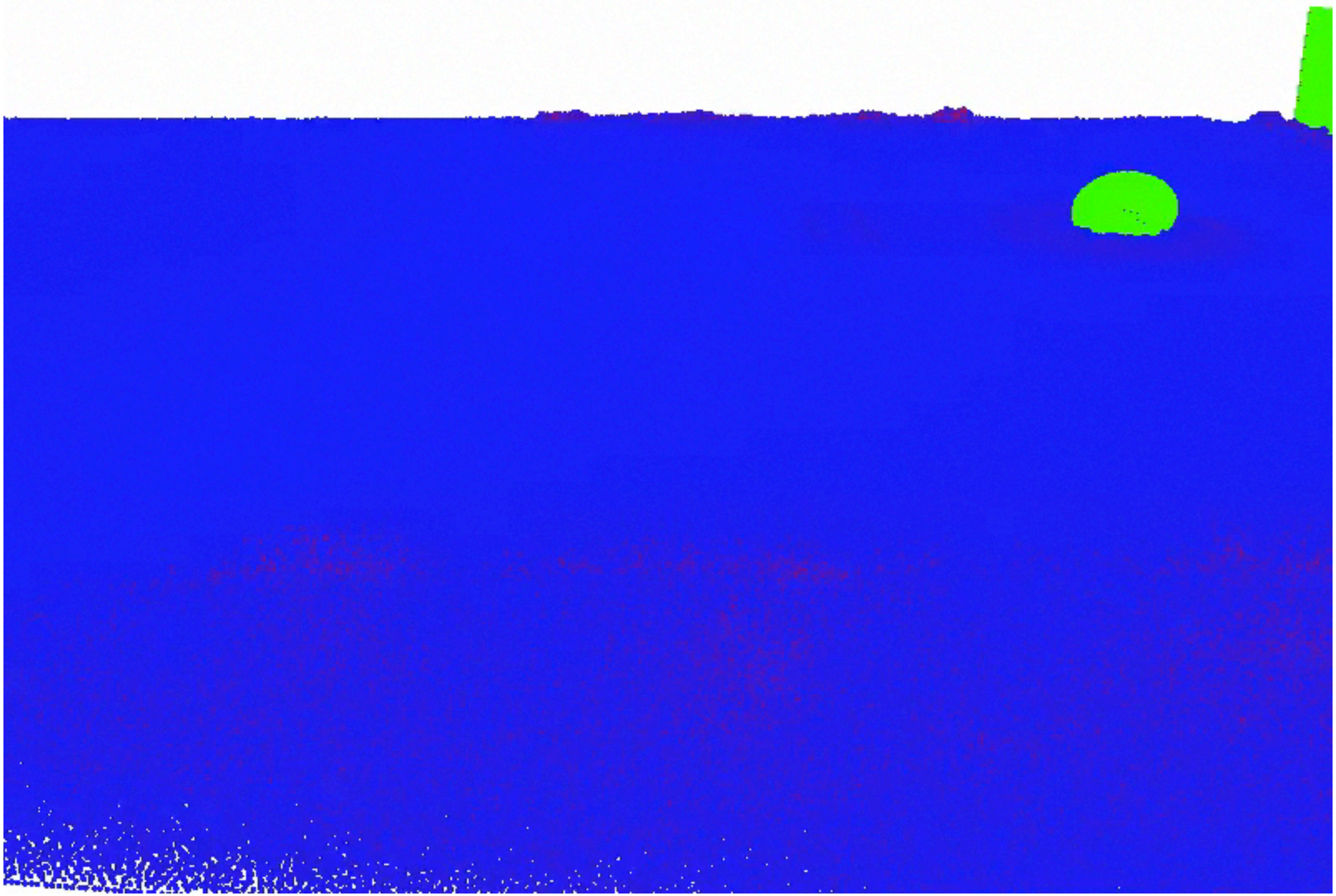
GPUSPH: a CUDA program

A free surface Smoothed Particle Hydrodynamics Code



Smoothed Particle Hydrodynamics for a weakly compressible fluid

Model nodes are irregularly spaced particles, each with mass, m_i

$$\text{Mass conservation as } m_i \text{ fixed and } \sum m_i = \text{constant}$$

$$\text{Volume of particle} = \frac{m_i}{\rho_i}$$

<u>Nodes move with fluid</u>: mesh-free Lagrangian method

Numerical Basis of Smoothed Particle Hydrodynamics

SPH is based on weighted interpolation:

