

# On mesh-particle techniques

Rainald Löhner · Fernando Camelli · Joseph D. Baum ·  
Fumiya Togashi · Orlando Soto

Received: 22 December 2013 / Revised: 31 March 2014 / Accepted: 3 April 2014 / Published online: 9 May 2014  
© Springer International Publishing Switzerland 2014

**Abstract** The treatment of dilute solid (or liquid) phases via Lagrangian particles within mesh-based gas-dynamics (or hydrodynamic) codes is common in computational fluid dynamics. While these techniques work very well for a large spectrum of physical parameters, in some cases, notably for very light or very heavy particles, numerical instabilities appear. The present paper examines ways of mitigating these instabilities, and summarizes important implementational issues.

**Keywords** Particle methods · Finite elements · Computational fluid dynamics

## 1 Introduction

When solving the compressible two-phase equations, the gas, as a continuum, is best represented by a set of partial differential equations (the Navier–Stokes equations) that are numerically solved on a mesh. Thus, the gas characteristics are calculated at the mesh points within the flowfield. However, as the particles (or fragments) may be relatively sparse in the flowfield, they can be modeled by either:

- (a) A continuum description, i.e. in the same manner as the fluid flow, or
- (b) A particle (or Lagrangian) description, where individual particles (or groups of particles) are monitored and tracked in the flow.

Although the continuum (so-called multi-fluid) method has proven relatively successful for compressible two-phase flows, the inherent assumptions of the continuum approach lead to several disadvantages which may be countered with a Lagrangian treatment for dilute flows [15, 16, 23, 30, 70]. The continuum assumption cannot robustly account for local differences in particle characteristics, particularly if the particles are polydispersed. In addition, the only boundary conditions that can be considered in a straightforward manner are slipping and sticking, whereas reflection boundary conditions, such as specular and diffuse reflection, may be additionally considered with a Lagrangian approach. Turbulent dispersion can also be treated on a more fundamental basis. Finally, numerical diffusion of the particle density can be eliminated by employing Lagrangian particles due to their pointwise spatial accuracy.

While a Lagrangian approach offers many potential advantages, this method also creates problems that should be addressed by the model. For instance, large numbers of particles may cause a Lagrangian analysis to be memory intensive. This problem is circumvented by treating parcels of particles, i.e. doing the detailed analysis for one particle and then applying the effect of many. In addition, continuous mapping and remapping of particles to their respective elements may increase computational requirements, particularly for unstructured grids.

The present paper summarizes the procedures used, as well as some of the difficulties encountered when implement-

---

**Electronic supplementary material** The online version of this article (doi:10.1007/s40571-014-0022-7) contains supplementary material, which is available to authorized users.

---

R. Löhner (✉) · F. Camelli  
Department of Computational and Data Science M.S. 6A2,  
CFD Center, College of Science, George Mason University Fairfax,  
Fairfax, VA 22030-4444, USA  
e-mail: rlohner@gmu.edu

J. D. Baum · F. Togashi · O. Soto  
Advanced Technology Group, SAIC, McLean, VA 22020, USA

ing a particle description of diluted phases in a flow solver based on unstructured grids.

### 2 Equations describing the motion of particles

In order to describe the interaction of particles with the flow, the mass, forces and energy/ work exchanged between the flowfield and the particles must be defined. Before going on, we need to define the physical parameters involved. For the fluid, we denote by  $\rho, p, e, T, k, v_i, \mu, \gamma$  and  $c_v$  the density, pressure, specific total energy, temperature, conductivity, velocity in direction  $x_i$ , viscosity, ratio of specific heats and the specific heat at constant volume. For the particles, we denote by  $\rho_p, T_p, v_{pi}, d, c_p$  and  $Q$  the density, temperature, velocity in direction  $x_i$ , equivalent diameter, and heat transferred per unit volume. In what follows, we will refer to particles, fragments, or chunks collectively as particles.

Making the classic assumptions that the particles may be represented by an equivalent sphere of diameter  $d$ , the drag forces  $\mathbf{D}$  acting on the particles will be due to the difference of fluid and particle velocity:

$$\mathbf{D} = \frac{\pi d^2}{4} \cdot c_D \cdot \frac{1}{2} \rho |\mathbf{v} - \mathbf{v}_p| (\mathbf{v} - \mathbf{v}_p). \tag{1}$$

The drag coefficient  $c_D$  is obtained empirically from the Reynolds-number  $Re$ :

$$Re = \frac{\rho |\mathbf{v} - \mathbf{v}_p| d}{\mu} \tag{2}$$

as:

$$c_D = \max \left( 0.1, \frac{24}{Re} \left( 1 + 0.15 Re^{0.687} \right) \right)$$

The lower bound of  $c_D = 0.1$  is required to obtain the proper limit for the Euler equations, where  $Re \rightarrow \infty$ . The heat transferred between the particles and the fluid is given by

$$Q = \frac{\pi d^2}{4} \cdot \left[ h \cdot (T - T_p) + \sigma^* \cdot (T^4 - T_p^4) \right], \tag{3}$$

where  $h$  is the film coefficient and  $\sigma^*$  the radiation coefficient. For the class of problems considered here, the particle temperature and kinetic energy are such that the radiation coefficient  $\sigma^*$  may be ignored. The film coefficient  $h$  is obtained from the Nusselt-Number  $Nu$ :

$$Nu = 2 + 0.459 Pr^{0.333} Re^{0.55}, \tag{4}$$

where  $Pr$  is the Prandtl-number of the gas

$$Pr = \frac{k}{\mu}, \tag{5}$$

as

$$h = \frac{Nu \cdot k}{d}. \tag{6}$$

Having established the forces and heat flux, the particle motion and temperature are obtained from Newton’s law and the first law of thermodynamics. For the particle velocities, we have:

$$\rho_p \frac{\pi d^3}{6} \cdot \frac{d\mathbf{v}_p}{dt} = \mathbf{D}. \tag{7}$$

This implies that:

$$\frac{d\mathbf{v}_p}{dt} = \frac{3\rho}{4\rho_p d} \cdot c_d |\mathbf{v} - \mathbf{v}_p| (\mathbf{v} - \mathbf{v}_p) = \alpha_v |\mathbf{v} - \mathbf{v}_p| (\mathbf{v} - \mathbf{v}_p). \tag{8}$$

The particle positions are obtained from:

$$\frac{d\mathbf{x}_p}{dt} = \mathbf{v}_p. \tag{9}$$

The temperature change in a particle is given by:

$$\rho_p c_p \frac{\pi d^3}{6} \cdot \frac{dT_p}{dt} = Q, \tag{10}$$

which may be expressed as:

$$\frac{dT_p}{dt} = \frac{3k}{4c_p \rho_p d^2} \cdot Nu \cdot (T - T_p) = \alpha_T (T - T_p). \tag{11}$$

Equations (8, 9, 11) may be formulated as a system of Ordinary Differential Equations (ODEs) of the form:

$$\frac{d\mathbf{u}_p}{dt} = \mathbf{r}(\mathbf{u}_p, \mathbf{x}, \mathbf{u}_f), \tag{12}$$

where  $\mathbf{u}_p, \mathbf{x}, \mathbf{u}_f$  denote the particle unknowns, the position of the particle and the fluid unknowns at the position of the particle.

