Fig. 12). CoupledHypSAMRSolver interpolates the pressure values $p^F|_{\mathcal{I}}$ along the surface mesh and communicates them to the CoupledSolidSolver through the coupling module InterSolverCommunication (see next section for details). CoupledHypSAMRSolver receives an updated interface mesh $\mathcal{I}$ that it passes to the ClosestPointTransform which is naturally made available as a concrete class based on LevelSetEvaluation. Further, CoupledHypSAMRSolver receives updated interface velocities $\mathbf{u}^S|_{\mathcal{I}}$ to be used in EmbeddedMovingWalls as the necessary concretization of EmbeddedBoundaryConditions. In order to re-use our standard TimeStepControler we have incorporated CoupledHypSAMRSolver and CoupledSolidSolver as attributes into a single CoupledSolver that encapsulates the extended method.

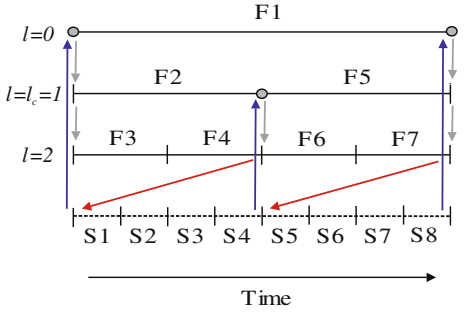

Fig. 11 Data exchange between the recursive fluid SAMR solver and the linear solid solver throughout one root level time step. *Red* and *blue arrows*: flow of interface data from fluid to solid and vice-versa. *Gray arrows*: regridding of higher SAMR levels, the base level (*gray circles*) stays fixed

5.2 Efficient inter-solver communication

Critical to 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 InterSolverCommunication as a dedicated asynchronous communication module 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 triangle surface mesh (cf. Sect. 3.4). A simple approach to coupling is to assemble the global boundary of the solid (i.e. gather the pieces from each process and merge them into a single mesh) and broadcast it to all fluid processes. Each fluid process then uses the relevant portion of the global boundary for the CPT and supplies pressure information for a portion of the boundary. Next, the pressure information is merged and broadcast to each solid process. Unfortunately, this simple strategy is not efficient for large solid meshes owing to the costs of assembling, storing, and communicating the global boundary and global pressure data structures.

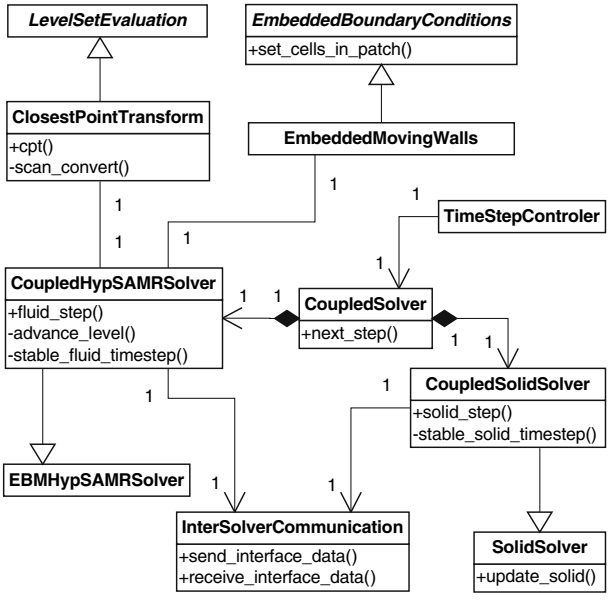

Fig. 12 Class structure of the fluid-structure coupling method realized as a concrete embedded boundary method in AMROC, see Fig. 5 for base classes

A better approach is to use point-to-point communications between the solid and fluid processes. Each solid process sends its portion of the boundary only to those fluid processes which require it on order to compute the CPT. Consider a single fluid process whose grids lie in a domain Ω and suppose that the CPT will be computed to a distance of δ. Then the fluid process needs only those portions of the interface that are within a distance δ of Ω. If the fluid process receives only the relevant portions of the interface, it can assemble them into a local triangle 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. Likewise, each fluid process constructs a Cartesian bounding box around its domain and enlarges it by δ. The solid bounding boxes define which portion of the interface the solid process has; the fluid bounding boxes define which portion of the interface the fluid process needs. These bounding boxes are gathered to 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 communication and computation.

This strategy is much more efficient than assembling the global boundary. It is far less costly to gather and broadcast bounding boxes than to gather, assemble, and broadcast the boundary itself. Intersecting the bounding boxes to determine the communication pattern is also inexpensive. Note that this pattern is computed anew for each inter-solver communication and will in general change as the simulation progresses.

