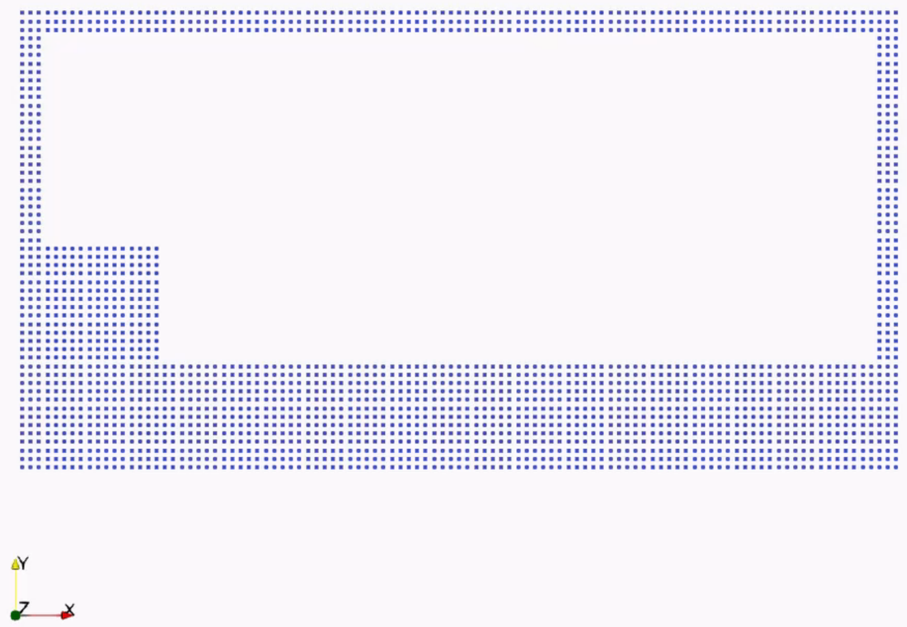
Imperial College London

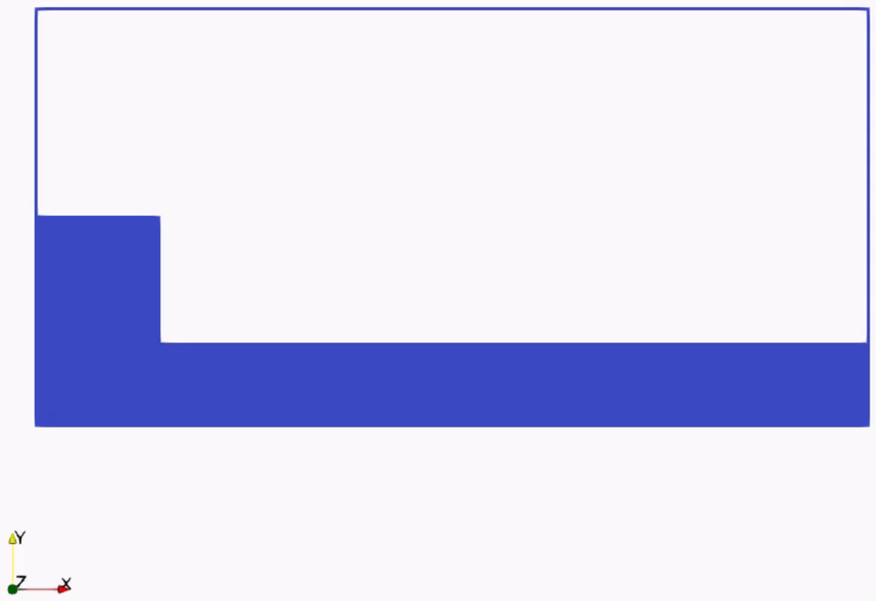
Using Smoothed Particle Hydrodynamics (SPH) to Model Multiphase Systems

Stephen Neethling

Imperial College London

What you are going to do





Imperial College
London

Basics of SPH

Convert values at discrete points into a continuous function using a smoothing kernel:

$$\langle A\rangle(\mathbf{x}) = \sum_{j=1}^{N} m_j \frac{A_j}{\rho_j} W\left(\mathbf{x} - \mathbf{x}_j, h\right)$$

