Application of B-Splines and Curved Geometries to Boundaries in SPH

Daniel J. Barker[1], Stephen J. Neethling[1]

[a]*Dept. of Earth Science & Engineering, Royal School of Mines, Imperial College London, London, SW7 2AZ, U.K.*

Abstract

Smoothed Particle Hydrodynamics (SPH) has been increasing in popularity rapidly in recent years and is being used for an ever wider range of applications. Central to almost all of these application is the inclusion of accurate wall boundaries. We present here a discussion of boundaries in SPH, in particular focusing on reflected ghost-particle boundaries. We show how one can include curved shapes as geometrical objects and more generally as parametric NURBS curves. By properly considering the reflection operation we derive a correction factor which demonstrably improves the accuracy of the SPH solution and present examples to confirm this. NURBS are standard for representing both 2D curves and 3D surfaces. We detail how they can be practically included in an SPH implementation, including how to calculate various required quantities and reflect particles in the NURBS object.

Keywords: Smoothed Particle Hydrodynamics, B-Splines, Wall Boundaries, Fluid Dynamics, Particle Methods, Lagrangian

1. Introduction

Smoothed Particle Hydrodynamics (SPH) is becoming an increasingly important and robust numerical method for a wide range of technical and engineering applications [1–5]. It is a meshless and Lagrangian method for solving, amongst others, the Navier Stokes (NS) equation.

SPH was developed to study the collapse of proto-stellar gas clouds [6, 7] meaning originally wall boundary conditions were not considered. However, one of the major challenges presented by the increasingly demanding engineering applications of SPH is the ability to accurately include complex boundary geometries. In the finite element or lattice Boltzmann methods the edge of the mesh or grid respectively, naturally defines the boundary of the simulation domain and various quantities or their gradients can be prescribed if desired. By contrast boundaries are relatively complicated to include in SPH due to it being a particle based method. Though there has been some progress in this area recently [8, 9]

Boundaries are typically handled in 3 main ways:

*Corresponding author

Repulsive Walls A repulsive force is included in the equations of motion which acts to push all particles away from the walls when they approach.

Ghost Particles Extra particles are placed outside the simulation domain which act through pressure to repel fluid particles.

Kernel Correction The missing kernel support is accounted for explicitly in the SPH equations.

Here we use a reflected particle form of ghost particle boundaries, see [10]. We present a discussion of the issues involved in efficiently implementing such a boundary treatment and present its application to surfaces represented by mathematical objects, e.g. spheres, and a parametric representation; non-uniform rational B-splines (NURBS). NURBS's are the most common way of parametrically representing surfaces or curves in CAD packages so their inclusion in an SPH implementation is a natural bonus.

1.1. SPH Interpolation and Gradients

The integration nodes (particles) in SPH are a set of N unconnected points in $\mathbb{R}^d$, X. Because of this unconnectedness one cannot use shape functions or similar to interpolate a function's value at each $\mathbf{x} \in X$ to the whole simulation domain. Instead an approximate interpolation based solely on distance is used; the interpolated quantity $A(\mathbf{x})$ is given by

$$A(\mathbf{x}) = \sum_{j=1}^{N} A_j W_h(|\mathbf{x}_j - \mathbf{x}|), \tag{1}$$

where $W_h(r)$ is the smoothing kernel, i.e. how much weight we give to a point based on its distance away and A_j is the value of A at particle with position $\mathbf{x}_j$. One can show [11] that the gradient of a smoothed quantity transfers onto the smoothing kernel giving

$$\nabla A(\mathbf{x}) = \sum_{j=1}^{N} A_j \nabla W_h(|\mathbf{x}_j - \mathbf{x}|). \tag{2}$$

This is the most basic form of the SPH interpolation and gradient estimates. It is possible to apply corrections which ensure zero order completeness [10, 12] or higher [11]. It is also possible to derive symmetric forms of the gradients which yield conservative SPH formulations [13]. The gradient used in this study is

$$\nabla A_i = \sigma_i \sum_{j=1}^{N} \left(\frac{A_j}{\sigma_j^2} + \frac{A_i}{\sigma_i^2} \right) \nabla W_{ij}, \tag{3}$$

where

$$\sigma_i = \sum_{j=1}^{N} W_{ij} \tag{4}$$

is the weighted number density of particles around the point $\mathbf{x}_i$ (see [10]) and $W_h(|\mathbf{x}_i - \mathbf{x}_j|)$ has been written as W_{ij} for clarity.

In order to make the method computationally tractable the smoothing kernel is chosen to have finite support. Typically the support of the smoothing kernel is defined to be a ball of radius $2h$. In this paper the quintic Wendland kernel [14] defined by

$$W(r,h) = C \begin{cases} \left(1 - \frac{r}{2h}\right)^4 \left(1 + \frac{2r}{h}\right) & \text{if } 0 \leq \frac{r}{h} < 2 \\ 0 & \text{otherwise} \end{cases}, \tag{5}$$

is used.

1.2. Boundaries

A recent investigation by Cummins et al [15] found that of the boundary formulation, the kernel used, kernel correction, time-stepping scheme and the compressibility of the SPH fluid, the boundary formulation had the largest effect on the simulation results. The earliest (and still widely used boundary conditions) are repulsive boundary conditions [2, 3, 16, 17], in which a repulsive force is applied to any particles which approach the wall boundary. Typically the repulsive force takes the form of a truncated Lennard-Jones type force. The advantages of this type of boundaries are that it is simple to implement and is good at ensuring particle containment within the domain.

One disadvantage, however, is that due to the stiff nature of the Lennard-Jones potential it can need a very small time-step to ensure stability. Further, because it acts only normal to the wall it cannot correctly capture shear stresses due to velocity gradients parallel to the wall. This becomes an issue at smaller scales or higher viscosities. Worse still because the force acts only outwards from the boundaries they cannot reproduce any system under tension, for instance a hanging droplet.

The support of smoothing kernels for particles near the boundaries extends out of the domain leading to so-called 'kernel boundary deficiency'. The deficiency leads to mis-estimation of quantities near the boundaries. While repulsive boundaries contain the particles in the domain they do not attempt to remedy this problem; unlike so-called 'ghost particle' boundaries.

Ghost particles are SPH particles which sit outside the domain, up to a distance of $2h$, thereby eliminating the kernel support deficiency. Their positions are fixed but they interact with the fluid through the usual pressure and viscosity forces. This method is usually preferable to the repulsive boundaries as it is possible to capture shear-stresses and since the interaction with the fluid particles is through the usual equations of motion the time-step is not unduly affected. There are two different methods which are classed as using ghost particles; *static* and *reflected*.