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.

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 Eqs. 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 1–2 m and lead to strong impulses of 1–2 ms duration. Under such conditions, conventional ballistic protective armor has proven 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. 13). 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 $p_a = 100$ kPa and the temperature is $T_a = 293$ K. The ideal gas relation $p = \rho RT$ then yields an ambient density of $\rho_a = 1.212$ 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 $E_i = 2{,}260$ kJ/kg, which, for simplicity, is assumed to be uniformly distributed over a small sphere of radius 5 cm with its gas initially at rest. The initial temperature in this sphere is assumed to be $T_i = 1{,}465$ K. Using Eq. 5 and the ideal gas relation together we evaluate pressure and density within the small sphere to be $p_i \approx 1{,}700$ kPa and $\rho_i \approx 4{,}122$ kg/m^3. These initial condi-

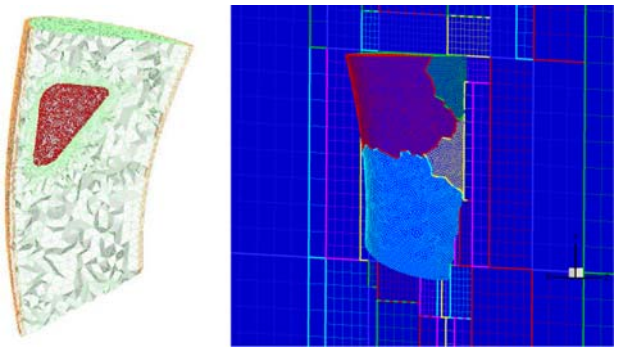

Fig. 13 Mesh of the liver inside the torso (*left*). Domain decomposition of the solid mesh and fluid mesh adaptation at the boundary (*right*)

tions result in the formation of a blast wave in the fluid (cf. Fig. 14).

The solid mesh used for this simulation has 19,562 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. 13). The fluid domain is 5 m × 5 m × 5 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.[3] This setting is sufficient to allow the construction of two internal ghost cells according to Sect. 2.2. With target CFL conditions 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 16,800 update steps in the solid solver. The simulation is completed in about 10 h real time on ten dual-processor 2 GHz G5 nodes of a Linux Beowulf cluster connected with Myrinet. 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 [66]:

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

Table 1 Material properties of soft tissues

	Liver	Torso
Young modulus	5 MPa	100 MPa
Poisson coefficient	0.3	0.3
Density	1,500 kg/m^3	950 kg/m^3

$$\Delta p_s = K_1 \frac{E_i}{d^3}, \quad t = K_2 \sqrt{\frac{d^5 \rho_a}{E_i}}. \tag{36}$$

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 values have been given by Brode who conducted extensive numerical investigations and summarized the results in the relations [67]

$$\begin{aligned} \Delta p_s[\text{bar}] &= \frac{6.7}{z^3} + 1 \text{ bar}, \\ &\quad \text{if } \Delta p_s > 10 p_a, \\ \Delta p_s[\text{bar}] &= \frac{0.975}{z} + \frac{1.455}{z^2} + \frac{5.85}{z^3} - 0.019 \text{ bar}, \\ &\quad \text{if } 0.1 p_a < \Delta p_s < 10 p_a, \end{aligned} \tag{37}$$

where all pressures are in bar and $z = d/W^{\frac{1}{3}}$ in mkg$^{-1/3}$ is the distance scaled with the charge mass W expressed in kilogram of equivalent TNT. In our simulation, we set W to the value 0.5. Figures 14, 15, 16 and 17 show the simulation results. The shock wave reaches the torso at $t = 1.55$ ms (see Fig. 16). This value is in reasonable agreement with the prediction from Eq. 36 considering the fact that Eq. 36 is derived for an ideal spherical shock wave emanating from a point source. At a distance $d = 1.5$ m, the overpressure peak according to Eq. 36 is $\Delta p_s \approx 104$ kPa. Our computation gives a value $\Delta p_s \approx 175$ kPa, which is close to Brode's approximation (Eq. 37) with $\Delta p_s \approx 177$ kPa at $z = 1.89$. It leads to an overall pressure peak of $p_s = p_a + \Delta p_s \approx 275$ kPa. An estimate of the reflected overpressure Δp_r can be calculated from the standard Rankine–Hugoniot relations for the Euler equations (cf. [23]). For the normal reflection of a planar shock on a rigid 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 formular