$$\boxed{\hat{A}(\mathbf{r}_i) = \frac{1}{\gamma_i} \sum_j A(\mathbf{r}_j)\, W(\mathbf{r_i} - \mathbf{r_j}, h)\, \frac{m_j}{\rho_j}}$$

$$\text{with } \gamma_i = \sum_j W(\mathbf{r}_i - \mathbf{r}_j, h) \left(\frac{m_j}{\rho_j}\right)$$

where $\gamma = 1$, except near boundaries

$W(\mathbf{r}_i - \mathbf{r}_j, h)$ usually taken as Wendland kernel

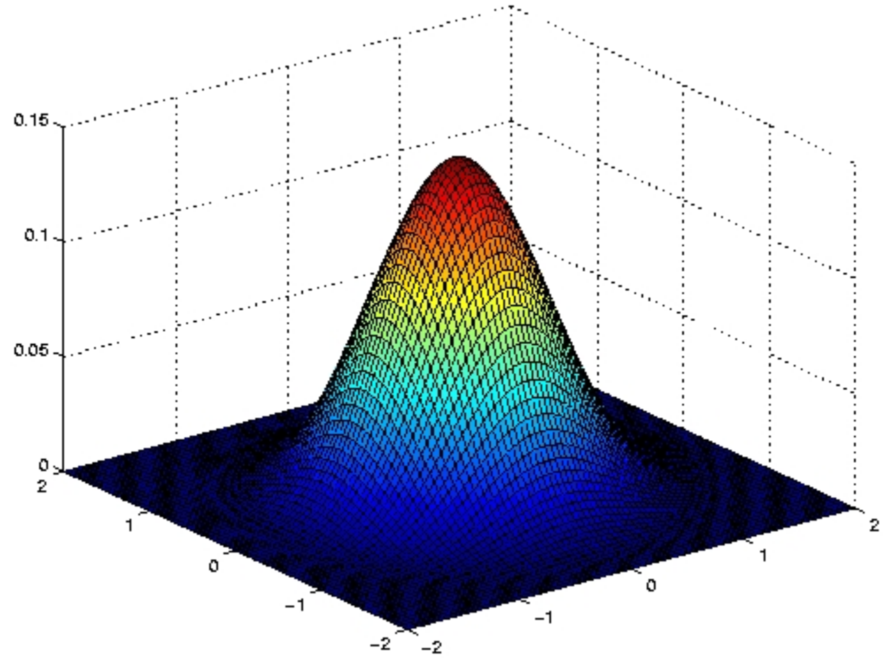
$$\boxed{\nabla \hat{A}(\mathbf{r}_i) = \frac{1}{\gamma_i} \sum_j A(\mathbf{r}_j) \nabla W(\mathbf{r}_i - \mathbf{r}_j) \frac{m_j}{\rho_j}}$$

Kernel Requirements (Monaghan)

