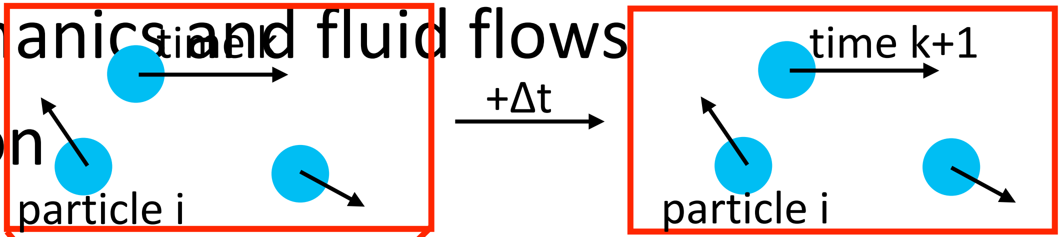
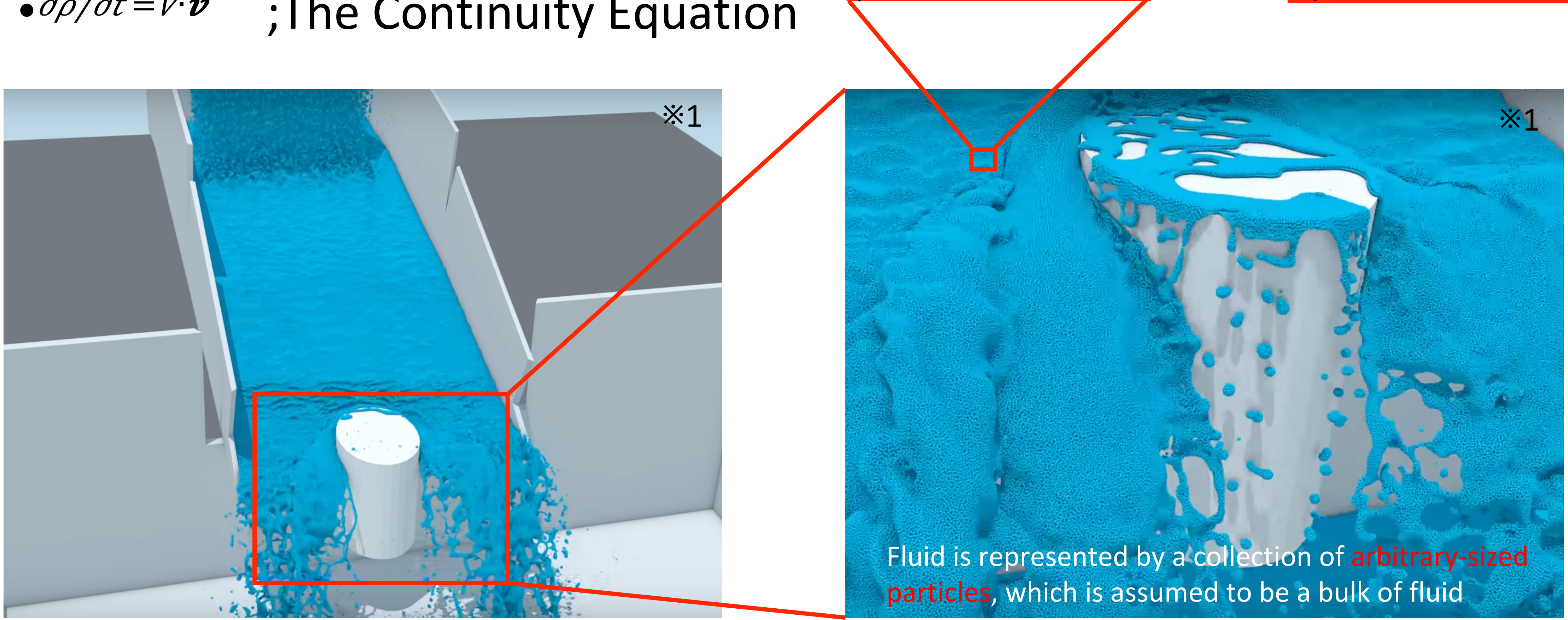
Smooth Particle Hydrodynamics

Takahiro Shinohara, Duo Hao, Timothy Witham and Dorivaldo Santos

What is SPH?

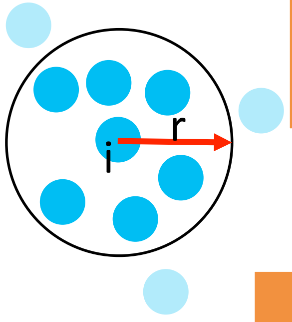
- SPH (Smoothed Particle Hydrodynamics) is a mesh free computational method used for simulating the dynamics of continuum media, such as solid mechanics and fluid flows
- $\rho \boldsymbol{dv}/dt = -\nabla P + \boldsymbol{f}$;The Euler Equation
- $\partial \rho / \partial t = \nabla \cdot \boldsymbol{v}$;The Continuity Equation





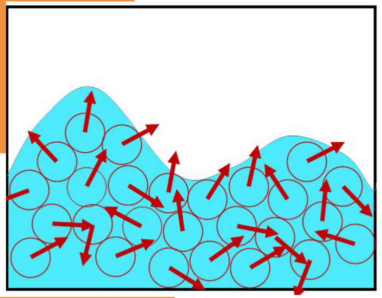
Discretization Methods in Numerical Simulation

Methods



Discretization Method

Mesh free methods



※2

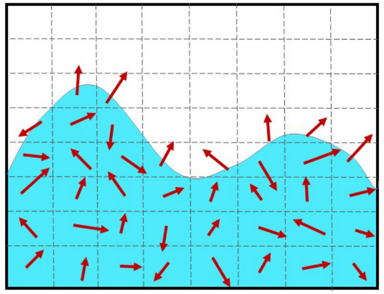
Discrete element

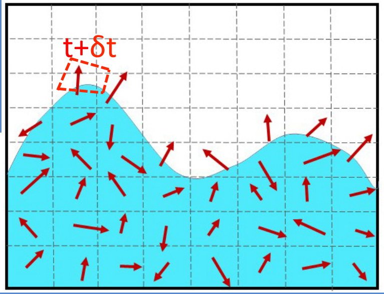
Inherently lagrangian

Suitable for problems involving

- Very large displacements
- Deformable boundaries
- Multiphase problems

Grid-Based methods





Eulerian Grid

Grid fixed in the space

Lagrangian Grid

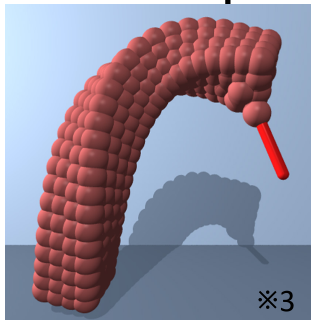
Grid attached on moving material

Not suitable for problems involving

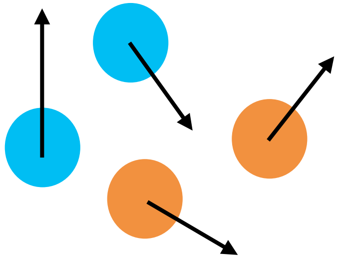
- Large displacements
- Deformable boundaries
- Multiphase Problems

Mesh Free Methods – Pros and Cons

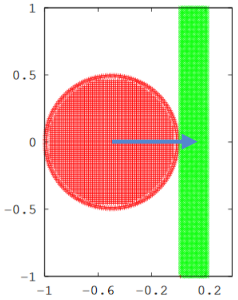
- Large displacements – since the connectivity between particles are generated as part of the computation and can change with time
- Deformable boundaries – inherently achieved regardless of the complexity of movement of the particles
- Multiphase problems – much less interface problems due to the fact that each material can be described by its own set of particles

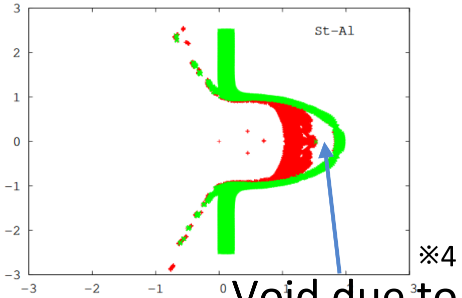


Large Deformation
& Deformable boundary

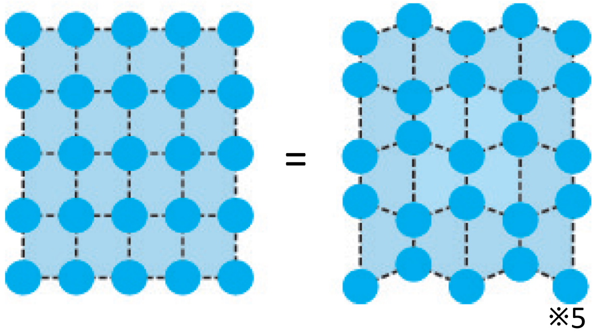

Multi Phase Flow

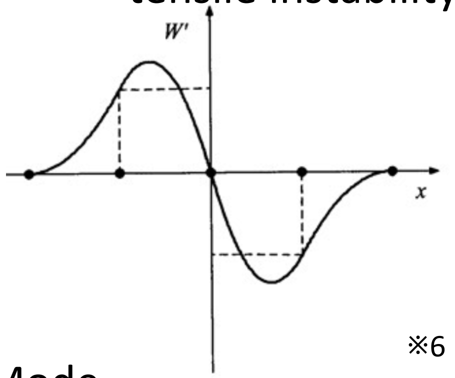
- Computational instability such as tensile instability - particle clustering at a region with tensile stress state - and zero energy mode - deformation energy become zero even when there is deformation


Tensile instability



Void due to tensile instability

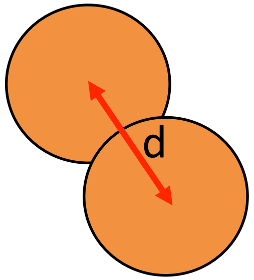




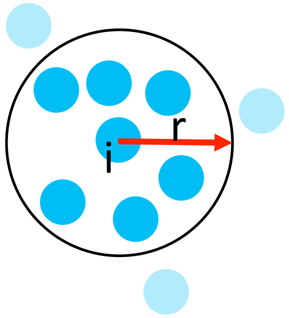
Zero Energy Mode

What's the difference between DEM and SPH?

SPH is a method where continuum is assumed to be a collection of imaginary particles, as opposed to DEM where a collection of particles is modeled as a collection of particles.



DEM



SPH

History - SPH was developed in the late 1970s for treating astrophysics problem

(Lucy, 1977; Gingold & Monaghan, 1977)

SPH was then implemented in many problems in fluid mechanics and solid mechanics, such as for modelling of viscous flow, and for modelling of strength of materials.

Takeda et al. (1994) Libersky & Petschek (1990)

Petroleum Engineering

- Problem on incompressible flow in single phase (Morris et al., 1997)
- Problem on multiphase flow and reactive transport in porous media (Tartakovsky, 2016)

Geosciences

- Ice sheet modelling (Pan et al., 2013)
- Landslide modelling (Huang et al., 2014)

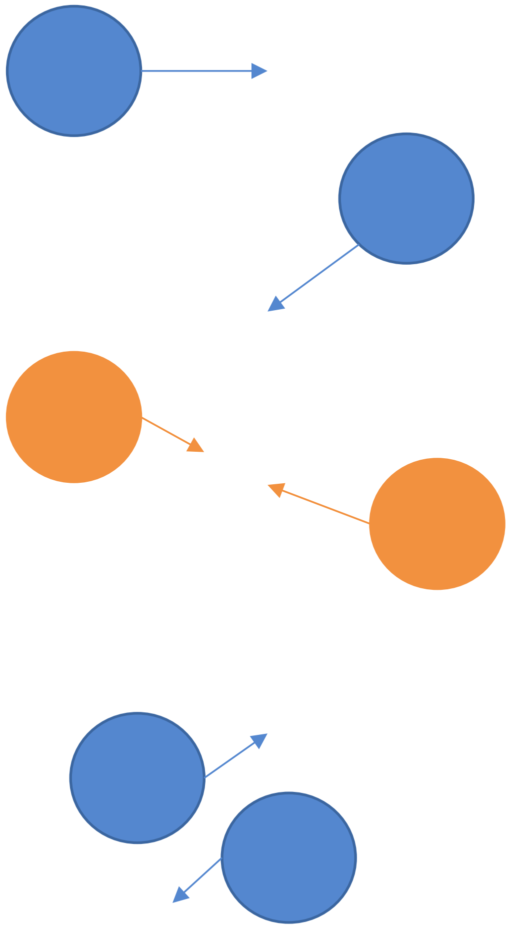
SPH has also found a number of applications in other fields such as computer graphics and civil engineering.

