Particle based Viscoelastic Fluid Simultaion

Kasi Sai Teja
Stony Brook University
Stony Brook, NY, US
skasi@cs.stonybrook.edu

Abstract

As a truly mesh-free particle method based upon the Lagrangian formulation, smoothed particle hydrodynamics (SPH) has been applied to a variety of different areas in science, computer graphics and engineering. It has been extended to successfully simulate various phenomena such as multi-phase flows, rigid and elastic solids, and fluid features such as air bubbles and foam. The study is concerned about the application of the (SPH) method within the computational fluid dynamics and elastodynamics. The implemented model of incompressible fluid is tested for internal flows as well as for flows involving a free surface of the fluid. In this paper a new technique to model fluid-fluid interaction based on the Smoothed Particle Hydrodynamics (SPH) method is proposed. We present a new particle-based method for viscoelastic fluid simulation. Incompressibility and particle anti-clustering are enforced with a double density relaxation procedure which updates particle positions according to two opposing pressure terms. Elastic and non-linear plastic effects are obtained by adding springs with varying rest length between particles.

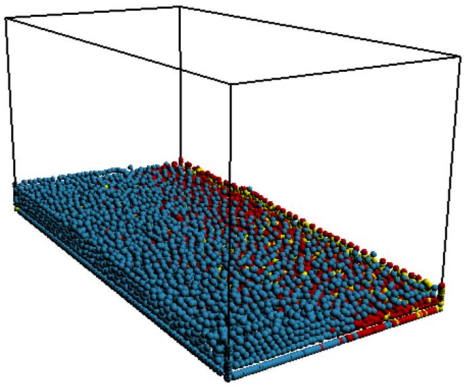

Figure : Particles with varied densities

1. Introduction

Fluid simulation is utilized in many applications including video games, special effects in both film and television, medical and military simulations, and virtual reality. Some applications use a technique known as offline rendering, where the scene or effect is rendered by a large array of computers prior to its use in the application. This can take a varying amount of time depending on the complexity of the simulation, and how many particles are used. The smoothed particle hydrodynamics (SPH) method is a truly meshless Lagrangian numerical technique first introduced to solve gas dynamics problems in astrophysics. The first papers on the SPH method were published by Gingold and Monaghan [2] and Lucy [5] in late 1970s. The simulation of fluids in real-time is a challenging area, with early examples using SPH reproducing relatively simple scenes such as calm bodies of water, gas interaction, and static smoke or fog. More recent implementations consider more of the properties that influence fluid behavior such as splashing, bubbles, correct pressure representation, mixing of different fluid viscosities, and boiling and evaporation. As the available computing power has increased, achieving reliable frame rates in real-time simulations has become possible. However, because the complexity of the simulations has also increased there is still many areas that could benefit from research and refinement.

Smoothed Particle Hydrodynamics is one of the most popular methods in physically-based fluid simulation. Its meshless character makes the method very flexible and enables simulations of physical problems which might be difficult to capture by conventional grid-based methods. This contribution is concerned about its

application within computational continuum mechanics with a focus on the problems of hydrodynamics and elasticity.

1.1 Previous Work

As the subject of this paper, SPH was chosen for a number of reasons: realistic simulations in real-time are achievable; a number of commercial 3D graphical applications for simulating fluid already use SPH; and it is not only limited to fluid based applications, some recent research using SPH include - the calculation of protein-ligand binding rates in BioPhysics (Pan et al., 2015), modelling the sound of a rigid body falling on water in Acoustics (Zhang et al., 2015), and for the estimation of sea wave impact on coastal structures in Coastal Engineering (Altomare et al., 2015).

In 1822 Claude Navier and in 1845 George Stokes formulated the famous Navier-Stokes Equations that describe the dynamics of fluids. Besides the Navier-Stokes equation which describes conservation of momentum, two additional equations namely a continuity equation describing mass conservation and a state equation describing energy conservation are needed to simulate fluids. Stam's method 18 that is grid based is certainly an important step towards real-time simulation of fluids.