$$\lim_{h \to 0} W(\mathbf{x}, h) = \delta(\mathbf{x})$$

Monotonically decreasing with distance |***s***-***x***|

Symmetric with distance



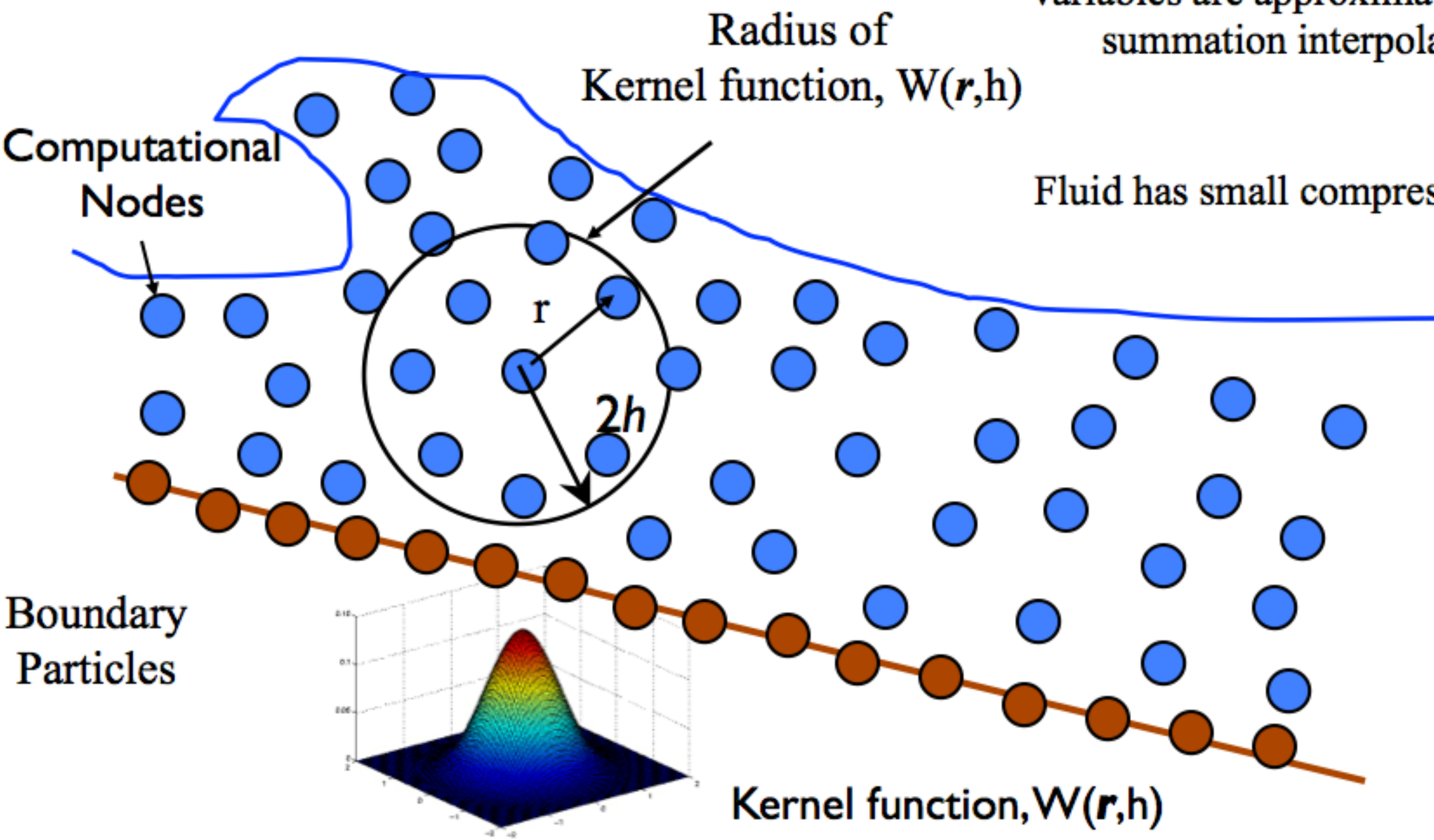
Smoothed Particle Hydrodynamics (SPH): Mesh-free Lagrangian Numerical Method

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

Fluid is described by quantities at discrete Lagrangian nodes

Variables are approximated by a summation interpolant;

Fluid has small compressibility



Governing Equations for the Fluid Flow

$$\frac{d\rho}{dt} + \rho\nabla\cdot\mathbf{v} = 0$$

Conservation of mass

$$\frac{d\mathbf{v}}{dt} = -\frac{1}{\rho}\nabla p + \nu_e\nabla^2 v - \mathbf{g}$$

Conservation of momentum

$$p = \frac{\rho_o c_s^2}{\gamma}\left[\left(\frac{\rho}{\rho_o}\right)^\gamma - 1\right]$$

Equation of State for compressible fluid

where ρ = density, $\mathbf{v}$ = velocity vector, p = pressure and $\mathbf{g}$ = gravity, $\nu_{\mathbf{e}}$ = viscosity, c_s = speed of sound

Governing Equations in Particle Form

For particle i:

$$\frac{d\rho_i}{dt} = \frac{1}{\gamma_i} \sum_j m_j \left(\mathbf{v}_i - \mathbf{v}_j\right) \cdot \nabla_i W_{ij}$$

$$\frac{d\mathbf{v}_i}{dt} = -\frac{1}{\gamma_i} \sum_j m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} + \text{visc} \right) \nabla_i W_{ij} \; + \mathbf{g}$$

$$\text{where } \gamma_i = \sum_j W_{ij} \left(\frac{m_j}{\rho_j}\right)$$

$$\frac{d\mathbf{r}_i}{dt} = \mathbf{v}_i$$

Closure Submodels for Water

Equation of state (Batchelor 1974):

$$p = \frac{\rho c_s^2}{\beta}\left[\left(\frac{\rho}{\rho_o}\right)^{\beta} - 1\right]$$