For the static type, the ghost particles are placed on a regular grid outside the fluid domain and their positions remain fixed for the entirety of the simulation (barring the movement of any boundaries with time). Examples of their application can be found in [4, 5]. This method is useful as it is also computationally efficient and is a simple extension to any SPH code. However, while it enforces some shear at the boundaries it is not capable of properly enforcing no-slip conditions. To see why this is the case consider the smoothed velocity field $\langle \mathbf{u} \rangle(\mathbf{x})$, as we sit on a boundary the smoothing operation sees the fluid with a linear velocity profile, but on the other side the ghost particles by definition have velocity zero. This means the averaged velocity at the boundary is non-zero.

To counter this non-zero smoothed velocity at the boundaries, one can use reflected ghost-particles [10, 18]. The method works much like the static ghost-particle method, i.e. by extending the computational domain by $2h$ outwards to overcome the lack of kernel support, but instead of static ghost particles the fluid particles are reflected across the wall-boundary at each time-step. The velocity of each particle is also reflected to become $-\mathbf{u}$. This requires slightly more computational effort but it ensures the velocity profile is correctly reproduced at the wall. In this paper we use the reflected ghost particle boundary condition and consider its applications.

Recent work has taken another approach which involves correcting the kernel boundary deficiency without extending the domain [9, 19, 20]. These look a promising way to capture boundary behaviour and are an active area of development.

1.3. Equations of Motion

SPH is used here to solve the Navier-Stokes equation, which in Lagrangian form and for incompressible Newtonian fluids, reads

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

with pressure P, fluid velocity $\mathbf{u}$, density ρ, dynamic viscosity μ and any addition body forces $\mathbf{f}$ for example gravity. Conservation of mass is captured by the incompressible continuity equation

$$\nabla \cdot \mathbf{u} = 0. \tag{7}$$

The density of a particle is calculated by multiplying the weighted number density by the mass associated with each particle;

$$\rho_i = m_i \sum_j W_{ij} = m_i \sigma_i. \tag{8}$$

This can be seen to be the case because $\sigma_i \approx 1/V_i$, giving $\rho_i = m_i/V_i$. Equation (8) has the advantage over the normal SPH density estimate, $\rho_i = \sum_j m_j W_j$, that any density jumps across fluid interfaces are not smoothed out.

We use a weakly compressible form of SPH where the pressure P_i is directly related to the density ρ_i by the equation of state

$$P_i = \frac{c_0^2 \rho_0}{\gamma} \left(\left(\frac{\rho}{\rho_0} \right)^{\gamma} - 1 \right) + P_0, \tag{9}$$

where c_0 is the numerical speed of sound, γ is a constant equal to 7 and ρ_0 is the reference density of the fluid. P_0 is a background pressure which is found to improve stability and remove tensile issues. If there are any free surfaces in the system, however, P_0 must equal 0. Equation (9) is called the Tait equation. The speed of sound should be set to be approximately 10 times the maximum velocity of the fluid in the system so as to ensure density fluctuations are no larger than 1% of the reference density [21].

The gradient of pressure at a particle i is then given by

$$\nabla P_i = \sigma_i \sum_{j=1}^{N} \left(\frac{P_j}{\sigma_j^2} + \frac{P_i}{\sigma_i^2} \right) \nabla W_{ij}. \tag{10}$$

1.3.1. Viscosity

To compute the viscosity a finite difference approximation is combined with the SPH gradient giving

$$\mu\nabla^2\mathbf{u}_i = \mu\sigma_i \sum_{j=1}^{N} \left(\frac{1}{\sigma_j^2} + \frac{1}{\sigma_i^2} \right) \frac{\mathbf{u}_{ij}}{r_{ij}} \frac{dW_{ij}}{dr_{ij}}, \tag{11}$$

with $\mathbf{u}_{ij} = \mathbf{u}_i - \mathbf{u}_j$ and $r_{ij} = |\mathbf{x}_i - \mathbf{j}|$. To see why (11) has this form, consider $\nabla^2\mathbf{u} = \nabla \cdot (\nabla\mathbf{u})$. We can approximate this with the SPH divergence estimate of the velocity gradient tensor, i.e.

$$\nabla \cdot (\nabla)u \approx \sum_j \nabla W_{ij} \cdot \nabla\mathbf{u}, \tag{12}$$

but

$$\nabla W_{ij} = \mathbf{e}_{ij} \frac{\mathrm{d}W_{ij}}{\mathrm{d}r_{ij}} \tag{13}$$

giving

$$\nabla \cdot (\nabla)u \approx \sum_j \mathbf{e}_{ij} \cdot \nabla\mathbf{u} \frac{\mathrm{d}W_{ij}}{\mathrm{d}r_{ij}}. \tag{14}$$

physically the vector $\mathbf{e}_{ij} \cdot \nabla\mathbf{u}$ can be thought of as the rate of change of the velocity vector in the direction $\mathbf{e}_{ij}$ which can clearly be approximated as

$$\mathbf{e}_{ij} \cdot \nabla\mathbf{u} \approx \frac{\mathbf{u}_{ij}}{r_{ij}}, \tag{15}$$

Combining this with equation (14) and symmetrizing yields equation (11).

2. Geometric Objects

While faceted geometries present a very general and portable solution to including wall-boundaries in SPH simulations, allowing complex and arbitrary shapes, they are not without issues [22]. In many situations simplified or idealized geometries suffice to elucidate salient elements of the flows at hand. For this reason we also consider the inclusion of basic geometric objects into our SPH code.

By geometric objects we mean objects whose shapes are easily representable as equations, such as spheres, cylinders, torii, etc. Such objects have several advantages over faceted surfaces; firstly the only values which need to be stored in memory are the relevant parameters. For instance for a sphere we need only store four floating point numbers, the radius R and the 3D centre $\mathbf{r}_c$. While not generally an issue for CPU based implementations this could be considered a benefit for GPU based codes which have access, relatively, to less RAM.

The second and primary advantage is that such simulations are robust to changes in resolution. For a complex faceted geometry it is often necessary to coarsen an initially fine mesh so that the extent of the facets roughly matches the resolution of the simulation. If the resolution then changes, further processing might be required. Since geometric objects are purely defined by their parameters this is clearly not an issue.