### 3 Numerical integration

As seen above, the equations describing the position, velocity and temperature of a particle or fragment may be formulated as a system of nonlinear Ordinary Differential Equations (see above) of the form:

$$\frac{d\mathbf{u}_p}{dt} = \mathbf{r}(\mathbf{u}_p, \mathbf{x}, \mathbf{u}_f). \tag{13}$$

They can be integrated numerically in a variety of ways. Due to its speed, low memory requirements and simplicity, we have chosen the following k-step low-storage Runge-Kutta procedure to integrate them:

$$\begin{aligned} \mathbf{u}_p^{n+i} &= \mathbf{u}_p^n + \alpha_i \Delta t \cdot \mathbf{r}(\mathbf{u}_p^{n+i-1}, \mathbf{x}^{n+i-1}, \mathbf{u}_f^{n+i-1}), \\ i &= 1, k, \quad \Delta \mathbf{u}^0 = 0. \end{aligned} \tag{14}$$

For linear ODEs the choice

$$\alpha_i = \frac{1}{k + 1 - i}, \quad i = 1, k \tag{15}$$

leads to a scheme that is  $k$ -th order accurate in time. Note that in each step the location of the particle with respect to the

fluid mesh needs to be updated in order to obtain the proper values for the fluid unknowns. The default number of stages used is  $k = 4$ . This would seem unnecessarily high, given that the flow solver is of second-order accuracy, and that the particles are integrated separately from the flow solver before the next (flow) timestep, i.e. in a staggered manner. However, it was found that the 4-stage particle integration preserves very well the motion in vortical structures and leads to less ‘wall sliding’ close to the boundaries of the domain. The stability/ accuracy of the particle integrator should not be a problem as the particle motion will always be slower than the maximum wave speed of the fluid (fluid velocity + speed of sound).

The transfer of forces and heat flux between the fluid and the particles must be accomplished in a conservative way, i.e. whatever is added to the fluid must be subtracted from the particles and vice-versa. The Finite Element Discretization of the the fluid equations will lead to a system of ODE’s of the form:

$$\mathbf{M}\Delta\mathbf{u} = \mathbf{r}, \tag{16}$$

where  $\mathbf{M}$ ,  $\Delta\mathbf{u}$  and  $\mathbf{r}$  denote, respectively, the consistent mass matrix, increment of the unknowns vector and right-hand side vector. Given the ‘host element’ of each particle, i.e. the fluid mesh element that contains the particle, we add the forces and heat transferred to  $\mathbf{r}$  as follows:

$$\mathbf{r}_D^i = \sum_{elsurri} N^i(\mathbf{x}_p)\mathbf{D}_p. \tag{17}$$

Here  $N^i(\mathbf{x}_p)$  denotes the shape-function values of the host element for the point coordinates  $\mathbf{x}_p$ . As the sum of all shape-function values is unity at every point:

$$\sum N^i(\mathbf{x}) = 1 \quad \forall \mathbf{x}, \tag{18}$$

this procedure is strictly conservative.

The change in momentum and energy for one particle is given by:

$$\mathbf{f}_p = \rho_p \frac{\pi d^3}{6} \frac{(\mathbf{v}_p^{n+1} - \mathbf{v}_p^n)}{\Delta t}, \tag{19}$$

$$q_p = \rho_p c_{pp} \frac{\pi d^3}{6} \frac{(T_p^{n+1} - T_p^n)}{\Delta t}. \tag{20}$$

These quantities are multiplied by the number of particles in a packet in order to obtain the final values transmitted to the fluid. Before going on, we summarize the basic steps required in order to update the particles one timestep:

- Initialize Fluid Source-Terms:  $\mathbf{r} = 0$
- DO: For Each Particle:
  - DO: For Each Runge-Kutta Stage:
    - Find Host Element of Particle: IELEM,  $N^i(\mathbf{x})$

- Obtain Fluid Variables Required
- Update Particle: Velocities, Position, Temperature, ...

- – ENDDO
- Transfer Loads to Element Nodes
- ENDDO

#### 4 Particle parcels

For a large number of very small particles, it becomes impossible to carry every individual particle in a simulation. The solution to this dilemma is to:

- (a) Agglomerate the particles into so-called packets of  $N_p$  particles;
- (b) Integrate the governing equations for one individual particle; and
- (c) Transfer back to the fluid  $N_p$  times the effect of one particle.

Beyond a reasonable number of particles per element (typically  $> 8$ ), this procedure produces accurate results without any deterioration in physical fidelity.

##### 4.1 Agglomeration/subdivision of particle parcels

As the fluid mesh may be adaptively refined and coarsened in time, or the particle traverses elements of different sizes, it may be important to adapt the parcel concentrations as well. This is necessary to ensure that there is sufficient parcel representation in each element and yet, that there are not too many parcels as to constitute an inefficient use of CPU and memory. For example, as an element with parcels is refined by one level (the maximum is typically four or five levels of refinement) to yield eight new elements, the number of parcels per new element will be significantly reduced if no parcel adaption is employed. This can lead to a reduction in local spatial accuracy, especially if no parcels are left in one or more of the new elements.

In order to locally determine if a refinement or a coarsening of parcels is to be performed, the number of parcels in each element is checked and modified either after a set number of timesteps or after each mesh adaptation/ change.

#### 5 Limiting during particle updates

As the particles are integrated independently from the flow solver, it is not difficult to envision situations where for the extreme cases of very light or very heavy particles physically meaningless or unstable results may be obtained.

### 5.1 Small/light particles

In order to see the difficulties that can occur with very small and/or light particles, consider an impulsive start from rest. This situation can happen when a shock enters a dusty zone. The friction forces are proportional to the difference of fluid and particle velocities to the 2nd power, and to the diameter of the particle to the 2nd power. The mass of the particle, however, is proportional to the diameter of the particle to the 3rd power. If the timestep is large and the particle very light, after a timestep (or Runge–Kutta substep) the velocity of the particle may exceed the velocity of the fluid. This is clearly impossible and is only due to the discretization error of the numerical integration in time (i.e. the timestep is too large). The same can happen to the temperature (and diameter, in the case of burning particles) of the particle.