Avoids solving the <u>Poisson equation</u>

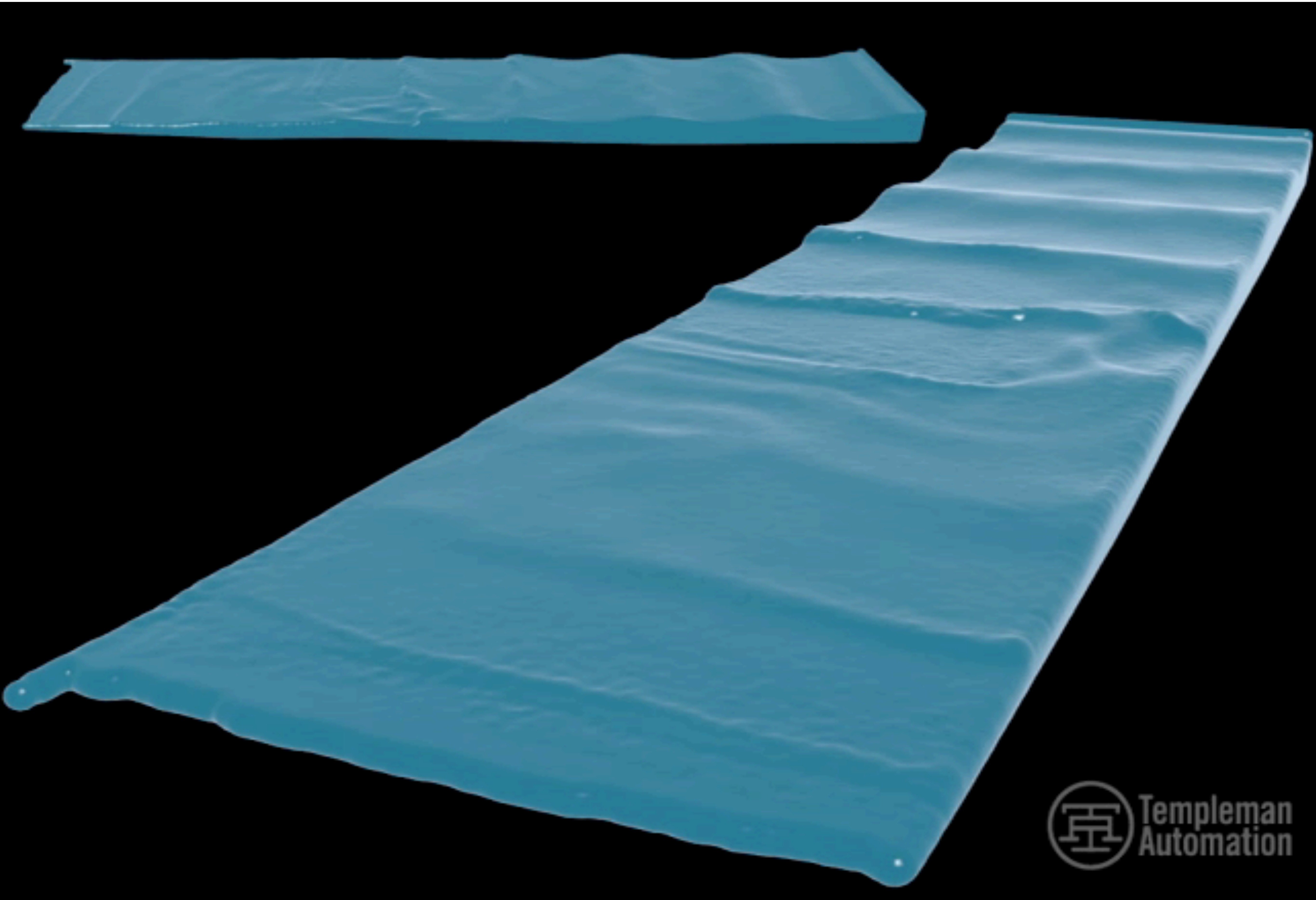
Use slow speed to get bigger time steps, but not too much; keep density variations small:

