A R C H I V E O F M E C H A N I C A L E N G I N E E R I N G

VOL. LXI 2014 Number 2

10.2478/meceng-2014-0014

Key words: fluid-solid interaction, high performance computing, smoothed particle hydrodynamics, rigid body dynamics, flexible body dynamics

ARMAN PAZOUKI[*]*, RADU SERBAN*[*]*, DAN NEGRUT*[*]

A HIGH PERFORMANCE COMPUTING APPROACH TO THE SIMULATION OF FLUID-SOLID INTERACTION PROBLEMS WITH RIGID AND FLEXIBLE COMPONENTS

This work outlines a unified multi-threaded, multi-scale High Performance Computing (HPC) approach for the direct numerical simulation of Fluid-Solid Interaction (FSI) problems. The simulation algorithm relies on the extended Smoothed Particle Hydrodynamics (XSPH) method, which approaches the fluid flow in a Lagrangian framework consistent with the Lagrangian tracking of the solid phase. A general 3D rigid body dynamics and an Absolute Nodal Coordinate Formulation (ANCF) are implemented to model rigid and flexible multibody dynamics. The two-way coupling of the fluid and solid phases is supported through use of Boundary Condition Enforcing (BCE) markers that capture the fluid-solid coupling forces by enforcing a no-slip boundary condition. The solid-solid short range interaction, which has a crucial impact on the small-scale behavior of fluid-solid mixtures, is resolved via a lubrication force model. The collective system states are integrated in time using an explicit, multi-rate scheme. To alleviate the heavy computational load, the overall algorithm leverages parallel computing on Graphics Processing Unit (GPU) cards. Performance and scaling analysis are provided for simulations scenarios involving one or multiple phases with up to tens of thousands of solid objects. The software implementation of the approach, called Chrono::Fluid, is part of the Chrono project and available as an open-source software.

1. Introduction

The last decade witnessed a paradigm shift in the microprocessor industry towards chip designs that provide strong support for parallel computing. Teraflop computing, i.e. computing at the rate of 10^{12} FLoating-point Operation Per Second, until recently the privilege of a select group of large research centers, is becoming a commodity due to inexpensive GPU cards and multi-

[*] *University of Wisconsin-Madison, 1513 University Avenue, Madison, WI-53706, USA; E-mail: pazouki@wisc.edu, serban@wisc.edu, negrut@wisc.edu*

to many-core x86 processors. The steady improvements in processor speed and architecture, memory layout and capacity, and power efficiency have motivated a trend of re-evaluating simulation algorithms with an eye towards identifying solutions that map well onto these new hardware platforms. In this context, a scrutinizing of the existing Fluid-Solid Interaction (FSI) solutions reveals that they are mostly inadequate since they either rely on algorithms that do not map well on emerging computer architectures, or they are unable to capture phenomena of interest which may require the simulation of tens of thousands of rigid and deformable bodies that interact directly or through the fluid media.

FSI problems are usually simulated in either an Eulerian-Lagrangian framework, where the Lagrangian solid phase moves with/over the Eulerian grid used for fluid discretization, or an Eulerian-Eulerian framework, where the solid phase is considered as an average of ensembles instead of a specific state. While the former approach delivers the flexibility required by general multibody dynamics problems, it does not provide a level of performance suitable for the simulation of dense fluid-solid problems such as suspensions. This can be traced back to the costly processes that overlaps the two different representations of the fluid and solid phases.

The Lagrangian representation of the fluid flow dovetails well with the Lagrangian approach used for the motion of the solid phase. This approach has recently been employed in the context of multibody dynamics and particle suspension in [12, 25, 29-31, 33]. Specifically, the FSI methodology relying on Smoothed Particle Hydrodynamics (SPH) [16, 19, 22] and rigid body Newton-Euler equation of motion [10] was discussed in detail in [29, 30]. This or a similar framework was used for the investigation and validation of rigid body suspensions [30], as well as flexible beams interacting with fluid [12, 31, 33]. Additionally, support for the short-range solid-solid interaction was introduced in [30, 31], which was leveraged in the simulation of dense suspensions.

This contribution concentrates on the high performance computing approach required for the efficient simulation of FSI problems. Although the approach and algorithms explained herein can leverage any multi-threaded architecture, the CUDA library [26] was employed for the execution of all solution components on the GPU, with negligible data transfer between the host (CPU) and the device (GPU). The paper is organized as follows. The numerical methods adopted for the simulation of fluid and solid phases, fluid-solid coupling, and solid-solid short range interaction are explained in Sect. 2. The computational scheme, including the time integration algorithm, HPC-based implementation, and parallel neighbor search required for SPH

are described in Sect. 3. Section 4 contains a performance analysis of the algorithm using Kepler-type GPU cards.

2. Numerical approach

The proposed FSI simulation framework combines SPH for the simulation of the fluid flow, the Newton-Euler formalism for rigid body dynamics, and ANCF for modeling flexible body dynamics. These algorithmic components are described in more detail in the following subsections, including a discussion on the approach adopted for resolving fluid-solid and solid-solid interactions.

2.1. Smoothed Particle Hydrodynamics

The term *smoothed* in SPH refers to the approximation of point properties via a smoothing kernel function W, defined over a support domain S. This approximation reproduces functions with up to 2^{nd} order of accuracy, provided the kernel function: (1) approaches the Dirac delta function as the size of the support domain tends to zero, that is $\lim_{h\to 0} W(\mathbf{r}, h) = \delta(\mathbf{r})$, where $\mathbf{r}$ is the spatial distance and h is a characteristic length that defines the kernel smoothness; (2) is symmetric, i.e., $W(\mathbf{r}, h) = W(-\mathbf{r}, h)$; and (3) is normal, i.e., $\int_S W(\mathbf{r}, h)\mathrm{d}\mathbb{V} = 1$, where $\mathrm{d}\mathbb{V}$ denote the differential volume. A typical spatial function $f(\mathbf{x})$ is then approximated by $\langle f(\mathbf{x})\rangle$ as

$$\begin{aligned} f(\mathbf{x}) &= \int_S f(\mathbf{x}')\delta(\mathbf{x}-\mathbf{x}')\mathrm{d}\mathbb{V} \\ &= \int_S f(\mathbf{x}')W(\mathbf{x}-\mathbf{x}', h)\mathrm{d}\mathbb{V} + O(h^2) \\ &= \langle f(\mathbf{x})\rangle + O(h^2)\ . \end{aligned} \tag{1}$$

To simplify notation, in the remainder of this document we use $f(\mathbf{x})$ to represent $\langle f(\mathbf{x})\rangle$. Using Eq. (1) and the divergence theorem, the spatial derivatives of a function can be mapped to the derivatives of the kernel function. For instance, the gradient of a function can be written as

$$\begin{aligned} \nabla f(\mathbf{x}) = &\int_{\partial S} f(\mathbf{x}')W(\mathbf{x}-\mathbf{x}', h)\cdot\mathbf{n}\mathrm{d}\mathbb{A} \\ &- \int_S f(\mathbf{x}')\nabla W(\mathbf{x}-\mathbf{x}', h)\mathrm{d}\mathbb{V}\ , \end{aligned} \tag{2}$$

where ∂S is the boundary of the integration volume $\mathbb{V}$ and $\mathrm{d}\mathbb{A}$ is the differential area. By imposing an additional property for the kernel function, namely that (4) it approaches zero as $\mathbf{r}$ increases, i.e., $\lim_{\mathbf{r}\to\infty} W(\mathbf{r}, h) = 0$, the first term on the right hand side of Eq. (2) vanishes. Note that some additional considerations, which will be addressed later, are required for the SPH approximation near boundaries.