The heart of the SPH method

Imperial College London

The Smoothing Kernel

Must have compact support, but ideally should be near Gaussian in shape

We will be using a cubic spline:

$$W(\mathbf{r},h) = \frac{10}{7\pi h^2}\begin{cases} 1-\frac{3}{2}q^2+\frac{3}{4}q^3 & if\ 0 \le q \le 1 \\ \frac{1}{4}(2-q)^3 & if\ 1 \le q \le 2 \\ 0 & if\ q > 2 \end{cases}$$

$$q = \frac{|\mathbf{r}|}{h}$$

Imperial College London

Gradients in SPH

We need to be able to approximate differentials:

$$\nabla A_i \approx \sum_{j=1}^{N} m_j \frac{A_j}{\rho_j} \nabla W\left(\mathbf{r}_{ij}, h\right)$$

Where for a symmetric kernel:

$$\nabla W\left(\mathbf{r}_{ij}, h\right) = \frac{dW}{dr}\left(\left|\mathbf{r}_{ij}\right|, h\right) \mathbf{e}_{ij} \qquad \mathbf{e}_{ij} = \frac{\mathbf{r}_{ij}}{\left|\mathbf{r}_{ij}\right|}$$

$\frac{dW}{dr}$ is known analytically by differentiating the kernel (you will need to do this)

$$\nabla A_i \approx \sum_{j=1}^{N} m_j \frac{A_j}{\rho_j} \frac{dW}{dr}\left(\left|\mathbf{r}_{ij}\right|, h\right) \mathbf{e}_{ij}$$

Imperial College London

(Anti)Symmetric Gradients

This approximation, though, is not symmetric or anti-symmetric (required for exact conservation of mass or momentum)

- Effect of *i* on *j* is not the same as the effect of *j* on *i*

Can make (anti)symmetric using the following relationships:

$$\rho\nabla A = \nabla(\rho A) - A\nabla\rho \qquad \frac{\nabla A}{\rho} = \nabla\left(\frac{A}{\rho}\right) + \frac{A}{\rho^2}\nabla\rho$$

Resulting in symmetric and anti-symmetric forms for the gradient:

$$\rho\nabla A \approx \sum_{j=1}^{N} m_j\left(A_j - A_i\right)\frac{dW}{dr}\left(|\mathbf{r}_{ij}|, h\right)\mathbf{e_{ij}}$$

Symmetric

$$\mathbf{e_{ij}} = -\mathbf{e_{ji}}$$

$$\frac{\nabla A}{\rho} \approx \sum_{j=1}^{N} m_j\left(\frac{A_i}{{\rho_i}^2} + \frac{A_j}{{\rho_j}^2}\right)\frac{dW}{dr}\left(|\mathbf{r}_{ij}|, h\right)\mathbf{e_{ij}}$$

Anti-symmetric

Imperial College London

Navier-Stokes and Continuity Equation

$$\frac{D\mathbf{v}}{Dt} = -\frac{\nabla P}{\rho} + \frac{\mu}{\rho}\nabla^2\mathbf{v} + \mathbf{g} \qquad \frac{D\rho}{Dt} = -\rho\nabla\cdot\mathbf{v}$$

We are solving this in a Lagrangian reference frame

If the time derivative is taken w.r.t. a reference frame that moves with a differential fluid volume (or, equivalently, in our approximation with a fluid “particle”):

$$\frac{D}{Dt} \equiv \frac{\partial}{\partial t}$$

Imperial College London

Approximating the Laplacian of the Velocity

We require a 2nd derivative for the shear stress term – could take a 2nd derivative of the kernel, but requires a lot of particles in the support region to be accurate

We will use the following approximations

$$\nabla^2 \mathbf{v}_i = \nabla \cdot \nabla \mathbf{v}_i \approx \sum_{j=1}^{N} m_j \left(\frac{1}{{\rho_i}^2} + \frac{1}{{\rho_j}^2} \right) \frac{dW}{dr} \left(|\mathbf{r}_{ij}|, h \right) \nabla \mathbf{v}_{ij} \cdot \mathbf{e}_{ij}$$

$\nabla \mathbf{v}_{ij} \cdot \mathbf{e}_{ij}$ can be interpreted as the gradient of the velocity in the $\mathbf{e}_{ij}$ direction, which we can approximate using a finite difference to yield:

$$\nabla^2 \mathbf{v}_i \approx \sum_{j=1}^{N} m_j \left(\frac{1}{{\rho_i}^2} + \frac{1}{{\rho_j}^2} \right) \frac{dW}{dr} \left(|\mathbf{r}_{ij}|, h \right) \frac{\mathbf{v}_{ij}}{|\mathbf{r}_{ij}|}$$