These geometric objects, in general, have curved surfaces. This fact should be taken into account to ensure the correct interpolation of quantities over the boundary. Consider

reflecting a particle in a circular boundary with radius R and centre $\mathbf{r}_c$. The signed distance of the particle to the circle is d. The sign of d indicates whether we are inside the circle or not; $d > 0$ means inside and $d < 0$ means outside. If the fluid particle is at $\mathbf{r}$ then $d = R - |\mathbf{r} - \mathbf{r}_c|$ and the reflected particle will be placed at

$$\mathbf{r}' = \mathbf{r} + \frac{(\mathbf{r} - \mathbf{r}_c)}{|\mathbf{r} - \mathbf{r}_c|} 2d. \tag{16}$$

Assuming a regular spacing of SPH particles we would expect that around the circle with radius $R - d$ there would be $2\pi(R - d)/\Delta x$ particles, where Δx is the average particle spacing in the domain. This means that around the reflected circle there will also be $2\pi(R-d)/\Delta x$ reflected particles. However the reflected circle has a radius of $R+d$ so we would instead expect $2\pi(R+d)/\Delta x$ particles to be present if the particles were regularly spaced. Thus for each reflected particle expected we actually have $(R - d)/(R + d)$ reflected particles, meaning each reflected particle represents a different volume then its matching fluid particle. The ratio of the reflected particle j's volume to its matching fluid particle's volume is

$$\beta_j = \frac{R + d}{R - d}. \tag{17}$$

This can be written in terms of the curvature $\kappa = 1/R$:

$$\beta_j = \frac{1 + \kappa d}{1 - \kappa d}. \tag{18}$$

Writing V_f in this form allows us to apply this correction to any arbitrary shape. The volume correction factor for the reflected particle is given by (18) where κ is the curvature of the object at the reflection point. This is generalizable to three dimensions as

$$\beta_j^{\mathrm{3D}} = \left(\frac{1 + \kappa_1 d}{1 - \kappa_1 d}\right)\left(\frac{1 + \kappa_2 d}{1 - \kappa_2 d}\right), \tag{19}$$

where κ_1 and κ_2 are the two principle curvatures of the surface at the closest point to the fluid particle.

Particle spacing is accounted for by σ which is inversely proportional to the volume V. Because curved boundaries lead to either more or fewer reflected particles than would otherwise be expected, we weight the contribution to the fluid particle's σ value from reflected particles by β. This gives a slightly modified form for calculating σ;

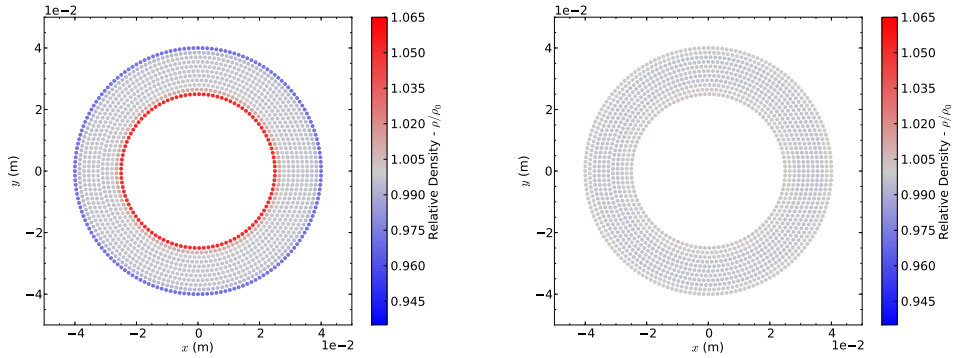
$$\sigma_i = \sum_j W(r_{ij})\beta'_j. \tag{20}$$

where

$$\beta'_j = \begin{cases} \frac{1+\kappa d}{1-\kappa d} & \text{if } j \text{ is a reflected particle} \\ 1 & \text{otherwise} \end{cases}, \tag{21}$$

and similarly for 3D. This ensures σ_i is calculated correctly for fluid particles which are within $2h$ of the boundary.

For reflected particles, σ_i must be copied from the particle from which it was reflected. This is because we cannot calculate σ at the reflected location due to the lack of kernel



(a) Standard Reflection

(b) Volume Correction

Figure 1: The calculated density field using only reflection (1a) and reflection with volume correction (1b). It is clear that the density is better estimated when using β.

support. If we are reflecting in a straight boundary simply setting $\sigma_{\text{refl}} = \sigma_{\text{fluid}}$ is correct because the σ's for the reflected and matching fluid particles will be equal. For curved boundaries however we must weight the reflected particles' σ's because the spacing of the reflected particles is different from that of the fluid particles from which σ is being copied. The correct weighting is again given by the volume factor:

$$\sigma_{\text{refl}} = \sigma_{\text{fluid}}\beta. \tag{22}$$

The correction becomes less important as the spatial resolution is increased. Since we expect $d \approx \Delta x$ (or $\Delta x/2$ if not using boundary particles) as we decrease Δx we have

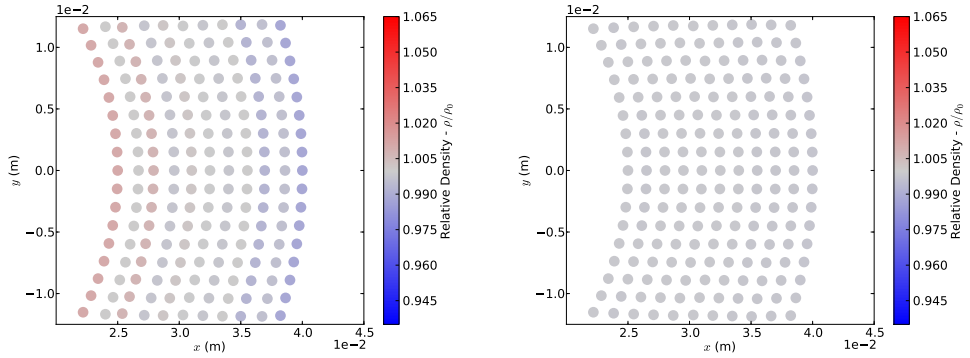
$$\lim_{d\to 0} \beta = 1 \tag{23}$$

as expected, however results show that even for moderately high resolutions this correction still confers a benefit over simply reflecting.

2.1. Example: Couette Flow

An example where curved boundaries are useful is that of a rotating flow between two cylinders, known as Couette flow. In this example a fluid with density $\rho = 1000\text{kgm}^{-3}$, dynamic viscosity $\mu = 0.001\text{Pas}$ is contained between two cylinders of radius $R_i = 2.5\text{cm}$ and $R_o = 4.0\text{cm}$. For the hydrostatic case $\omega = 0\text{s}^{-1}$; figure 1 shows the relative density at the zeroth timestep of two simulations, one with the volume correction factor applied and one without. It is clear that the volume correction factor has improved the density estimate significantly.

After a time $T = 1.0\text{s}$ (Figure 2) in the volume corrected simulation the particles maintain the correct spacing, whereas without this correction factor the particles rearrange themselves to attempt to even out the density. Even with this rearrangement the density has not fully corrected itself by 1s; the particles near the inner wall are pushed



(a) Standard Reflection

(b) Volume Correction

Figure 2: The calculated density field using only reflection (2a) and reflection with volume correction (2b). It is clear that the density is better estimated when using β and that the particles correctly maintain their spacing.