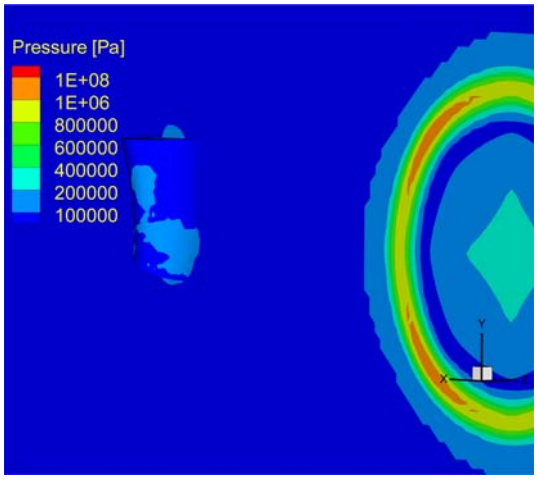

Fig. 14 Formation and propagation of the blast wave (time = 0.31 ms)

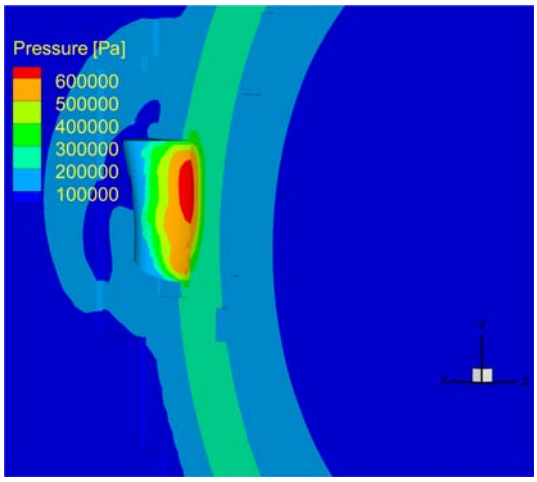

Fig. 16 Interaction of the shock wave with the body (time = 1.55 ms)

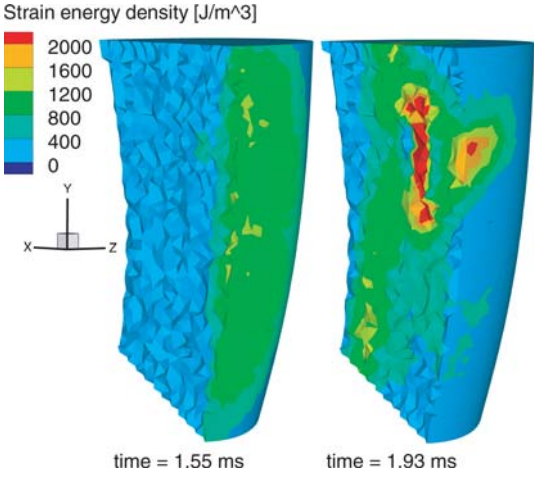

Fig. 15 Stress waves in the torso after the impact

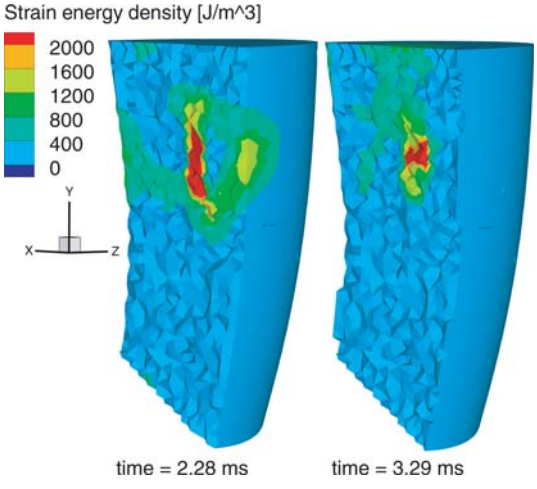

Fig. 17 Stress wave reflection in the liver

$$\Delta p_r \simeq 2\Delta p_s \frac{7p_a + 4\Delta p_s}{7p_a + \Delta p_s} \in [2\Delta p_s; 8\Delta p_s], \tag{38}$$

which yields $\Delta p_r \approx 560$ kPa for $\Delta p_s \approx 175$ kPa. The absolute reflected pressures obtained in the simulation are shown in Fig. 16. The peak values are in the order of 600 kPa, which is lower than the value $p_r = p_a + \Delta p_r \approx 660$ kPa obtained from Eq. 38, 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. Figures 15, 16 and 17 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 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 (Eq. 17) of tantalum is modeled by recourse to Vinet's EOS as fitted to first-principle calculations by Cohen et al. [68]. The deviatoric part of Eq. 18 is computed by considering a plastic flow (Eq. 20) of the form of a power-law rate-sensitivity, hardening and Steinberg–Guinan [69] pressure dependence