General Principles

Initial Position and Velocity

Governing Equations

New Position and Velocity



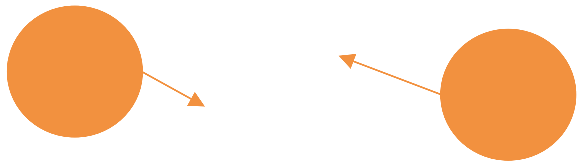
$\boldsymbol{r}, \boldsymbol{v}$

$A(\boldsymbol{r}, t)$

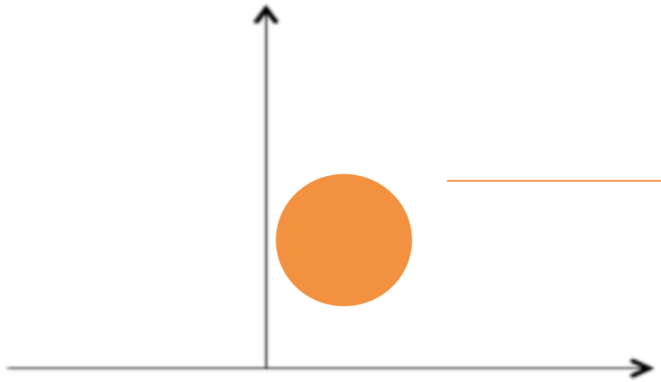
$\nabla^k A(\boldsymbol{r}, t)$

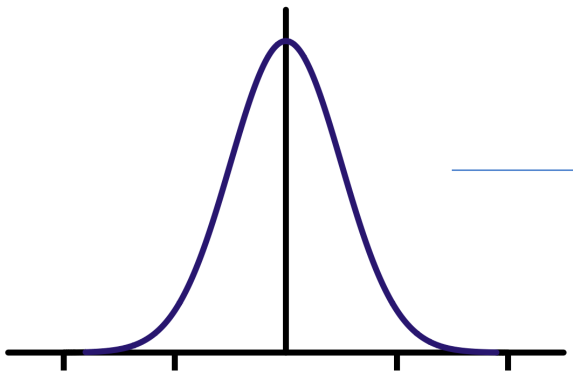
$\boldsymbol{r}_1, \boldsymbol{v}_1$

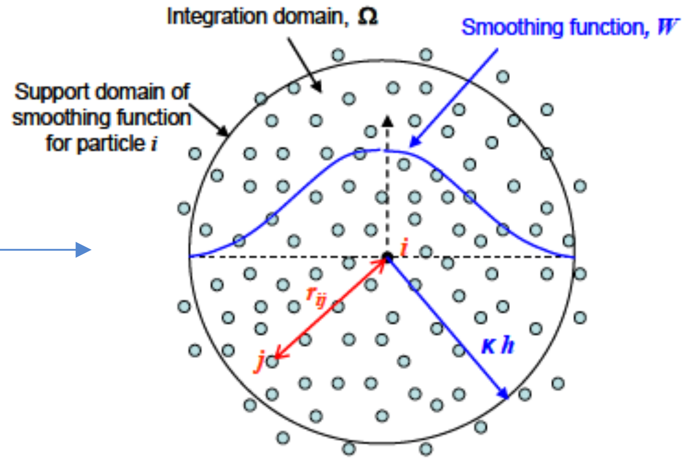
General Principles (Interpolation)



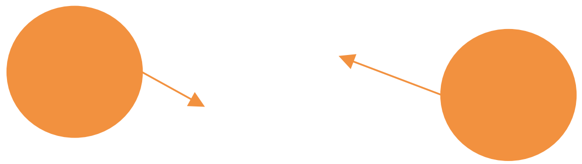
$A(\boldsymbol{r},t)$ → Kernel function → $\int_{\Omega} A(\boldsymbol{r}',t)\, w_h(\boldsymbol{r}-\boldsymbol{r}')\, d^n \boldsymbol{r}'$







General Principles (derivatives)

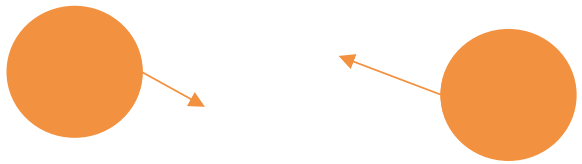


$\nabla A(\boldsymbol{r},t)$ → Kernel function → $\int_{\Omega} A(\boldsymbol{r}',t)\nabla w_h(\boldsymbol{r}-\boldsymbol{r}')\,d^n\boldsymbol{r}'$

$\nabla\cdot\boldsymbol{A}(\boldsymbol{r},t)$ → Kernel function → $\int_{\Omega} \boldsymbol{A}(\boldsymbol{r}',t)\nabla\cdot w_h(\boldsymbol{r}-\boldsymbol{r}')\,d^n\boldsymbol{r}'$

$\nabla^2 A(\boldsymbol{r},t)$ → Kernel function → $\int_{\Omega} A(\boldsymbol{r}',t)\nabla^2 w_h(\boldsymbol{r}-\boldsymbol{r}')\,d^n\boldsymbol{r}'$

General Principles (SPH Discretization)



$$V_b A_b w_h(r_{ab})$$

$$\nabla A(r_a,t) \approx \sum_b V_b A_b \nabla_a w_h(r_{ab}) = \sum_b V_b \frac{\rho_b^{2k} A_a + \rho_a^{2k} A_b}{(\rho_a \rho_b)^k} \cdot w_{ab}' \mathbf{e}_{ab}$$

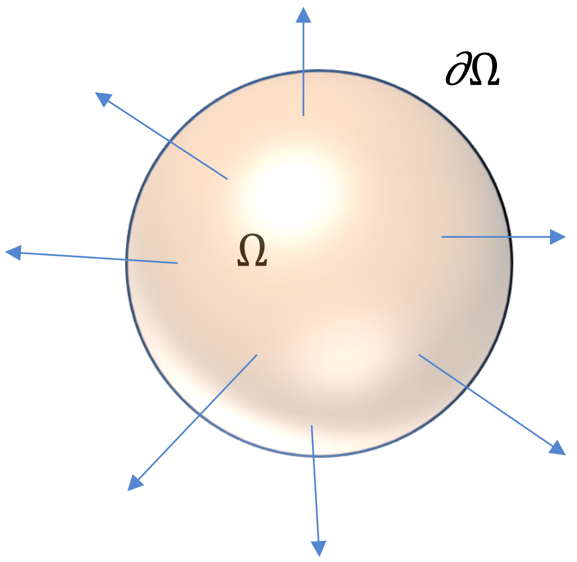
$$\cdot \mathbf{A}(r_a,t) \approx \sum_b V_b \mathbf{A}_b \cdot \nabla_a w_h(r_{ab}) = \sum_b V_b \frac{\rho_b^{2k} \mathbf{A}_a + \rho_a^{2k} \mathbf{A}_b}{(\rho_a \rho_b)^k} \cdot w_{ab}' \mathbf{e}_{ab}$$

$$\nabla^2 A(r_a,t) \approx \sum_b V_b A_b \nabla_a^2 w_h(r_{ab})$$

Governing equations (Euler and Continuity)

$$\boldsymbol{F}_{tot} = \boldsymbol{F}_{ext} + \boldsymbol{F}_{int}$$

$$\frac{d}{dt}\int_{\Omega} d\boldsymbol{p} = \int_{\partial\Omega} \boldsymbol{\sigma}\cdot n \; d\Gamma + \int_{\Omega} \rho\boldsymbol{g} \; d\Omega$$



$$\frac{d}{dt}\int_{\Omega} \boldsymbol{v}\, dm = \int_{\Omega} \nabla\cdot\boldsymbol{\sigma} \, d\Omega + \int_{\Omega} \rho\boldsymbol{g} \; d\Omega$$

$$\int_{\Omega} \frac{D\boldsymbol{v}}{Dt}\, \rho\, d\Omega = \int_{\Omega} (\nabla\cdot\boldsymbol{\sigma} + \rho\boldsymbol{g}) \; d\Omega$$

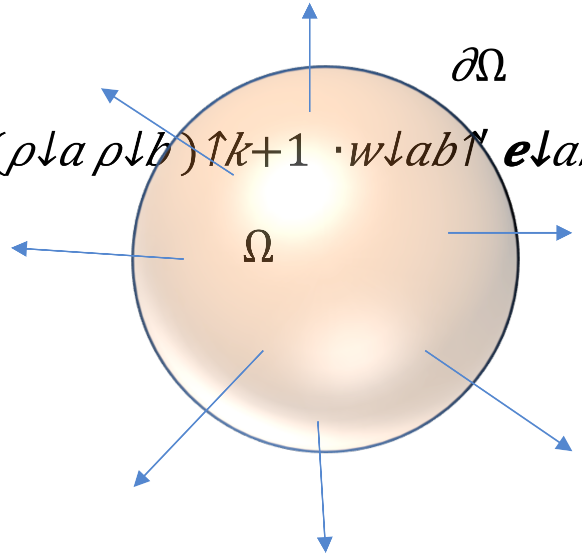
$$\frac{D\boldsymbol{v}}{Dt} = \frac{1}{\rho}\nabla\cdot\boldsymbol{\sigma} + \boldsymbol{g}$$

$$\frac{\partial\rho}{\partial t} = \nabla\cdot\boldsymbol{v}$$

Governing Equations: SPH form

Euler Equation

$$\frac{D\boldsymbol{v}_a}{Dt} = -\sum_b m_b \frac{\rho_b^{2k} p_a + \rho_a^{2k} p_b}{(\rho_a \rho_b)^{k+1}} \cdot w_{ab}' \boldsymbol{e}_{ab} + \boldsymbol{g}$$