The term *particle* in SPH terminology indicates the discretization of the domain by a set of Lagrangian particles. To remove the ambiguity caused by the use of the term *rigid particles* in the context of FSI problems, we use the term *marker* to refer to the SPH discretization unit. Each marker has mass m associated to the representative volume $\mathrm{d}\mathbb{V}$ and carries all of the essential field properties. As a result, any field property at a certain location is shared and represented by the markers in the vicinity of that location. The value of a certain unknown at a given location is calculated according to the distance between that location and the collection of markers in its vicinity using the field values at these markers and the expression of the kernel function W. This leads for the second approximation embedded in SPH, which can be expressed as

$$\begin{aligned} f(\mathbf{x}) &= \int_S \frac{f(\mathbf{x}')}{\rho(\mathbf{x}')} W(\mathbf{x}-\mathbf{x}', h)\rho(\mathbf{x}')\mathrm{d}\mathbb{V} \\ &\simeq \sum_b \frac{m_b}{\rho_b} f(\mathbf{x}_b) W(\mathbf{x}-\mathbf{x}_b, h)\ , \end{aligned} \tag{3}$$

where b is the marker index and ρ_b is the fluid density, smoothed at the marker location $\mathbf{x}_b$. The summation in Eq. (3) is over all markers whose support domain overlaps the location $\mathbf{x}$. Several other properties of the kernel functions are provided in [16]. For instance, the kernel function should be a positive and monotonically decreasing function of $\mathbf{r}$, which implies that the influence of distant markers on field properties at a given location is less than that of nearby markers. Moreover, for acceptable computational performance and to avoid a quadratic computational complexity, kernel functions have a compact domain of influence with a radius R_s defined as some finite multiple κ of the characteristic length h, as shown in Figure 1. The methodology adopted herein relies on a cubic spline,

$$W(q,h) = \frac{1}{4\pi h^3} \times \begin{cases} (2-q)^3 - 4(1-q)^3, & 0 \le q < 1 \\ (2-q)^3, & 1 \le q < 2 \\ 0, & q \ge 2 \end{cases}, \tag{4}$$

where $q \equiv |\mathbf{r}|/h$, has a support domain with radius $2h$.

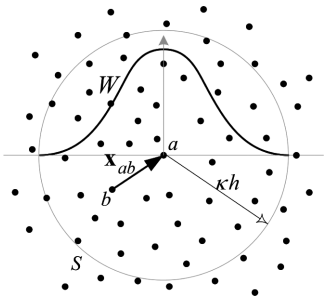

Fig. 1. Illustration of the kernel W and its support domain S. SPH markers are shown as black dots. For 2D problems the support domain is a circle, while for 3D problems it is a sphere

2.1.1. SPH for fluid dynamics

For fluid dynamics, the continuity and momentum equations are given in Lagrangian form as

$$\frac{d\rho}{dt} = -\rho\nabla\cdot\mathbf{v} \tag{5}$$

and

$$\frac{d\mathbf{v}}{dt} = -\frac{1}{\rho}\nabla p + \frac{\mu}{\rho}\nabla^2\mathbf{v} + \mathbf{f}\ , \tag{6}$$

where μ is the fluid viscosity, while $\mathbf{v}$ and p are the flow velocity and pressure, respectively. Within the SPH framework described earlier, Eqs. (5) and (6) are discretized at an arbitrary location $\mathbf{x} = \mathbf{x}_a$ within the fluid domain as [24]

$$\dot{\rho}_a = \frac{d\rho_a}{dt} = \rho_a \sum_b \frac{m_b}{\rho_b}\left(\mathbf{v}_a - \mathbf{v}_b\right)\cdot\nabla_a W_{ab}\ , \tag{7}$$

and

$$\dot{\mathbf{v}}_a = \frac{d\mathbf{v}_a}{dt} = -\sum_b m_b\left[\left(\frac{p_a}{{\rho_a}^2} + \frac{p_b}{{\rho_b}^2}\right)\nabla_a W_{ab} + \Pi_{ab}\right] + \mathbf{f}_a\,. \tag{8}$$

In the above equations, quantities with subscripts a and b are associated with markers a and b (see Figure 1), respectively. It is important to note that these quantities are different from the corresponding physical quantities at locations $\mathbf{x}_a$ and $\mathbf{x}_b$. The viscosity term Π_{ab} is defined as

$$\Pi_{ab} = -\frac{(\mu_a + \mu_b)\mathbf{x}_{ab}\cdot\nabla_a W_{ab}}{\bar{\rho}_{ab}^2(x_{ab}^2 + \varepsilon\bar{h}_{ab}^2)}\mathbf{v}_{ab}\ , \tag{9}$$

where $\mathbf{x}_{ab} = \mathbf{x}_a - \mathbf{x}_b$, $W_{ab} = W(\mathbf{x}_{ab}, h)$, ∇_a is the gradient with respect to $\mathbf{x}_a$, i.e., $\partial/\partial\mathbf{x}_a$, and ε is a regularization coefficient. Quantities with an over-bar are averages of the corresponding quantities for markers a and b.

Alternative viscosity discretizations include:

1. the model suggested in [5]:
   $\Pi^*_{ab} = -\mu_a\mu_b \mathbf{x}_{ab}\cdot\mathbf{v}_{ab}/[\rho_a\rho_b(\mu_a+\mu_b)(x^2_{ab}+\varepsilon\bar{h}^2_{ab})]\nabla_a W_{ab}$;
2. direct discretization of ∇^2 operator [22, 24]; and
3. the class of artificial viscosity models introduced in [17, 19, 23]

However, using Eq. 9 is preferred since it has the following properties: (1) it ensures that the viscous force is along the shear direction $\mathbf{v}_{ab}$, instead of the particles center line $\mathbf{x}_{ab}$; (2) it is less sensitive to local velocities; (3) it allows for better computational efficiency by removing the nested loop required for the computation of the ∇^2 operator; and (4) it is stated in terms of physical properties, rather than model parameters like those in artificial viscosity, which are introduced primarily for numerical stabilization through damping. In the simulation of transient Poiseuille flow, although we managed to virtually obtain exact results using Eq. 9 [30], the error caused by implementing either Π^*_{ab} or artificial viscosity were non-negligible.

In the weakly compressible SPH model, the pressure p is evaluated using an equation of state [22]

$$p = \frac{{c_s}^2\rho_0}{\gamma}\left\{\left(\frac{\rho}{\rho_0}\right)^{\gamma} - 1\right\}, \tag{10}$$

where ρ_0 is the reference density of the fluid, γ tunes the stiffness of the pressure-density relationship, and c_s is the speed of sound. The value c_s is adjusted depending on the maximum speed of the flow, $V_{\max}$, to keep the flow compressibility below some arbitrary value. We use $\gamma = 7$ and $c_s = 10V_{\max}$, which allows 1% flow compressibility [22].

The fluid flow equations (7) and (8) are solved together with

$$\dot{\mathbf{x}}_a = \frac{d\mathbf{x}_a}{dt} = \mathbf{v}_a \tag{11}$$

to update the position of the SPH markers.

The original SPH summation formula calculates the density according to

$$\rho_a = \sum_b m_b W_{ab}. \tag{12}$$

Equation (7), which evaluates the time derivative of the density, was preferred to the above since it produces a smooth density field, works well for markers close to the boundaries (the free surface, solid, and wall), and does not exhibit the large variations in the density field introduced when using Eq. (12) close to the boundaries. However, Eq. (7) does not guarantee consistency between

a marker's density and the associated mass and volume [3, 21, 24]. The so-called *density re-initialization technique* [6] attempts to address this issue by using Eq. (7) at each time step and periodically; i.e., every n time steps, use Eq. (12) to correct the mass-density inconsistency. The results reported herein were obtained with $n = 10$. The Moving Least Squares method or a normalized version of Eq. (12) could alternatively be used to address the aforementioned issues, see [6, 7].

Finally, the methodology proposed employs the extended SPH approach (XSPH), which prevents extensive overlap of markers' support domain and enhances flow incompressibility [20]. This correction takes into account the velocity of neighboring markers through a mean velocity evaluated within the support of a nominal marker a as

$$\hat{\mathbf{v}}_a = \mathbf{v}_a + \Delta\mathbf{v}_a, \tag{13}$$

where

$$\Delta\mathbf{v}_a = \zeta \sum_b \frac{m_b}{\bar{\rho}_{ab}}(\mathbf{v}_b - \mathbf{v}_a)W_{ab}, \tag{14}$$

and $0 \leq \zeta \leq 1$ adjusts the contribution of the neighbors' velocities. All the results reported herein were obtained with $\zeta = 0.5$. The modified velocity calculated from Eq. (13) replaces the original velocity in the density and position update equations, but not in the momentum equation [6].