It would be impractical (and unnecessary) to reduce the timestep so as to achieve high temporal accuracy throughout the calculation. After all, for the case of a shock entering a quiescent dusty zone the timestep would have to be reduced until the shock has traversed the complete region. In order to prevent this, the changes in particle velocities and temperatures are limited in order not to exceed the differences in velocities and temperature between the particles and the fluid. Assume (in 1D) a difference of velocities at time  $t = t^n$ :

$$\Delta v^n = v^n - v_p^n. \quad (21)$$

Furthermore, assume that the particles are updated before the flow. The particle velocity is then limited as follows:

- If:  $v_p^n < v^n \Rightarrow v_p^{n+1} \leq v^n$
- If:  $v_p^n > v^n \Rightarrow v_p^{n+1} \geq v^n$

This limiting procedure is applied to each of the Runge–Kutta stages.

### 5.2 Large/heavy/many particles

Consider now the opposite case as before. Assume that the particles are started impulsively from rest (e.g. by a shock entering a quiescent dusty region), but that there are many or these and/or they are large or heavy. In this case, when the drag force is added back to the fluid, if the timestep is too large a flow reversal could occur (if the particles are accelerated the flow is decelerated). To prevent this unphysical (and unstable) phenomenon to happen, the source-terms are limited. This is done by comparing the resulting source-terms for the momentum and energy equations of the fluid with the fluid velocities and temperature. Assuming we know the source-terms for the particles  $\mathbf{s}_v, s_T$  and the current timestep  $\Delta t$ , the procedure is as follows:

- Obtain the average particle velocities and temperatures at the points of the flow mesh  $\mathbf{v}_p, T_p$ .
- Obtain the change in flow velocities and temperatures if only the source-terms from the particles are added, e.g. for the velocities:

$$\mathbf{M} \rho \Delta \mathbf{v} = \mathbf{s}_v; \quad (22)$$

- Obtain the allowed increase/decrease factors from:

$$\alpha_v = \frac{|(\mathbf{v}_p - \mathbf{v}) \cdot \Delta \mathbf{v}|}{\Delta \mathbf{v} \cdot \Delta \mathbf{v}}; \quad \alpha_T = \frac{|(T_p - T)|}{\Delta T}; \quad (23)$$

- Limit the allowed increase/decrease factor:

$$\alpha_v = \max(0, \min(1, \alpha_v)), \quad \alpha_T = \max(0, \min(1, \alpha_T)). \quad (24)$$

This assures that the source-terms added to the momentum and energy equation remain bounded. While this procedure works very well, avoiding instabilities, it is non-conservative.

## 6 Particle contact

In some situations, the density of the particles increases to a point that they basically occupy all the volume available. Although such high density situations are outside the scope of the underlying theory, production runs require techniques that can cope with them. What happens physically is that at some point particles contact with one another, thereby limiting the achievable density and volume-fill ratio of particles.

### 6.1 Particle forces due to contact

In order to approximate the forces exerted by the contact, the first measure that has to be obtained is the equivalent radius. After all, we are computing packets of particles. Some of these packets represent hundreds or thousands of actual particles. Given  $n_p$  particles of diameter  $d_p$ , the volume occupied by them is given by:

$$V = \frac{n_p \pi}{\alpha_K} \frac{d_p^3}{6}, \quad (25)$$

where  $\alpha_K$  is the maximum filling factor (whose theoretical limit for spheres is the Kepler limit of  $\alpha_K \approx 0.74$ ). The equivalent radius is therefore given by:

$$r^a = \left[ \frac{3}{4\pi} V \right]^{1/3} = \left[ \frac{n_p}{8\alpha_K} \right]^{1/3} d_p. \quad (26)$$

Given two particle packets with positions  $\mathbf{x}_i, \mathbf{x}_j$ , the overlap distance is given by:

$$do_{ij} = r_i^a + r_j^a - d_{ij}, \quad d_{ij} = |\mathbf{x}_i - \mathbf{x}_j|. \quad (27)$$

The average overlap between particles is then:

$$do_{ij}^s = do_{ij} \frac{1}{2} \left( \frac{1}{n_{p_i}} + \frac{1}{n_{p_j}} \right). \tag{28}$$

Defining a unit normal  $\mathbf{n}$  in the direction  $i, j$  as:

$$\mathbf{n}_{ij} = \frac{\mathbf{x}_i - \mathbf{x}_j}{|\mathbf{x}_i - \mathbf{x}_j|}, \tag{29}$$

the relative velocity of the particles defines a tangential direction  $\mathbf{t}$ :

$$\begin{aligned} \mathbf{v}_{ij} &= \mathbf{v}_j - \mathbf{v}_i, \quad v_{ij}^n = \mathbf{v}_{ij} \cdot \mathbf{n}_{ij}, \quad \mathbf{v}_{ij}^t = \mathbf{v}_{ij} - v_{ij}^n \cdot \mathbf{n}_{ij}, \\ \mathbf{t}_{ij} &= \frac{\mathbf{v}_{ij}^t}{|\mathbf{v}_{ij}^t|}. \end{aligned} \tag{30}$$

The normal and tangential forces are:

$$f_{ij}^n = \frac{1}{2} (k_i + k_j) do_{ij}^s, \tag{31}$$

$$f_{ij}^t = \frac{1}{2} (h_i + h_j) f_{ij}^n, \tag{32}$$

where  $k, h$  refer to the stiffness and damping specified. The tangential force is limited so as to avoid a reversal in relative tangential velocities:

$$f_{ij}^t = \min \left( f_{ij}^t, \frac{|\mathbf{v}_{ij}^t|}{\Delta t \cdot \max(m_i, m_j)} \right), \tag{33}$$

A damping force is added in the normal direction in order to avoid ‘ringing’. This force is given by:

$$f_{ij}^{nd} = -v_{ij}^n \frac{(h_i + h_j) \cdot (k_i + k_j)}{(m_i + m_j)}, \tag{34}$$

and is limited to the lowest possible value of damping in order to avoid reversion of contact force due to velocity damping:

$$f_{ij}^{nd} = \max \left( -f_{ij}^n, f_{ij}^{nd} \right). \tag{35}$$

The complete force is then given by:

$$\mathbf{f}_{ij} = (f_{ij}^n + f_{ij}^{nd}) \cdot \mathbf{n} + f_{ij}^t \cdot \mathbf{t}. \tag{36}$$

The particles are stored in a bin in order to quickly find the particles in the vicinity of any given particle.