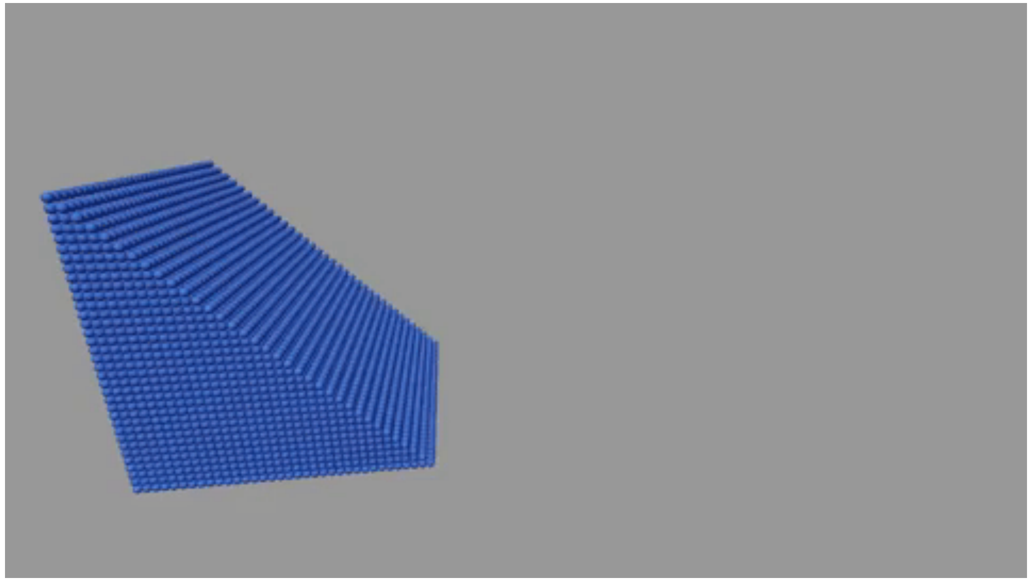


Tait Equation

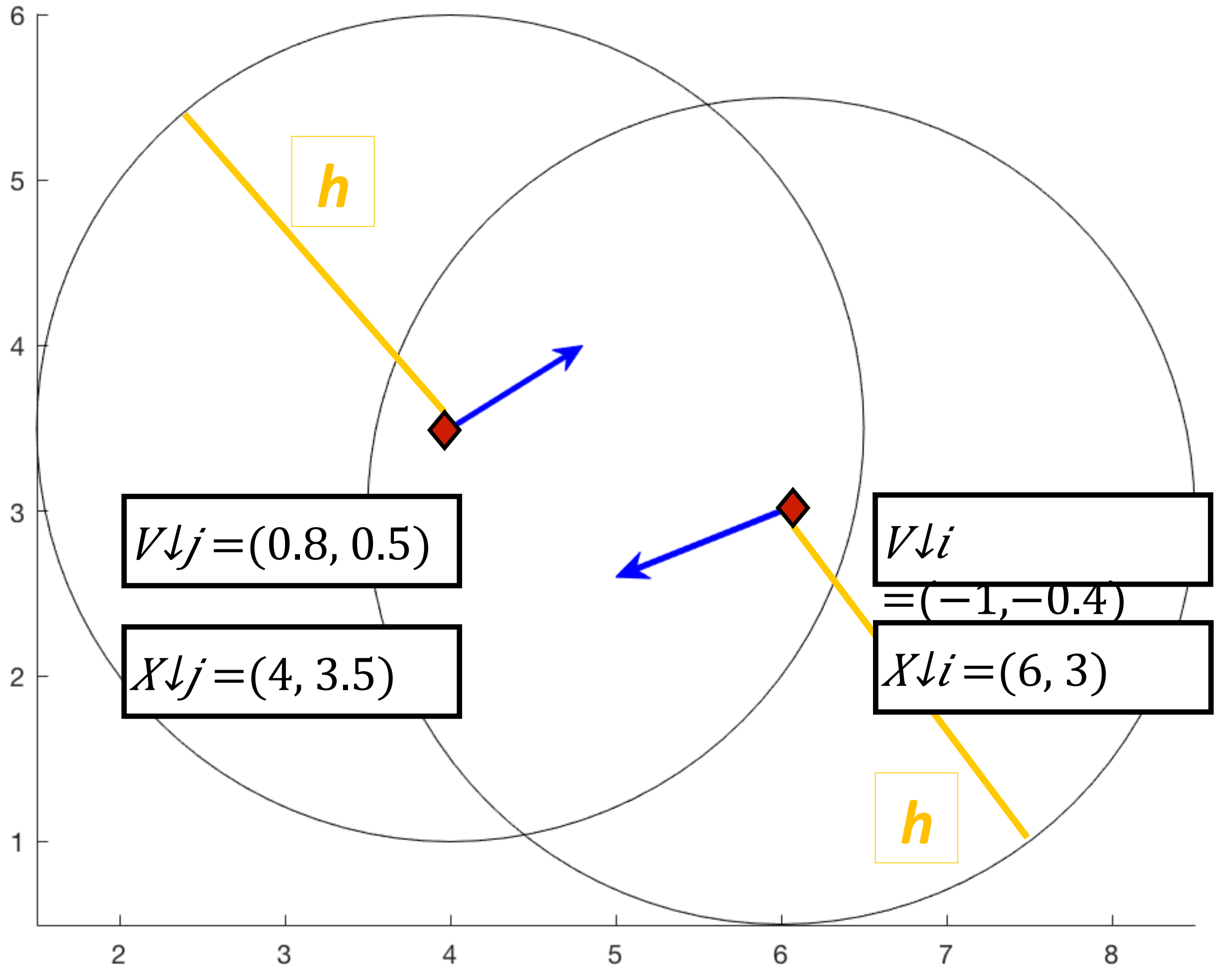
$$p_a = \frac{\rho_o c_o^2}{\gamma}\left[\left(\frac{\rho_a}{\rho_o}\right)^{\gamma} - 1\right]$$

Continuity Equation

$$\frac{\partial \rho}{\partial t} = \sum_b V_b \boldsymbol{v}_b \cdot w_{ab}' \boldsymbol{e}_{ab}$$



Hand Calculation: Introduction



- Given two particles with an initial position and velocity, what will their new position and velocity be after a time step?
- The two particles influence each other via a smoothing length of h and a kernel function.

Initial Parameters

	Mass	Density	Pressure	Velocity	Location	h	Δt
Particle i	1	1	1	$(-1, -0.4)$	$(6, 3)$	2	1
Particle j	1	1	1	$(0.8, 0.5)$	$(4, 3.5)$	2	1

Governing Equation/Start of Solution

$$\frac{dV_i}{dt} = -m_j * \left(\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2} + \Pi_{i,j}\right) * \nabla W_{i,j} + g$$

Assume no influence of gravity and viscosity is insignificant:

$$\frac{dV_i}{dt} = -m_j * \left(\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2}\right) * \nabla W_{i,j}$$

Calculation of $\nabla W_{i,j}$

Step 1.) Solve for the Kernel Function:

B-Spline, Order 3

$$W(r,h)=\frac{1}{\pi * h^3} * \begin{cases} 1+\frac{3}{4} * q^3 + \frac{3}{2} * q^2 & \text{if } 0 \le q \le 1 \\ \frac{1}{4} * (2-q)^3 & \text{if } 1 \le q \le 2 \\ 0 & \text{otherwise} \end{cases}$$

$$q = \frac{r_{ij}}{h}$$

In this case, $r_{ij} = |X_i - X_j| = 2.0616$

Therefore, q = $\sqrt{2}/2 = 0.8246$, and this means we will use the topmost portion of the equation.

Calculation of $\nabla W_{i,j}$

In our situation,

$$W(r,h)=\frac{1}{\pi * h^3} * \begin{cases} 1+\frac{3}{4} * q^3 + \frac{3}{2} * q^2 & \text{if } 0\leq q\leq 1 \\ \frac{1}{4} * (2-q)^3 & \text{if } 1\leq q\leq 2 \\ 0 & \text{otherwise} \end{cases}$$

$$q = r_{ij} / h$$

Plugging in r_{ij} / h for q yields:

So, $\nabla W_{i,j} = \frac{1}{\pi * h^3} * \left(1+ \frac{3}{4} * q^3 + \frac{3}{2} * q^2\right) = \frac{1}{\pi * h^3} + \frac{3r^3}{4\pi * h^6} + \frac{3r^2}{2\pi * h^5}$

Calculation of $\nabla W_{i,j}$

For particle "i"

Using the numerical method,

$$\partial W_{ij}/\partial x = \left(1/\pi * h^{3} + 3*((1-0.001)^{2}+1^{2})^{3/2}/4\pi * h^{6} + 3*((1-0.001)^{2}+1^{2})^{2/2}/2\pi * h^{5}\right) - \left(1/\pi * h^{3} + 3(\sqrt{2.0616})^{3}/4\pi * h^{6} + 3*2.0616/2\pi * h^{5}\right)/0.001$$

$\partial W_{ij}/\partial x$ = -0.0497

$$\partial W_{ij}/\partial y = \left(1/\pi * h^{3} + 3*((1+0.001)^{2}+1^{2})^{3/2}/4\pi * h^{6} + 3*((1+0.001)^{2}+1^{2})^{2/2}/2\pi * h^{5}\right) - \left(1/\pi * h^{3} + 3(\sqrt{2})^{3}/4\pi * h^{6} + 3*2/2\pi * h^{5}\right)/0.001$$

$\partial W_{ij}/\partial y$ = 0.0497

Calculation of Acceleration, Velocity, and Position for Particle "i"

Step 2.) Calculate the new positions and velocities:

$$a_i = dV_i/dt = -m_i * \left(P_i/\rho_i^2 + P_j/\rho_j^2 + \Pi_{i,j}\right) * \nabla W_{i,j}$$

$$= -1 * (1/1^2 + 1/1^2 + 0) * (-0.0497, 0.0497) = (0.0995, -0.0995)$$

$$V_{i,new} = V_i + (a_i * \Delta t) = (-0.9005, -0.4995)$$

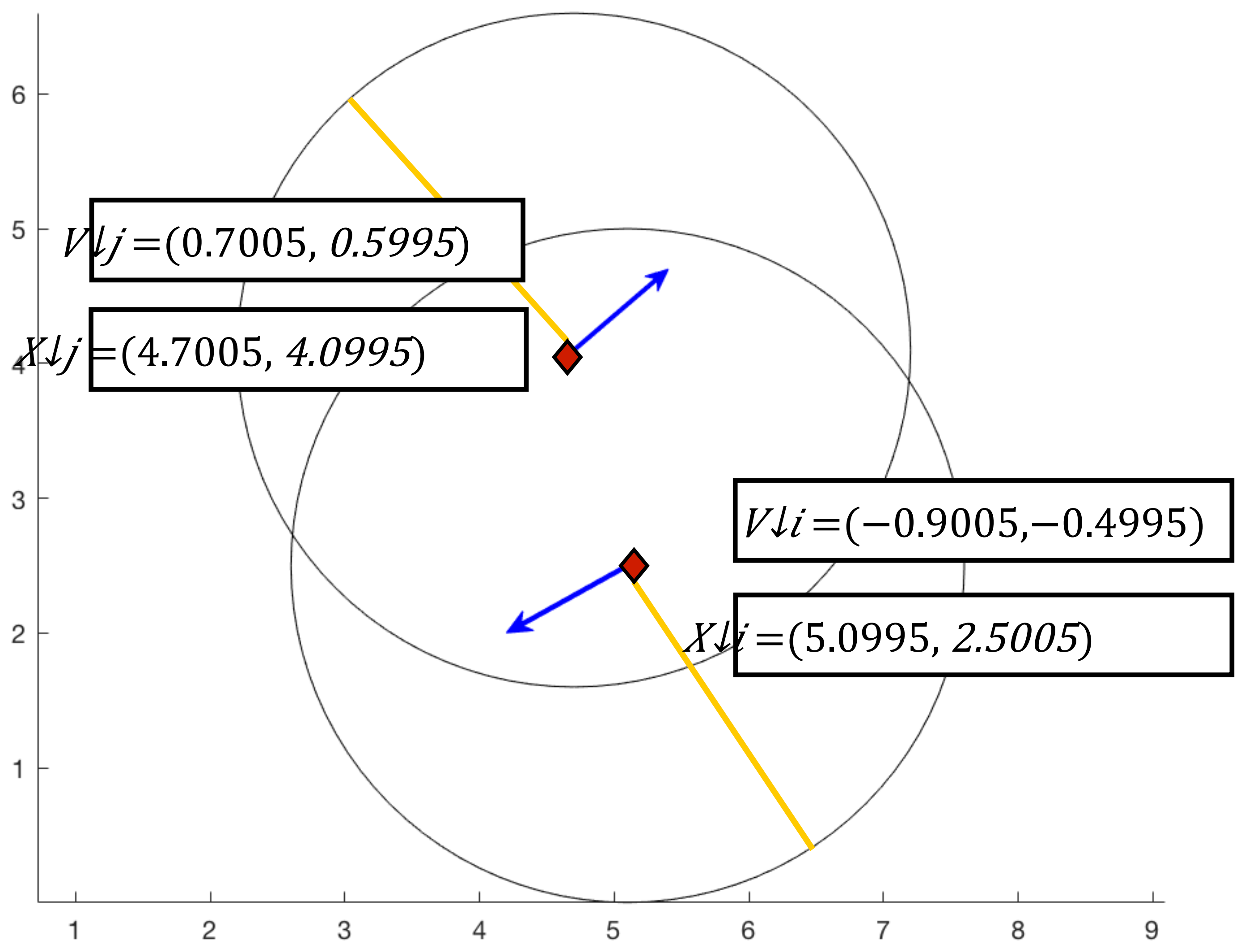
$$X_{i,new} = X_i + (V_{i,new} * \Delta t) = (5.0995, 2.5005)$$

Using the same method, the new velocity and position for Particle "j" can be found too.