$$\bar{\sigma}(\epsilon^p, \dot{\epsilon}^p, J) = \frac{\mu(J)}{\mu_0}\sigma_y\left(1+\frac{\epsilon^p}{\epsilon_0^p}\right)^{\frac{1}{n}}\left(1+\frac{\dot{\epsilon}^p}{\dot{\epsilon}_0^p}\right)^{\frac{1}{m}} \tag{39}$$

in which ε_0^p denotes a reference plastic strain, $\dot{\epsilon}_0^p$ a reference plastic strain rate, m the rate sensitivity exponent, n the hardening exponent, σ_y the initial yield stress, and μ_0 the shear modulus at zero pressure. The material parameters used in the calculation are collected in Table 2. The dependence of the shear modulus of tantalum on pressure has been computed by Cohen from first principles [68]. We neglect the temperature effect (which is small according to [68]), the anisotropy of the crystal and assume isotropic elastic behavior in terms of a shear modulus. The material behavior is assumed to be adiabatic.

The fluid part of this simulation uses the entire set of Eqs. 1–5 and in particular the reaction term (Eq. 6). The cylinder has length 100 mm and an outer radius of 18.5 mm. An inner detonation chamber filled with HMX with radius 8.5 mm and depth 55 mm 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 $x = 10$ mm. 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. 18). No deformations are allowed in the entire solid for $x < 10$ mm 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 10 mm $\le x \le$ 30 mm.

According to Mader [70], unreacted HMX has a density of $\rho_0 = 1{,}900$ kg/m^3, and we assume atmospheric pressure $p_0 = 100$ kPa in the unreacted material. The detonation velocity for a freely propagating Chapman–Jouguet detonation (cf. [21]) in HMX is experimentally known to be approximately 9,100 m/s and the entire hydrodynamic flow can be described with reasonable accuracy with a constant adiabatic exponent of $\gamma = 3$ [70]. The rate factor T_R is unknown; we therefore set it to $T_R = 1$ μs, which is a reasonable value for most solid explosives [21].

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

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 combustion toward chemical equilibrium. The simplified Chapman–Jouguet theory can be used to evaluate the energy release of our configuration to be $q = 5{,}176$ kJ/kg and to predict the hydrodynamic values in the equilibrium state $p_{\mathrm{CJ}} \approx 39.3$ GPa and $\rho_{\mathrm{CJ}} \approx 2{,}533$ 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. 1–6. Detailed derivations of the ZND solution can be found in the book by Fickett and Davis [21] or, for instance, in [32]. According to the ZND solution, the peak values at the head of the detonation are $p_{\mathrm{vN}} \approx 78.7$ GPa and $\rho_{\mathrm{vN}} \approx 3{,}800$ kg/m^3. We use the analytic ZND solution as our hydrodynamic initial conditions. Figure 18 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 makes the application of very effective dynamic mesh adaptation possible.

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 of pressure and mass fraction λ). Its effective resolution corresponds to the uncoupled one-dimensional simulation shown in Fig. 18. To allow for large deformations of the cylinder walls, the fluid domain spans 30 mm × 30 mm × 60 mm, 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 the Figs. 19, 20 and 21 uses a solid mesh of 56,080 elements. With target CFL conditions 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 backward through the fully reacted material. The final time is set to 5.8 μs, 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 3,200 solid update steps. The distance parameter δ is chosen as in Sect. 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 1,000. Compared with a uniform fluid mesh of $480 \times 480 \times 960 \simeq 221$ M cells, that would otherwise be necessary to capture the detonation with similar accuracy, mesh adaptation clearly provides enormous savings. Figure 21 shows the highly localized fluid mesh refinement in the midplane for the second snapshot shown in Fig. 19.