1.2 Present contributions

Desbrun used SPH to animate highly deformable bodies 2 . We extend his method focusing on the simulation of fluids. To this end, we derive the viscosity and pressure force fields directly from the Navier-Stokes equation and propose a way to model surface tension forces. For the purpose of interactivity, we designed new special purpose smoothing kernels. Two main categories of simulation methods exist: Eulerian grids and Lagrangian particles. Eulerian methods discretize the problem using a subdivision of the spatial domain and control fluid flow in each cell. Lagrangian methods discretize the fluid mass using particles. We propose a novel procedure for incompressibility and particle anti-clustering. We call this procedure double density relaxation. Loosely speaking, double density relaxation computes two different particle densities: one quantifying the number of neighbors, and another quantifying the number of close neighbors. The force between two particles depends on these two context-dependent measures, instead of depending only on the distance between them. We also present a simple method to simulate viscoelastic substances by insertion and removal of springs between pairs of particles.

In this paper, we also propose a particle-based method to simulate the phenomena emerging from fluid-fluid interaction. With Eulerian, grid-based methods, the simulation of multiple fluids or multiple phases is a difficult problem. Particles seem to be the natural choice in this case. In a particle system, each individual particle can have its own attribute values and particles with different properties can be mixed arbitrarily. In addition, particles can easily be generated and deleted dynamically as needed. The reason for the complexity of fluid behavior is the complex interplay of various phenomena such as convection, diffusion, turbulence and surface tension.

Some set of extensions to the regular SPH method targeting the following phenomena:
Multiple fluids: To simulate multiple fluids with different particle types, we store parameters which are global in standard SPH-models on individual particles and extend the equations to cope with additional particle attributes.

2. SPH Model

SPH is an interpolation method for particle systems. With SPH, field quantities that are only defined at discrete particle locations can be evaluated anywhere in space. For this purpose, SPH distributes quantities in a local neighborhood of each particle using radial symmetrical smoothing kernels. According to SPH, a scalar quantity A is interpolated at location r by a weighted sum of contributions from all particles:

$$A_S(\mathbf{r}) = \sum_j m_j \frac{A_j}{\rho_j} W(\mathbf{r} - \mathbf{r}_j, h),$$

where j iterates over all particles, mj is the mass of particle j, r j its position, ρj the density and Aj the field quantity at r j .

The function W(r,h) is called the smoothing kernel with core radius h. Since we only use kernels with finite support, we use h as the radius of support in our formulation. If W is even (i.e. W(r,h) = W(−r,h)) and normalized, the interpolation is of second order accuracy. The kernel is normalized if

$$\int W(\mathbf{r}) d\mathbf{r} = 1.$$

The particle mass and density appear in Eqn. (1) because each particle i represents a certain volume Vi = mi/ρi . While the mass mi is constant throughout the simulation and, in our case, the same for all the particles, the density ρi varies and needs to be evaluated at every time step. Through substitution into Eqn. (1) we get for the density at location r:

$$\rho_S(\mathbf{r}) = \sum_j m_j \frac{\rho_j}{\rho_j} W(\mathbf{r} - \mathbf{r}_j, h) = \sum_j m_j W(\mathbf{r} - \mathbf{r}_j, h).$$