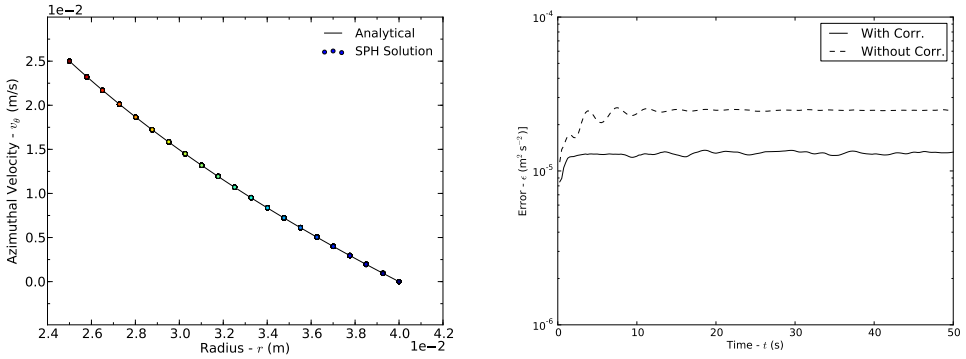


Figure 3: 3a shows the analytical solution for the Couette flow compared to the SPH solution showing good agreement. 3b shows the error shown as a function of time for Couette flow. The solid line represents the solution with volume-factor correction and the dotted line without. It is apparent that the correction improves the solution markedly.

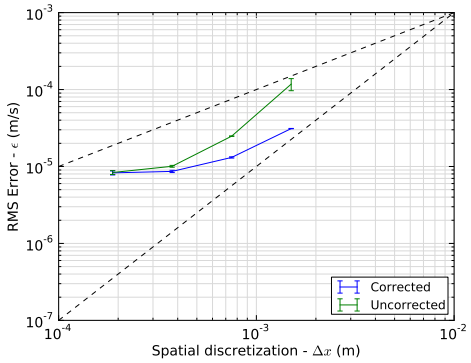


Figure 4: The error in the SPH solution as a function of spatial discretization length for both the simple reflection and the corrected reflection. One can see that the correction significantly improves the solution at low to moderate resolutions

away whereas the particles near the outer wall are pulled towards it. The effect is smaller for the outer wall simply because it has a lower curvature.

Next consider the case where the inner cylinder rotates with angular velocity $\omega = 1\text{s}^{-1}$. At steady state the analytical velocity profile as a function of radius - r - is

$$v_r = 0, \qquad v_\theta = \frac{\omega}{1-\eta^2}\left(R_i^2\, r^{-1} - \eta^2\, r\right), \tag{24}$$

where $\eta = R_i/R_o$, see [23]. The error of the SPH solution is

$$\varepsilon = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(v_{\theta,i} - v_\theta(r_i))}, \tag{25}$$

where the sum is over SPH fluid particles only. Figure 3b shows the error in the solution against time for both the standard reflection and volume-corrected reflection. It is clear that the correction leads to large improvement in the solution demonstrating that this is not something which can be ignored. Figure 4 shows the error for both corrected and uncorrected reflection as a function of the spatial discretization length Δx. It shows that, as expected, the correction makes less difference as we increase the resolution. However, except for at the very highest resolution (>85,000 particles) the correction still provides a noticeable improvement to the solution. Since the domain (and hence fluid flow) for this example is curved, a pressure gradient is set up which depends on r. Because of this pressure gradient a small background pressure of $P_0 = 2.5$ is added to improve the stability and avoid tensile instability. For consistency this pressure gradient was kept constant at 2.5 across all Δx, which leads to a slightly strange rate of convergence since at higher resolutions the pressure required to avoid tensile instability actually decreases.

3. B-Splines Curves

Non-uniform rational B-splines (NURBSs) are a superset of the perhaps better known Bezier curves. They have become well established as the *de facto* standard for represent-

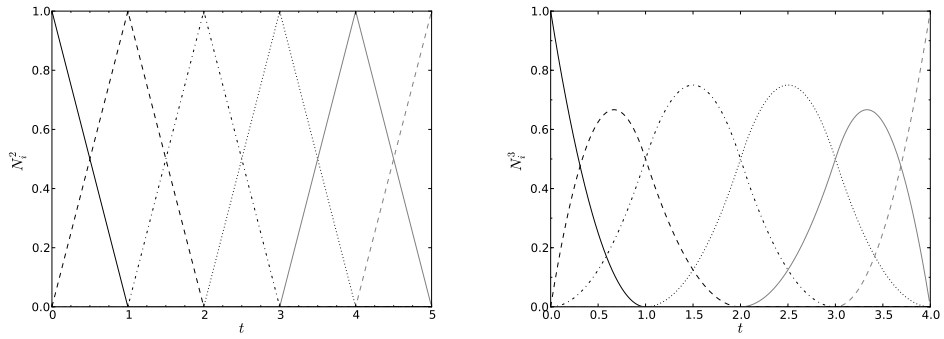


Figure 5: B-spline basis functions N_i^k for $k = 2, 3$ with the non-periodic knot vector $\mathbf{T} = [0\,0\,0\,1\,2\,3\,4\,4\,4]$.

ing and drawing curves and surfaces in CAD packages. The ability to natively include them in a CFD simulation would be therefore very useful.

A NURBS curve is defined by its control points $P = \{\mathbf{P}_i, i = 1...n\}$, their weights $\Omega = \{w_i, i = 1...n\}$ and the so-called knot vector $\mathbf{T} \in \mathbb{R}^{k+n}$. The curve is then parameterized as

$$\mathbf{C}(t) = \frac{\sum_{i=1}^{n} w_i N_{i,k}(t)\mathbf{P}_i}{\sum_{i=1}^{n} w_i N_{i,k}(t)} = \sum_{i=1}^{n} R_{i,k}(t)\mathbf{P}_i, \tag{26}$$

where k is the order of the B-spline interpolation, $k = 1$ being piecewise constant, 2 being piecewise linear, and so on. Where we have set

$$R_{i,k}(t) = \frac{w_i N_{i,k}(t)}{\sum_{j=1}^{n} w_j N_{j,k}(t)} \tag{27}$$

for brevity. The B-spline basis function of order k for control point i is defined by the recursive relationship [24, 25]

$$N_{i,k}(t) = N_{i,k-1}(t)\frac{t - T_i}{T_{i+k-1} - T_i} + N_{i+1,k-1}(t)\frac{T_{i+k} - t}{T_{i+k} - T_{i+1}} \tag{28}$$

for $k > 1$ and for $k = 1$

$$N_{i,1}(t) = \begin{cases} 1 & \text{if } t \in [T_i, T_{i+1}) \\ 0 & \text{otherwise} \end{cases} \tag{29}$$

The basis functions satisfy the partition of unity property meaning $\sum_i N_{i,k}(t) = 1\ \forall k, t$. A similar relation holds for defining a NURBS surface in 3D as a function of two parameters

$$\mathbf{S}(t, u) = \frac{\sum_{i=1}^{p}\sum_{j=1}^{q} N_{i,k}^1(t) N_{j,l}^2(u) w_{i,j}\mathbf{P}_{i,j}}{\sum_{i=1}^{p}\sum_{j=1}^{q} N_{i,k}^1(t) N_{j,l}^2(u) w_{i,j}}, \tag{30}$$

where in 3D the control points have the form of a regular grid with the number of points in the t and u directions respectively being p and q. The basis functions N^1 and N^2 are

different in the two directions only by virtue of having possibly different knot vectors. Figure 5 shows an example of the basis functions $N_{i,k}$ for $k = 2$ and 3.