2.2. Rigid body dynamics

Once the fluid-solid interactions between individual markers; i.e., the right hand side of Eqs. (7) and (8), are accounted for, the total rigid body force and torque due to the interaction with the fluid can be obtained by respectively summing the individual forces and their induced torques over the entire rigid body. They are then added to any other forces, including external and contact forces. The dynamics of the rigid bodies is fully characterized by the Newton-Euler equations of motion (EOM), see for instance [10]:

$$\frac{d\mathbf{V}_i}{dt} = \frac{\mathbf{F}_i}{M_i}, \tag{15}$$

$$\frac{d\mathbf{X}_i}{dt} = \mathbf{V}_i, \tag{16}$$

$$\frac{d\boldsymbol{\omega}'_i}{dt} = \mathbf{J}'^{-1}_i \left(\mathbf{T}'_i - \tilde{\boldsymbol{\omega}}'_i \mathbf{J}'_i \boldsymbol{\omega}'_i\right), \tag{17}$$

$$\frac{d\mathbf{q}_i}{dt} = \frac{1}{2}\mathbf{G}_i^T \boldsymbol{\omega}'_i, \tag{18}$$

and

$$\mathbf{q}_i^T \mathbf{q}_i - 1 = 0, \tag{19}$$

where $\mathbf{F}_i$, $\mathbf{T}'_i$, $\mathbf{X}_i$, $\mathbf{V}_i$, $\boldsymbol{\omega}'_i \in \mathbb{R}^3$ denote the force, torque, position, velocity, and angular velocity associated to body i, $i = 1, 2, \ldots, n_b$, respectively. The quantity $\mathbf{q}_i$ denotes the rotation quaternion, while M_i and $\mathbf{J}'_i$ are the mass and moment of inertia, respectively. Quantities with a prime symbol are represented in the rigid body local reference frame. Given $\mathbf{a} = \left[a_x,\, a_y,\, a_z\right]^T \in$ $\in \mathbb{R}^3$ and $\mathbf{q} = \left[q_x,\, q_y,\, q_z,\, q_w\right]^T \in \mathbb{R}^4$, the auxiliary matrices $\tilde{\mathbf{a}}$ and $\mathbf{G}$ are defined as [10]

$$\tilde{\mathbf{a}} = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \quad \text{and} \quad \mathbf{G} = \begin{bmatrix} -q_y & q_x & q_w & -q_z \\ -q_z & -q_w & q_x & q_y \\ -q_w & q_z & -q_y & q_x \end{bmatrix}. \tag{20}$$

2.3. Flexible body dynamics

The ANCF formulation [36], which allows for large deformations and large rigid body rotations, is adopted herein for the simulation of flexible bodies suspended in the fluid. While extension to other elastic elements is straightforward, the current Chrono::Fluid implementation only supports gradient deficient ANCF beam elements, which are used to model slender flexible bodies composed of n_e adjacent ANCF beam elements. The flexible bodies are modeled using a number $n_n = n_e + 1$ of equally-spaced node beam elements, each represented by 6 coordinates, $\mathbf{e}_j = [\mathbf{r}_j^T,\ \mathbf{r}_{j,x}^T]^T$, $j = 0, 1, \ldots, n_e$; i.e., the three components of the global position vector of the node, and the three components of its slope. This is therefore equivalent to a model using n_e ANCF beam elements with $6 \times n_n$ continuity constraints, but is more efficient in that it uses a minimal set of coordinates. We note that formulations using gradient deficient ANCF beam elements display no shear locking problems [9, 34, 35] and, due to the reduced number of nodal coordinates, are more efficient than fully parametrized ANCF elements. However, gradient deficient ANCF beam elements cannot describe a rotation about its axis and therefore cannot model torsional effects.

Consider first a single ANCF beam element of length ℓ. The global position vector of an arbitrary point on the beam center line, specified through its element spatial coordinate $0 \leq x \leq \ell$, is then obtained as

$$\mathbf{r}(x, \mathbf{e}) = \mathbf{S}(x)\mathbf{e}\,, \tag{21}$$

where $\mathbf{e} = [\mathbf{e}_l^T,\ \mathbf{e}_r^T]^T \in \mathbb{R}^{12}$ is the vector of element nodal coordinates. With $\mathbf{I}$ being the 3×3 identity matrix, the shape function matrix $\mathbf{S} = [S_1\mathbf{I} \quad S_2\mathbf{I} \quad S_3\mathbf{I} \quad S_4\mathbf{I}] \in \mathbb{R}^{3\times 12}$ is defined using the shape functions [36]

$$\begin{aligned} S_1 &= 1 - 3\xi^2 + 2\xi^3 \\ S_2 &= \ell\left(\xi - 2\xi^2 + \xi^3\right) \\ S_3 &= 3\xi^2 - 2\xi^3 \\ S_4 &= \ell\left(-\xi^2 + \xi^3\right), \end{aligned} \tag{22}$$

where $\xi = x/\ell \in [0, 1]$.

The element EOM are then written as

$$\mathbf{M}\ddot{\mathbf{e}} + \mathbf{Q}^e = \mathbf{Q}^a, \tag{23}$$

where $\mathbf{Q}^e$ and $\mathbf{Q}^a$ are the generalized element elastic and applied forces, respectively, and $\mathbf{M} \in \mathbb{R}^{12\times 12}$ is the symmetric consistent element mass matrix defined as

$$\mathbf{M} = \int_\ell \rho_s A \mathbf{S}^T \mathbf{S} dx. \tag{24}$$

The generalized element elastic forces are obtained from the strain energy expression [36] as

$$\mathbf{Q}^e = \int_\ell EA\varepsilon_{11}\left(\frac{\partial \varepsilon_{11}}{\partial \mathbf{e}}\right)^T dx + \int_\ell EI\kappa\left(\frac{\partial \kappa}{\partial \mathbf{e}}\right)^T dx, \tag{25}$$

where $\varepsilon_{11} = \left(\mathbf{r}_x^T\mathbf{r}_x - 1\right)/2$ is the axial strain and $\kappa = \|\mathbf{r}_x \times \mathbf{r}_{xx}\|/\|\mathbf{r}_x\|^3$ is the magnitude of the curvature vector. The required derivatives of the position vector $\mathbf{r}$ can be easily obtained from Eq. (21) in terms of the derivatives of the shape functions as $\mathbf{r}_x(x, \mathbf{e}) = \mathbf{S}_x(x)\mathbf{e}$ and $\mathbf{r}_{xx}(x, \mathbf{e}) = \mathbf{S}_{xx}(x)\mathbf{e}$.

External applied forces, in particular the forces due to the interaction with the fluid, see Sect. 2.4, are included as concentrated forces at a BCE marker. The corresponding generalized forces are obtained from the expression of the virtual work as

$$\mathbf{Q}^a = \mathbf{S}^T(x_a)\mathbf{F}, \tag{26}$$

where $\mathbf{F}$ is the external point force and the shape function matrix is evaluated at the projection onto the element's center line of the force application point. The generalized gravitational force can be computed as

$$\mathbf{Q}^g = \int_\ell \rho_s A \mathbf{S}^T \mathbf{g} dx. \tag{27}$$

In the above expressions, ρ_s represents the element mass density, A is the cross section area, E is the modulus of elasticity, and I is the second moment of area.

The EOM for a slender flexible body composed of n_e ANCF beam elements are obtained by assembling the elemental EOMs of Eq. (23) and taking into consideration that adjacent beam elements share 6 nodal coordinates.

Let $\hat{\mathbf{e}} = [\mathbf{e}_0^T\ ,\ \mathbf{e}_1^T\ ,\ \dots\ \mathbf{e}_{n_e}^T]^T$ be the set of independent nodal coordinates; then the nodal coordinates for the j-th element can be written using the mapping

$$\begin{bmatrix}\mathbf{e}_l\\ \mathbf{e}_r\end{bmatrix}_j = \mathbf{B}_j\hat{\mathbf{e}}\,,\ \text{with}\ \mathbf{B}_j = \begin{bmatrix}\mathbf{0} & \mathbf{0} & \dots\mathbf{I}_3 & \mathbf{0} & \dots\mathbf{0}\\ \mathbf{0} & \mathbf{0} & \dots\mathbf{0} & \mathbf{I}_3 & \dots\mathbf{0}\end{bmatrix} \tag{28}$$

and the assembled EOMs are obtained, from the principle of virtual work, as follows. Denoting by $\mathbf{M}_j$ be the element mass matrix of Eq. (24) for the j-th ANCF beam element, it can be written in block form as

$$\mathbf{M}_j = \begin{bmatrix}\mathbf{M}_{j,ll} & \mathbf{M}_{j,lr}\\ \mathbf{M}_{j,rl} & \mathbf{M}_{j,rr}\end{bmatrix}, \tag{29}$$

where $\mathbf{M}_{j,lr} = \mathbf{M}_{j,rl}^T$ and all sub-blocks have dimension 6×6. Here, l denotes the *left* end of the beam element, i.e., the node characterized by the nodal coordinates $\mathbf{e}_{j-1}$, while r corresponds to the node with coordinates $\mathbf{e}_j$. With a similar decomposition of a generalized element force into