### 6.2 Estimating contact stiffness and damping parameters

The estimation of the required particle contact stiffness and damping parameters presents an interesting challenge. The measured values for contact stiffness may be very high, forcing a reduction of the allowable timestep. Therefore, an attempt was made to obtain values that would avoid penetration, yet allow the usual CFL-based flowfield timesteps to be kept. Let us consider a stationary particle in a flow with

density  $\rho_f$  and velocity  $v_f$ . In this case the force exerted by the fluid is given by:

$$D = \frac{\pi d^2}{4} \cdot c_D \cdot \frac{1}{2} \rho_f v_f^2 \tag{37}$$

$$c_D = \max \left( 0.1, \frac{24}{Re} \left( 1 + 0.15 Re^{0.687} \right) \right), \quad Re = \frac{\rho_f v_f d}{\mu}. \tag{38}$$

The stiffness required in order to avoid a penetration distance of  $\xi d$  is then:

$$k \xi d = D, \tag{39}$$

or

$$k = \frac{D}{\xi d}. \tag{40}$$

For the low and high Reynolds-number regimes, we obtain:

$$k_{Re < 1} \approx \frac{3\pi\mu v}{\xi}, \quad k_{Re \gg 1} \approx \frac{0.1\pi d \rho v^2}{8\xi}. \tag{41}$$

## 7 Accounting for void fractions

The amount the fluid can occupy in any given volume is reduced by the presence of particles. In the original derivation of the theory the assumption of a very dilute solid phases was made. This implied that all volume (or void fraction) effects could be neglected. As users keep pushing up the void fraction, these assumptions are no longer valid and the effect of the void fraction has to be accounted for in the flow solver. Given a volume  $V$ , occupied to the extent  $V_f$  by a fluid and  $V_p$  by particles, the void fraction  $\epsilon$  is defined as:

$$\epsilon = \frac{V_f}{V} = \frac{V - V_p}{V}. \tag{42}$$

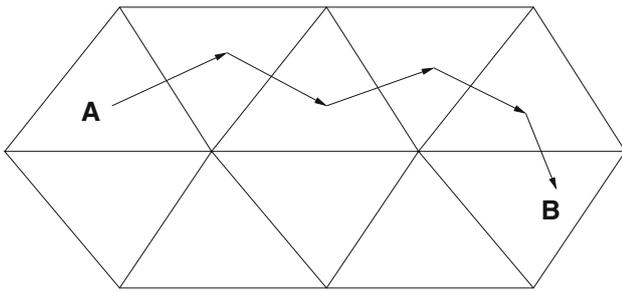
The Navier–Stokes equations for the case of noticeable void fractions are given by [28,69]:

$$\mathbf{u}_{,t} + \nabla \cdot (\mathbf{F}^a - \mathbf{F}^v) = \mathbf{S}, \tag{43}$$

where

$$\begin{aligned} \mathbf{u} &= \epsilon \{ \rho, \rho v_i, \rho e \}, \\ \mathbf{F}_j^a &= \epsilon \{ \rho v_j, \rho v_i v_j + p \delta_{ij}, v_j (\rho e + p) \}, \\ \mathbf{F}_j^v &= \epsilon \{ 0, \sigma_{ij}, v_i \sigma_{lj} + k T_{,j} \}, \\ \mathbf{S} &= \{ 0, p \epsilon_{,x} + \epsilon \rho g_x, p \epsilon_{,y} \\ &\quad + \epsilon \rho g_y, p \epsilon_{,z} + \epsilon \rho g_z, (p \epsilon_{,j} + \epsilon \rho g_j) v_j \}. \end{aligned} \tag{44}$$

Here  $\rho, p, e, T, k, v_i, g_i$  denote the density, pressure, specific total energy, temperature, conductivity, fluid velocity and gravity in direction  $x_i$  respectively. This set of equations is closed in the usual way by providing an equation of state for the pressure, relating the stress tensor  $\sigma_{ij}$  to the deformation rate tensor.



**Fig. 1** Particle tracing on unstructured grid

## 8 Particle tracking

A common feature of all particle-grid applications is that the particles do not move far between timesteps. This makes physical sense: if a particle jumped ten gridpoints during one timestep, it would have no chance to exchange information with the points along the way, leading to serious errors. Therefore, the assumption that the new host elements of the particles are in the vicinity of the current ones is a valid one. For this reason, the most efficient way to search for the new host elements is via the vectorized neighbour-to-neighbour algorithm described in [33,45] (see Fig. 1). The idea is to search for the host elements of as many particles as possible. The obstacle to this approach is that not every particle will find its host element in the same number of attempts or passes over the particles. The solution is to reorder the particles to be interpolated after each pass so that all particles that have not yet found their host element are at the top of the list.

## 9 Particles and shared memory parallel machines

For shared memory parallel machines, the ‘find host element’ technique described above can be used directly. One only has to make sure that sufficiently long vectors are obtained so that even on tens of cores the procedure is efficient. For the ‘particle loads to fluid nodes’ assembly, the particles loads are first accumulated according to elements. Thereafter, these element loads are added to the fluid nodes using the standard mesh coloring techniques [45].

## 10 Particles and GPUs

Almost all physics subroutines employed by the authors in the FEFLO code have been ported to GPUs using the semi-automatic tool F2CUDA [49,51,58,59]. Particles on an unstructured mesh represent two ‘irregular’ data structures. Therefore, it is not surprising that porting the particle modules to GPUs required some effort. In order to pre-sort these particles heavy use was made of the CUDA Thrust

library, in particular the `thrust::copy_if` option. A further algorithmic difficulty was encountered during the step that adds the source-terms (forces, source-terms) from particles to points. After all, many particles could reside in an element, adding repeatedly to points. Such memory contention inhibits straightforward parallelization directly over the particles, necessitating a more advanced algorithm.

One solution is to use the colouring/grouping of elements (used to avoid memory conflicts during scatter-add operations) and the host element information of each particle to presort the order in which the particle source-terms are added. The problem with this approach is that the number of particles in each element is not the same. Therefore, besides being difficult to vectorize, large load imbalances may occur.

An alternative approach is to apply data-parallel algorithms provided by the Thrust library [29]. In particular, prior to scattering particle-point contributions, they are first written to a temporary array along with the point index to which the contributions should be scattered. Next, the contributions array is sorted by the point indexes using `thrust::sort_by_key`, which is based on the Merrill-radix-sort algorithm [57]. With particle-point contributions now arranged consecutively in memory, the next step is to sum the contributions to each point using the `thrust::reduce_by_key` algorithm. Given that not necessarily all points will receive a contribution from particles, it is necessary to then perform a `thrust::scatter` to map the computed particle contributions to the appropriate points. While most of FEFLO is automatically ported to the GPU via F2CUDA [49,51], this is one instance where this is not the case. The employed translator allows for incorporating manual overrides of the original Fortran code. This is done here by expressing the algorithms in terms of standardized data-parallel primitives. These primitives are highly non-trivial to implement efficiently, but are easily accessible via the Thrust library. An advantage of this approach is that future performance improvements to the Thrust library will be immediately reflected in the GPU version of FEFLO.

We remark here, as we have done on several occasions before, that favourable GPU timings require that all operations be performed on the GPU.