The simulation ran on four 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 the Figs. 19 and 20. The graphics 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. 19), the ensuing large

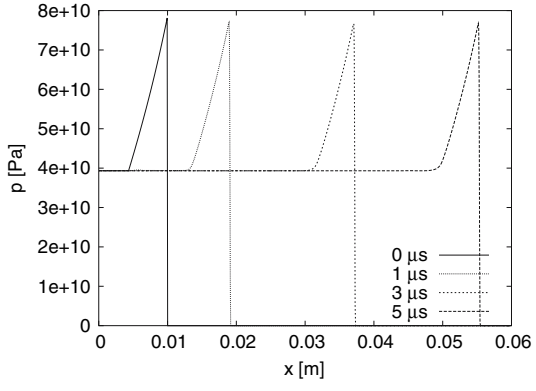

Fig. 18 Pressure distributions of the detonation wave in HMX in the inner detonation chamber from a purely hydrodynamic one-dimensional simulation

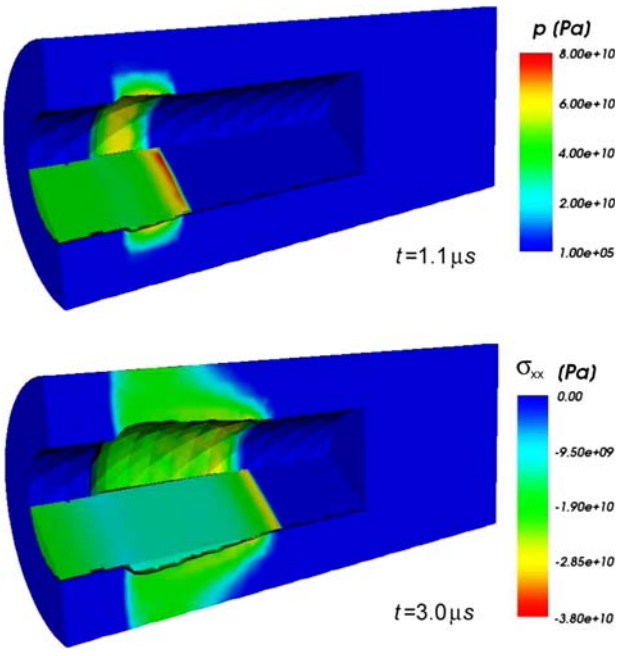

Fig. 19 Initiation of stress waves in the solid and compression of the wall material next to the detonation chamber due to the detonation passage

deformations of the cylinder wall (Fig. 20 and lower graphic of Fig. 19), the reflection of the shocks in the solid at the constrained outer cylinder wall (Fig. 20), and the transmission of a high-intensity shock into the

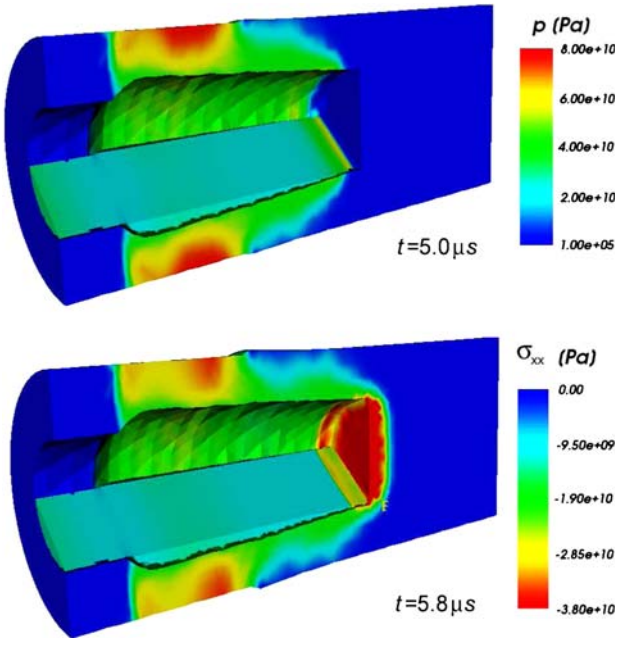

Fig. 20 Strong material compression in constrained and outward movement of unconstrained walls and the strong compression in axial direction due to the impact event

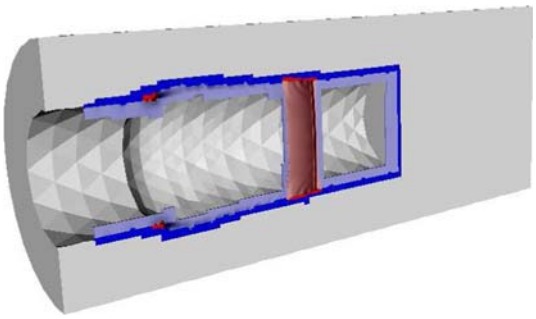

Fig. 21 Schlieren plot of the density on regions covered by SAMR level 1 (*blue*) and 2 (*red*) inside the deforming cylinder for $t = 3.0\mu s$

solid target (lower graphic of Fig. 20). 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 high-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 capability to deal with moving embedded boundaries via a ghost fluid approach. Both solvers have been parallelized for distributed memory machines utilizing domain decomposition, and an effective inter-solver communication module has been outlined. An algorithm has been presented that transforms the triangular solid surface mesh very efficiently into a signed distance function on the Cartesian grid. The application of the methodology to two distinct fluid–structure interaction problems has also been described. We note that the combined approach can lead to high efficiencies in the solution of coupled problems. Particular, our second computational example, in which a detonation wave in a high-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 >10,000 h CPU.