In most fluid equations, derivatives of field quantities appear and need to be evaluated. With the SPH approach, such derivatives only affect the smoothing kernel. The gradient of A is simply

$$\nabla A_S(\mathbf{r}) = \sum_j m_j \frac{A_j}{\rho_j} \nabla W(\mathbf{r} - \mathbf{r}_j, h)$$

while the Laplacian of A evaluates to

$$\nabla^2 A_S(\mathbf{r}) = \sum_j m_j \frac{A_j}{\rho_j} \nabla^2 W(\mathbf{r} - \mathbf{r}_j, h).$$

where ρi is the density of particle i and W(r,h) is a smoothing kernel. The density ρi can be computed with Eq. (1) yielding

$$\rho_i = \rho(\mathbf{x}_i) = \sum_j m_j W(\mathbf{x}_i - \mathbf{x}_j, h).$$

Substituting Eq. (1) into the Navier-Stokes equations and symmetrization yield the following particle body forces

$$\mathbf{f}_i^{\text{pressure}} = -\sum_j m_j \frac{p_i + p_j}{2\rho_j} \nabla W(\mathbf{r}_{ij}, h)$$

$$\mathbf{f}_i^{\text{viscosity}} = \mu \sum_j m_j \frac{\mathbf{v}_j - \mathbf{v}_i}{\rho_j} \nabla^2 W(\mathbf{r}_{ij}, h),$$

where vi are the particles' velocities, μ the viscosity coefficient of the fluid and ri j the distance vector xi −xj . We use the kernels proposed in [MCG03] to compute these body forces. The pressures pi are computed via the constitutive equation

$$p_i = k(\rho_i - \rho_0),$$

where k is the stiffness of the fluid and ρ0 its rest density. Finally, for the acceleration ai of a particle i we have

$$\mathbf{a}_i = 1/\rho_i (\mathbf{f}_i^{\text{pressure}} + \mathbf{f}_i^{\text{viscosity}} + \mathbf{f}_i^{\text{external}}),$$

where f external i are external body forces such as gravity or forces due to surface tension.

2.1 Multiple Fluids

The SPH fluid model given by Eq. (5) and Eq. (6) is already designed to handle individual particle masses mi . However, because each particle now carries its own viscosity coefficient, we modify the computation of the viscosity force in Eq. (6) as follows

$$\mathbf{f}_i^{\text{viscosity}} = \sum_j \frac{\mu_i + \mu_j}{2} m_j \frac{\mathbf{v}_j - \mathbf{v}_i}{\rho_j} \nabla^2 W(\mathbf{r}_{ij}, h),$$

meaning that the individual viscosity coefficients are averaged between interacting particles.

3. Simulation Step

Our fluid is represented by particles evolving through space and time. A typical particle simulation goes through the following steps

Algorithm 1: Simulation step.

```
foreach particle i
    // apply gravity
    v_i ← v_i + Δt g
// modify velocities with pairwise viscosity impulses
applyViscosity                          // (Section 5.3)
foreach particle i
    // save previous position
    x_i^prev ← x_i
    // advance to predicted position
    x_i ← x_i + Δt v_i
// add and remove springs, change rest lengths
adjustSprings                           // (Section 5.2)
// modify positions according to springs,
// double density relaxation, and collisions
applySpringDisplacements                // (Section 5.1)
doubleDensityRelaxation                 // (Section 4)
resolveCollisions                       // (Section 6)
foreach particle i
    // use previous position to compute next velocity
    v_i ← (x_i − x_i^prev)/Δt
```

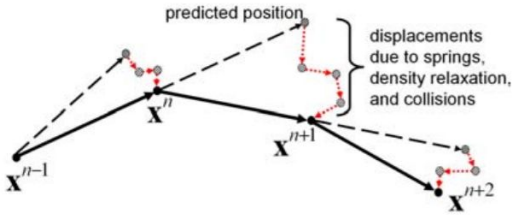

Figure : Prediction-relaxation scheme

4. Interaction Methods

4.1 Buoyancy

Buoyancy is a phenomenon which simply emerges from the fact that we use individual rest densities ρ0 per particle. The parameter ρ0 in the constitutive law defined in Eq. (7) has the effect that the pressure term not only pushes close particles apart but also contracts particles in order to reach a non-zero rest density. When two fluids with different rest densities are mixed, a density gradient and, thus, a pressure gradient will emerge at their common interface. This pressure gradient will cause the less dense fluid to rise inside the denser fluid.