$$c_s > 10\,|\mathbf{v}|_{\mathrm{max}}$$

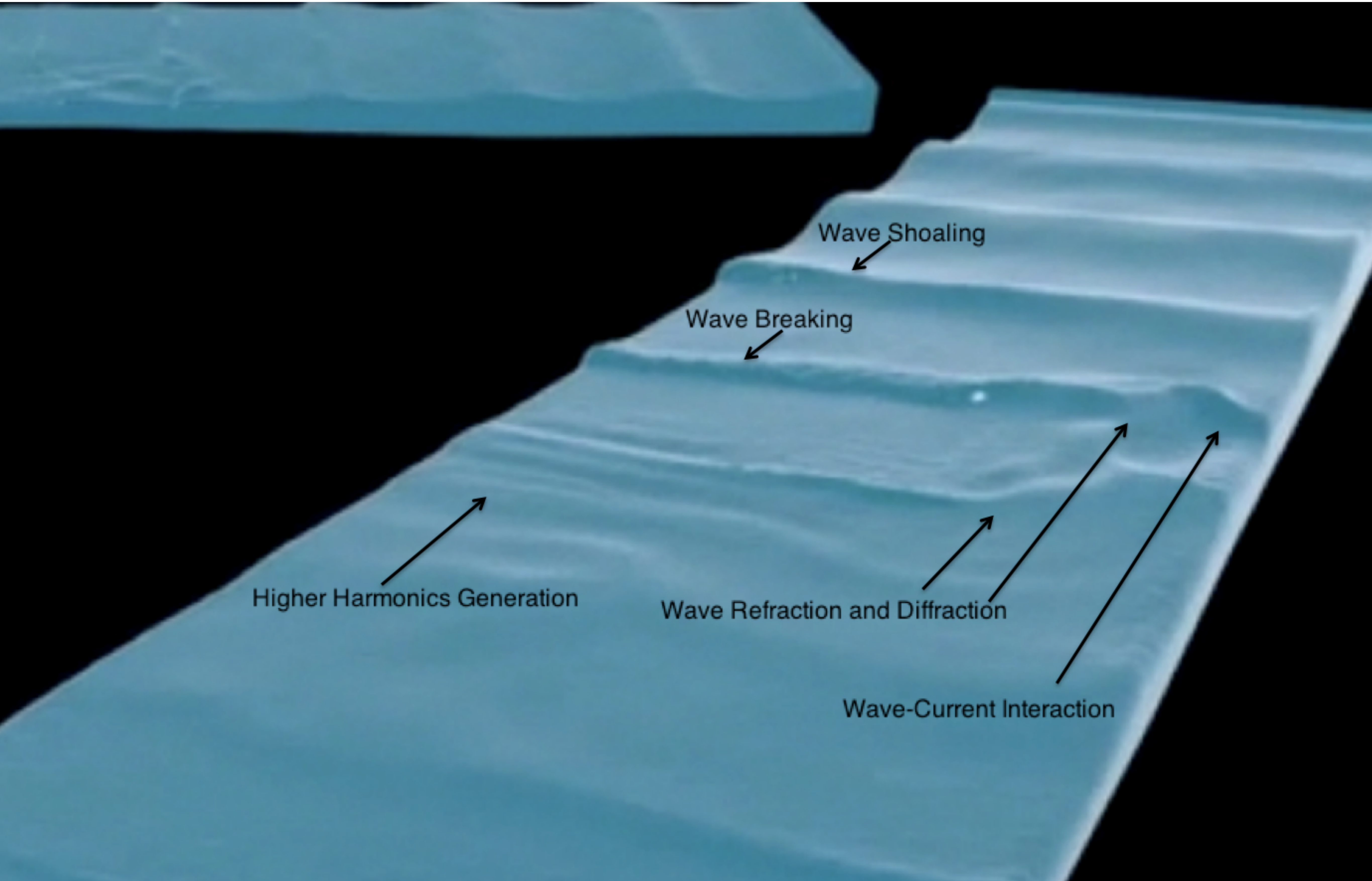
$$\mathbf{M} < 0.1$$

Variety of viscosity terms--originally needed for stability of the method.

