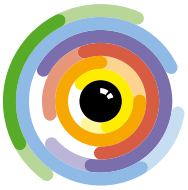
# Fluid Simulation in Computer Graphics Exercise Sheet 4



RWTH AACHEN UNIVERSITY

Prof. Dr. Jan Bender, MSc. José Fernández

Contact: {bender, fernandez}@cs.rwth-aachen.de

Deadline: **18/12/19** (three weeks)

In this assignment sheet, we will study and implement an implicit pressure solver based on a modified variant of Position Based Fluids, due to Macklin and Müller [MM13]. Their publication can be downloaded from L2P. It is a short paper - you are encouraged to spend some time reading it.

## An introduction to Position Based Fluids

In this exercise, we will use a modified version of Position Based Fluids (PBF) [MM13], as an alternative solver to our current implementation of Weakly compressible SPH (WCSPH). Whereas WCSPH simply adds pressure forces and integrates the equations of motion explicitly (i.e., one does not have to solve a system of equations), PBF imposes constraints on the particles positions and solves for the *constraint forces* which approximately lead to fulfillment of the constraints. Since we seek to simulate nearly incompressible fluids, it is natural to pose constraints that require the density measured at various particles to be equal to rest density. Mathematically, we write

$$C_i(\hat{\mathbf{x}}) = \frac{\rho_i(\hat{\mathbf{x}})}{\rho_0} - 1 = 0. \tag{1}$$

The above *constraint function* $C_i$ requires that the density of particle $i$ should be equal to rest density $\rho_0$. Here, we've denoted by $\hat{\mathbf{x}} \in \mathbb{R}^{3n}$ the concatenated vector of **all** particle positions involved in the computation of $\rho_i$, including boundary particles. More precisely, we have that

$$\hat{\mathbf{x}} = \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_n \end{pmatrix},$$

where $n$ is the number of particles involved in the computation of $\rho_i$.

PBF follows the general idea of Position Based Dynamics [MHHR07], in which all constraints are solved in a local Newton-like solver. After evolving all particles forward in time according to the influence of external forces such as gravity and viscosity, we seek to find $\Delta\hat{\mathbf{x}}_i \in \mathbb{R}^{3n}$ such that, for each *fluid* particle $i = 1, \ldots, n_f$,

$$C_i(\hat{\mathbf{x}} + \Delta\hat{\mathbf{x}}_i) = 0. \tag{2}$$

In the above, we have a single scalar equation and $3n$ variables $\Delta\hat{\mathbf{x}}_i$. As it turns out, for such constrained systems, a particular choice of direction for the position corrections is necessary to prevent the constraint forces from performing *work* on the system (i.e. changing the overall energy of the system). Conveniently, this choice can be written

$$\Delta\hat{\mathbf{x}}_i = M^{-1}\nabla C_i \, \lambda_i, \tag{3}$$

for some $\lambda_i \in \mathbb{R}$ and $M$ is a diagonal matrix containing the masses of the particles as diagonal $3 \times 3$ block matrices, in the form

$$M = \begin{pmatrix} m_1 I & 0 & \ldots & 0 \\ 0 & m_2 I & \ldots & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & \ldots & m_n I \end{pmatrix},$$

with $I$ denoting the $3 \times 3$ identity matrix. There is a slight complication with the above definition of the mass matrix: Our boundary particles are static, and so, from the point of view of a dynamical system, they essentially have infinite

mass (i.e., no amount of force will make them move). However, as it turns out, we will not need the mass matrix directly - instead, we will require only its inverse, which is again a diagonal matrix, with the inverse masses on the diagonal. In this case, we *define* the inverse mass of a boundary particle to be zero, which turns out to be consistent with the equations of motion.