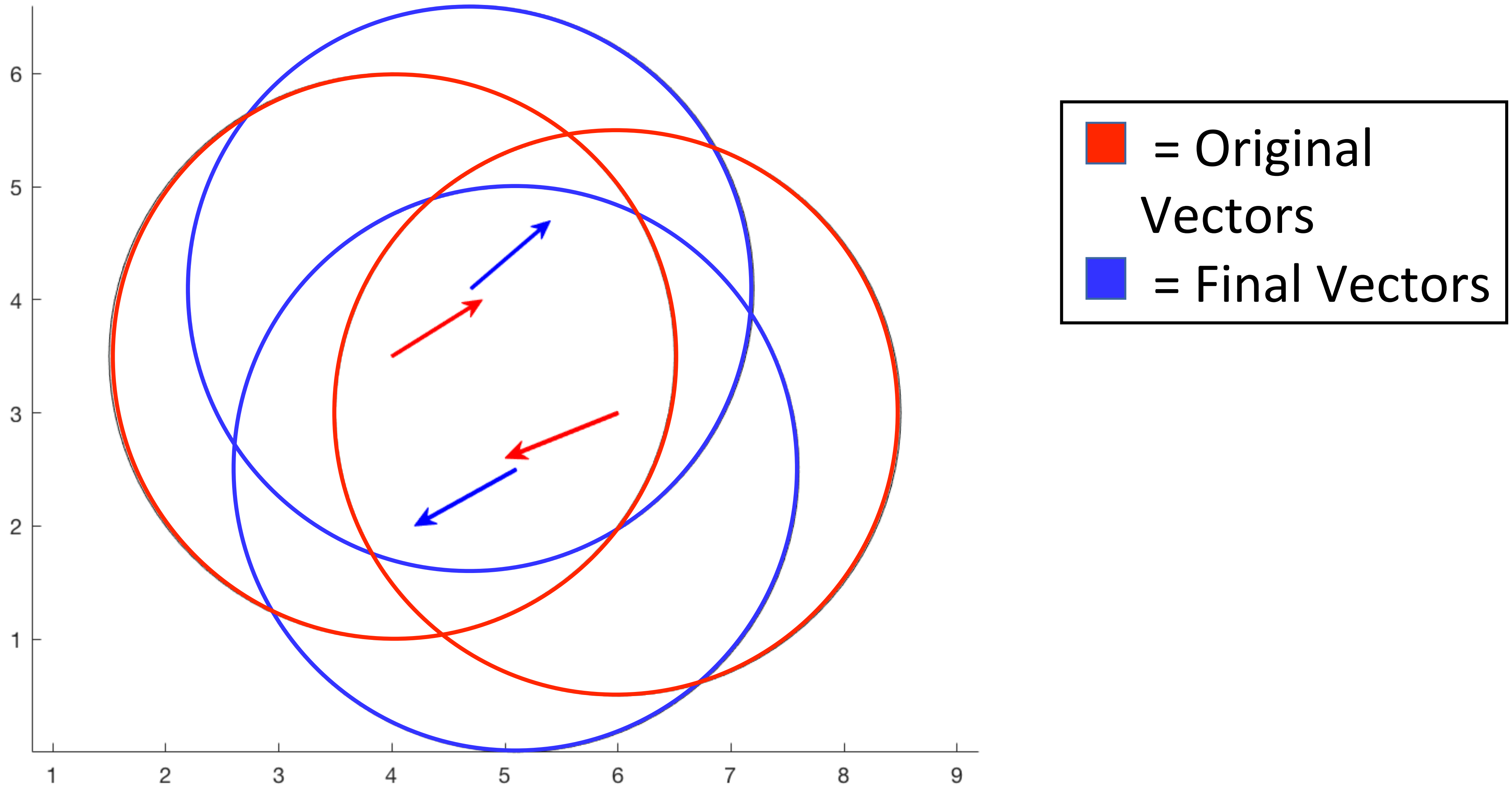
Final Velocities and Positions:

	Mass	Density		Velocity	Location	h	Δt
Particle i	1	1	1	$(-0.9005, -0.4995)$	$(5.0995,$ 2.5005)	2	1
Particle j	1	1	1	$(0.7005,$ 0.5995)	$(4.7005,$ 4.0995)	2	1

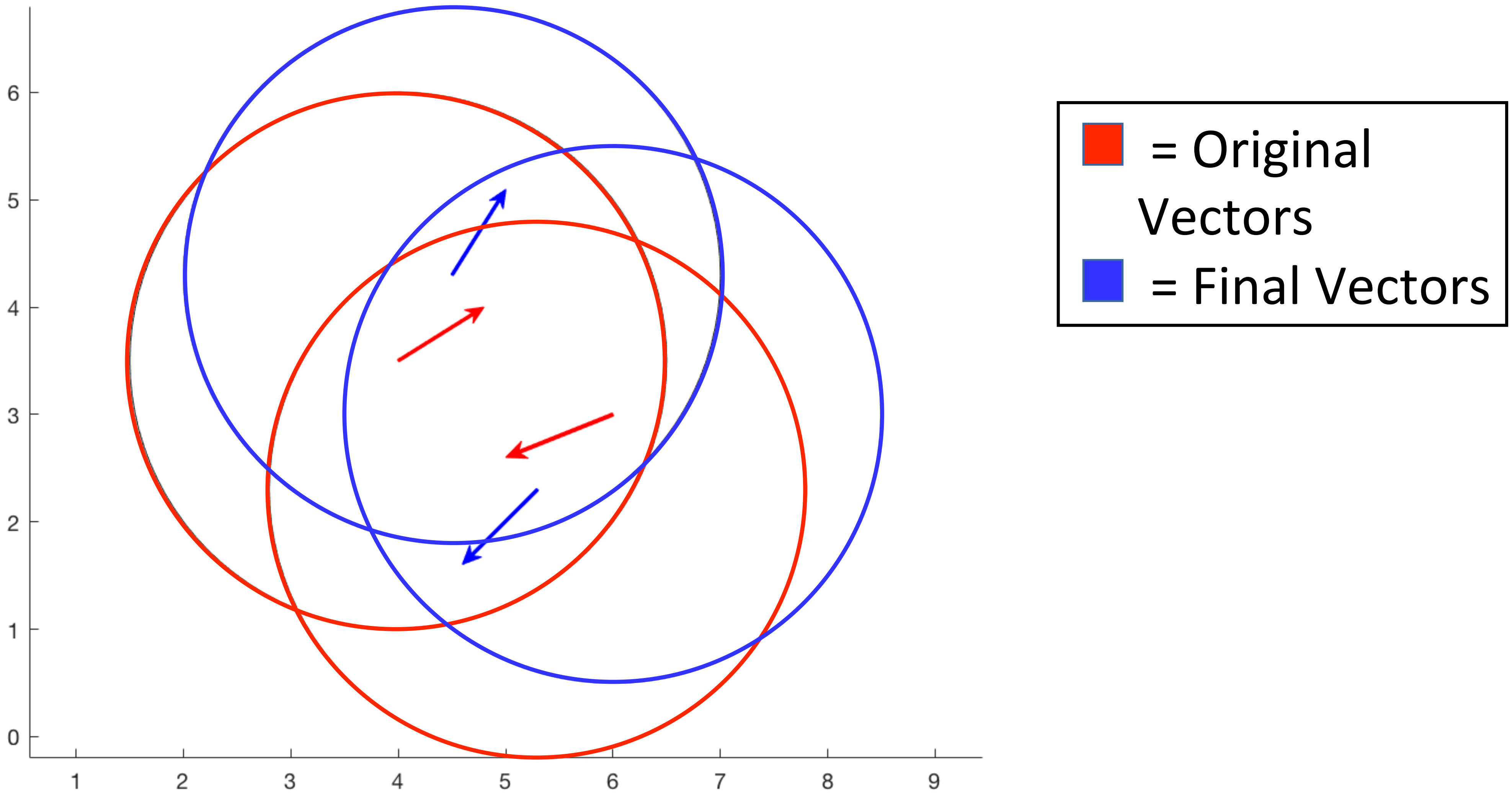
Final Position



Superimposed



When Pressure is a value over 1:



MATLAB Code for Hand Calculation:

```
%Define line vectors:
xj1=4;
yj1=3.5;
r1=2.5;
%
xi1=6;
yi1=3;
r2=2.5;
%
vj1x=0.8;
vj1y=0.5;
%
vi1x=-1;
vi1y=-0.4;
%
mass=1;
p=1;
dens=1;
%
% %final stuff:
% x12=
% y12=
% r1=
%
% x22=
% y22=
% r22=
%
% v12x=
% v12y=
%
% v22x=
% v22y=

%First Position:
% circle2(4,3,2); hold on;
% circle2(1,5,2); hold on;
% drawArrow([4;3],[7;5],'b'); hold
on;
% drawArrow([1;5],[2;9],'b')
%Above, the end of the arrows is the
vector x, y components +
%the center x value (ie: 2 + 0.2)
figure
circle2(xj1,yj1,r1); hold on;
circle2(xi1,yi1,r2); hold on;
drawArrow([xj1;yj1],
[xj1+vj1x;yj1+vj1y],'b'); hold on;
drawArrow([xi1;yi1],
[xi1+vi1x;yi1+vi1y],'b')
%% Calc the Kernel Function:
q=((((xi1-xj1)^2)+((yi1-yj1)^2))^0.5)/
r1;
W=0;
if 0<=q<=1
   W=(1/(3.14*(r1^3)))*(1+(3*(q^3)/
4)+(3*(q^2)/2))
elseif 1<=q<=2
   W=(1/(3.14*(r1^3)))*(0.25*((2-
q)^3))
else
   W=0
end
%% Calc acceleration for both
particles
ai=zeros(1,2);
aj=zeros(1,2);
ai=[-1*(-mass*((p/(dens^2))+(p/
(dens^2)))*W);-mass*((p/(dens^2))+
(p/(dens^2)))*W]
aj=[(-mass*((p/(dens^2))+(p/
(dens^2)))*W);-1*(-mass*((p/
(dens^2))+(p/(dens^2)))*W)]
%% Calc New Velo and New Pos:

vi2x=[vi1x+(ai(1,1))*1];
vi2y=[vi1y+(ai(2,1))*1];
vi2=[vi2x;vi2y]
% v12x=[vj1x+(ai(1,1)*1)]
% v12y=[vj1y+(ai(2,1)*1)]
%
vj2x=[vj1x+(aj(1,1))*1];
vj2y=[vj1y+(aj(2,1))*1];
vj2=[vj2x;vj2y]
%
xi2=[xi1+(vi2x*1)];
yi2=[yi1+(vi2y*1)];
pos_i_2=[xi2;yi2]
%
xj2=[xj1+(vj2x*1)];
yj2=[yj1+(vj2y*1)];
pos_j_2=[xj2;yj2]
% %r22=
% %
% v12x=
% v12y=
% %
% v22x=
% v22y=
%% Final Figure
figure
circle2(xj2,yj2,r1); hold on;
circle2(xi2,yi2,r2); hold on;
drawArrow([xj2;yj2],
[xj2+vj2x;yj2+vj2y],'b'); hold on;
drawArrow([xi2;yi2],
[xi2+vi2x;yi2+vi2y],'b')
%% Superimposed
figure
circle2(xj2,yj2,r1); hold on;
circle2(xi2,yi2,r2); hold on;
drawArrow([xj2;yj2],
[xj2+vj2x;yj2+vj2y],'b'); hold on;
drawArrow([xi2;yi2],
[xi2+vi2x;yi2+vi2y],'b'); hold on;
circle2(xj1,yj1,r1); hold on;
circle2(xi1,yi1,r2); hold on;
drawArrow([xj1;yj1],
[xj1+vj1x;yj1+vj1y],'r'); hold on;
drawArrow([xi1;yi1],
[xi1+vi1x;yi1+vi1y],'r')
```

Example Applications

- Apply SPH to study the time evolution of a **toy star model** and find its equilibrium state
- Show steps of progressive ordering of particles towards equilibrium (density & location) in different **toy star** simulation case examples (2D & 3D)
- Show steps of progressive ordering of particles (density & location) of a toy collision problem of two polytropic bodies (2D)

What is a “toy star model”?

- “A simple model of a star where compressibility is retained but the gravitational force is replaced by a force between pairs which is directed a long their line of centres and proportional to their separation.”
- **Toy stars in one dimension** (Monaghan & Price (2006))

What is a "**toy star model**"?

$$\frac{d\mathbf{v}_i}{dt} = -\nu\mathbf{v}_i - \sum_{j,j\neq i} m_j \left(\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2}\right) \nabla W(\mathbf{r}_i - \mathbf{r}_j;\ h) - \lambda\mathbf{x}_i$$

$$m = M/N$$

$$P = k\rho^{1+1/n}$$

$$0 = -\frac{1}{\rho}\nabla P - \lambda\mathbf{x} = -\frac{k(1+1/n)}{\rho}\rho^{1/n}\nabla\rho - \lambda\mathbf{x}$$

$$\lambda = \begin{cases} 2k\pi^{-1/n}\left(M(1+n)/R^2\right)^{1+1/n}/M & d=2 \\ 2k(1+n)\pi^{-3/(2n)}\left(\frac{M\Gamma(\frac{5}{2}+n)}{R^3\Gamma(1+n)}\right)^{1/n}/R^2 & d=3 \end{cases}$$

$$\rho(r) = \left(\frac{\lambda}{2k(1+n)}(R^2 - r^2)\right)^n$$

Apply SPH to Toy Star Model

$$\rho_i = \sum_j m_j W(\mathbf{r}_i - \mathbf{r}_j;\ h).$$

```
Calculate_Density(x, m, h){
for i = 1 : N
    % initialize density with i = j contribution
    rho(i) = m * kernel(0, h);
    for j = i + 1 : N
        % calculate vector between two particles
        uij = x(i, :) - x(j, :);
        rho_ij = m * kernel(uij, h);
        % add contribution to density
        rho(i)+ = rho_ij;
        rho(j)+ = rho_ij;
    end
end
}
```

where $\mathbf{x}$ are the particle positions, m is the mass of each particle, and h is the smoothing length.