Governing Equations

Substituting the appropriate gradient approximations into the Navier-Stokes and continuity equation yields the following governing equations:

$$\frac{\partial \mathbf{v}_{\boldsymbol{i}}}{\partial t} \approx \mathbf{a}_i = -\sum_{j=1}^{N} m_j \left(\frac{P_i}{\rho_i{}^2} + \frac{P_j}{\rho_j{}^2}\right) \frac{dW}{dr}\left(|\mathbf{r}_{ij}|, h\right) \mathbf{e}_{\boldsymbol{ij}} + \mu \sum_{j=1}^{N} m_j \left(\frac{1}{\rho_i{}^2} + \frac{1}{\rho_j{}^2}\right) \frac{dW}{dr}\left(|\mathbf{r}_{ij}|, h\right) \frac{\mathbf{v}_{ij}}{|\mathbf{r}_{ij}|} + \mathbf{g}$$

$$\frac{\partial \rho_i}{\partial t} \approx D_i = \sum_{j=1}^{N} m_j \frac{dW}{dr}\left(|\mathbf{r}_{ij}|, h\right) \mathbf{v}_{ij} \cdot \mathbf{e}_{\boldsymbol{ij}}$$

Imperial College London

Closing the problem

As we are solving this in 2D we now have 3 equations (momentum equations in the x and y directions and a continuity equation), but 4 unknowns (velocity in the x and y direction, density and pressure) at each point

We will close the system by assuming that it is slightly compressible and use an equation of state to relate the pressure to the density:

$$P = B\left(\left(\frac{\rho}{\rho_0}\right)^{\gamma} - 1\right)$$

$$B = \frac{\rho_0\, {c_0}^2}{\gamma}$$

Imperial College London

Integrating the equations

We now need to integrate these equations by updating the particle's position, velocity and density (with the pressure being calculated based on the density)

The simplest time integration scheme is forward Euler:

$$\mathbf{x_i}^{t+1} = \mathbf{x_i}^{t} + \Delta t\ \mathbf{v_i}^{t}$$

$$\mathbf{v_i}^{t+1} = \mathbf{v_i}^{t} + \Delta t\ \mathbf{a_i}^{t}$$

$$\rho_i^{\ t+1} = \rho_i^{\ t} + \Delta t\ D_i^{\ t}$$

You should implement this scheme first as it will be the simplest based on the code stub that you have been given

Imperial College London

Integrating the equations (more advanced and accurate)

A more accurate scheme is this predictor-corrector scheme:

$$\mathbf{x_i}^{t+1/2} = \mathbf{x_i}^{t} + 0.5\Delta t\ \mathbf{v_i}^{t}$$

$$\mathbf{v_i}^{t+\frac{1}{2}} = \mathbf{v_i}^{t} + 0.5\Delta t\ \mathbf{a_i}^{t}$$

$$\rho_i{}^{t+1/2} = \rho_i{}^{t} + 0.5\Delta t\ D_i{}^{t}$$

$$\bar{\mathbf{x}}_\mathbf{i}{}^{t+1/2} = \mathbf{x_i}^{t} + 0.5\Delta t\ \mathbf{v_i}^{t+1/2}$$

$$\bar{\mathbf{v}}_\mathbf{i}{}^{t+\frac{1}{2}} = \mathbf{v_i}^{t} + 0.5\Delta t\ \mathbf{a_i}^{t+\frac{1}{2}}$$

$$\bar{\rho}_i{}^{t+1/2} = \rho_i{}^{t} + 0.5\Delta t\ D_i{}^{t+1/2}$$

$$\mathbf{x_i}^{t+1} = 2\ \bar{\mathbf{x}}_\mathbf{i}{}^{t+1/2} - \mathbf{x_i}^{t}$$

$$\mathbf{v_i}^{t+1} = 2\ \bar{\mathbf{v}}_\mathbf{i}{}^{t+1/2} - \mathbf{v_i}^{t}$$

$$\rho_i{}^{t+1} = 2\ \bar{\rho}_i{}^{t+1/2} - \rho_i{}^{t}$$

Need to store 2 sets of data for each particle

Imperial College London

Stable Time Steps

Because the method is explicit there are strict limits on the time step:

$$\Delta t_{CFL} = min\left\{\frac{h}{|\mathbf{v}_{ij}|}\right\}$$

$$\Delta t_{F} = min\left\{\sqrt{\frac{h}{|\mathbf{a}_{i}|}}\right\}$$

$$\Delta t_{A} = min\left\{\frac{h}{c_0\sqrt{(\rho/\rho_0)^{\gamma-1}}}\right\}$$

$$\Delta t = C_{CFL}\ min\{\Delta t_{CFL}, \Delta t_F, \Delta t_A\}$$

To get started you can assume that the acoustic time step is the dominant one and that the system is not very compressible:

$$\Delta t = 0.1\frac{h}{c_0}$$

Imperial College London

Smoothing the Density

The density (and thus pressure) will develop numerical fluctuations due to local variations in particle spacing. These should be smoothed every 10 to 20 time steps.