This lets us express the position update for particle $i$ in terms of a single variable $\lambda_i$. We now take the first order approximation of our constraint function and obtain

$$C_i(\hat{\mathbf{x}} + \Delta\hat{\mathbf{x}}_i) \approx C_i(\hat{\mathbf{x}}) + (\nabla C_i)^T \Delta\hat{\mathbf{x}}_i = C_i(\hat{\mathbf{x}}) + \underbrace{(\nabla C_i)^T M^{-1} \nabla C_i}_{S_i} \lambda_i = 0. \tag{4}$$

In the above, $S_i$ is a scalar, and clearly, if $S_i \neq 0$, then we may directly obtain $\lambda_i$ by rearranging the equations:

$$\lambda_i = -\frac{C_i(\hat{\mathbf{x}})}{S_i}. \tag{5}$$

However, the above expression may lead to instabilities when neighboring particles are all very close to the boundary of the kernel support. To cope with this, we introduce a regularization parameter $\epsilon \approx 10^{-4}$ and write

$$\lambda_i = -\frac{C_i(\hat{\mathbf{x}})}{S_i + \epsilon}. \tag{6}$$

Having obtained $\lambda_i$ for each constraint (particle), we could in principle compute the position changes from (3). However, we will show that there is a more convenient form available that more closely resembles the typical SPH formulas that we have worked with so far. Let us next derive an expression for the derivative of $C_i$ with respect to a particle $k$. Note that $k$ can represent either a fluid particle or a boundary particle. Recall that $\rho_i = \sum_{j \in \mathcal{N}_i} m_j W_{ij} + \sum_{j \in \mathcal{N}_i^b} \rho_0 V_j W_{ij}$, with $V_j$ the volume of boundary particle $j$. We obtain,

$$\begin{aligned}
\nabla_{\mathbf{x}_k} C_i &= \frac{1}{\rho_0} \sum_{j \in \mathcal{N}_i} m_j \nabla_{\mathbf{x}_k} W_{ij} + \sum_{j \in \mathcal{N}_i^b} V_j \nabla_{\mathbf{x}_k} W_{ij} \\
&= \begin{cases}
\dfrac{1}{\rho_0} \displaystyle\sum_{j \in \mathcal{N}_i} m_j \nabla W_{ij} + \sum_{j \in \mathcal{N}_i^b} V_j \nabla W_{ij} & \text{if } k = i \ \ (k \text{ is a fluid particle}) \\
-\frac{m_k}{\rho_0} \nabla W_{ik} & \text{if } k \neq i \text{ and } k \text{ is fluid particle} \\
-V_k \nabla W_{ik} & \text{if } k \neq i \text{ and } k \text{ is boundary particle}
\end{cases}
\end{aligned} \tag{7}$$

where $\nabla W_{ij}$ and $\nabla W_{ik}$ correspond simply to the standard kernel gradient that we have worked with up to now, $\mathcal{N}_i$ represents the fluid particle neighborhood of fluid particle $i$, and $\mathcal{N}_i^b$ represents the boundary particle neighborhood of fluid particle $i$. However, it is not necessary to assemble $\nabla_{\mathbf{x}_k} C_i$ explicitly since it is only used to compute $S_i$ which can be done by directly summing up the contributions of each particle in the neighborhood:

$$S_i = (\nabla C_i)^T M^{-1} \nabla C_i = \frac{1}{m_i} \left\| \sum_{j \in \mathcal{N}_i} \frac{m_j}{\rho_0} \nabla W_{ij} + \sum_{k \in \mathcal{N}_i^b} V_k \nabla W_{ik} \right\|^2 + \sum_{j \in \mathcal{N}_i} \frac{1}{m_j} \left\| -\frac{m_j}{\rho_0} \nabla W_{ij} \right\|^2 + \sum_{k \in \mathcal{N}_i^b} \frac{1}{m_k} \left\| -V_k \nabla W_{ik} \right\|^2 \tag{8}$$

Note that the last term corresponding to the static boundary particles doesn't need to be computed since Position Based methods assign infinite mass to static bodies, and therefore $\frac{1}{m_k} = 0$.

Summing up all the contributions from all constraint forces on a single particle $i$, we can show that the position update for particle $i$ is:

$$\Delta\mathbf{x}_i = \underbrace{\frac{1}{\rho_0} \sum_{j \in \mathcal{N}_i} \left( \frac{m_j}{m_i} \lambda_i + \lambda_j \right) \nabla W_{ij}}_{\text{fluid contribution}} + \underbrace{\sum_{k \in \mathcal{N}_i^b} \frac{V_k}{m_i} \lambda_i \nabla W_{ik}}_{\text{boundary contribution}}. \tag{9}$$

So far, we have shown how to perform a single position update for all particles such that they approximately satisfy the *linearized* constraint (4). However, our constraint is non-linear. The idea of PBF is then to perform multiple iterations of what we have outlined so far. That is, we perform a sequence of position updates which - given enough iterations - should converge towards the exact solution. We note that in practice, PBF converges extremely slowly to a high accuracy solution. However, usually, this is also not necessary for a satisfactory result: a handful of iterations typically suffices.