Apply SPH to Toy Star Model

$$\frac{d\mathbf{v}_i}{dt} = -\nu\mathbf{v}_i - \sum_{j,j\neq i} m_j \left(\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2}\right) \nabla W(\mathbf{r}_i - \mathbf{r}_j;\ h) - \lambda\mathbf{x}_i$$

```
Calculate_Acceleration(x, v, m, rho, P, nu, lambda, h){
% initialize accelerations
a = zeros(N, dim);
% add damping and gravity
for i = 1 : N
    a(i, :)+ = −nu ∗ v(i, :) − lambda ∗ x(i, :);
end
% add pressure
for i = 1 : N
    for j = i + 1 : N
        % calculate vector between two particles
        uij = x(i, :) − x(j, :);
        % calculate acceleration due to pressure
        p_a = −m ∗ (P(i)/rho(i)^2 + P(j)/rho(j)^2) ∗ gradkernel(uij, h);
        a(i, :)+ = p_a;
        a(j, :)+ = −p_a;
    end
end
}
```

where $\mathbf{v}$ are the particle velocities and $\mathbf{P}$ are the calculated pressures from the density.

Apply SPH to Toy Star Model

$$\mathbf{v}(t+\Delta t/2) = \mathbf{v}(t-\Delta t/2) + \mathbf{a}(t)\Delta t$$

$$\mathbf{x}(t+\Delta t/2) = \mathbf{x}(t) + \mathbf{v}(t+\Delta t/2)\Delta t$$

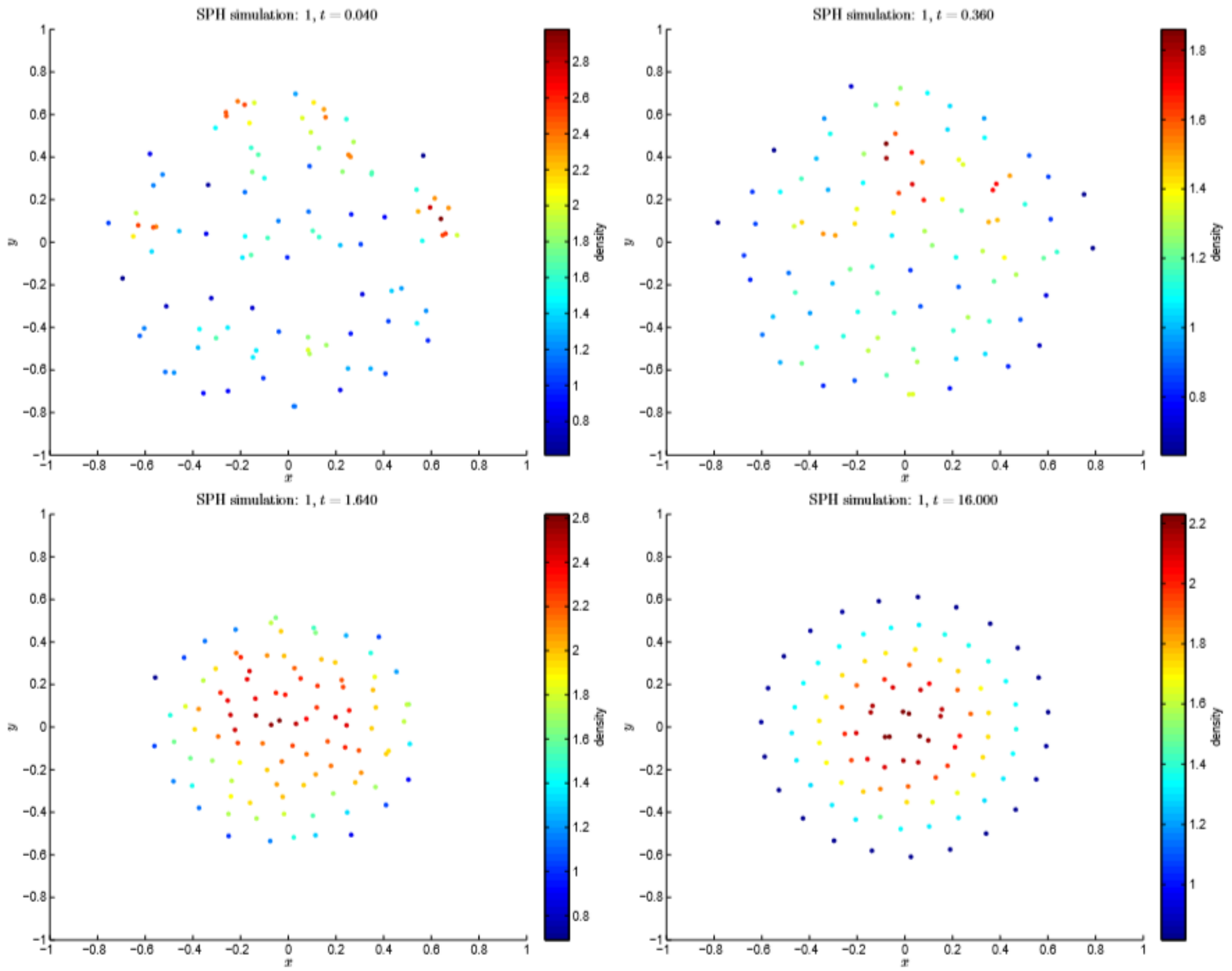
$$\mathbf{v}(t+\Delta t) = \frac{\mathbf{v}(t-\Delta t/2) + \mathbf{v}(t+\Delta t/2)}{2}$$

$$P = k\rho^{1+1/n}$$

```
Main_Loop
for i = 1 : max_time_step
    v_phalf = v_mhalf + a * dt;
    x+ = v_phalf * dt;
    v = 0.5 * (v_mhalf + v_phalf);
    v_mhalf = v_phalf;
    % update densities, pressures, accelerations
    rho = Calculate_Density(x, m, h);
    P = k * rho. ∧ (1 + 1/npoly);
    a = Calculate_Acceleration(x, v, m, rho, P, nu, lambda, h);
end
```

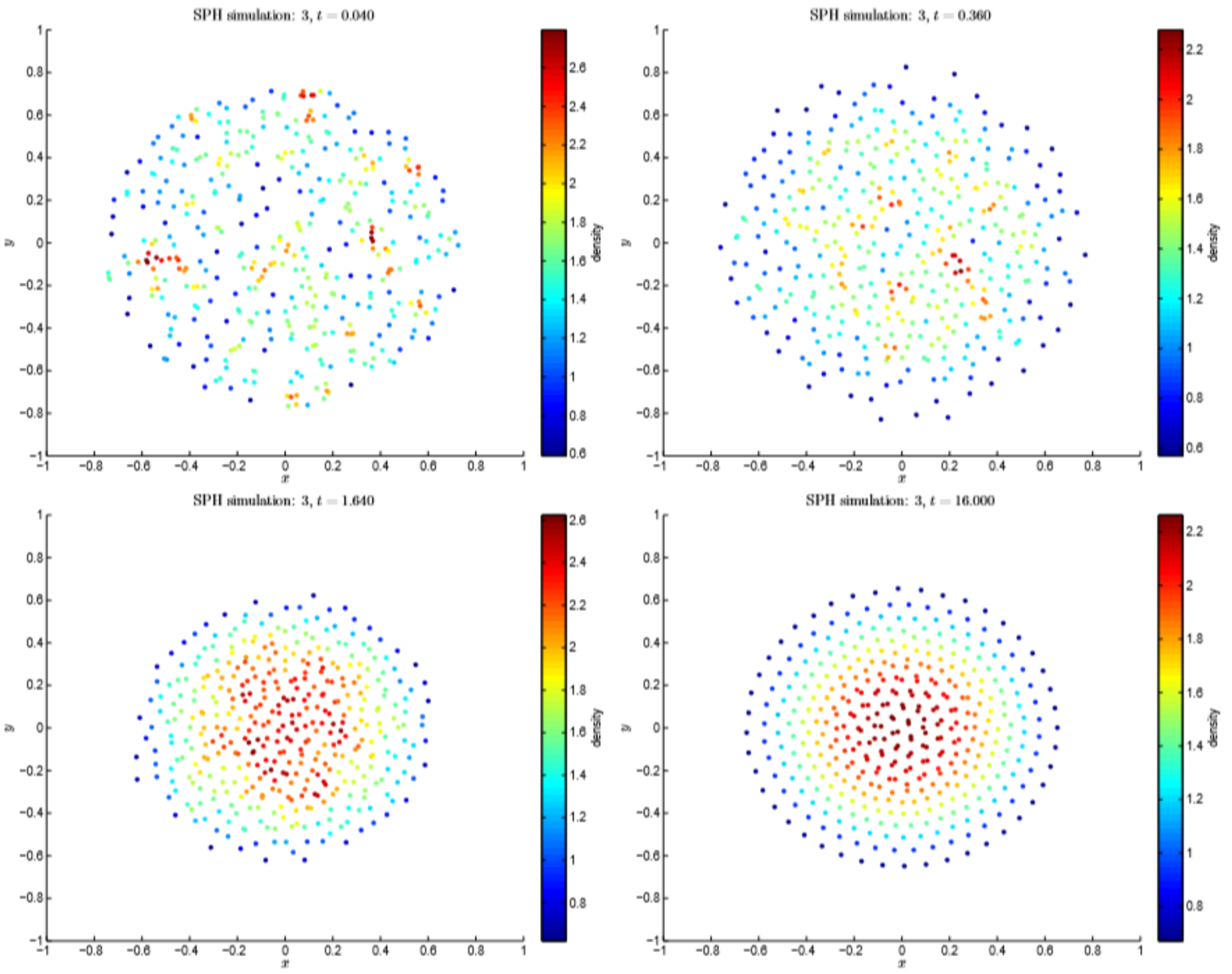
Case 1: typical 2D star collapse into equilibrium

Parameter	Value
number of particles	$N = 100$
dimension	$d = 2$
star mass	$M = 2$
star radius	$R = 0.75$
smoothing length	$h = 0.04/\sqrt{N/1000}$
time step	$\Delta t = 0.04$
damping	$\nu = 1$
pressure constant	$k = 0.1$
polytropic index	$n = 1$
max time steps	400
kernel	spline
initial config.	random inside circle radius R