$$\mathbf{Q}_j = \begin{bmatrix}\mathbf{Q}_{j,l}\\ \mathbf{Q}_{j,r}\end{bmatrix} \tag{30}$$

one obtains

$$\hat{\mathbf{M}}\ddot{\hat{\mathbf{e}}} = \hat{\mathbf{Q}}^a - \hat{\mathbf{Q}}^e \tag{31}$$

where

$$\hat{\mathbf{M}} = \begin{bmatrix}\mathbf{M}_{1,ll} & \mathbf{M}_{1,lr} & & & \\ \mathbf{M}_{1,rl} & \mathbf{M}_{1,rr}+\mathbf{M}_{2,ll} & & & \\ & \mathbf{M}_{2,rl} & \ddots & & \\ & & & & \mathbf{M}_{n_e,rr}\end{bmatrix} \tag{32}$$

$$\hat{\mathbf{Q}}^a - \hat{\mathbf{Q}}^e = \begin{bmatrix}\sum\mathbf{Q}_{1,l}^a\\ \sum\mathbf{Q}_{1,r}^a + \sum\mathbf{Q}_{2,l}^a\\ \sum\mathbf{Q}_{2,r}^a + \sum\mathbf{Q}_{3,l}^a\\ \vdots\\ \sum\mathbf{Q}_{n_e,r}^a\end{bmatrix} - \begin{bmatrix}\mathbf{Q}_{1,l}^e\\ \mathbf{Q}_{1,r}^e + \mathbf{Q}_{2,l}^e\\ \mathbf{Q}_{2,r}^e + \mathbf{Q}_{3,l}^e\\ \vdots\\ \mathbf{Q}_{n_e,r}^e\end{bmatrix}. \tag{33}$$

Inclusion of additional kinematic constraints, e.g., anchoring the beam at one end to obtain a flexible cantilever or fixing its position to obtain a flexible pendulum, can be done either by formulating the EOM as differential-algebraic equations or by deriving an underlying ODE after explicitly eliminating the corresponding constrained nodal coordinates. The latter approach was used in all simulations involving flexible cantilevers discussed in Sect. 4.

2.4. Fluid-solid interaction

The two-way fluid-solid coupling was implemented based on a methodology described in [29]. The state update of any SPH marker relies on the properties of its neighbors and resolves shear as well as normal inter-marker forces. For the SPH markers close to solid surfaces, the SPH summations presented in Eqs. (7), (8), (12), and (14) capture the contribution of fluid markers. The contribution of solid objects is calculated via Boundary Condition Enforcing (BCE) markers placed on and close to the solid surface as shown in Figure 2. The velocity of a BCE marker is obtained from the rigid/deformable body motion of the solid thus ensuring the no-slip condition on the solid surface. Including BCE markers in the SPH summation equations (7) and (8) results in fluid-solid interaction forces that are added to both fluid and solid markers.

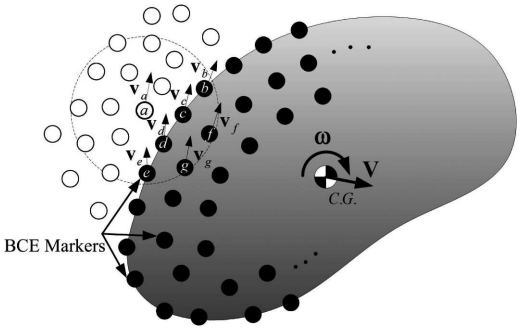

Fig. 2. Fluid-solid interaction using BCE markers attached to a rigid body. BCE and fluid markers are represented by black and white circles, respectively. The BCE markers positioned in the interior of the body (markers g and f in the figure) should be placed to a depth less than or equal to the size of the compact support associated with the kernel function W. A similar concept yet with different kinematics is used for flexible bodies

2.5. Solid-solid short range interaction

Dry friction models typically used to characterize the dynamics of granular materials [1, 13 14] do not capture accurately the impact of solid surfaces in hydrodynamics media. In practice, it is infeasible to resolve the

short-range, high-intensity impact forces arising in wet media due to computational limits on space and time resolution. Ladd [15] proposed a normal lubrication force between two spheres that increases rapidly as the distance between particles approaches zero and therefore prevents the actual touching of the spheres:

$$\mathbf{F}_{ij}^{lub} = \min\left\{-6\pi\mu\left(\frac{a_i a_j}{a_i + a_j}\right)^2\left(\frac{1}{s} - \frac{1}{\Delta_c}\right), 0\right\} \cdot \mathbf{v}_{n_{ij}}, \tag{34}$$

where, a_i and a_j are the sphere radii, $\mathbf{v}_{n_{ij}}$ is the normal component of the relative velocity, and s is the distance between surfaces. For $s > \Delta_c$, $\mathbf{F}_{ij}^{lub} = 0$, and the spheres are subject only to hydrodynamic forces.

Equation (34) provides a basic model for the estimation of the lubrication force in normal direction. The generalization of this model to non-spherical objects requires the calculation of the minimum distance and the curvature of the two contact surfaces. The calculation of the partial lubrication force between non-spherical surfaces follows the approach proposed in [8] for a lattice Boltzmann method but is amended to fit the Lagrangian formulation adopted herein. Accordingly, the force model provided in Eq. (34) is modified as

$$\begin{aligned} \mathbf{F}_{ij}^{lub} &= \sum_k \mathbf{f}_{ij}^k, \\ \text{with } \mathbf{f}_{ij}^k &= \min\left\{-\frac{3}{2}\pi\mu h^2\left(\frac{1}{s^*} - \frac{1}{\Delta_c}\right), 0\right\} \cdot \mathbf{v}_{n_{ij}}^*, \end{aligned} \tag{35}$$

where s^* and $\mathbf{v}_{n_{ij}}^*$ denote the markers' relative distance and velocity, respectively, and the summation is over all interacting markers of the two solid objects.

3. HPC implementation

Chrono::Fluid [4], an open-source simulation framework for fluid-solid interaction, provides an implementation that executes all phases of the solution method on the GPU. A brief overview of the GPU hardware and programming model adopted is followed below by a detailed discussion of the critical kernels that implement the proposed modeling and solution approach.

3.1. GPU hardware and programming model

To a very large extent, the performance of today's simulation engines is dictated by the memory bandwidth of the hardware solution adopted. Recent

numerical experiments conducted by the authors revealed that only about 5 to 10% of the peak flop rate is reached by the large number of cores present on today's architectures since these cores most of the time idle waiting for data from global memory or RAM. It is this observation that motivated the selection of the GPU as the target hardware for implementing Chrono::Fluid. At roughly 300 GB/s, the GPU memory bandwidth stands four to five times higher than what one could expect on a fast CPU.

To describe the hardware organization of the GPU, we consider here the NVIDIA GeForce GTX 680 [18, 27]. This GPU is based on the first generation of Kepler architecture, code name GK104, which is also implemented in Tesla K10. The Kepler architecture relies on a Graphics Processing Cluster (GPC) as the defining high-level hardware block. There are a total of 4 GPCs on the GK104. Each GPC includes two Stream Multiprocessors (SM), each of which has 192 Scalar Processors (SP), for a total of 1536 SPs, and 3.1 TFlops processing rate.

In addition to the processing cores, the second important aspect of GPU hardware is that of the memory hierarchy. The memory on the GPU is divided into several types, each with different access patterns, latencies, and bandwidths:

Registers (read/write per thread): 65536, 32-bit memory units. Very low latency, high bandwidth ($\simeq$ 10 TB/s cumulative) memory used to hold thread-local data.

Shared memory/L1 cache (read/write per block): 64 KB. Low latency, high bandwidth ($\simeq$ 1.5-2 TB/s cumulative) memory divided between shared memory and L1 cache.

Global memory (read/write per grid): 2 GB. Used to hold input and output data. Accessible by all threads, with a bandwidth of 192 GB/s and a higher latency ($\simeq$ 400-800 cycles) than shared memory and registers. All accesses to global memory pass through the L2 cache. The latter is 512 KB large and has a bandwidth of 512 B/clock cycle.

Constant memory (read only per grid): 48 KB per Kepler SM. Used to hold constants, serviced at the latency and bandwidth of the L1 cache upon a cache hit, or those of the global memory upon a cache miss.