The overall coupling method has been realized as a natural extension of the object-oriented C++ framework on which the adaptive finite volume fluid solver with embedded boundary capability has been built. We have discussed its design in detail and highlighted particular the implementation efficiency that we have gained by utilizing framework concepts. Further, all newly developed components are now generically available within the fluid framework, which already has enabled new, industry-strength, non-coupled fluid applications in which very complicate surface meshes derived from CAD drawings have been used as complex embedded boundaries.

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 time-implicit methods for both fluid and solid into the VTF.

References

1. Aivazis M, Goddard W, Meiron D, Ortiz M, Pool J, Shepherd J (2000) A virtual test facility for simulating the dynamic response of materials. Comput Sci Eng 2(2):42–53
2. Cummings J, Aivazis M, Samtaney R, Radovitzky R, Mauch S, Meiron D (2002) A virtual test facility for the simulation of dynamic response in materials. J Supercomput 23:39–50
3. Mauch S, Meiron D, Radovitzky R, Samtaney R (2003) Coupled Eulerian–Lagrangian simulations using a level set method. In: Bathe K (ed) 2nd M.I.T. conference on computational fluid and solid mechanics, Cambridge, MA, June 17–20
4. Arienti M, Hung P, Morano E, Shepherd J (2003) A level set approach to Eulerian–Lagrangian coupling. J Comput Phys 185:213–251
5. Cirak F, Radovitzky R (2003) A general algorithm for coupling Lagrangian-shell with Eulerian-fluid formulations. 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. Löhner R, Baum J, Charman C, Pelessone D (2003) Fluid-structure interaction simulations using parallel computers. In: Lecture notes in computer science, vol 2565. Springer, Berlin Heidelberg New York, pp 3–23
7. Cirak F, Radovitzky R (2005) A Lagrangian–Eulerian shell-fluid coupling algorithm based on level sets. Comput Struct 83:491–498
8. Löhner R, Cebral J, Yang C, Baum J, Mestreau E, Charman C, Pelessone D, (2004) Large-scale fluid-structure interaction simulations. Comput Sci Eng 6(3):27–37
9. Hu H, Patankar N, Zhu M (2001) Direct numerical simulations of fluid-solid systems using the arbitrary Lagrangian–Eulerian technique. J Comput Phys 169(2):427–462
10. Mittal R, Iaccarino G (2005) Immersed boundary methods. Annu Rev Fluid Mech 37:239–261
11. Quirk J (1994) An alternative to unstructured grids for computing gas dynamics flows around arbitrarily complex two-dimensional bodies. Comput Fluids 23:125–142