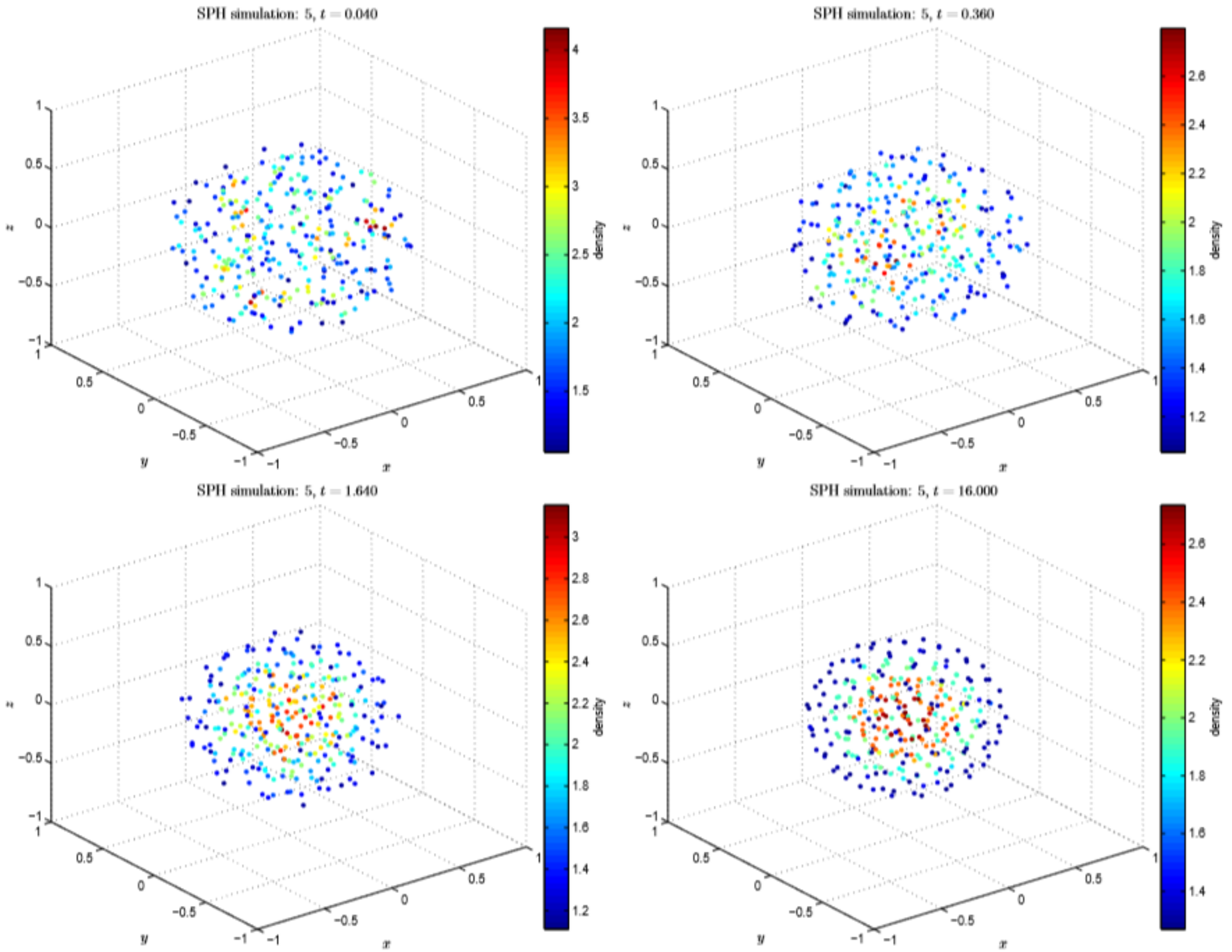
Case 2: Number of Particles increased (2D)

Parameter	Value
number of particles	$N = 400$
dimension	$d = 2$
star mass	$M = 2$
star radius	$R = 0.75$
smoothing length	$h = 0.04/\sqrt{N/1000}$
time step	$\Delta t = 0.04$
damping	$\nu = 1$
pressure constant	$k = 0.1$
polytropic index	$n = 1$
max time steps	400
kernel	spline
initial config.	random inside circle radius R



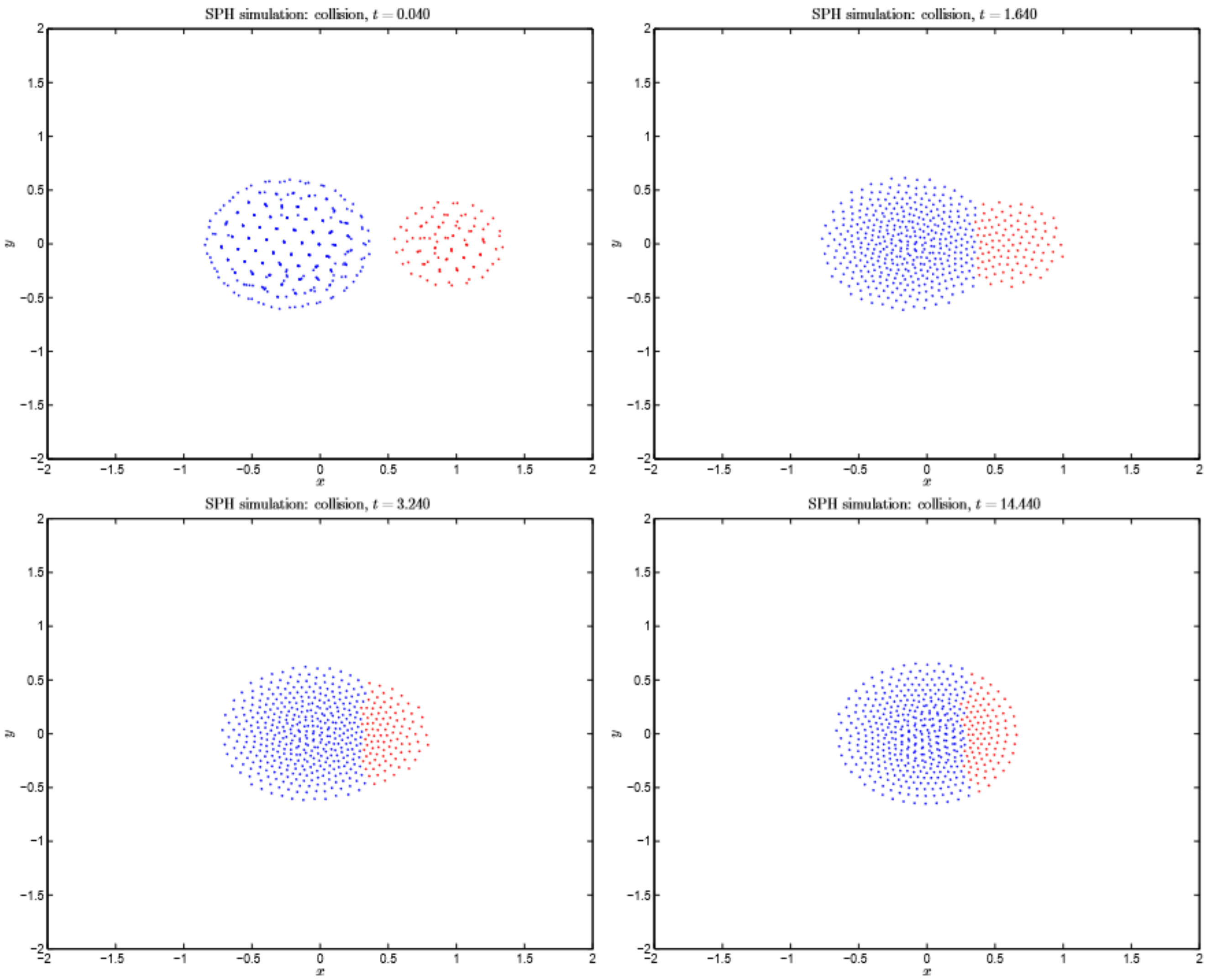
Case 3: Number of Particles increased (3D)

Parameter	Value
number of particles	$N = 300$
dimension	$d = 3$
star mass	$M = 2$
star radius	$R = 0.75$
smoothing length	$h = 0.04/\sqrt{N/1000}$
time step	$\Delta t = 0.04$
damping	$\nu = 1$
pressure constant	$k = 0.1$
polytropic index	$n = 1$
max time steps	400
kernel	spline
initial config.	random inside circle radius R



Case 4: Soft collision of 2 stars-head on (2D)

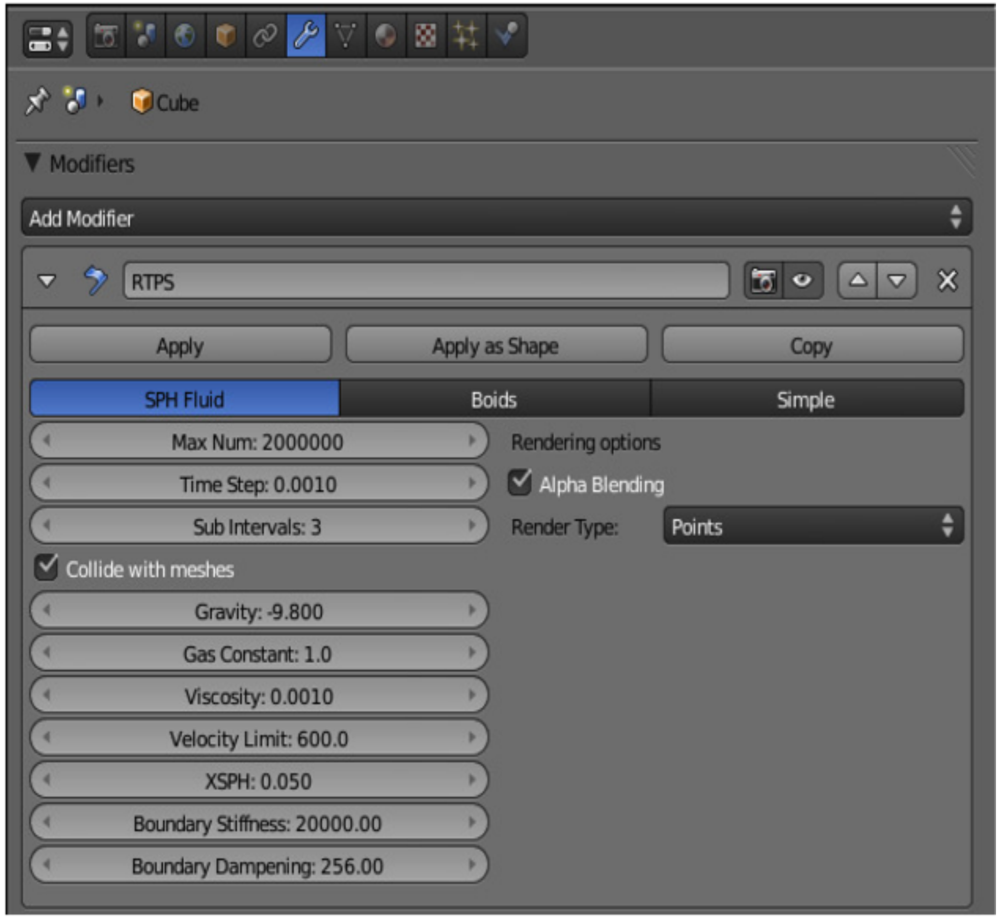
Parameter	**Value**
number of particles	$N = 500$
dimension	$d = 2$
total mass	$M = 2$
final star radius	$R = 0.75$
particles per star	400, 100
smoothing length	$h = 0.04/\sqrt{N/1000}$
time step	$\Delta t = 0.04$
damping	$\nu = 6$
pressure constant	$k = 0.1$
polytropic index	$n = 1$
max time steps	400
kernel	spline
initial config.	two stars in equilibrium center distance 1.2 apart relative velocity 0.4

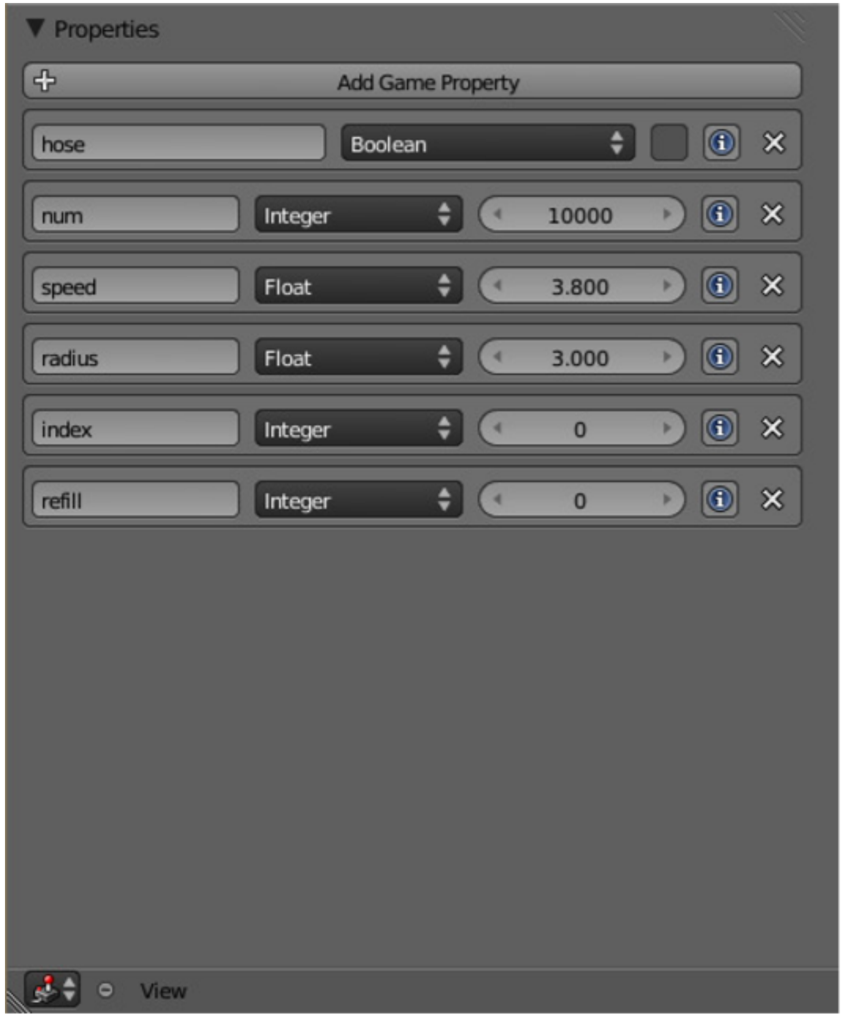


A Further Example: SPH in ‘Blender’