## 11 Particles and distributed memory parallel machines

Porting particle tracing and particle/fluid interaction options to a distributed memory environment within a domain decomposition framework/ approach requires the proper transfer of particles from one domain to the other, with the associated extra coding. FEFLO uses overlap of domains that is one layer thick. While this has many advantages, in the case of particles one faces the problem that a particle could be in

several domains. In order to arrive at a fast particle transmission algorithm, the following procedure was implemented:

- Obtain all the elements that border/overlap other domains;
- Order these border elements according to the communication passes that exchange mesh information with neighbours;
- Obtain all the particles in each element;
- In each exchange pass:
  - From the list of border elements and the list of particles in each element: assemble all particles that need to be sent to the neighbouring domain;
  - Exchange the information of how many particles will be sent and received in this pass;
  - Send/receive the particles from the neighbouring domains;
- Remove the duplicate particles residing in the domain.

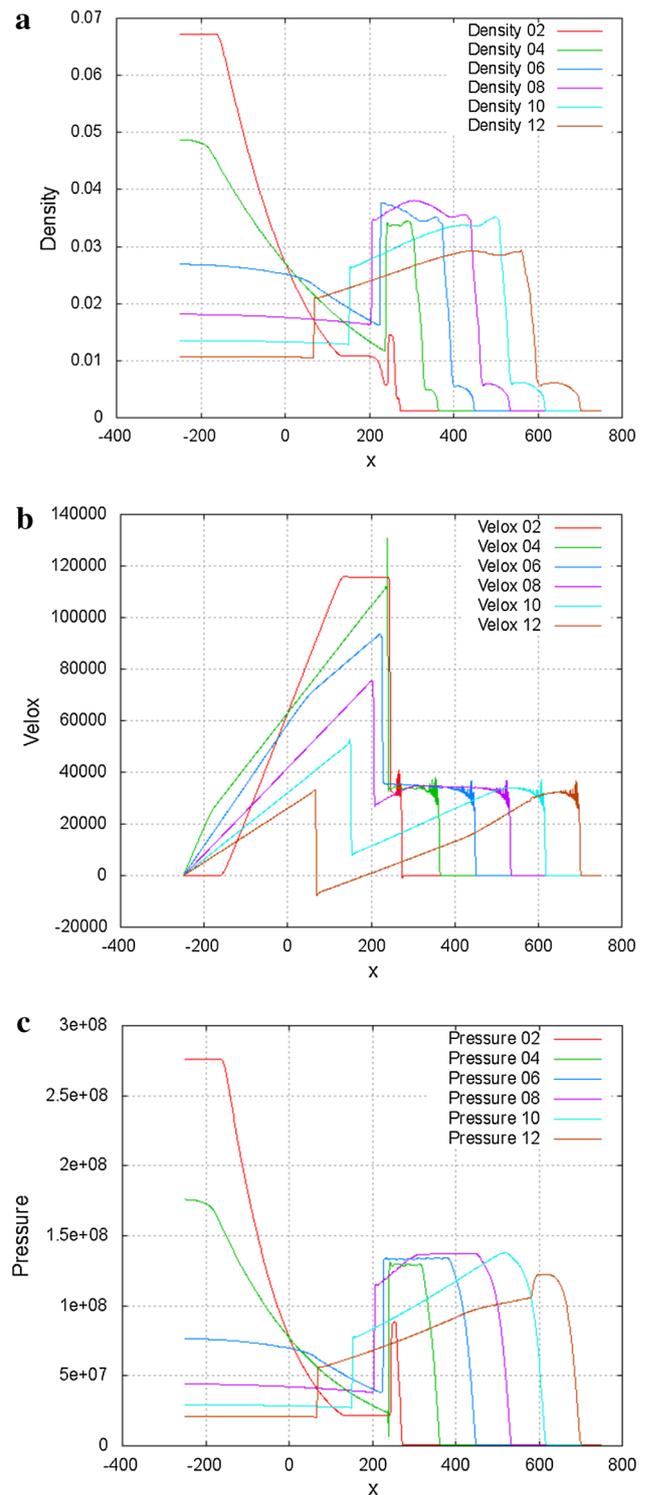
A number of problems had to be overcome before this procedure would work reliably and without excessive CPU requirements:

- *Duplicate particles* after exchanging the particles, the same particle may appear repeatedly in the same processor. For example, the particle may be moving along the border of two domains, or may move slowly (which implies it was also sent from the neighbouring domain in the previous timestep). The best solution for this dilemma is to assign to each particle a so-called ‘unique universal number’ (UUN). The particles are then traversed, and those whose UUN is already in the list are discarded.
- *Particles with same location* due to flow physics and/or geometric singularities distinct particles may end up in the same location. This is avoided by traversing all elements, finding the particles in each, and then separating in each element those that are too close together. This last step is done by assigning a small random change to the shape-functions of the particles at the current location.

It was observed that the parallel particle update modules required a considerable amount of CPU resources. For cases where the particle count per processor was in the millions (and this happens frequently), an update could take 2–3 min (!). After extensive recoding and optimization for both MPI and OMP, the same update was reduced to 2–3 s per update.