$$\rho_i = \frac{\sum_{j=1}^{n} W\left(\mathbf{r}_{ij}, h\right)}{\sum_{j=1}^{n} \frac{W\left(\mathbf{r}_{ij}, h\right)}{\rho_j}}$$

When doing your smoothing remember to NOT use a combination of old and new values in the smoothing

Imperial College London

Other Relationships

We need an appropriate ratio between the particle spacing and the smoothing length:

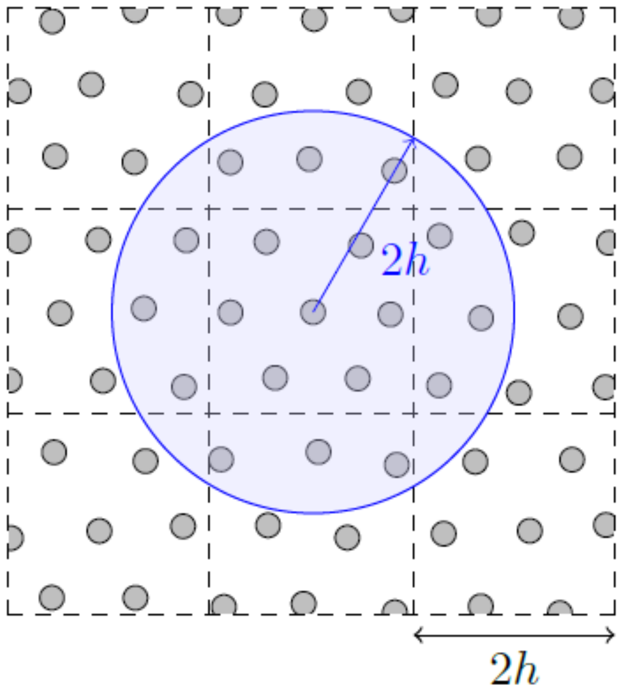
$$h = 1.3\,\Delta x$$

The mass associated with each particle is related to the initial particle spacing and density:

$$m_i = \Delta x^2 \rho_0$$

Imperial College London

Finding Neighbours



Probably the most complex bit of the code and so you have been given a basic version – there are modifications that can be made to significantly increase the speed