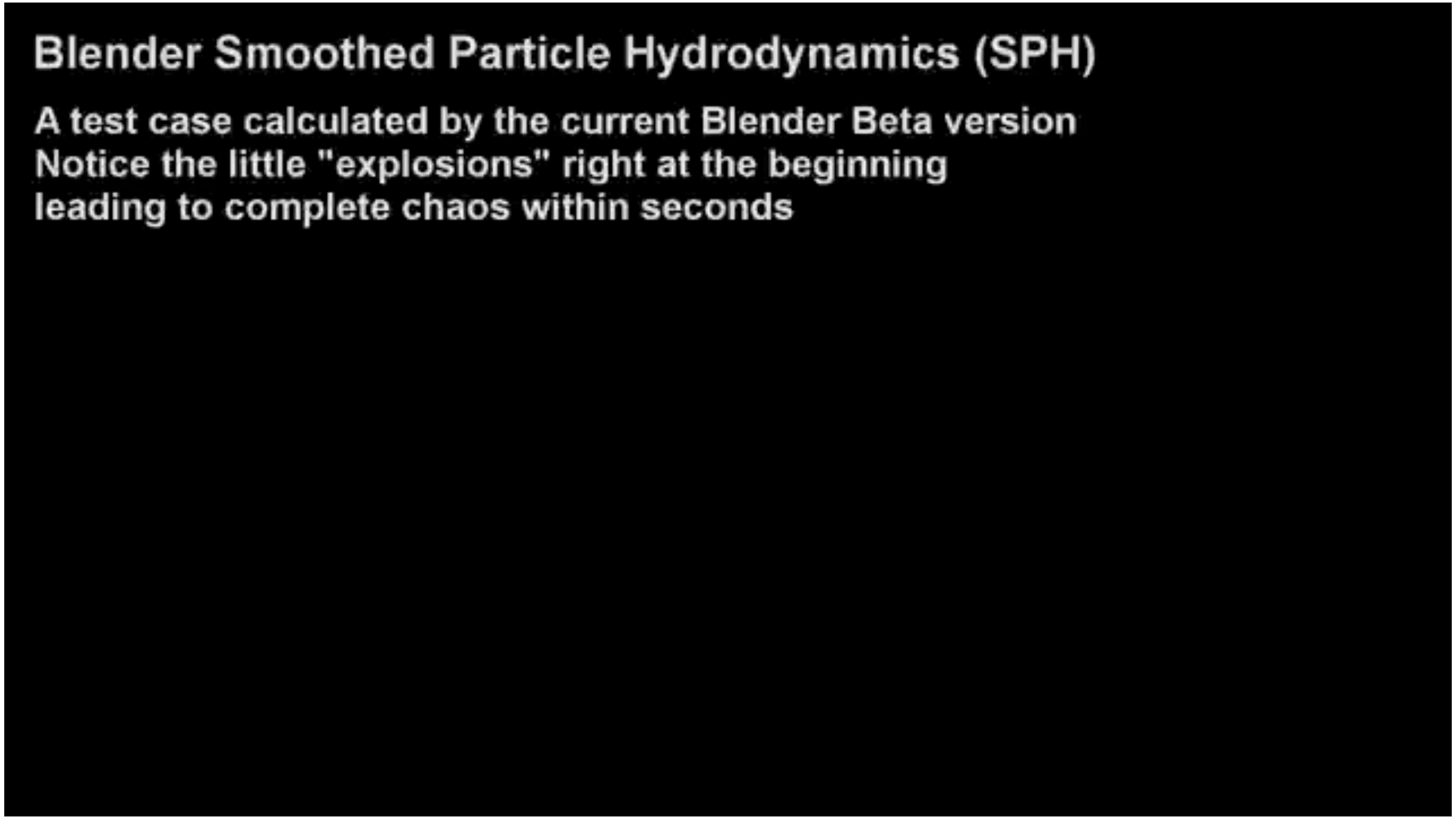
- A popular 3D content creation suite, and the software used as a platform includes an SPH fluid simulator
- Provides a comprehensive Python scripting interface to the base functionality
- Python scripts are used to manipulate objects and their properties