Immiscible Liquids: The solubility of two liquids depends mainly on the interactions of the different types of molecules. Water and oil are immiscible because the water molecules are polar while the oil molecules are not.

4.2 Diffusion

ısion effects play an important role in connection with fluids. The diffusion equation describes how heat gets distributed in a fluid or how cream dissolves in a cup of coffee. The evolution of an attribute A(x,t) due to diffusion is given by

$$\frac{\partial A_i}{\partial t} = c\sum_j m_j \frac{A_j - A_i}{\rho_j}\nabla^2 W(\mathbf{r}_{ij}, h).$$

4.3 Pressure

ery simple solution for our purposes of speed and stability

$$\mathbf{f}_i^{\text{pressure}} = -\sum_j m_j \frac{p_i + p_j}{2\rho_j}\nabla W(\mathbf{r}_i - \mathbf{r}_j, h)..$$

The so computed pressure force is symmetric because it uses the arithmetic mean of the pressures of interacting particles. Since pressure forces depend on the gradient of the pressure field, the offset mathematically has not effect on pressure forces. However, the offset does influence the gradient of a field smoothed by SPH and makes the simulation numerically more stable.

4.4 Viscosity

Viscosity has the effect of smoothing the velocity field. It is applied as radial pairwise impulses between neighboring particles. These impulses modify particle velocities at the beginning of the timestep, before moving them to their predicted positions.

orithm 5: Viscosity impulses.

```
foreach neighbor pair ij, (i < j)
    q ← r_ij/h
    if q < 1
        // inward radial velocity
        u ← (v_i − v_j) · r̂_ij
        if u > 0
            // linear and quadratic impulses
            I ← Δt(1 − q)(σu + βu^2) r̂_ij
            v_i ← v_i − I/2
            v_j ← v_j + I/2
```

4.5 Surface Tension

Surface tension is physically caused by attraction between molecules. Inside the fluid, this attraction cancels out, but for molecules near the surface, asymmetry in neighbor distribution results in a non-zero net force towards the fluid.

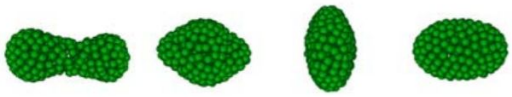

Figure : *Oscillating drop*

4.6 Elasticity

To simulate elastic behavior, we add springs between pairs of neighboring particles. Springs create displacements on the two attached particles. The displacement magnitude is proportional to L−r, where r is the distance between the particles and L is the spring rest length.

Algorithm 3: Spring displacements.

1. foreach spring ij
2. $\quad \mathbf{D} \leftarrow \Delta t^2 k^{\text{spring}}(1 - L_{ij}/h)(L_{ij} - r_{ij})\hat{\mathbf{r}}_{ij}$
3. $\quad \mathbf{x}_i \leftarrow \mathbf{x}_i - \mathbf{D}/2$
4. $\quad \mathbf{x}_j \leftarrow \mathbf{x}_j + \mathbf{D}/2$

4.7 Plasticity

A perfectly elastic substance always remembers its fixed rest shape, and fights external forces to recover it. On the other hand, a perfectly plastic substance considers its current shape as its rest shape. In general, plasticity can be thought of as the rate at which a substance forgets its past.

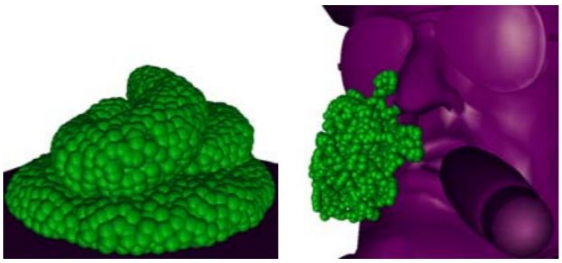

Figure : *Various plastic behaviours*

5 Double Density Relaxation

The impulses exchanged between particles depend on two different measures of their neighbor density.