For ease of discussion in all examples presented here the weights w_i are set equal to unity. This simplifies the maths but retains the key aspects for application to SPH. For the full derivations and expressions we refer the interested reader to [26] for example.

3.1. Gradients and Curvature

Because B-Spline curves are defined mathematically it is easy to calculate quantities such as normal vectors, tangents and curvatures given a parameter value t. The gradient of the curve $\mathbf{C}(t)$ is just the tangent vector and is given by

$$\frac{\mathrm{d}\mathbf{C}}{\mathrm{d}t} = \sum_{i=1}^{n} N'_{i,k}(t) P_i, \tag{31}$$

with

$$N'_{i,k}(t) = \frac{N_{i,k-1}(t) + (t - T_i)N'_{i,k-1}(t)}{T_{i+k-1} - T_i} + \frac{(T_{i+k} - t)N'_{i+1,k+1}(t) - N_{i+1,k-1}(t))}{T_{i+k} - Ti + 1}, \tag{32}$$

for $k > 2$. For $k = 2$

$$N'_{i,2}(t) = \frac{N_{i,1}(t)}{T_{i+1} - T_i} - \frac{N_{i+1,1}(t)}{T_{i+2} - T_{i+1}}. \tag{33}$$

Similarly the second derivative is

$$\frac{\mathrm{d}^2\mathbf{C}}{\mathrm{d}t^2} = \sum_{i=1}^{n} N''_{i,k}(t) P_i. \tag{34}$$

with

$$N''_{i,k}(t) = \frac{2N'_{i,k-1}(t) + (t - T_i)N''_{i,k-1}(t)}{T_{i+k-1} - T_i} + \frac{(T_{i+k} - t)N''_{i+1,k-1}(t) - 2N'_{i+1,k+1}(t)}{T_{i+k} - T_{i+1}} \tag{35}$$

for $k > 3$. For $k = 3$

$$N''_{i,3}(t) = 2\left(\frac{N'_{i,2}(t)}{T_{i+2} - T_i} - \frac{N'_{i+1,2}(t)}{T_{i+3} - T_{i+1}}\right). \tag{36}$$

In 2 dimensions the normal to the curve can be found by taking the cross product with a unit vector pointing along the third (z) axis, i.e.

$$\mathbf{n}(t) = \frac{\mathbf{t}(t) \times \hat{\mathbf{z}}}{|\mathbf{t}(t)|}, \tag{37}$$

and the curvature at any point is given by

$$\kappa(t) = \frac{|\mathbf{t}(t) \times \mathbf{C}''(t)|}{|\mathbf{t}(t)|^3}. \tag{38}$$

3.2. Computational Issues

There are several computational factors to consider before using NURBS with reflected boundaries in SPH. Foremost amongst these is the ability to quickly reflect the fluid particle accurately across the NURBS curve or surface. To perform this operation we need to know the normal, closest point to the surface and, to properly account for the curved nature of these boundaries, the curvature. Given the parameter value which corresponds to the closest point we can calculate all these auxiliary quantities directly from the analytical forms (37), (26) and (38). The difficulty lies in quickly yet accurately calculating the value $t_{\min}$ which locates the closest point on a curve to an arbitrary point $\mathbf{x}$ in $\mathbb{R}^d$.

It is the experience of the authors that in order to obtain satisfactory containment of SPH particles within the domain it is necessary to place a stationary layer of particles along the boundary. We utilize these boundary particles to quickly locate $t_{\min}$.

Each boundary particle stores the value of t which corresponds to its location. When reflecting a fluid particle, a nearby boundary particle can be located quickly using the linked-cell-grid method used in SPH codes and its value of t is taken as an initial guess for $t_{\min}$.

We use the Newton-Raphson method to find the minimum of the function Δ with respect to t.

$$\Delta(t) = ||\mathbf{x} - \mathbf{C}(t)||^2 \tag{39}$$

$$= (\mathbf{x} - \mathbf{C}(t)) \cdot (\mathbf{x} - \mathbf{C}(t)) \,. \tag{40}$$

$\Delta(t)$ gives the squared distance between a point $\mathbf{x}$ and the point on the NURBS curve defined by parameter value t.

To perform Newton-Raphson optimization we need to know both the first and second derivatives of Δ. By expanding the dot product in (40) it is simple to show that

$$\frac{\mathrm{d}\Delta}{\mathrm{d}t} = 2\,(\mathbf{C} - \mathbf{x}) \cdot \frac{\mathrm{d}\mathbf{C}}{\mathrm{d}t}. \tag{41}$$

The quantity $\frac{\mathrm{d}\mathbf{C}}{\mathrm{d}t}$ represents the tangent vector to the curve at t, so at any extrema the fact $\frac{\mathrm{d}\Delta}{\mathrm{d}t} = 0$ implies that the vector between the point $\mathbf{x}$ and the extremum is normal to the curve as expected. By differentiating again we get the second derivative

$$\frac{\mathrm{d}^2\Delta}{\mathrm{d}t^2} = 2\left[\mathbf{t} \cdot \mathbf{t} + (\mathbf{C} - \mathbf{x}) \cdot \frac{\mathrm{d}^2\mathbf{C}}{\mathrm{d}t^2}\right] \tag{42}$$

where we have set $\frac{\mathrm{d}\mathbf{C}}{\mathrm{d}t} = \mathbf{t}$ for conciseness. The iterative sequence for optimizing t is given by

$$t_{n+1} = t_n - \frac{\Delta'(t_n)}{\Delta''(t_n)}, \tag{43}$$

which, using the results above, is

$$t_{n+1} = t_n - \frac{(\mathbf{C} - \mathbf{x}) \cdot \mathbf{t}}{\mathbf{t} \cdot \mathbf{t} + (\mathbf{C} - \mathbf{x}) \cdot \frac{\mathrm{d}\mathbf{t}}{\mathrm{d}t}}. \tag{44}$$