RTPS Modifier UI Panel & Logic Panel for emitter

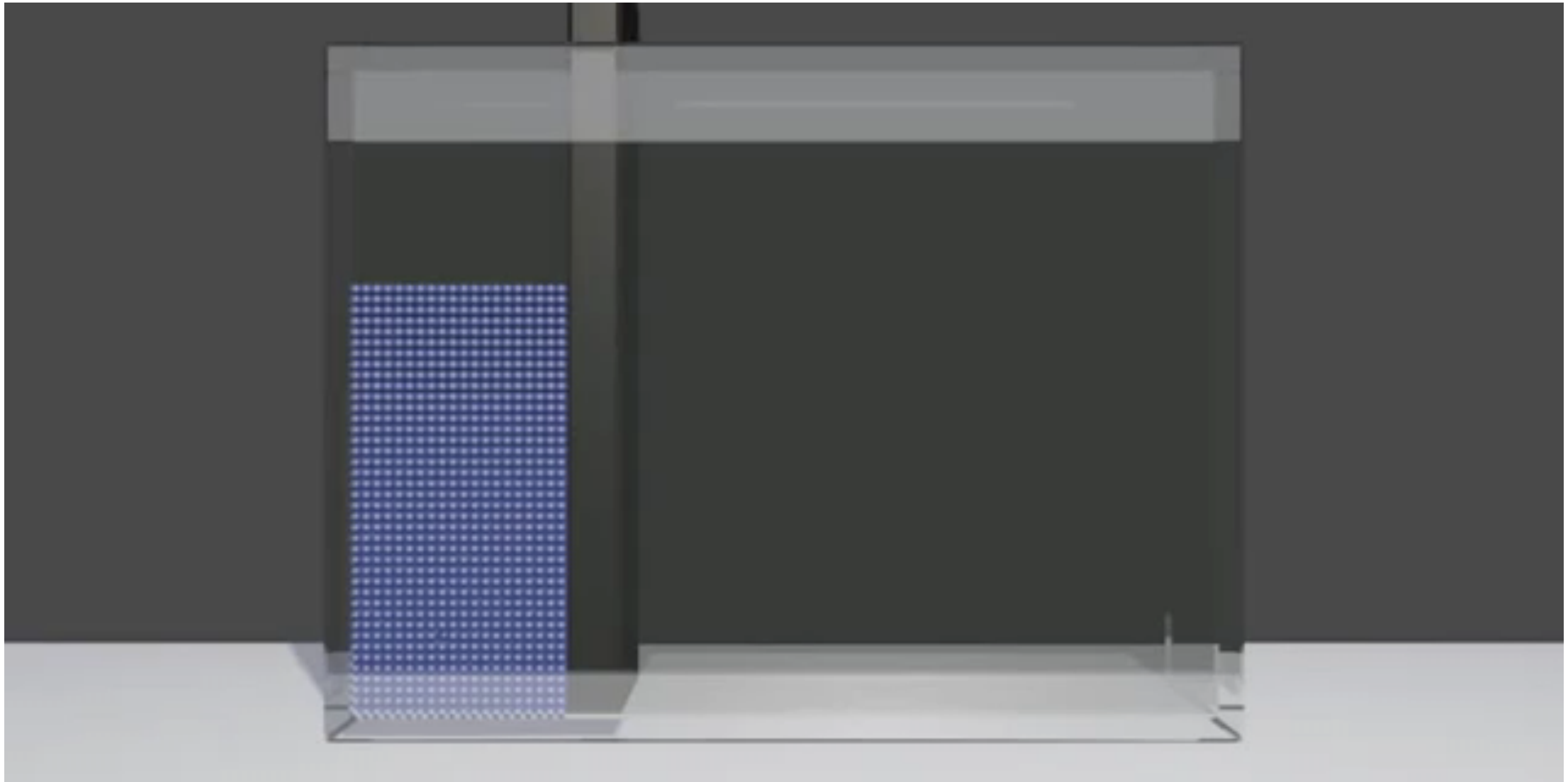




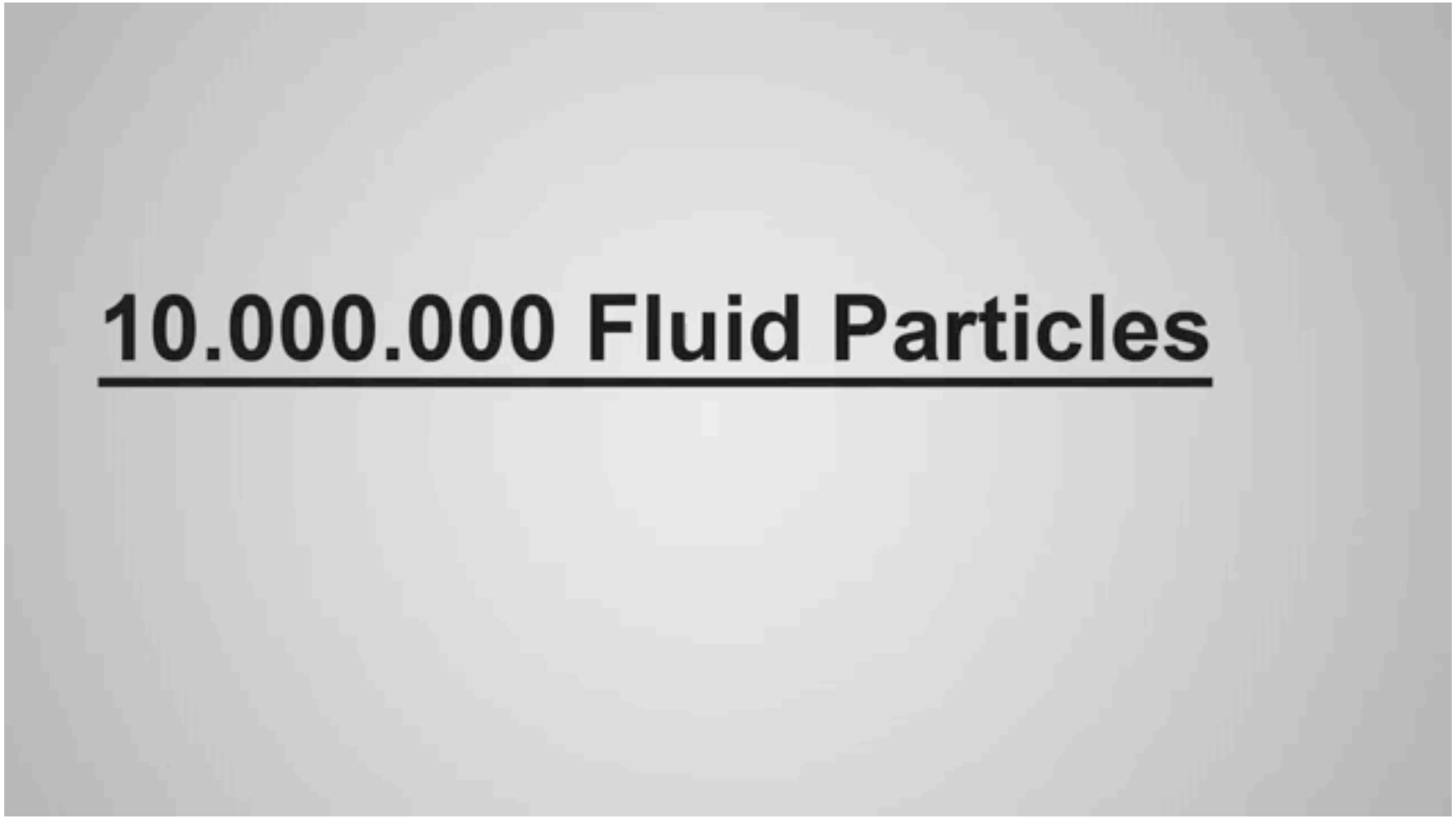
Blender SPH Model



Dam Failure

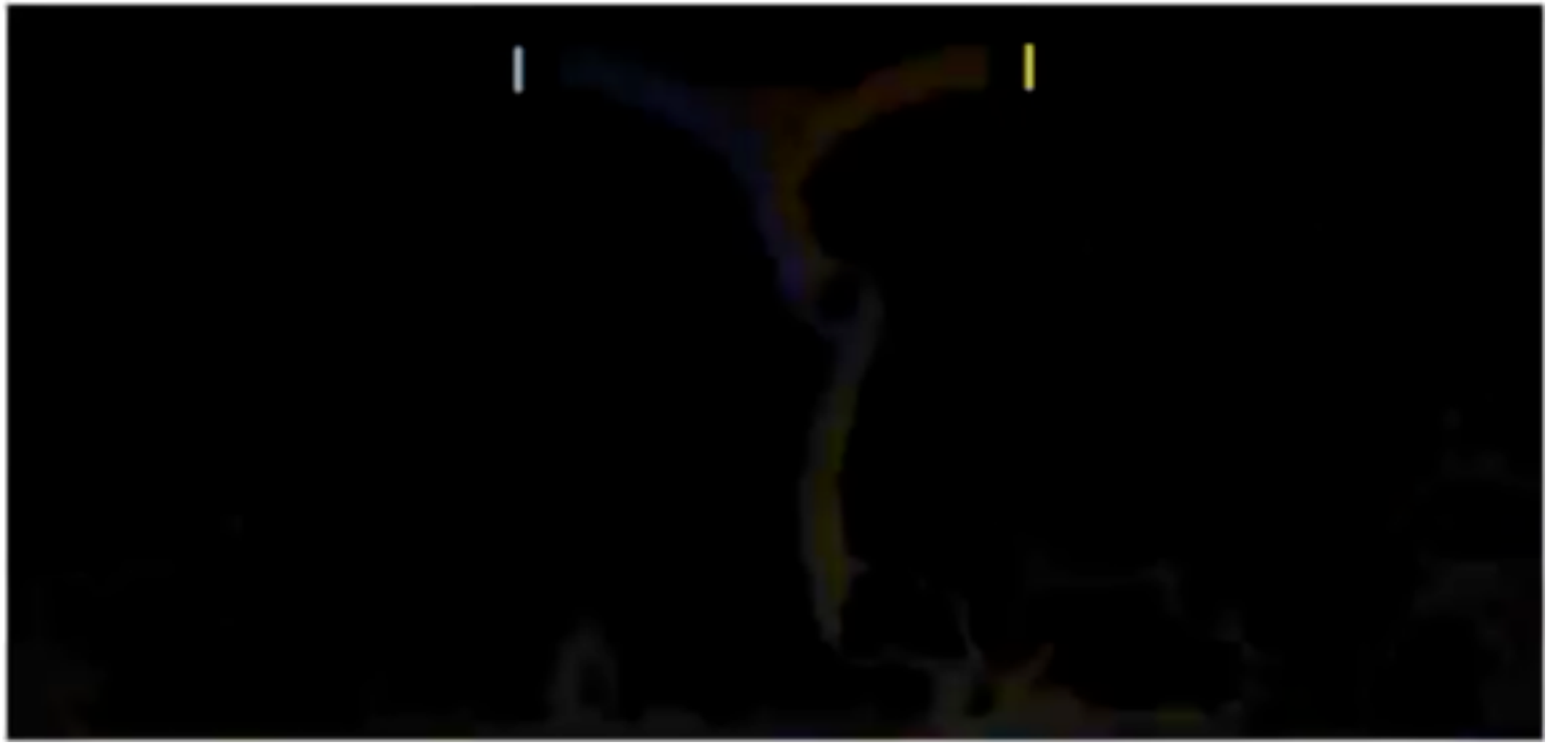


Dam Failure, Part 2.



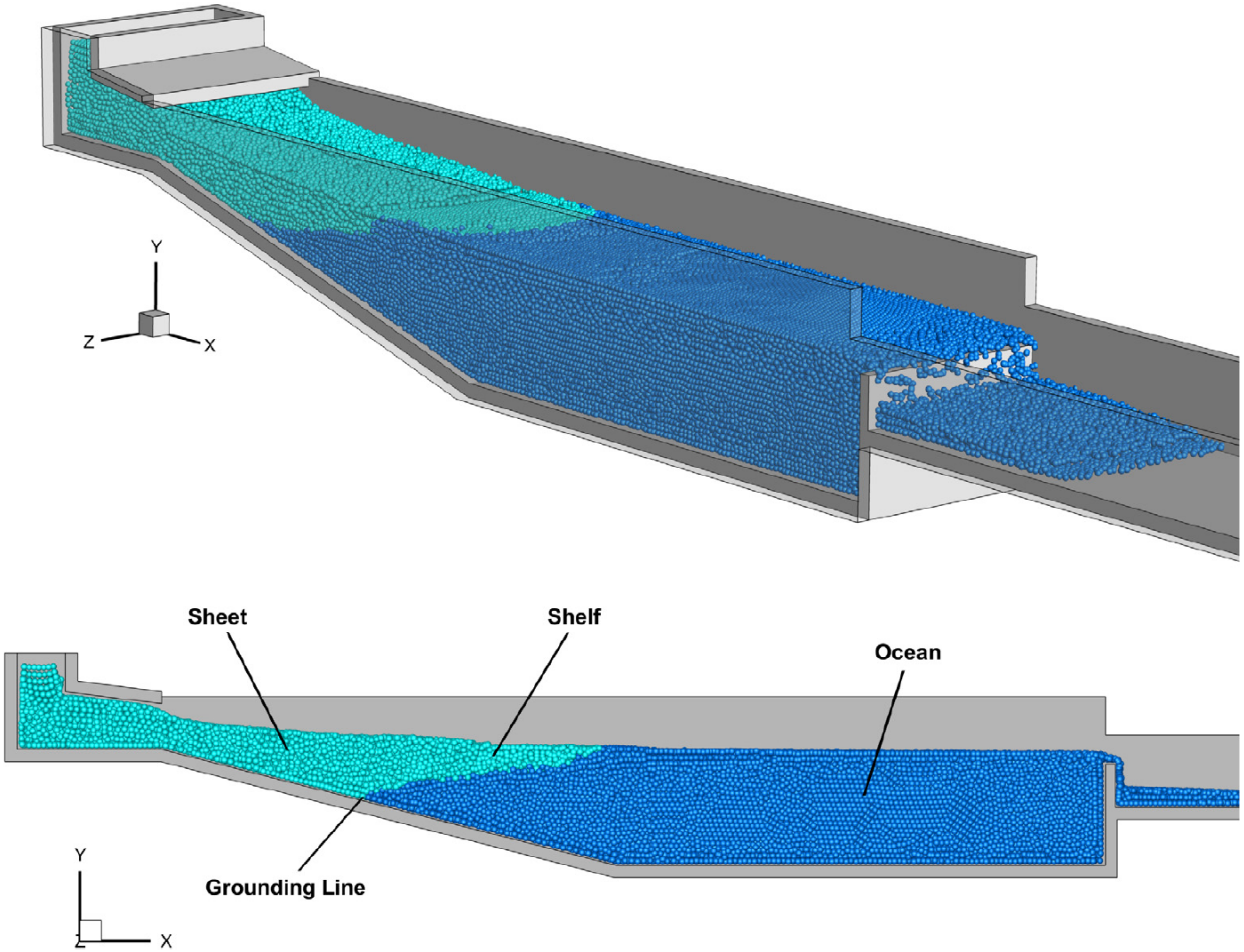
Ocean Wave Dynamics

Multiphase Fluid Flow

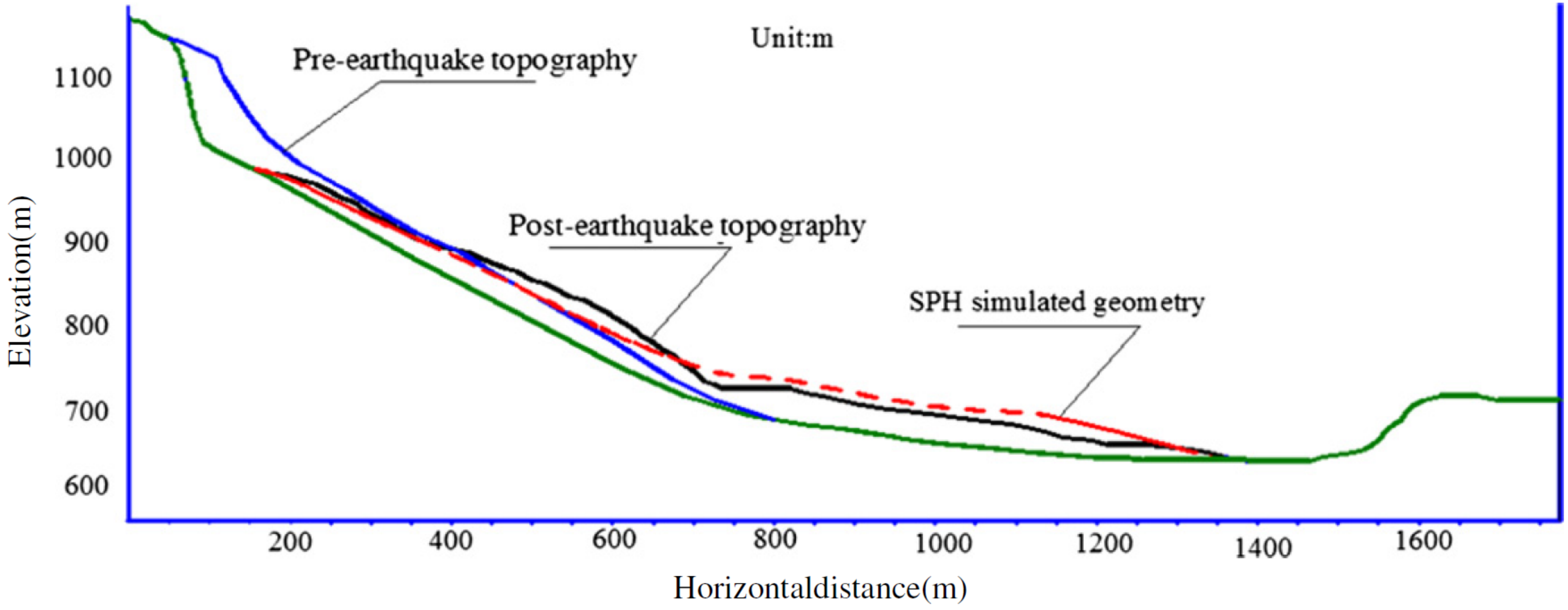


References

1. Smoothed Particle Hydrodynamics: Theory, Implementation, and Application to Toy Stars, by Philip Mocz (2011)
2. Toy Stars in one dimension, by J.J. Monaghan and D. J. Price (2003)
3. Real-Time Particle Systems in the Blender Game Engine, by Ian Johnson (2011)
4. https://www.youtube.com/watch?v=Qve54Z71VYU
5. http://blog.media.teu.ac.jp/2015/04/cg-07ff.html
6. Becker et al. (2009)
7. Mehra et al. (2012)
8. http://www.slis.tsukuba.ac.jp/~fujisawa.makoto.fu/pdf/ipsj2017_saida.pdf
9. http://www.astro.lu.se/~david/teaching/SPH/notes/ComputationalAstrophysicsL6
10. Violeau et al. (2012)
11. Previous SPH Presentation (2014)
12. https://www.youtube.com/watch?v=WyvKdaC0Ejs
13. https://www.youtube.com/watch?v=SWP25SefHHo
14. https://www.youtube.com/watch?v=e2Jk96wTqmY
15. https://www.youtube.com/watch?v=Msf_khjCQZk



Pan et al. (2013)



Huang et al. (2014)