The parallel execution paradigm best supported on GPUs is "single instruction multiple data" (SIMD). In this model, if for instance two arrays of length one million are to be added, one million threads are launched with each executing the same instruction; i.e., adding two numbers, but on different data – each thread adding two different numbers. Fortunately, SIMD computing is very prevalent in the solution methodology proposed, where each SPH marker is handled by a thread in the same way in which hundreds

of thousands of other threads handle their markers using different data. While strongly leveraging the SIMD model owing to the fine grain parallelism that it exposes, the methodology adopted is prone to lead to memory access patterns that do not display high spatial and/or temporal locality. This is because the SPH markers move in time leading to less structured memory accesses that adversely impact the effective bandwidth reached by the code.

3.2. Time integration

Chrono::Fluid uses a second order explicit mid-point Runge-Kutta (RK2) scheme [2] for the time integration of the fluid and solid phases, the latter in its rigid or flexible representation.

Algorithm 1 summarizes the steps required for the calculation of the force on the SPH markers, rigid bodies, and deformable beams at time step k.

Algorithm 1 Force Calculation

▷ *Calculate modified XSPH velocities*
for $a := 0$ to $(N_m - 1)$ **do**
  $\hat{\mathbf{v}}_a^k \equiv \hat{\mathbf{v}}_a\left(\boldsymbol{\rho}^k, \mathbf{x}^k, \mathbf{v}^k\right)$
end for
▷ *SPH forces*
for $a := 0$ to $(N_m - 1)$ **do**
  $\dot{\rho}_a^k \equiv \dot{\rho}_a\left(\boldsymbol{\rho}^k, \mathbf{x}^k, \hat{\mathbf{v}}^k\right)$
  $\dot{\mathbf{x}}_a^k = \hat{\mathbf{v}}_a^k$
  $\dot{\mathbf{v}}_a^k \equiv \dot{\mathbf{v}}_a\left(\boldsymbol{\rho}^k, \mathbf{x}^k, \mathbf{v}^k\right)$
end for
▷ *Rigid body forces*
for $i := 0$ to $(N_r - 1)$ **do**
  $\dot{\mathbf{V}}_i^k \equiv \dot{\mathbf{V}}_i\left(\dot{\mathbf{v}}^k\right)$
  $\dot{\mathbf{X}}_i^k = \dot{\mathbf{V}}_i^k$
  $\dot{\boldsymbol{\omega}}_i^k \equiv \dot{\boldsymbol{\omega}}_i\left(\dot{\mathbf{v}}^k, \boldsymbol{\omega}_i^k\right)$
  $\dot{\mathbf{q}}_i^k \equiv \dot{\mathbf{q}}_i\left(\boldsymbol{\omega}_i^k, \mathbf{q}_i^k\right)$
end for
▷ *ANCF forces*
for $j := 0$ to $\left(N_f - 1\right)$ **do**
  $\hat{\mathbf{Q}}_j^k = \hat{\mathbf{Q}}_j^a\left(\dot{\mathbf{v}}^k\right) - \hat{\mathbf{Q}}_j^e\left(\hat{\mathbf{e}}^k\right) + \hat{\mathbf{Q}}_j^g\left(\hat{\mathbf{e}}^k\right)$
  $\dot{\hat{\mathbf{e}}}_1^k = \hat{\mathbf{e}}_2^k$
  $\dot{\hat{\mathbf{e}}}_2^k = \hat{\mathbf{M}}^{-1}\hat{\mathbf{Q}}_j^k$
end for

The variables N_m, N_r, and N_f denote the number of markers, rigid bodies, and flexible beams, respectively. The arrays $\boldsymbol{\rho}$, $\mathbf{x}$, $\mathbf{v}$, and $\hat{\mathbf{v}}$ store the density, position, velocity, and modified velocity for all markers, respectively; for example, $\boldsymbol{\rho} = \{\rho_a | a = 0, 1, 2, ..., N_m - 1\}$.

The external forces on the rigid and flexible bodies include the FSI forces captured via BCE markers at distributed locations on the solid surfaces. The distributed forces need to be accumulated into a single force and torque at the center of each rigid body, or point forces at node locations of each flexible body. The reduction (summation of the forces and torques) is handled by parallel scan operations available through the Thrust library [11], which exposes a scan algorithm that scales linearly.

The RK2 integration scheme requires the calculation of the force at the beginning as well as middle of the time step. Algorithm 2 lists the steps required for the time integration of a typical FSI problem.

To improve the code vectorization and use of fast memory; i.e., L1/L2 cache, shared memory, and registers, each computation task was implemented as a sequence of light GPU kernels. For instance, different computation kernels are implemented to update the attributes of the solid bodies, including force, moment, rotation, translation, linear and angular velocity, and locations of the associated BCE markers. A similar coding philosophy was maintained for the density re-initialization, boundary condition implementation, and mapping of the marker data on an Eulerian grid for post processing.

3.2.1. Multi-rate integration

Stable integration of the SPH fluid equations requires step-sizes which are also appropriate for propagating the dynamics of any rigid solids in the FSI system. However, integration of the dynamics of deformable bodies, especially as their stiffness increases, calls for very small time steps. To alleviate the associated computational cost, we use a dual-rate integration scheme using intermediate steps for the integration of the flexible dynamics EOMs, typically with $\Delta t_{SPH}/\Delta t_{ANCF} = 10$, although stiffer problems may require ratios of up to 50.

This aspect is noteworthy given that typical FSI models involve a number of SPH markers orders of magnitude larger than the number of ANCF nodal coordinates required for the flexible bodies. Without a multi-rate implementation, the numerical solution would visit the fluid phase at each integration step, an approach that led to very large solution times. This aspect is further discussed in Sect. 4.

Algorithm 2 RK2 Time Integration

▷ *Force calculation at beginning of step (Algorithm 1)*
Calculate $\{\hat{\mathbf{v}}^k, \dot{\rho}^k, \dot{\mathbf{x}}^k, \dot{\mathbf{v}}^k, \dot{\mathbf{V}}^k, \dot{\mathbf{X}}^k, \dot{\omega}^k, \dot{\mathbf{q}}^k, \dot{\mathbf{Q}}^k\}$
▷ *Half-step updates for fluid, rigid body, and flexible body states*
for $a \in \{a | a$ is a fluid marker$\}$ **do**
  $\psi_a^{k+1/2} = \psi_a^k + \dot{\psi}_a^k \times \Delta t/2$, where $\psi_a \in \{\rho_a, \mathbf{x}_a, \mathbf{v}_a\}$
end for
for $i := 0$ to $(N_r - 1)$ **do**
  $\Psi_i^{k+1/2} = \Psi_i^k + \dot{\Psi}_i^k \times \Delta t/2$, where $\Psi_i \in \{\mathbf{V}_i, \mathbf{X}_i, \boldsymbol{\omega}_i, \mathbf{q}_i\}$
end for
for $j := 0$ to $\left(N_f - 1\right)$ **do**
  $\hat{\mathbf{e}}_1^{k+1/2} = \hat{\mathbf{e}}_1^k + \dot{\hat{\mathbf{e}}}_1^k \times \Delta t/2$
  $\hat{\mathbf{e}}_2^{k+1/2} = \hat{\mathbf{e}}_2^k + \dot{\hat{\mathbf{e}}}_2^k \times \Delta t/2$
end for
▷ *Half-step update for BCE marker positions and velocities*
for $a \in \{a | a$ is a BCE marker$\}$ **do**
  Obtain $\mathbf{x}_a^{k+1/2}$ and $\mathbf{v}_a^{k+1/2}$ according to associated rigid, flexible, or immersed boundary motion.
end for
▷ *Force calculation at half-step (Algorithm 1)*
Calculate $\{\hat{\mathbf{v}}^{k+1/2}, \dot{\rho}^{k+1/2}, \dot{\mathbf{x}}^{k+1/2}, \dot{\mathbf{v}}^{k+1/2}, \dot{\mathbf{V}}^{k+1/2}, \dot{\mathbf{X}}^{k+1/2}, \dot{\omega}^{k+1/2}, \dot{\mathbf{q}}^{k+1/2}, \dot{\mathbf{Q}}^{k+1/2}\}$
▷ *Full-step updates for fluid, rigid body, and flexible body states*
for $a \in \{a | a$ is a fluid marker$\}$ **do**
  $\psi_a^{k+1} = \psi_a^k + \dot{\psi}_a^{k+1/2} \times \Delta t$
end for
for $i := 0$ to $(N_r - 1)$ **do**
  $\Psi_i^{k+1} = \Psi_i^k + \dot{\Psi}_i^{k+1/2} \times \Delta t$
end for
for $j := 0$ to $\left(N_f - 1\right)$ **do**
  $\hat{\mathbf{e}}_1^{k+1} = \hat{\mathbf{e}}_1^k + \dot{\hat{\mathbf{e}}}_1^{k+1/2} \times \Delta t$
  $\hat{\mathbf{e}}_2^{k+1} = \hat{\mathbf{e}}_2^k + \dot{\hat{\mathbf{e}}}_2^{k+1/2} \times \Delta t$
end for
▷ *Full-step update for BCE marker positions and velocities*
for $a \in \{a | a$ is a BCE marker$\}$ **do**
  Obtain $\mathbf{x}_a^{k+1}$ and $\mathbf{v}_a^{k+1}$.
end for