The iterative scheme (44) is only useful if it converges quickly. In typical SPH simulations

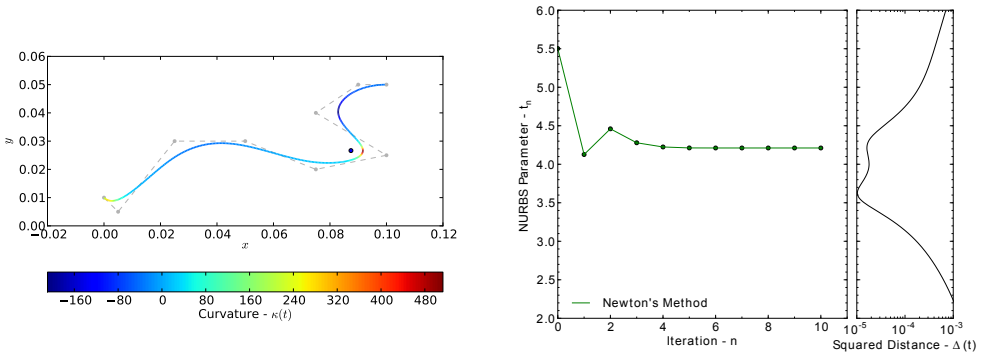


Figure 6: An example NURBS curve. If the location $\mathbf{x}$ to which the closest point on the curve $\mathbf{x}_{np}$ is desired is near a region of high curvature, then finding $\mathbf{x}_{np}$ using Newton-Raphson can fail due to a local minimum appearing in the function $\Delta(t)$.

it is found that with a tolerance of $0.01\Delta x$ (44) always converges within 10 iterations and usually with as few as 3. This is partly due to the strength of the Newton-Raphson method for the function $\Delta(t)$ and partly due to the initial guesses for t_0 being close to $t_{\min}$.

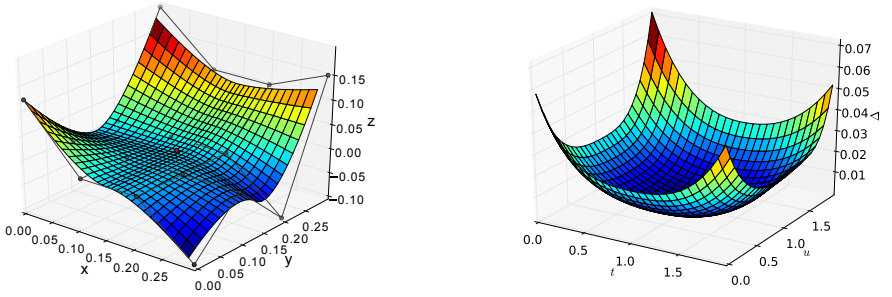
However, certain situations can lead to difficulty for the location of $t_{\min}$. If there are areas of high positive curvature there can exist two local minimia for the distance function. Figure 6 illustrates this and shows how this causes the optimization to converge to the incorrect value for $t_{\min}$.

For certain choices of t_0 the method diverges. There are two possible causes for this; either it is spurious or the closest point is truly one of the end points. This can be checked by comparing t_n to our initial guess t_0. Since we know t_0 must be close to the true value for $t_{\min}$ we check to see if our point is close to either end. If this is the case we can assume the method has not spuriously diverged.

If, however, we hit either $t = 0$ or $t = (N - K + 1)$ and t_0 is not close to either end we assume this divergence is spurious, move t_0 slightly and then try again. In these situations we define t_0 to be close to the endpoints if it is within $(N - K + 1)/(2N)$ of either limit. If there are control points bunched near either end point this might be a poor definition but in practice it was found to be adequate to distinguish the cases.

An additional check which was found to improve stability is to look at the gradient. For all cases except those of high curvature, the function $\Delta(t)$ is observed to be quasi-convex meaning we always wish to be moving in the direction of negative gradient. If a Netwon-Raphson step is hit which would move up the gradient it is instead moved the same distance in the direction of negative gradient.

The boundary particles must be placed at regular intervals of Δx along the curve. If the control points for the B-spline curve are all the same distance apart, and the knot vector intervals are equal (i.e. $T_{i+1} - T_i = T_{i+2} - T_{i+1}$ for all $i = 1, ...(N + K - 3)$) then moving a set distance in t will move a set amount along C in $\mathbb{R}^d$ at all points along C. However, this is quite a strict requirement on the curve which would significantly



(a) NURBS Surface

(b) $\Delta(t, u)$

Figure 7: An example of a NURBS surface $S(t, u)$ in 3D. It has 16 control points and $K = 3$. Also shown in the squared distance from the red point $\Delta(t, u)$ as a function of the two parameter values.

reduce its usefulness. For this reason we numerically integrate the arc-length $s(t)$ along the curve and place boundary particles at regular intervals.

$$s(t) = \int_0^t \left.\frac{\mathrm{d}\mathbf{C}}{\mathrm{d}t'}\right|_{t'=t} dt' \tag{45}$$

$$\approx \sum_{i=1}^{n} |\mathbf{C}_i - \mathbf{C}_{i-1}| \, (t_i - t_{i-1}). \tag{46}$$

While this is a relatively slow operation, it only needs to be performed once at the very beginning of the simulation and placing boundary particles in this manner removes any restrictions on the control-points and knot-vector.

3.3. Application in Three Dimensions

The examples and methods shown in this paper focus on 2D NURBS curves, but, as mentioned above, it is also possible to define NURBS surfaces in 3D. The Newton-Raphson method used to locate closest points also works for a two-dimensional parameter space. The function Δ is still well behaved for a NURBS surface meaning the Newton-Raphson method should be easily applicable to 3D simulations. Figure 7b shows the behaviour of Δ over the parameter space for the example NURBS surface shown in 7a.

The larger issue in 3D comes when placing the boundary particles. In 2D it is possible to simply numerically integrate the arc length along the curve with a fine step size and place particles at regular intervals of Δx. Because the mapping between (t, u) and the position on the surface is non-trivial it is difficult to say what pattern one must follow in (t, u) space to give an even distribution of boundary particles. This does not mean that the NURBS surfaces cannot be used as boundaries in SPH. It is reasonable to expect that the distances between neighbouring control points be similar. By taking a relatively fine grid in (t, u) this would allow us to place points onto the surface with a good coverage.