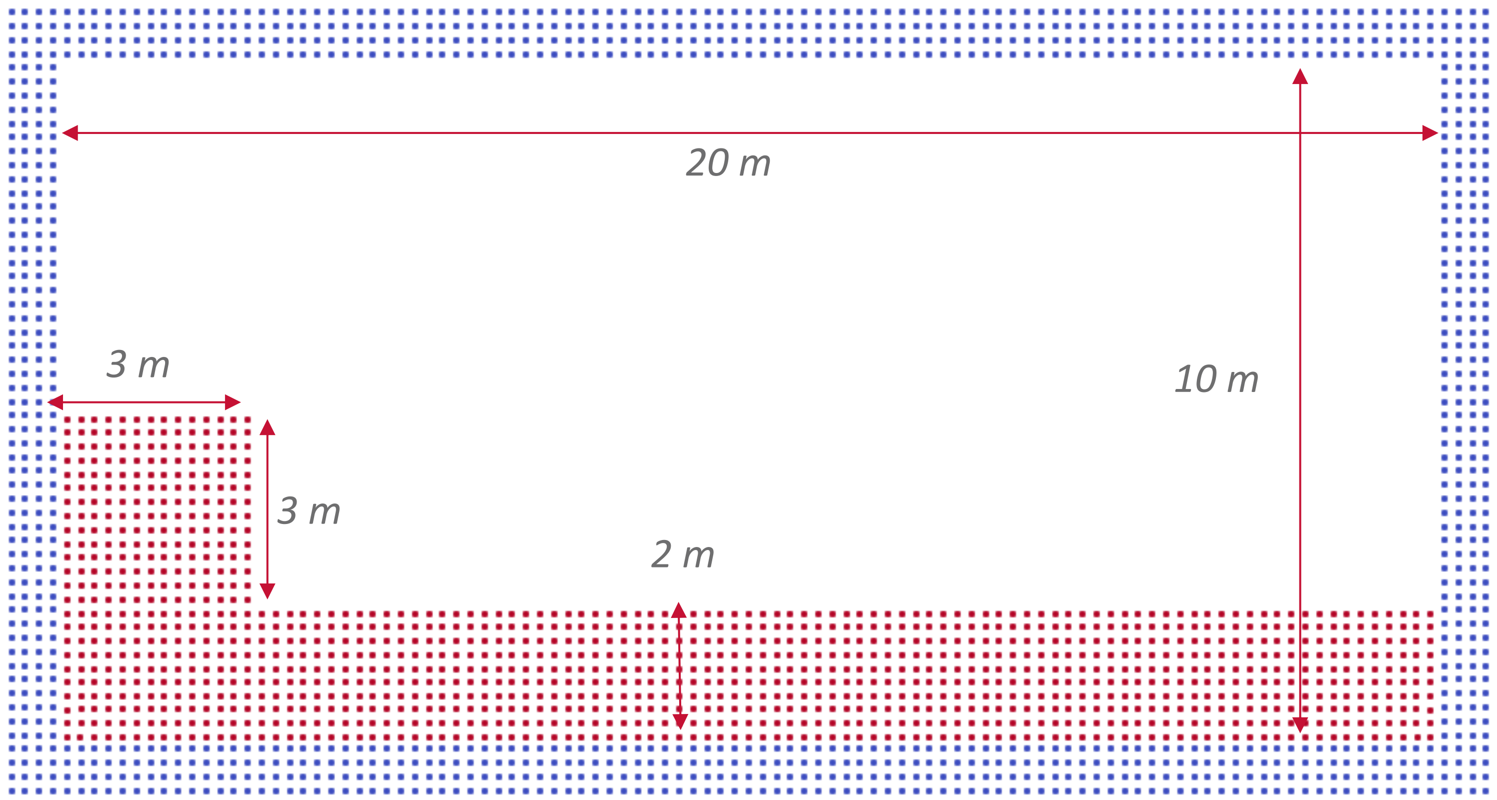
Imperial College London

The Simulation Domain

Use $\Delta x = 0.2\ m$

$\rho_0 = 1000 \frac{kg}{m^3}$

$\mu = 0.001\ Pa\ s$



THE CODE YOU HAVE BEEN GIVEN

The Tasks

- Create an SPH simulator that can run based on the initial problem geometry and conditions described above. These simulations should run until t=30 s (this may take a while under Python and so test the code over much shorter intervals). Start by implementing a simple forward Euler timestep method, for which you may need to use a very small timestep.
- Output particle data to file in such a form that it can be used for subsequent post-processing. Remember that the time steps will be quite small and so only produce outputs at a user set simulation time output interval.
- Create a post-processing code to plot the particle data stored in the output files produced by the SPH simulator. The output from this code should be an animation of the simulation results, saved in a sensible format.

Imperial College London

The Tasks – Continued

- Improve your timestepping method by implementing the predictor-corrector scheme.
- Measure the wave height and location of the crest as a function of time. What is the sloshing interval across the domain and how does this compare to the expected shallow water wave speed?
- Simulate different shaped domains, especially ones with non-vertical and horizontal walls. Try and simulate a shoaling wave (e.g. running up onto a beach).
- Carry out a convergence study in which the resolution (Δx) of the simulation is varied. You will need to decide upon a suitable metric for the error (for example, timescale for wave transits). What is the order of convergence?

Suggestion on how to proceed

Initially divide the team in two

- One sub-team to start working on numerical implementation
- One sub-team to work on saving data to file, data visualisation and post-processing

These tasks need to occur in parallel as you can’t tell if the code is working if you can’t see the results

- Once the basic simulation is working various aspects need to be implemented:
  - Particle containment
  - Re-averaging the density
  - New time stepping scheme
  - Etc….

Code Structure

- Input simulation specifications either hard-coded or via an input file
- Numerical simulation should output particle data to file at specified simulation time intervals
- Output files should be loaded by post-processing code which will carry out visualisation and data analysis
  - Visualisation should ideally be saved as real-time video

Code testing

- You are developing a new simulator and so want have test case results to compare against developed early enough

- Do your initial testing by checking that the code is not going nuts! Some suggested checks:
  - Check that no particle is going faster than the specified speed of sound
  - Check that no particle has a density significantly different from the initial density
  - Check that no particle is a significant distance outside the simulation domain (some particles are likely to leak and your code needs to deal with them)