5.1 Density and pressure

In an SPH framework, the global goal of minimizing compressibility translates into a local constraint to maintain constant density. The density at particle i is approximated by summing weighted contributions from each neighbor j. We choose the density at particle i to be:

$$\rho_i = \sum_{j \in N(i)} (1 - r_{ij}/h)^2$$

where ri j = |ri j |, ri j = xj − xi , and N(i) denotes the set of neighboring particles that are closer than the interaction radius h.
We define a pseudo-pressure Pi proportional to the difference between the current density ρi and a rest density ρ0

$$P_i = k(\rho_i - \rho_0)$$

5.2 Incompressibility Relaxation

The density relaxation displacement between two particles is proportional to the pseudo-pressure, weighted by the linear kernel function:

$$\mathbf{D}_{ij} = \Delta t^2 P_i (1 - r_{ij}/h)\hat{\mathbf{r}}_{ij}$$

5.3 Near Density and Near Pressure

This new force does not simply depend on the distance between particles, but depends on a new density – the near-density – computed with a different kernel function. We define near-density as

$$\rho_i^{\text{near}} = \sum_{j \in N(i)} (1 - r_{ij}/h)^3$$

In order to make the corresponding force exclusively repulsive, we define near-pressure as:

$$P_i^{\text{near}} = k^{\text{near}} \rho_i^{\text{near}}$$

Algorithm 2: Double density relaxation.

```
foreach particle i
    ρ ← 0
    ρ^near ← 0
    // compute density and near-density
    foreach particle j ∈ neighbors( i )
        q ← r_ij / h
        if q < 1
            ρ ← ρ + (1 − q)^2
            ρ^near ← ρ^near + (1 − q)^3
    // compute pressure and near-pressure
    P ← k(ρ − ρ_0)
    P^near ← k^near ρ^near
    dx ← 0
    foreach particle j ∈ neighbors( i )
        q ← r_ij / h
        if q < 1
            // apply displacements
            D ← Δt^2 (P(1 − q) + P^near (1 − q)^2) r̂_ij
            x_j ← x_j + D/2
            dx ← dx − D/2
    x_i ← x_i + dx
```

6 Implementations

The time integration of the SPH equations is performed by an explicit numerical integration scheme (the predictor-corrector scheme is applied within this study). When rigid boundaries are involved in the simulation, either of two implemented SPH rigid boundary representations can be applied, the ghost particles or the boundary particles.

6.1. **Ghost particles** The ghost particles are created by reflecting the fluid particles in the vicinity of the boundary across its interface at every timestep, [6]. The width of the reflected region is equal to the width of the smoothing function support. In order to prevent the boundary penetration by fluid and to model the no-slip boundary conditions, the ghost particle's velocity vector is opposite to the fluid particle's one.

6.2. **Boundary particles**

The boundary particles consist of two sets of particles, the interface and the wall particles, [7]. The position of the boundary particles does not change in time. The interface particles are placed along the boundary surface, while the wall particles fill the boundary region to the width of the smoothing function support. The density and pressure of the wall particles is evolved through the governing equations, but the density and pressure of the interface particles is not updated. When a static rigid boundary is modelled, the velocity of all the boundary particles is equal zero.

6.3 Surface tracking and visualization

The color field cS and its gradient field n = ∇cS defined in section 3.3 can be used to identify surface particles and to compute surface normals. We identify a particle i as a surface particle if

$$|\mathbf{n}(\mathbf{r}_i)| > l,$$

where l is a threshold parameter. The direction of the surface normal at the location of particle i is given by

$$-\mathbf{n}(\mathbf{r}_i).$$

6.4.1 Point Splatting

We now have a set of points with normals but without connectivity information. This is exactly the type of information needed for point splatting techniques.