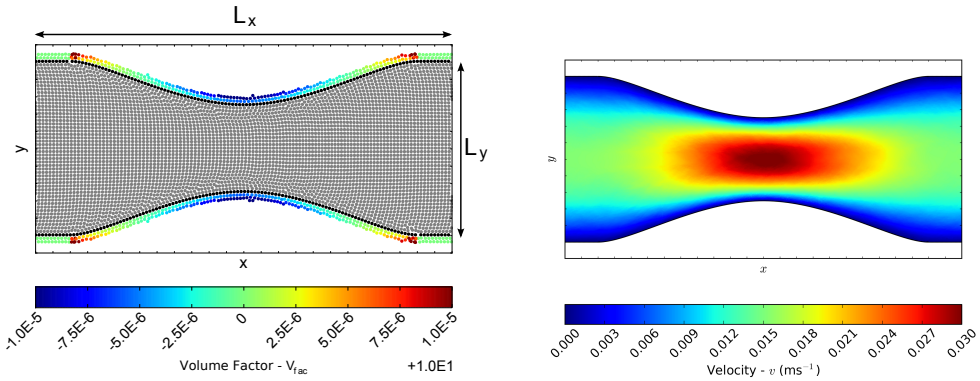


Figure 8: An example of a simulation using B-spline wall boundaries. In 8a The boundary particles are shown in black, fluid particles in grey. The reflected ghost particles are coloured according their corresponding volume-factor correction which is calculated based on the curvature of the B-spline curve. 8b shows the final steady-state velocity magnitude.

These points could then be used as in 2D to provide initial guesses for t and u when locating closest points *but* they could not be used as boundary particles as their spacing would be uneven.

4. Examples

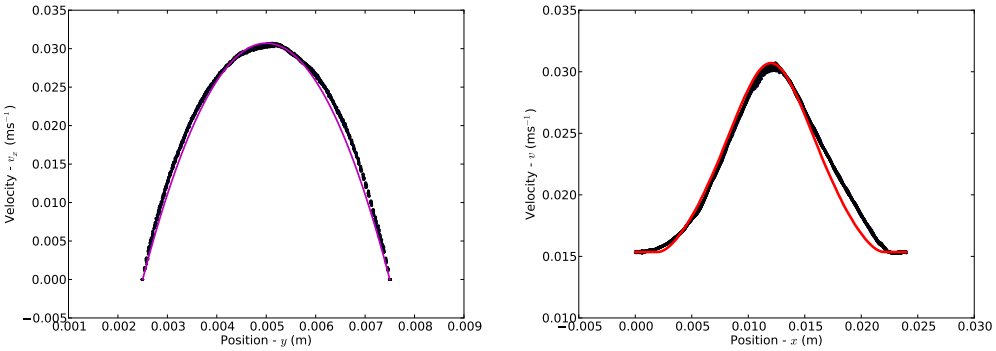
4.1. Throttled Pipe Flow

As a simple example of using B-spline curves we show a variant of the plane Pousielle flow in which fluid is driven through a pipe under a pressure gradient but where the pipe has a contraction. Figure 8 shows the simulation setup with fluid particles shown in grey, boundary particles in black and the reflected particles coloured by their volume-factor correction. The size of the system is $L_x = 0.024$m, $L_y = 0.01$m and at the narrowest point of the constriction the width is $L_y/2$, the pressure gradient applied is 50Pa and the viscosity μ is chosen to be 0.01, this gives a Reynold's number of $Re \approx 30$. The final velocity magnitude is shown in Figure 8b, we can see that as it should the velocity goes to zero at the boundaries correctly and increases in the constriction. Defining the half-width of the channel to be $R(x)$ with $R(0) = L_y/2$ we have the flux

$$Q(x) = \int_{-R(x)}^{+R(x)} v_x(x, y)\, dy. \tag{47}$$

The flux $Q(x)$ must be constant for all x due to conservation of mass. If we assume that at each x in the system the velocity profile is quadratic, in line with standard Pousielle flow, then we can relate the velocity at the each x to the velocity at $x = 0$ via equation (47);

$$v_x(x, y) = \frac{R(0)}{R(x)} v_x(0, y\frac{R(0)}{R(x)}) \tag{48}$$



(a) y Midline Velocity

(b) x Midline Velocity

Figure 9: Velocity profiles for the x and y midlines, that is lines of constant y and x respectively.

This is only an approximation since in reality we must have non-zero y components to the velocity field. At the two midlines[1], however, we would expect v_y to be zero due to symmetries. If we look at Figure 9b, which shows the profile of v_x along the line $x = L_x/2$, we can see that it closely matches the expected shape of a parabola. Similarly figure 9a shows how v_x varies along the line $y = 0$ we can see it nicely matches $v_x(x, 0)$ given by (48) ($R(x)$ is calculated numerically).

4.2. Interface Forces

In this example we show a droplet of fluid spreading along a boundary due to surface forces, which are included by the continuum surface force model [10, 27]. The two fluids have a density ratio of 10, with the droplet being heavier, and equal viscosities of 0.001 Pa s. There is no gravity in this simulation. The droplet is partially wetting with a contact angle $\theta = 30°$. We can see the that the surface tension forces at the NURBS boundary pull the droplet from it's initial configuration and it becomes held in an indentation with the correct contact angle. This demonstrates that as well as shear-forces in single phase flows the NURBS boundaries can handle additional body forces.

5. Conclusions

In this paper we have highlighted issues involved with implementing reflected boundary conditions for curved objects in SPH. It was shown that when using geometric objects care must be taken to correctly renormalize the weightings for both the fluid particle and the corresponding reflected particle and that without accounting for the 'volume-factor' the simulations are demonstrably less accurate. Including objects as geometrical representations is desirable as often one desires a simplified setup to mimic experiments or to

[1] Lines defined by $x = L_x/2$ and $y = 0$

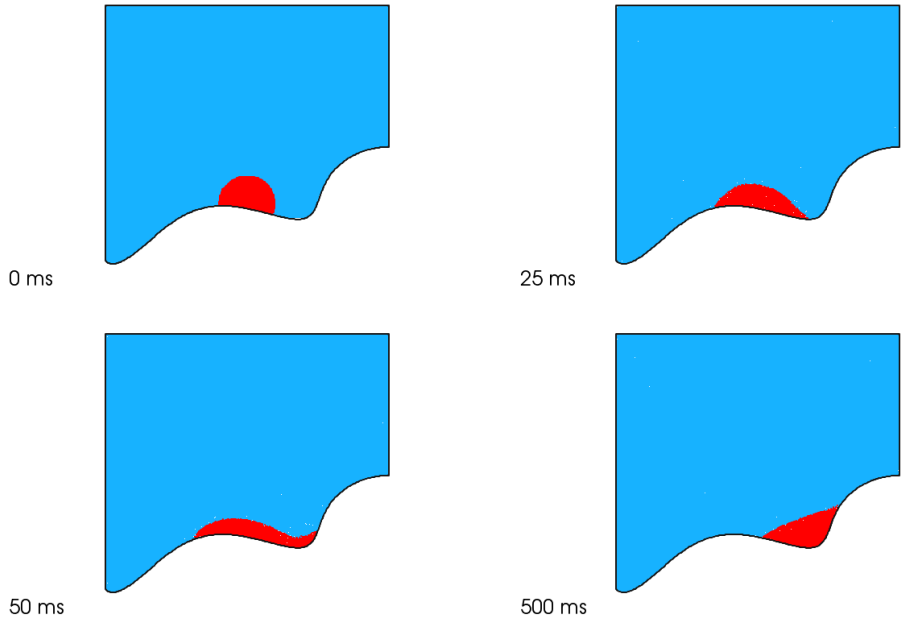


Figure 10: A multi-phase simulation of two fluids with contact angle of 30° and $\rho_1/\rho_2 = 10$

remove extraneous factors from consideration. Further this framework could easily be extended to include more complicated objects such as superquadrics.