3.3. Proximity calculation and neighbor search

In the current implementation, each marker has a list of neighbor markers; i.e., markers that fall within its support domain. Given this set of lists, the calculation of $\hat{\mathbf{v}}_a^k \equiv \hat{\mathbf{v}}_a\left(\boldsymbol{\rho}^k, \mathbf{x}^k, \mathbf{v}^k\right)$, $\dot{\rho}_a^k \equiv \dot{\rho}_a\left(\boldsymbol{\rho}^k, \mathbf{r}^k, \hat{\mathbf{v}}^k\right)$, and $\dot{\mathbf{v}}_a^k \equiv \dot{\mathbf{v}}_a\left(\boldsymbol{\rho}^k, \mathbf{r}^k, \mathbf{v}^k\right)$ is straightforward. The loop iterations in Algorithms 1 and 2 have no overlap and can be executed in parallel. The computational bottleneck thus becomes the determination of the neighbor lists through proximity calculation, a step that requires about 70% of the entire computational budget and thus critically impacts the overall performance of the simulation. Herein, we adopt a spatial binning algorithm which is sub-optimal in terms of amount of net work but superior in terms of memory usage. In this approach, the computation domain is divided into a collection of bins. The bins have side lengths equal to the maximum influence distance of a marker, i.e. κh. This localizes the search for the possible interacting markers to the bin and all of its 26 immediate 3D neighbors. Figure 3 shows the binning approach in 2D.

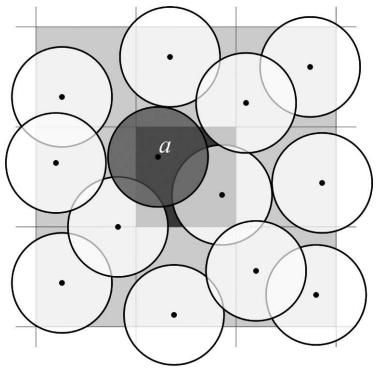

Fig. 3. Illustration of the spatial subdivision method used for proximity computation in 2D. The circles represent the domain of influence of each marker; i.e., the support domain. For clarity, a coarse distribution of markers is shown. In reality, the concentration of markers per bin is much larger

In the neighbor search approach adopted herein, neighbor lists are not saved in memory; instead, neighbors are evaluated whenever required. Alternatively [28], the principle of action and reaction could be leveraged to calculate and store the acceleration terms for each inter-penetration. Although the second algorithm reduces the amount of work by re-using the acceleration terms calculated for each inter-penetration, it can not be applied to the SPH method due to the massive amount of memory required to store the data associated with all inter-penetration events. Another advantage of the adopted neighbor search approach is the coalesced memory access achieved

by sorting and accessing the data based on the markers location. The steps required for accessing the neighbors are summarized in Algorithm 3.

Algorithm 3 Inner loop: accessing neighbor markers

1: Divide the solution space into n_b bins of $\left(\Delta_x, \Delta_y, \Delta_z\right)$ dimensions, where $n_b = n_x \times n_y \times n_z$, and (n_x, n_y, n_z) is the number of grid cells along (x, y, z) axis.
2: Construct the hash array: $= \{s_a | a = 0, 1, 2, ..., N_m - 1\}$ according to $s_a = = i \times n_y \times n_z + j \times n_z + k$, where (i, j, k) is the location of the bin containing the marker a.
3: Sort into $_{sorted}$ and obtain corresponding $\boldsymbol{\rho}_{sorted}$, $\mathbf{x}_{sorted}$, $\mathbf{v}_{sorted}$, and $\hat{\mathbf{v}}_{sorted}$.
4: Construct $\mathbf{c}^1 = \{c_p^1 | p = 0, 1, 2, ..., n_b - 1\}$ and $\mathbf{c}^2 = \{c_p^2 | p = 0, 1, 2, ..., n_b - 1\}$, where c_p^1 and c_p^2 denote the two indices in $_{sorted}$ that bound the sequence of hash values $s_a = p$.
5: Access markers data in bin (i, j, k) by loading $\left[c_p^1, c_p^2\right]$ portions of the sorted arrays $\boldsymbol{\rho}_{sorted}$, $\mathbf{x}_{sorted}$, $\mathbf{v}_{sorted}$, and $\hat{\mathbf{v}}_{sorted}$, as needed.

The data sorting in step 3 of Algorithm 3 is performed using the linear-complexity radix sort available in the Thrust library. As a result, this solution component does not affect the overall linear scaling of Algorithms 1 and 2. Working with the sorted arrays for the bulk of the computation has the additional advantage of increasing the memory spatial locality: SPH markers that share a neighborhood in the physical space, do so in their memory location as well.

4. Results

The simulation approach described in Sect. 3 was implemented to execute in parallel on GPU cards using the CUDA programming environment [26]. All simulations reported in this section were run on an NVIDIA GeForce GTX 680 GPU card described in sub-section 3.1. In all simulations, if present, the flexible beams were modeled using $n_e = 4$ ANCF beam elements, while the integrals appearing in the elastic forces $\mathbf{Q}^e$ in Eq. (25) were evaluated using 5 and 3 Gauss quadrature points for the axial and bending elastic forces, respectively.

A snapshot from a typical FSI problem involving flexible bodies is shown in Figure 4. Additional Chrono::Fluid simulation examples can be found at [32]. Comprehensive validation studies of the Chrono::Fluid simulation package are provided in [31]. The focus here is on numerical experiments

that illustrate the performance and scalability of the proposed algorithm and its parallel implementation.

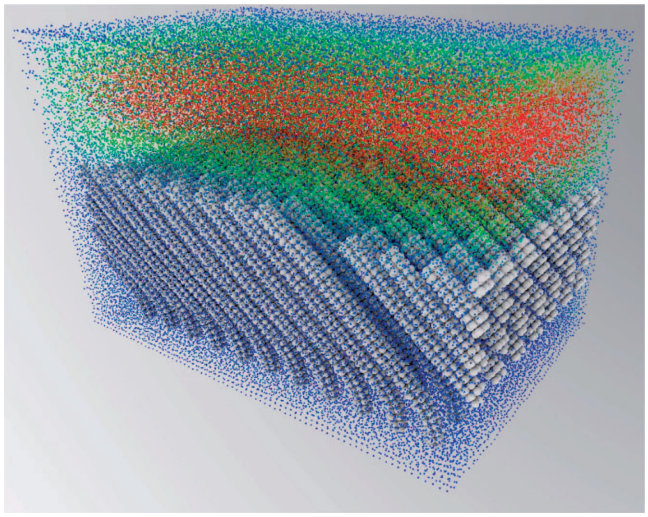

Fig. 4. Example of Chrono::Fluid simulation: channel flow over an array of flexible cantilevers. (See [32] for further details)

We begin by analyzing the efficiency and scaling attributes of the solution for systems composed exclusively of rigid bodies, flexible bodies, or a fluid phase. In each of these three scenarios, we provide the simulation times per time step for problems of increasing size. Data provided in Tables 1, 2, and 3 and illustrated in Figure 5 indicate that updates of the dynamics of each phase scale linearly with problem size; i.e., with the number of rigid bodies, flexible bodies, and SPH markers, respectively.

Table 1.

Time required for advancing the rigid body dynamics simulation by one time step as function of problem size (number of rigid bodies, N_r)

$N_m = 0$, $N_f = 0$						
N_r	($\times 10^3$)	0.49	2.87	16.59	56.77	118.23
t	(ms)	5	8	16	44	78

While the previous results show linear scaling for rigid and flexible body dynamics, in actual FSI problems in which rigid and/or flexible bodies interact with the fluid, the simulation time is virtually independent of the number of solid objects. The results presented in Tables 4 and 5 were obtained on a system consisting of approximately 3 million SPH markers by varying the number of rigid bodies and flexible bodies. The small sensitivity of the

Table 2.