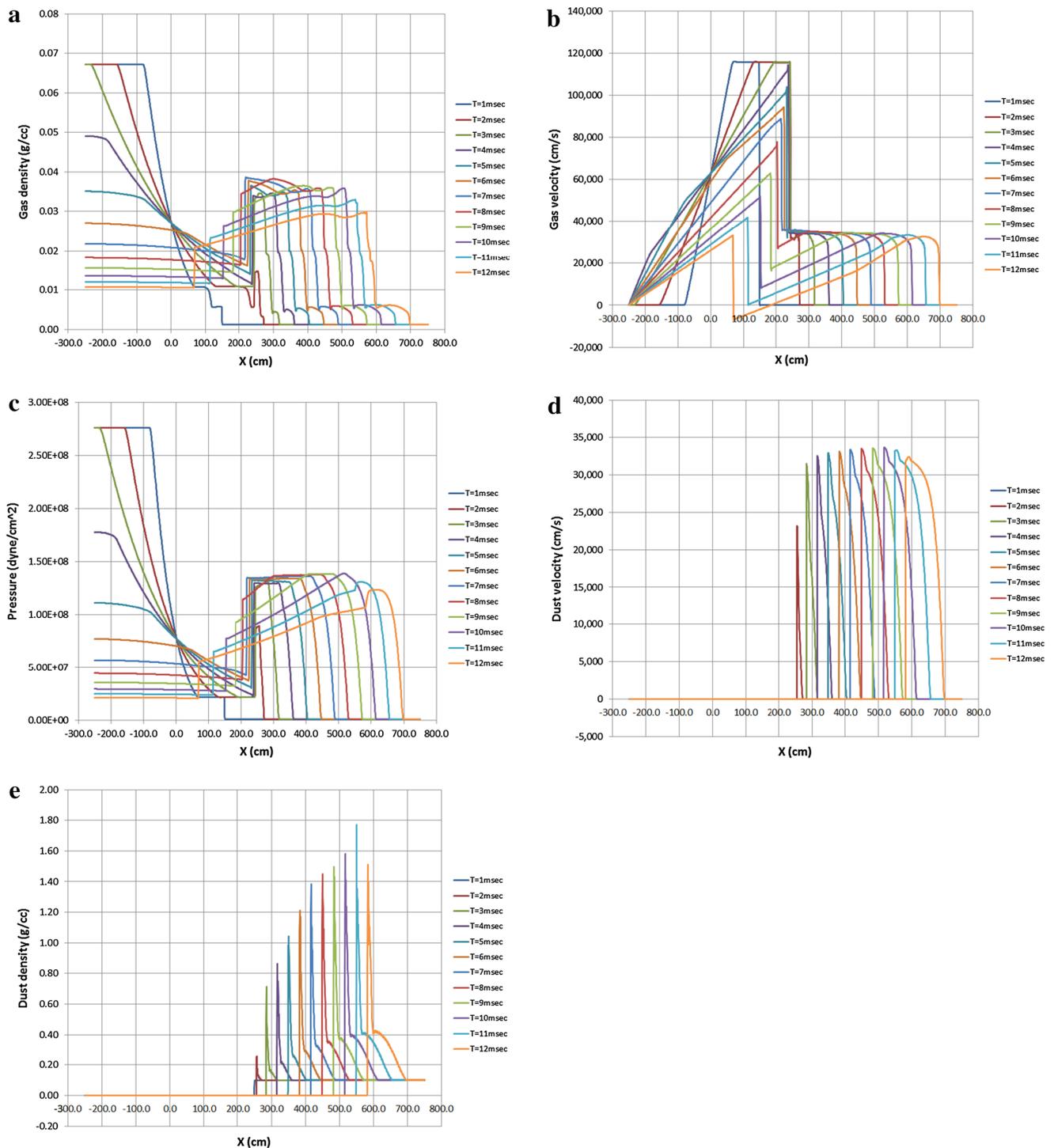
### 12 Examples

The techniques described above were implemented in FEFLO, a general-purpose CFD code based on the following general principles:



**Fig. 2** **a** Tube with particles: fluid density. **b** Tube with particles: fluid velocity. **c** Tube with particles: fluid pressure

- Use of unstructured grids (automatic grid generation and mesh refinement);
- Finite element discretization of space;
- Separate flow modules for compressible and incompressible flows;



**Fig. 3** **a** Tube with particles: fluid density. **b** Tube with particles: fluid velocity. **c** Tube with particles: fluid pressure. **d** Tube with particles: dust velocity. **e** Tube with particles: dust density

- Edge-based data structures for speed;
- Optimal data structures for different architectures;
- Bottom-up coding from the subroutine level to assure an open-ended, expandable architecture.

The code has had a long history of relevant applications involving compressible flow simulations in the areas of transonic flow [37,52–56], store separation [4,7,9,11,12], blast–structure interaction [3,5,6,8,10,13,14,40,46,63,65,67],

incompressible flows [2,39,42,50,60,62,66], free-surface hydrodynamics [36,43,44], dispersion [17–20,41], patient-based haemodynamics [1,21,22,37,47] and aeroacoustics [31]. The code has been ported to vector [38], shared memory [35,64,68], distributed memory [34,48,60,61] and GPU-based [24–27,49] machines.

### 12.1 Shock into dust

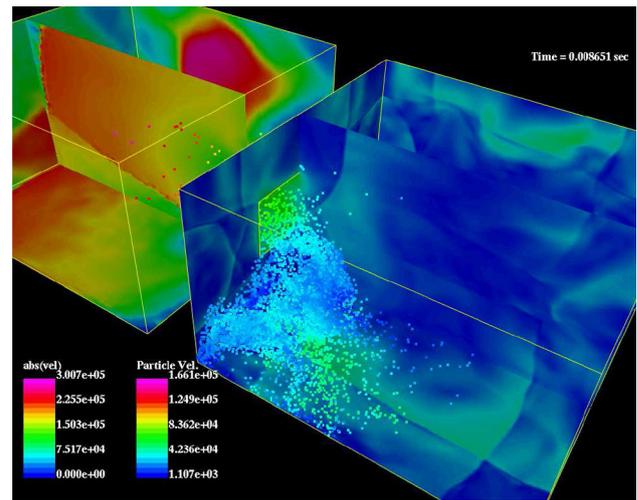
This case considers a 60-cm-square rigid shaft, with the axis running in the  $x$ -direction. The shaft is considered as semi-infinite starting at  $x = -250\text{ cm}$ . The gas is air, and treated as a perfect gas obeying  $p = \rho R T$  with  $R = 2.869 \cdot 10^6\text{ dynes} - \text{cm/g} - \text{K}$  and  $\gamma = 1.4$ . The air for  $x > 0$  is initially at  $p = 1.01 \cdot 10^6\text{ dynes/cm}^2$  and  $T = 15.15^\circ\text{C} = 288.3^\circ\text{K}$ . The air for  $x < 0$  is initially at  $p = 4000\text{ psi} = 2.7579 \cdot 10^8\text{ dynes/cm}^2$  and  $T = 1430.6^\circ\text{K}$ . Initially the region  $x > 250.0\text{ cm}$  is filled with a mixture of air and uniformly distributed dust. The dust particles have a density of  $\rho_p = 2.3\text{ g/cm}^3$  and  $D = 100\text{ nm}$ . The average mass loading of dust inside the disk is  $0.1\text{ g/cm}^3$ . The dust particles therefore occupy a volume fraction of  $0.1/2.3 = 0.0435$  within the dusty region, sufficiently low for a dilute species assumption. At time  $t = 0$  the gases are allowed to begin interacting as in a Riemann problem, launching a shock wave propagating in the  $+x$  direction. Although the problem is one-dimensional, it was run using the 3-D code. The results obtained have been summarized in Fig. 2a–c, which show the variables along the centerline of the tube for different times. The emergence of the classical Riemann problem is visible, followed by slowdown and partial reflection due to the presence of the particles. This is a high-loading case (the density of the particles are 100x the ambient density of air), and can lead to instabilities. In order to trigger these, we ran on a cartesian mesh split into tetrahedra, with  $2 \times 2 \times 2$  particles per cartesian cell. Note the emergence of oscillations in the velocities as the shock enters the region of quiescent particles. Without the limiters described above, this type of run would fail.

This case was repeated with  $4 \times 4 \times 4$  per cartesian cell. The results are shown in Fig. 3. Note that the oscillations have largely disappeared.

### 12.2 Blast in room with dilute material

This example considers the flow and particle transport resulting from a blast in a room where dilute material has been deposited. The powder-like material is modeled via particles. The geometry, together with the solution, can be discerned from Fig. 4.