NURBS curves and surfaces are the industry standard for much CAD and technical drawing work, therefore the fact that they can be directly included in a practical manner into an SPH implementation is a very desirable feature. This paper examined the issues involved with their application as boundaries in SPH using reflected particle boundaries. It was shown that in 2D that the Newton-Raphson method provides a very quick way of searching the parameter space and locating closest points, although care must be taken to detect the occasional divergences which occur as part of the method. In 3D it is expected that the same method should also work well.

It has been shown that the handling of boundaries in SPH has a large effect on the simulation results [15], for this reason any improvement to SPH in this regard is a useful addition.

[1] Ø ystein Eklund Krog. GPU-based Real-Time Snow Avalanche Simulations. (June), 2010.

[2] Matthias Müller, David Charypar, and Markus Gross. Particle-Based Fluid Simulation for Interactive Applications. *Fluid Dynamics*, pages 154–159, 2003.

[3] Pourya Omidvar, Peter K Stansby, and Benedict D Rogers. SPH for 3D floating bodies using variable mass particle distribution. *International Journal for Numerical Methods in Fluids*, 2012.

[4] Saeed Ovaysi and Mohammad Piri. Direct pore-level modeling of incompressible fluid flow in porous media. *Journal of Computational Physics*, 229(19):7456–7476, September 2010.

[5] R Das and P W Cleary. Effect of rock shapes on brittle fracture using Smoothed Particle Hydrodynamics. *Theoretical and Applied Fracture Mechanics*, 53(1):47–60, 2010.

[6] L B Lucy. A numerical approach to the testing of the fission hypothesis. *The Astronomical Journal*, 82(12):1013–1024, 1977.

[7] R A Gingold and J J Monaghan. Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Monthly Notices of the Royal Astronomical Society*, 181(181):375–389, 1977.

[8] Takahiro Harada, Seiichi Koshizuka, and Yoichiro Kawaguchi. Smoothed particle hydrodynamics in complex shapes. In *Proc of spring conference on computer graphics*, pages 26–28, 2007.

[9] S. Kulasegaram, J. Bonet, R. W. Lewis, and M. Profit. A variational formulation based contact algorithm for rigid boundaries in two-dimensional SPH applications. *Computational Mechanics*, 33(4):316–325, March 2004.

[10] X Y Hu and N A Adams. A multi-phase SPH method for macroscopic and mesoscopic flows. *Journal of Computational Physics*, 213(2):844–861, 2006.

[11] M. B. Liu and G. R. Liu. Smoothed Particle Hydrodynamics (SPH): an Overview and Recent Developments. *Archives of Computational Methods in Engineering*, 17(1):25–76, February 2010.

[12] Donald Shepard. A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the 1968 23rd ACM national conference*, ACM '68, pages 517–524, New York, NY, USA, 1968. ACM.

[13] Volker Springel. Smoothed Particle Hydrodynamics in Astrophysics. *Annual Review of Astronomy and Astrophysics*, 48:391–430, 2010.

[14] Holger Wendland. Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *Advances in Computational Mathematics*, 4(1):389–396, 1995.

[15] S J Cummins, T B Silvester, and P W Cleary. Three-dimensional wave impact on a rigid structure using smoothed particle hydrodynamics. *International Journal for Numerical Methods in Fluids*, 68(12):1471–1496, 2012.

[16] Ø ysteinE. Krog and AnneC. Elster. Fast GPU-Based Fluid Simulations Using SPH. In Kristján Jónasson, editor, *Applied Parallel and Scientific Computing*, volume 7134 of *Lecture Notes in Computer Science*, pages 98–109. Springer Berlin Heidelberg, 2012.

[17] GG Pereira and Mahesh Prakash. SPH modelling of fluid at the grain level in a porous medium. *Applied Mathematical Modelling*, (December):1–6, 2010.

[18] G Oger, M Doring, B Alessandrini, and P Ferrant. Two-dimensional SPH simulations of wedge water entries. *Journal of Computational Physics*, 213(2):803–822, April 2006.

[19] M Ferrand, D R Laurence, B D Rogers, D Violeau, and C Kassiotis. Unified semi-analytical wall boundary conditions for inviscid , laminar or turbulent flows in the meshless SPH method. *International Journal for Numerical Methods in Fluids*, 2012.

[20] J Feldman and J Bonet. Dynamic refinement and boundary contact forces in SPH with applications in fluid flow problems. *International Journal for Numerical Methods in Engineering*, 72(3):295–324, 2010.

[21] Markus Becker and Matthias Teschner. Weakly compressible SPH for free surface flows. *Eurographics*, pages 1–8, 2007.

[22] Daniel J. Barker, Stephen J. Neethling, and Gopalkrishnan Parameswaran. SPH Simulation of Packed-beds and Columns Applied to Heap-leaching. In C.B. Solnordal, P. Liovic, G.W. Delaney, and P.J. Witt, editors, *9th International Conference on CFD in the Minerals and Process Industries*, 2012.

[23] M C Wendl. General solution for the Couette flow profile. *Physical review. E, Statistical physics, plasmas, fluids, and related interdisciplinary topics*, 60(5 Pt B):6192–4, November 1999.

[24] M.G. Cox. The Numerical Evaluation of B-Splines. *IMA Journal of Applied Mathematics*, 10(2):134–149, 1971.

[25] Carl de Boor. On Calculating With B-Splines. *J. Approximation Theory*, 6:50–62, 1972.

[26] David F. Rogers. *An Introduction to NURBS: With Historical Perspective*. Morgan Kaufmann, San Francisco; London, 2001.

[27] J U Brackbill, D B Kothe, and Charles Zemach. A Continuum Method for modeling Surface Tension. *Journal of Computational Physics*, 100:335–354, 1992.