Time required for advancing the flexible body dynamics simulation by one time step as function of problem size (number of flexible bodies, N_f)

$N_m = 0$, $N_r = 0$						
N_f	($\times 10^3$)	0.78	3.51	17.55	56.94	115.05
t	(ms)	8	14	48	122	238

simulation time with respect to the number of solid objects is due to the fact that the number of BCE markers associated with solid bodies represents only a very small fraction of the number of SPH discretization markers, the latter overwhelmingly dictating the required computation time. Nevertheless, as the concentration of solid objects increases, smaller time steps are required since the probability of short-range, high-frequency interactions increases.

Table 3.

Time required for advancing a fluid dynamics simulation by one time step as function of problem size (number of SPH markers, N_m)

$N_r = 0$, $N_f = 0$						
N_m	($\times 10^6$)	0.06	0.32	0.93	1.79	4.13
t	(ms)	27	121	331	538	1150

Table 4.

Simulation time per time step for an FSI problem with fixed number of SPH markers and increasing number of rigid bodies

$N_m \simeq 3.0 \times 10^6$, $N_f = 0$							
N_r	0	36	120	480	1800	8400	33600
t (ms)	906	919	923	925	926	926	921

The two sets of results provided in Table 5 for different values of $\tau = = \Delta t_{SPH}/\Delta t_{ANCF}$ show a small increase in the simulation time when choosing a very small integration step size for flexible body dynamics. This illustrates the efficiency of the multi-rate integration scheme in improving the time integration stability at a small increase in computational cost. The small changes in the simulation times provided in Tables 4 and 5 are mainly due to the deviations in the magnitude of N_m as the number of solid objects changes. Linear scaling in Chrono::Fluid is also demonstrated in an experiment where a combined FSI problem, i.e. involving rigid and flexible bodies and including a lubrication force model and two-way coupling with fluid, is solved on domains of increasing size (see Figure 6). As the simulation domain volume

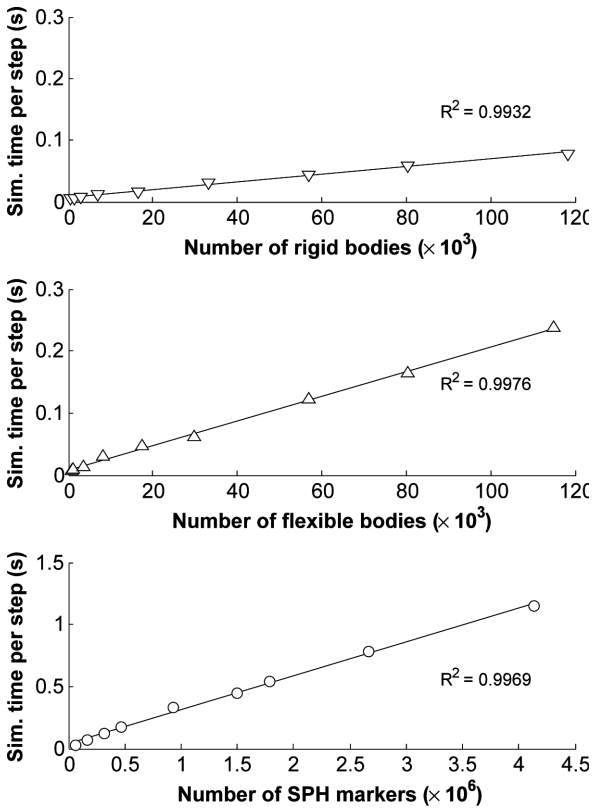

Fig. 5. Scaling analysis of Chrono::Fluid for rigid body dynamics, flexible body dynamics, and fluid dynamics as function of problem size (N_r, N_f, and N_m, respectively). The coefficients R^2 are specified to each linear regression

is increased by factors from 2 up to 32, the number of SPH markers varies from about 76,000 to more than 2.5 million. Simultaneously, the number of rigid and flexible bodies grow from 168 to more than 24,000 and from 160 to almost 10,000, respectively (see Table 6). As shown in Figure 6, Chrono::Fluid achieves linear scaling over the entire range of problem sizes.

Table 5.

Simulation time per time step for an FSI problem with fixed number of SPH markers and increasing number of flexible bodies, for two different values of the multi-rate integration factor $\tau = \Delta t_{SPH}/\Delta t_{ANCF}$

$N_m \simeq 3.0 \times 10^6$, $N_r = 0$								
	N_f	0	45	140	440	1152	2100	4704
$\tau = 10$	t (ms)	906	923	928	916	960	950	921
$\tau = 50$	t (ms)	906	973	978	965	1066	1060	1060

Table 6.

Simulation time per time-step for combined FSI problems of increasing size

N_m	($\times 10^6$)	0.08	0.16	0.29	0.63	0.95	1.54	2.50
N_r	($\times 10^3$)	0.17	0.52	1.12	4.48	7.84	14.56	24.64
N_f	($\times 10^3$)	0.16	0.42	0.84	2.10	3.36	5.88	9.66
t	(ms)	45	74	120	230	343	522	820

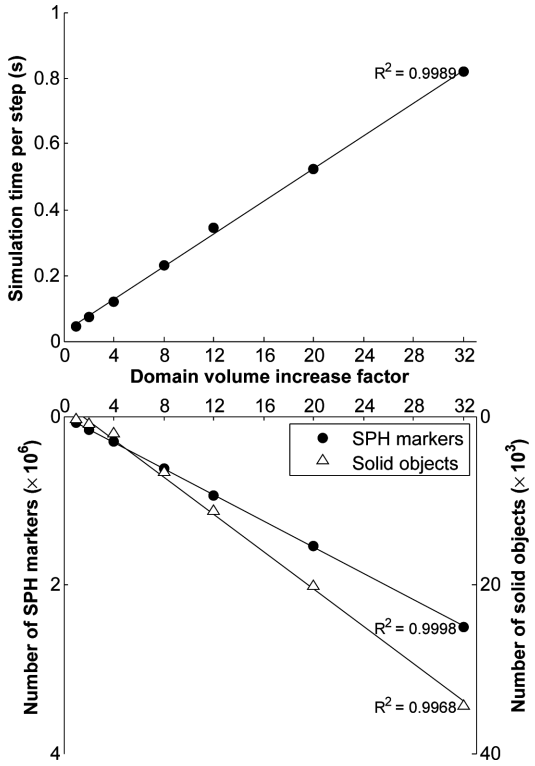

Fig. 6. Scaling analysis of Chrono::Fluid for FSI problems: simulation time as function of combined problem size. In this experiment, the volume of the simulation domain is increased, up to 32 times the volume of the initial domain, leading to proportional increases in the number of SPH markers and of solid objects (both rigid and flexible bodies) as shown in the bottom plot (see also Table 6). As illustrated by the top graph, the simulation time per time step varies linearly with problem size. The coefficients R^2 are specified for each linear regression

5. Conclusions

This contribution discusses details of an HPC approach to the simulation of FSI problems. Relying on: (i) a Lagrangian/Lagrangian formalism for the simulation of the fluid and solid phases, and (ii) GPU parallelism, Chrono::Fluid, the software implementation of the proposed solution method-

ology, is suitable for the study of multi-phase/multi-component problems such as particle suspensions, polymer flow, and biomedical applications.

Chrono::Fluid has been shown to scale linearly in the dynamics of each phase independently as well as in the overall FSI solution implementation. The dynamics of the fluid phase; i.e. the time propagation of the SPH markers, controls the overall simulation time, and neither cross-phase communications nor solid phase simulation have a notable effect on the simulation time. This represents one advantage of the adopted unified FSI methodology over co-simulation and asynchronous approaches. Chrono::Fluid, whose preliminary validation is discussed in [30], is open source and freely available under a BSD3 license.

Several directions of future work are as identified as follows: (*a*) revisit the SPH marker numbering scheme in order to improve the spatial and temporal memory access patterns, thus increasing the effective bandwidth of Chrono::Fluid; (*b*) augment the current implementation for use on cluster supercomputers that adopt a non-uniform memory access model and rely on the Message Passing Interface standard; (*c*) investigate problems at macroscale such as vehicle fording operations, which call for large scale simulations at low Reynolds numbers and possibly in the presence of very large viscosity; and (*d*) gauge the potential of Chrono::Fluid in biomechanics applications such as blood flow in deformable arteries or heart, channel occlusion in stroke, etc.

Acknowledgments

Financial support for this work was provided by National Science Foundation award CMMI-0840442. Support for the second and third authors was provided in part by Army Research Office awards W911NF-11-1-0327 and W911NF-12-1-0395.

Manuscript received by Editorial Board, March 11, 2014;
final version, April 14, 2014.