6.4.2 Marching Cubes

Another way to visualize the free surface is by rendering an iso surface of the color field cS. We use the marching cubes algorithm to triangulate the iso surface. In a grid fixed in space the cells that contain the surface are first identified. We start searches from all the cells that contain surface particles and from there recursively traverse the grid along the surface. With the use of a hash table we make sure that the cells are not visited more than once. For each cell identified to contain the surface, the triangles are generated via a fast table lookup.

For fast previsualization, particles are simply rendered as polygonized spheres. High quality rendering uses a surface mesh extracted with the marching cube algorithm. The marching cube extracts an iso-surface of a scalar function defined by the particles. We use the function:

$$\phi(\mathbf{x}) = \left(\sum_j (1 - r/h)^2 \right)^{\frac{1}{2}}$$

where r = |x−xj |. The square root ensures that the function behaves like a distance function, i.e., has an almost constant slope around the extracted iso-surface. This is important for moving iso-surfaces because the location of the surface in a cube is linearly interpolated.

6.4.3 Neighbor Search

Finding all particles that are within the interaction radius of a particle is a frequent task in any particle-based method. Since the interaction radius h is constant and identical for each particle, it makes sense to use a simple regular spatial hashing grid with cube size h.

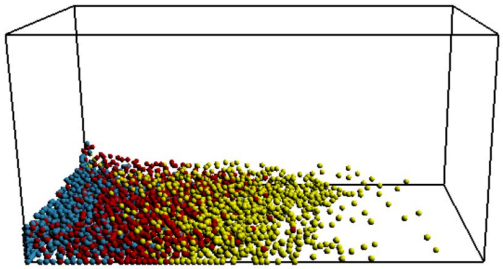

Figure: *Simulation of Fluid Particles*

6. Work

1. Rendered the fluid particles using OpenGL.
2. Implemented a 2D fluid simulation initially.
3. Then, implemented the fluid particle simulation into a 3D simulation as well.
4. Added camera features.
5. Worked on multiple fluids to observe the densities and other physical properties.
6. Future work would include usage of Matching Cube algorithm to create the surface of the fluid .

7. Conclusion

Smoothed Particle Hydrodynamics is a fully meshfree particle based method, where the particles carry the material properties of the medium it is simulating (fluid, gases). As SPH is a purely particle based method, to simulate actual fluid the particles need to be rendered as such. Surface reconstruction is a technique for achieving visualization of fluids and its complexity scales with the number of particles used in the simulation. We have presented a variety of techniques to enhance particle based fluid simulations by considering fluid-fluid interaction effects. The physical model is based on Smoothed Particle Hydrodynamics and uses special purpose kernels to increase stability and speed. We have presented techniques to track and render the free surface of fluids. Also, We have presented a particle-based method to interactively simulate the complex behavior of viscoelastic fluids. A simple and flexible method to simulate viscoelasticity with varying rest length springs. A new scheme for the robust simulation of surface tension in particle-based systems, without requiring computation of curvature over the liquid surface. We have proposed a novel SPH solver which allows us to enforce incompressibility in an efficient manner. Introduced the notion of particle slip in order to extend this SPH solver to simulate diffuse phenomena such as mixtures of spray and air.

References

[1] Simon Clavet, Philippe Beaudoin, and Pierre Poulin. 2005. Particle-based viscoelastic fluid simulation. In Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation (SCA '05). ACM, New York, NY, USA, 219-228. DOI: https://doi.org/10.1145/1073368.1073400

[2] Matthias Müller, David Charypar, and Markus Gross. 2003. Particle-based fluid simulation for interactive applications. In Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation (SCA '03). Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 154-159.

[3] Matthias Müller, Barbara Solenthaler, Richard Keiser, and Markus Gross. 2005. Particle-based fluid-fluid interaction. In Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation (SCA '05). ACM, New

York, NY, USA, 237-244. DOI: https://doi.org/10.1145/1073368.1073402