So far we have only considered the *equality constraint* $C_i = 0$. In the original PBF paper [MM13], they use this equality constraint along with some artificial pressure terms to prevent particle clumping and instabilities. Instead, we will do as we did with WCSPH and use pressure clamping to avoid attractive forces between particles due to underpressure. Thankfully, this modification is simple: Instead of treating $C_i$ as an equality constraint, we treat it as an inequality constraint of the form $C_i \leq 0$. The only change this needs to our procedure is in the computation of $\lambda_i$:

$$\lambda_i = \begin{cases} -\frac{C_i}{S_i + \epsilon} & \text{if } C_i > 0 \\ 0 & \text{otherwise} \end{cases}. \tag{10}$$

In other words, if the constraint is violated ($C_i > 0$), then we apply a position change through $\lambda_i$. Otherwise, the constraint is already fulfilled and we simply do nothing.

At the end of the algorithm, velocities must be corrected to match the changes to the particle positions. This is straightforward:

$$v_i^{n+1} = \frac{x_i^{n+1} - x_i^n}{\Delta t},$$

where $\mathbf{x}_i^n$ is the initial position at the very beginning of the time step (i.e. before any integration at all).

**Algorithm 1:** Simulation loop for the Position Based Fluids algorithm.

Sample fluid and boundary particles.
Perform a neighborhood search.
**while** $t_{simulation} < t_{end}$ **do**
  $\Delta t = \min(\Delta t_{CFL}, \Delta t_{default})$
  Compute fluid particles density.
  Compute accelerations with gravity and viscosity.
  Integrate particles positions in time using the modified Semi-Implicit Euler scheme.
  Update a neighborhood search.
  **while** *iter* $< n_{PBF\_iterations}$ **do**
    Compute fluid particles density.
    **for all** *particles* $i$ **do**
      Compute $S_i$ from (8)
    **for all** *particles* $i$ **do**
      Compute $\lambda_i$ from (10)
    **for all** *particles* $i$ **do**
      Compute $\Delta \mathbf{x}_i$ from (9)
    **for all** *particles* $i$ **do**
      $\mathbf{x}_i \leftarrow \mathbf{x}_i + \Delta \mathbf{x}_i$
  **for all** *particles* $i$ **do**
    $\mathbf{x}_i^{n+1} \leftarrow \mathbf{x}_i$
    $\mathbf{v}_i^{n+1} \leftarrow \frac{\mathbf{x}_i^{n+1} - \mathbf{x}_i^n}{\Delta t}$
  $t_{simulation} \mathrel{+}= \Delta t$

## Assignment 1 - Position Based Fluids implementation

In this assignment, you will extend your simulator with an implementation of the PBF algorithm that we have outlined above (see Algorithm 1 for reference). You should make your simulator capable of switching out the solver used, so that you can easily re-run the same scenes with different solvers. Come up with some non-trivial scenes to test your PBF solver and compare the solution with respect to the WCSPH solver from previous assignments. Try different number of particles, time step sizes and number of constraint iterations (PBF) or pressure stiffness (WCSPH) and comment on the stability, compressibility and run time differences between them.

Keep in mind that PBF and WCSPH will *not* produce a similar result for similar parameters. For example, since PBF is dissipative by construction, you will need much less (or none) viscosity to get similar results than WCSPH.

**Note:** Typical good test scenes are the *"dam break"* and the *"double dam break"* with consist on one or two boxes of fluid falling inside a closed box. The higher the fluid columns, the harder it is to get a stable simulation.

**Note:** Since comparing PBF and WCSPH should be an important part of your final presentation and report, it is recommended that you spend some time designing good test scenes where you can showcase their main differences. You should prepare plots to support your findings.

## References


[MHHR07] Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. Position based dynamics. *Visual Communication and Image Representation*, 18(2):109–118, 2007.

[MM13] Miles Macklin and Matthias Müller. Position Based Fluids. *ACM Transactions on Graphics*, 32(4):1–5, 2013.