REFERENCES

[1] Anitescu M., Hart G.D.: A constraint-stabilized timestepping approach for rigid multibody dynamics with joints, contact and friction. International Journal for Numerical Methods in Engineering 60(14), 2335–2371 (2004).

[2] Atkinson K.: An introduction to numerical analysis. John Wiley and Sons, USA (1989).

[3] Benz W.: Smoothed particle hydrodynamics: A review. In: Proceedings of the NATO Advanced Research Workshop on The Numerical Modelling of Nonlinear Stellar Pulsations Problems and Prospects, Les Arcs, France, March 20-24. Kluwer Academic Publishers (1986).

[4] Chrono::Fluid: An Open Source Engine for Fluid-Solid Interaction (2014). http://armanpazouki.github.io/chrono-fluid/

[5] Cleary P.W.: Modelling confined multi-material heat and mass flows using SPH. Applied Mathematical Modelling 22(12), 981–993 (1998).
[6] Colagrossi A., Landrini M.: Numerical simulation of interfacial flows by smoothed particle hydrodynamics. Journal of Computational Physics 191(2), 448–475 (2003).
[7] Dilts G.: Moving-least-squares-particle hydrodynamics–I. consistency and stability. International Journal for Numerical Methods in Engineering 44(8), 1115–1155 (1999).
[8] Ding E.J., Aidun C.K.: Extension of the lattice-Boltzmann method for direct simulation of suspended particles near contact. Journal of statistical physics 112(3-4), 685–708 (2003).
[9] Gerstmayr J., Shabana A.: Analysis of thin beams and cables using the absolute nodal coordinate formulation. Nonlinear Dynamics 45(1), 109–130 (2006).
[10] Haug E.: Computer aided kinematics and dynamics of mechanical systems. Allyn and Bacon Boston (1989).
[11] Hoberock J., Bell N.: Thrust: C++ template library for CUDA. http://thrust.github.com/
[12] Hu W., Tian Q., Hu H.: Dynamic simulation of liquid-filled flexible multibody systems via absolute nodal coordinate formulation and SPH method. Nonlinear Dynamics pp. 1–19 (2013).
[13] Kruggel-Emden H., Simsek E., Rickelt S., Wirtz S., Scherer V.: Review and extension of normal force models for the discrete element method. Powder Technology 171(3), 157–173 (2007).
[14] Kruggel-Emden H., Wirtz S., Scherer V.: A study on tangential force laws applicable to the discrete element method (DEM) for materials with viscoelastic or plastic behavior. Chemical Engineering Science 63(6), 1523–1541 (2008).
[15] Ladd A.J.: Sedimentation of homogeneous suspensions of non-Brownian spheres. Physics of Fluids 9, 491 (1997).
[16] Liu M.B., Liu G.R.: Smoothed particle hydrodynamics (SPH): An overview and recent developments. Archives of Computational Methods in Engineering 17(1), 25–76 (2010).
[17] Lucy L.B.: A numerical approach to the testing of the fission hypothesis. The Astronomical Journal 82, 1013–1024 (1977).
[18] Micikevicius P.: GPU performance analysis and optimization (2014). http://on-demand.gputechconf.com/gtc/2012/presentations/S0514-GTC2012-GPU-Performance-Analysis. pdf
[19] Monaghan J.J.: An introduction to SPH. Computer Physics Communications 48(1), 89–96 (1988).
[20] Monaghan J.J.: On the problem of penetration in particle methods. Journal of Computational Physics 82(1), 1–15 (1989).
[21] Monaghan J.J.: Smoothed particle hydrodynamics. Annual Review of Astronomy and Astrophysics. 30, 543–574 (1992).
[22] Monaghan J.J.: Smoothed particle hydrodynamics. Reports on Progress in Physics 68(1), 1703–1759 (2005).
[23] Monaghan J.J., Gingold R.: Shock simulation by the particle method SPH. Journal of Computational Physics 52(2), 374–389 (1983).
[24] Morris J., Fox P., Zhu Y.: Modeling low Reynolds number incompressible flows using SPH. Journal of Computational Physics 136(1), 214–226 (1997).
[25] Negrut D., Tasora A., Mazhar H., Heyn T., Hahn P.: Leveraging parallel computing in multibody dynamics. Multibody System Dynamics 27(1), 95–117 (2012).
[26] NVIDIA: CUDA developer zone (2014). https://developer. nvidia.com/cuda-downloads.
[27] NVIDIA: NVIDIA GeForce GTX 680 (2014). http://www.geforce.com/Active/en˙US/en˙US/pdf/GeForce-GTX-680-Whitepaper-FINAL.pdf.
[28] Pazouki A., Mazhar H., Negrut D.: Parallel collision detection of ellipsoids with applications in large scale multibody dynamics. Mathematics and Computers in Simulation 82(5), 879–894 (2012). URL http://www.sciencedirect. com/science/article/pii/S0378475412000080

[29] Pazouki A., Negrut D.: Direct simulation of lateral migration of buoyant particles in channel flow using GPU computing. In: Proceedings of the 32nd Computers and Information in Engineering Conference, CIE32, August 12-15, Chicago, IL, USA. American Society of Mechanical Engineers (2012).
[30] Pazouki A., Negrut D.: A numerical study of the effect of particle properties on the radial distribution of suspensions in pipe flow. Submitted to International Journal of Multiphase Flow (2014).
[31] Pazouki A., Serban R., Negrut D.: A Lagrangian-Lagrangian framework for the simulation of rigid and deformable bodies in fluid. Multibody Dynamics: Computational Methods and Applications, ISBN: 9783319072593, Springer, 2014, pp. 33-52.
[32] SBEL: Vimeo page (2014). https://vimeo.com/uwsbel.
[33] Schörgenhumer M., Gruber P.G., Gerstmayr J.: Interaction of flexible multibody systems with fluids analyzed by means of smoothed particle hydrodynamics. Multibody System Dynamics pp. 1–24 (2013).
[34] Schwab A., Meijaard J.: Comparison of three-dimensional flexible beam elements for dynamic analysis: finite element method and absolute nodal coordinate formulation. In: Proceedings of the ASME 2005 IDETC/CIE. Orlando, Florida (2005).
[35] Shabana A.: Flexible multibody dynamics: Review of past and recent developments. Multibody System Dynamics 1, 339–348 (1997).
[36] Shabana A.: Dynamics of Multibody Systems. Cambridge University Press, New York (2005).

Zastosowanie wysokowydajnej techniki obliczeniowej (HPC) do symulacji problemów interakcji między płynem i ciałem stałym z elementami sztywnymi i elastycznymi

S t r e s z c z e n i e

W pracy przedstawiono zarys jednolitego podejścia do bezpośredniej numerycznej symulacji problemów interakcji płyn – ciało stałe (FSI) z wykorzystaniem wielowątkowej wysokowydajnej techniki obliczeniowej (HPC) o wielkiej skali. Algorytm symulacji opiera się na rozszerzonej metodzie hydrodynamiki cząstek gładkich (XSPH), która opisuje przepływ płynu w formalizmie Lagrange'a zgodnym z metodą Lagrange'a śledzenia fazy stałej. W celu modelowania sztywnego i elastycznego układu wielu ciał implementowano ogólną, trójwymiarową dynamikę ciała sztywnego i zastosowano sformułowanie bezwzględnych współrzędnych węzłowych (ANCF). Dwukierunkowe sprzężenie między płynem i fazą stałą jest zamodelowane przez użycie znaczników wymuszenia warunków brzegowych (BCE) które oddają działanie sił sprzężenia między płynem a ciałem stałym wymuszając brak poślizgu w warunkach brzegowych. Problem interakcji bliskiego zakresu między płynem i ciałem stałym, która ma decydujący wpływ na zachowanie w małej skali mieszanin płynów i ciał stałych, rozwiązano przy pomocy modelu sił smarowania. Stany systemu zbiorczego są integrowane w czasie przy użyciu jawnego, wieloszybkościowego schematu. By zmniejszyć wielkie obciążenie obliczeniowe, w algorytmie ogólnym położono nacisk na obliczenia równoległe w kartach procesorów graficznych (GPU). W pracy przedstawiono analizę wydajności i skalowania dla scenariuszy symulacji obejmujących jedną lub wiele faz przy liczbie obiektów stałych sięgającej dziesiątek tysięcy. Implementacja oprogramowania przedstawionej metody, o nazwie Chrono::Fluid, jest częścią projektu Chrono i jest udostępniona do użytku nieodpłatnego.