The compressible Euler equations are solved using an edge-based FEM-FCT technique [32,45,52]. The initialization was performed by interpolating the results of a very



**Fig. 4** Blast in room: pressures and particle velocities

**Table 1** Blast in room with dilute material

nelem	CPU/GPU	mvecl	Time (sec)
4.0 M	Xeon E5530 (1)	32	305
4.0 M	Xeon E5530 (2)	32	178
4.0 M	Xeon E5530 (4)	32	113
4.0 M	Xeon E5530 (6)	32	79
4.0 M	Xeon E5530 (8)	32	68
4.0 M	Tesla C2070	51200	49

detailed 1D (spherically symmetric) run. The particles are transported using a 4th order Runge-Kutta technique. The timing studies (summarized in Table 1) were carried out with the following set of parameters:

- Compressible Euler
- Ideal Gas EOS
- Explicit FEM–FCT
- Initialization from a 1D file
- 4.0 Million elements, 93,552 particles
- Run for 60 steps

## 13 Conclusions and outlook

The treatment of dilute solid (or liquid) phases via Lagrangian particles within mesh-based gas-dynamics (or hydrodynamic) codes is common in computational fluid dynamics. While these techniques work very well for a large spectrum of physical parameters, in some cases, notably for very light or very heavy particles, numerical instabilities appear. The present paper has examined ways of mitigating these instabilities. Furthermore, important implementational issues were summarized.

Current efforts are directed at porting all compressible flow modules to account for volume blockage effects, and the link to chemical reactions with burning particles.

**Acknowledgments** This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725, and also resources of the DoD High Performance Computing Modernization Program. This support is gratefully acknowledged.

## References

1. Appanaboyina S, Mut F, Löhner R, Putman CM, Cebra JR (2008) Computational fluid dynamics of stented intracranial aneurysms using adaptive embedded unstructured grids. *Int J Numer Methods Fluids* 57(5):475–493
2. Aubry R, Mut F, Löhner R, Cebra JR (2008) Deflated preconditioned conjugate gradient solvers for the pressure–poisson equation. *J Comput Phys* 227(24):10196–10208
3. Baum JD, Löhner R (1991) Numerical simulation of shock interaction with a modern main battlefield tank AIAA-91-1666.
4. Baum JD, Löhner R (1993) Numerical simulation of pilot/seat ejection from an F-16 AIAA-93-0783.
5. Baum JD, Luo H, Löhner R (1993) Numerical simulation of a blast inside a boeing 747; AIAA-93-3091.
6. Baum JD, Luo H, Löhner R (1993) Numerical simulation of a blast withing a multi-room shelter. In: *Proceedings of the MABS-13 conference*. The Hague, pp 451–463
7. Baum JD, Luo H, Löhner R (1994) A new ALE adaptive unstructured methodology for the simulation of moving bodies; AIAA-94-0414
8. Baum JD, Luo H, Löhner R (1995) Numerical simulation of blast in the world trade center; AIAA-95-0085
9. Baum JD, Luo H, Löhner R (1995) Validation of a new ALE, adaptive unstructured moving body methodology for multi-store ejection simulations; AIAA-95-1792
10. Baum JD, Luo H, Löhner R, Yang C, Pelessone D, Charman C (1996) Coupled fluid/structure modeling of shock interaction with a truck; AIAA-96-0795
11. Baum JD, Luo H, Löhner R, Goldberg E, Feldhun A (1997) Application of unstructured adaptive moving body methodology to the simulation of fuel tank separation from an F-16 c/d fighter; AIAA-97-0166
12. Baum JD, Löhner R, Marquette TJ, Luo H (1997) Numerical simulation of aircraft canopy trajectory; AIAA-97-1885
13. Baum JD, Luo H, Mestreau E, Löhner R, Pelessone D, Charman C (1999) Coupled CFD/CSD methodology for modeling weapon detonation and fragmentation; AIAA-99-0794
14. Baum JD, Mestreau E, Luo H, Löhner R, Pelessone D, Giltrud ME, Gran JK (2006) Modeling of near-field blast wave, evolution; AIAA-06-0191
15. Balakrishnan K, Menon S (2010) On the role of ambient reactive particles in the mixing and afterburn behind explosive blast waves. *Combust Sci Technol* 182:186–214
16. Benkiewicz K, Hayashi K (2003) Two-dimensional numerical simulations of multi-headed detonations in oxygen–aluminum mixtures using an adaptive mesh refinement. *Shock Waves* 12(5):385–402
17. Camelli F, Löhner R (2004) Assessing maximum possible damage for contaminant release events. *Eng Comput* 21(7):748–760
18. Camelli F, Löhner R, Sandberg WC, Ramamurti R (2004) VLES study of ship stack gas, dynamics; AIAA-04-0072
19. Camelli F, Löhner R (2006) VLES study of flow and dispersion patterns in heterogeneous urban areas; AIAA-06-1419.
20. Camelli F, Lien J, Dayong D, Wong DW, Rice M, Löhner R, Yang C (2012) Generating seamless surfaces for transport and dispersion modeling in GIS. *GeoInformatica* 16(2):207–327
21. Cebra JR, Löhner R (2001) From medical images to anatomically accurate finite element grids. *Int J Numer Methods Eng* 51:985–1008
22. Cebra JR, Löhner R (2005) Efficient simulation of blood flow past complex endovascular devices using an adaptive embedding technique. *IEEE Trans Med Imaging* 24(4):468–476
23. Clift R, Grace JR, Weber ME (1978) *Bubbles, drops and particles*. Academic Press, New York
24. Corrigan A, Camelli F, Löhner R (2010) Porting of an edge-based CFD solver to GPUs; AIAA-10-0523.
25. Corrigan A, Camelli F, Löhner R, Mut F (2010) Porting of FEFLO to GPUs. In: *Proceedings of the ECCOMAS CFD 2010 conference*. Lisbon, June 14–17.
26. Corrigan A, Camelli FF, Löhner R, Wallin J (2011) Running unstructured grid based CFD solvers on modern graphics hardware. *Int J Numer Methods Fluids* 66:221–229
27. Corrigan A, Löhner R (2011) Porting of FEFLO to multi-GPU clusters; AIAA-11-0948.
28. Deen NG, Sint Annaland Mv, Kuipers JAM (2006) Direct numerical simulation of particle mixing in dispersed gas–liquid–solid flows using a combined volume of fluid and discrete particle approach. In: *Proceedings of the fifth international conference on CFD in the process industries CSIRO*, Melbourne, 13–15 Dec.
29. Hoberock J, Bell N (2010) Thrust: parallel template library. Version 1:3
30. Kim CK, Moon JG, Hwang JS, Lai MC, Im KS (2008) Afterburning of TNT explosive products in air with aluminum particles; AIAA-2008-1029.
31. Liu J, Kailasanath K, Ramamurti R, Munday D, Gutmark E, Löhner R (2009) Large-eddy simulations of a supersonic jet and its near-field acoustic properties. *AIAA J* 47(8):1849–1864
32. Löhner R, Morgan K, Peraire J, Vahdati M (1987) Finite element flux-corrected transport (FEM-FCT) for the Euler and Navier–Stokes equations. *Int J Numer Methods Fluids* 7:1093–1109
33. Löhner R, Ambrosiano J (1990) A vectorized particle tracer for unstructured grids. *J Comp Phys* 91(1):22–31
34. Löhner R, Ramamurti R (1995) A load balancing algorithm for unstructured grids. *Comput Fluid Dyn* 5:39–58
35. Löhner R (1998) Renumbering strategies for unstructured-grid solvers operating on shared-memory, cache-based parallel machines. *Comput Methods Appl Mech Eng* 163:95–109
36. Löhner R, Yang C, Oñate E (1998) Viscous free surface hydrodynamics using unstructured grids. In: *Proceedings of the 22nd Symposium Naval Hydrodynamics*. Washington, D.C., Aug
37. Löhner R., Yang Chi, Cebra J, Soto O, Camelli F, Baum JD, Luo H, Mestreau E, Sharov D, Ramamurti R, Sandberg W, Oh Ch (2001) *Advances in FEFLO*; AIAA-01-0592.
38. Löhner R, Galle M (2002) Minimization of indirect addressing for edge-based field solvers. *Commun Numer Methods Eng* 18:335–343
39. Löhner R (2004) Multistage explicit advective prediction for projection-type incompressible flow solvers. *J Comput Phys* 195:143–152
40. Löhner R, Baum JD, Rice D (2004) Comparison of coarse and fine mesh 3D Euler predictions for blast loads on generic building configurations. In: *Proceedings of the MABS-18 conference* bad. Reichenhall, Germany, Sept
41. Löhner R, Camelli F (2005) Optimal placement of sensors for contaminant detection based on detailed 3D CFD simulations. *Eng Comput* 22(3):260–273