12. Berger M, Helzel C (2002) Grid aligned H-box methods for conservation laws in complex geometries. In: Proceedings of the 3rd international symposium on finite volumes for complex applications, Porquerolles
13. Fedkiw R (2002) Coupling an Eulerian fluid calculation to a Lagrangian solid calculation with the ghost fluid method. J Comput Phys 175:200–224
14. Tam D, Radovitzky R, Samtaney R (2005) An algorithm for modelling the interaction of a flexible rod with a two-dimensional high-speed flow. Int J Numer Methods Eng 64(8):1057–1077
15. Pantano C, Deiterding R, Hill D, Pullin D (2005) A low-numerical dissipation patch-based adaptive mesh refinement method for large-eddy simulation of compressible flows. In: Proceedings of the Cyprus international symposium on complex effects in large eddy simulations, University of Cyprus, Nicosia
16. Deiterding R (2005a) Detonation structure simulation with AMROC. In: Lecture notes in computer science, vol 3726. Springer, Berlin Heidelberg New York, pp 916–927
17. Deiterding R, Bader G (2005) High-resolution simulation of detonations detailed chemistry. In: Warnecke G (ed) Analysis and numerics of conservation laws. Springer, Berlin Heidelberg New York, pp 69–91
18. Deiterding R (2005b) A high-resolution method for realistic detonation structure simulation. In: Asakura F et al (eds) Proceegins of the 10th international conference on hyperbolic problems: theory, numerics, applications, Yokohama Publishers
19. Deiterding R (2003) Parallel adaptive simulation of multi-dimensional detonation structures. PhD Thesis, Brandenburgische Technische Universität Cottbus
20. Deiterding R (2002) Efficient simulation of multi-dimensional detonation phenomena. In: Handlikova A et al (eds) Proceedings of the ALGORITMY 2002, 16th conference on scientific computing. Slovak University of Technology, Bratislava, pp 94–101
21. Fickett W, Davis W (1979) Detonation. University of California Press, Berkeley, Los Angeles
22. Clarke J, Karni S, Quirk J, Roe P, Simmonds L, Toro E (1993) Numerical computation of two-dimensional unsteady detonation waves in high energy solids. J Comput Phys 106:215–233
23. Toro E (1999) Riemann solvers and numerical methods for fluid dynamics. Springer, Berlin Heidelberg New York
24. Einfeldt B, Munz C, Roe P, Sjögreen B (1991) On Godunov-type methods near low densities. J Comput Phys 92:273–295
25. Fedkiw R, Aslam T, Merriman B, Osher S (1999) A non-oscillatory Eulerian approach to interfaces in multimaterial flows (the ghost fluid method). J Comput Phys 152:457–492
26. Osher S, Fedkiw R (2003) Level set methods and dynamic implicit surfaces. Springer, Berlin Heidelberg New York
27. Berger M, Colella P (1988) Local adaptive mesh refinement for shock hydrodynamics. J Comput Phys 82:64–84
28. Bell J, Berger M, Saltzman J, Welcome M (1994) Three-dimensional adaptive mesh refinement for hyperbolic conservation laws. SIAM J Sci Comp 15(1):127–138
29. Rendleman C, Beckner V, Lijewski M, Crutchfield W, Bell J (2000) Parallelization of structured, hierarchical adaptive mesh refinement algorithms. Comput Vis Sci 3:147–157
30. Parashar M, Browne J (1997) System engineering for high performance computing software: the HDDA/DAGH infrastructure for implementation of parallel structured adaptive mesh refinement. In: Structured adaptive mesh refinement grid methods, IMA volumes in mathematics and its applications. Springer, Berlin Heidelberg New York
31. Deiterding R (2005c) AMROC—blockstructured adaptive mesh refinement in object-oriented C++. http://www.amroc.sourceforge.net
32. Deiterding R (2005d) Construction and application of an AMR algorithm for distributed memory computers. In: Plewa T et al (eds) Lecture notes in computational science and engineering, vol 41. Springer, Berlin Heidelberg New York, pp 361–372
33. Parashar M, Browne J (1996) On partitioning dynamic adaptive grid hierarchies. In: Proceedings of the 29th annual Hawaii international conference on system sciences
34. MacNeice P, Olson K, Mobarry C, deFainchtein R, Packer C (2000) PARAMESH: a parallel adaptive mesh refinement community toolkit. Comput Phys Commun 126:330–354
35. Booch G, Rumbaugh J, Jacobsen I (1999) The unified modeling language user guide. Addison-Wesley, Reading
36. Marsden J, Hughes T (1993) Mathematical foundations of elasticity. Dover Publications, New York
37. Lew A, Radovitzky R, Ortiz M (2002) An artificial-viscosity method for the Lagrangian analysis of shocks in solids with strength on unstructured, arbitrary-order tetrahedral meshes. J Comput-Aided Mater Des 8:213–231
38. Von Neumann J, Richtmyer R (1950) A method for the numerical calculation of hydrodynamic shocks. J Appl Phys 21:232–243
39. Lubliner J (1972) On the thermodynamic foundations of non-linear solid mechanics. Int J Non-Linear Mech 7:237–254
40. Radovitzky R, Ortiz M (1999) Error estimation and adaptive meshing in strongly nonlinear dynamic problems. Comput Methods Appl Mech Eng 172:203–240
41. Ortiz M, Stainier L (1999) The variational formulation of viscoplastic constitutive updates. Comput Methods Appl Mech Eng 171:419–444
42. Cuitino A, Ortiz M (1992) Material-independent method for extending stress update algorithms from small-strain plasticity to finite plasticity with multiplicative kinematics. Eng Comput 9(4):437–451
43. Stainier L, Cuitino A, Ortiz M (2002) A micromechanical model of hardening, rate sensitivity and thermal softening in BCC single crystals. J Mech Phys Solids 50:1511–1545
44. Kambouchev N, Fernandez J, Radovitzky R (2005) Polyconvex model for materials with cubic anisotropy. http://www.arxiv.org/abs/cond-mat/0505178
45. Benson D (1992) Computational methods in Lagrangian and Eulerian hydrocodes. Comput Methods Appl Mech Eng 99:235–394
46. Ciarlet P (1976) Numerical analysis of the finite element method. Les Presses de L'Universite de Montreal, Quebec
47. Belytschko T (1983) An overview of semidiscretization and time integration procedures. In: Belytschko T, Hughes T (eds) Computational methods for transient analysis. North-Holland, Amsterdam, pp 1–65
48. Hughes T (1983) Analysis of transient algorithms with particular reference to stability behavior. In: Belytschko T, Hughes T (eds) Computational methods for transient analysis. North-Holland, Amsterdam, pp 67–155
49. Kane C, Marsden J, Ortiz M, West M (2000) Variational integrators and the Newmark algorithm for conservative and dissipative mechanical systems. Int J Numer Methods Eng 49:1295–1325
50. Deiterding R, Cirak F, Mauch S, Meiron D (2006) A virtual test facility for simulating detonation-induced fracture of thin flexible shells. In: Alexandrov V et al (eds) Lecture notes in computer science, vol 3992. Springer, Berlin Heidelberg New York, pp 122–130

51. Karypis G (2001) METIS—family of multilevel partitioning algorithms. http://www.users.cs.umn.edu/~karypis/metis
52. Liu A, Joe B (1996) Quality local refinement of tetrahedral meshes based on 8-subtetrahedron subdivision. Math Comput 65:1183–1200
53. Stainier L, Radovitzky R, Ortiz M (1999) Constitutive modeling of Tantalum crystals and polycrystals. In: Croitoro E (ed) Proceedings of the 1st Canadian conference on nonlinear solid mechanics, Victoria, BC, pp 203–213
54. Radovitzky R, Cuitino A (2003) Direct numerical simulation of polycrystals. In: Proceedings of the 44th AIAA/ASME/ASCE/AHS structures, structural dynamics, and materials conference, Norfolk, AIAA
55. Zhao Z, Radovitzky R, Kuchnicki S, Cuitino A (2003) Dynamical evolution of texture in polycrystalline metals under high strain-rate conditions. In: Proceedings of the international symposium on plasticity and its current applications, Quebec City
56. Cuitino A, Radovitzky R (eds) (2004) Multiscale material modeling and simulation. In: Sandia NM Proceedings of the symposia on multiscale material modeling and simulation, Cambridge, MA. Modelling Simul Mater Sci Eng, vol 12, issue 4, Institute of Physics Publishing
57. Zhao Z, Radovitzky R, Cuitino A (2004) A study of surface roughening in fcc metals using direct numerical simulation. Acta Mater 52:5791–5804
58. Kuchnicki S, Cuitino A, Radovitzky R (2005) Efficient and robust constitutive integrators for single-crystal plasticity modeling. Int J Plast 22:1988–2011
59. Sethian J (1999) Level set methods and fast marching methods. Cambridge University Press, Cambridge
60. Johnson D, Cohen E (1998) In: Proceedings of the IEEE international conference on robotics and automation, Leuven, pp 3678–3684
61. Mauch S (2003) Efficient algorithms for solving static Hamilton–Jacobi equations. PhD Thesis, California Institute of Technology
62. Foley J, van Dam A, Feiner S, Hughes J (1996) Computer graphics: principles and practice. Addison-Wesley, Reading
63. Watt A (1993) 3D computer graphics. Addison-Wesley, Reading
64. Park K, Felippa C, Deruntz J (1997) Stabilization of staggered solution procedures for fluid-structure interaction analysis. In: Belytschko T, Geers T (eds) Computational methods for fluid–structure interaction problems, New York, pp 94–124
65. Zhang Q, Hisada T (2004) Studies of the strong coupling and weak coupling methods in FSI analysis. Int J Numer Methods Eng 60:2013–2029
66. Taylor G (1950) The formation of a blast wave by a very intense explosion, I theorical discussion. Proc R Soc Ser A 201(1065):159–174
67. Brode H (1955) Numerical solution of spherical blast waves. J Appl Phys 26(6):766–775
68. Cohen R, Gülseren O (2001) Thermal equation of state of tantalum. Phys Rev B 63:3363
69. Steinberg D, Cochran S, Guinan M (1980) A constitutive model for metals applicale at high-strain rate. J Appl Phys 51:1498–1504
70. Mader C (1979) Numerical modeling of detonations. University of California Press, Berkeley, Los Angeles