42. Löhner R, Chi Yang JR, Camelli O Soto (2006) Improving the speed and accuracy of projection-type incompressible flow solvers. *Comput Methods Appl Mech Eng* 195(23–24):3087–3109
43. Löhner R, Yang Chi (2006) On the simulation of flows with violent free surface motion. *Comput Methods Appl Mech Eng* 195:5597–5620
44. Löhner R, Yang Chi (2007) Simulation of flows with violent free surface motion and moving objects using unstructured grids. *Int J Numer Methods Fluids* 53:1315–1338
45. Löhner R (2008) *Applied CFD techniques*, 2nd edn. Wiley, New York
46. Löhner R, Luo H, Baum JD, Rice D (2008) Improvements in speed for explicit, transient compressible flow solvers. *Int J Numer Methods Fluids* 56(12):2229–2244
47. Löhner R, Cebral JR, Camelli FF, Appanaboyina S, Baum JD, Mestreau EL, Soto O (2008) Adaptive embedded and immersed unstructured grid techniques. *Comput Methods Appl Mech Eng* 197:2173–2197
48. Löhner R, Mut F, Camelli FF (2011) Timings OF FEFLO on the SGI-ICE machines; AIAA-11-1064.
49. Löhner R, Corrigan A (2011) Semi-automatic porting of a general fortran CFD code to GPUs: the difficult modules; AIAA-11-3219.
50. Löhner R, Mut F, Cebral JR, Aubry R, Houzeaux G (2011) Deflated preconditioned conjugate gradient solvers for the pressure–poisson equation: extensions and improvements. *Int J Numer Methods Eng* 87(1–5):2–14
51. Löhner R (2012) F2GPU a general fortran to GPU translator. In: *Proceedings of the NVIDIA GTC conference*. San Jose, May
52. Luo H, Baum JD, Löhner R (1994) Edge-based finite element scheme for the Euler equations. *AIAA J* 32(6):1183–1190
53. Luo H, Baum JD, Löhner R, Cabello J (1994) Implicit finite element schemes and boundary conditions for compressible flows on unstructured grids; AIAA-94-0816.
54. Luo H, Baum JD, Löhner R (1999) An accurate, fast, matrix-free implicit method for computing unsteady flows on unstructured grids; AIAA-99-0937.
55. Luo H, Sharov D, Baum JD, Löhner R (2000) A class of matrix-free implicit methods for compressible flows on unstructured grids. In: *First international conference on computational fluid dynamics*, Kyoto, July 10–14
56. Luo H, Baum JD, Löhner R (2001) A fast, matrix-free implicit method for computing low mach number flows on unstructured grids. *Int J CFD* 14:133–157
57. Merrill D, Grimshaw A (2010) Revisiting sorting for GPGPU stream architectures. UVA CS Report CS2010-03 Charlottesville.
58. NVIDIA Corporation. *NVIDIA CUDA 3.2 Programming Guide* (2010).
59. Peterson P (2009) F2PY: tool for connecting Fortran and python programs. *Int J Comput Sci Eng* 4:296–305
60. Ramamurti R, Löhner R (1993) Simulation of flow past complex geometries using a parallel implicit incompressible flow solver. In: *Proceedings of the 11th AIAA CFD conference*, Orlando, pp 1049–1050
61. Ramamurti R, Löhner R (1996) A parallel implicit incompressible flow solver using unstructured meshes. *Comput Fluids* 5:119–132
62. Ramamurti R, Sandberg WC, Löhner R (1999) Computation of unsteady flow past deforming geometries. *Int J Comput Fluid Dyn* 13:83–99
63. Rice DL, Baum JD, Togashi F, Löhner R, Amini A (2008) First-principles blast diffraction simulations on a notebook: accuracy, resolution and turn-around issues. In: *Proceedings of the MABS-20 conference*, Oslo
64. Sharov D, Luo H, Baum JD, Löhner R (2000) Implementation of unstructured grid GMRES+LU-SGS method on shared-memory, cache-based parallel computers; AIAA-00-0927.
65. Stück A, Camelli F, Löhner R (2010) Adjoint-based design of shock mitigation devices. *Int J Numer Methods Fluids* 64:443–472
66. Tilch R, Tabbal A, Zhu M, Decker F, Löhner R (2008) Combination of body-fitted and embedded grids for external vehicle aerodynamics. *Eng Comput* 25(1):28–41
67. Togashi F, Baum JD, Mestreau E, Löhner R, Sunshine D (2009) Numerical modeling of long-duration blast wave evolution in confined facilities; AIAA-09-1531.
68. Tuszynski J, Löhner R (1998) Parallelizing the construction of indirect access arrays for shared-memory machines. *Commun Appl Numer Methods Eng* 14:773–781
69. Vreman AW, Geurts BJ, Deen NG, Kuipers JAM (2004) Large-eddy simulation of a particle laden turbulent channel flow. In: *Proceedings of the direct and large-eddy simulation V. ERCOFTAC Series* 9, pp 271–278
70. Zhang F (ed.) (2009) *Shock wave science and technology reference library*, vol. 4: heterogeneous detonation. Springer, New York.