Waves over a Sand Bar with Rip Current

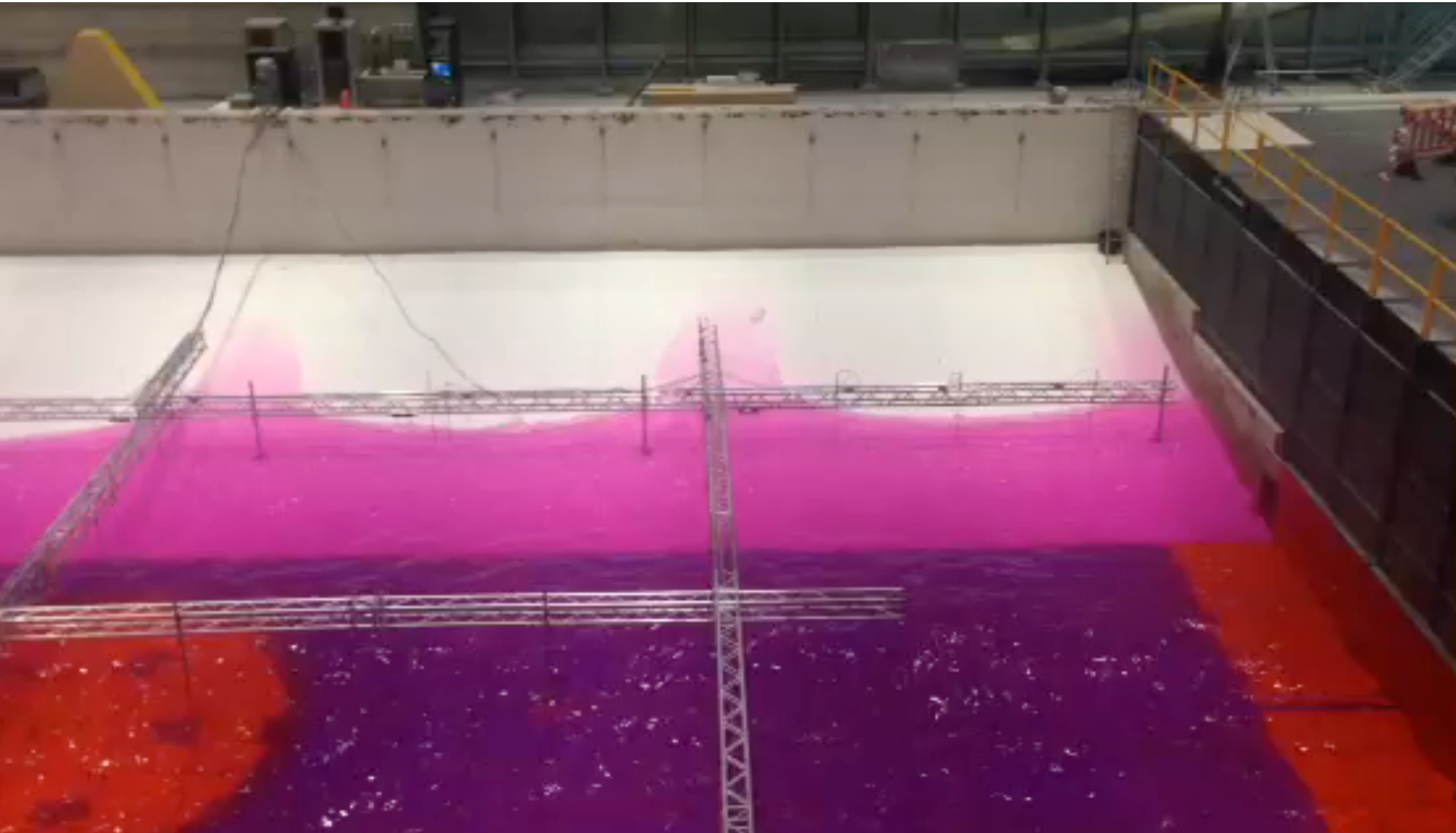


Wave Processes + Wave Setup



Instituto de Hidráulica Cantabria

ANIMO Project—Giovanni Coco



Instituto Hidráulica Ambiental: “IHCantabria”

Subharmonic edge wave generation by incident waves

Munan Xu

www.gpusph.org

GPUSPH

About Features Downloads References Gallery Contact

Posts

Jun 3, 2014

GPUSPH v3 released

GPUSPH version 3.0 is out! Head over to the release tag on GitHub.

May 31, 2014

New GPUSPH website

With the upcoming 3.0 release of GPUSPH, the first implementation of weakly-compressible SPH to run entirely on CUDA GPUs, we decided to design a new website for your viewing pleasure.

subscribe via RSS

GPUSPH

info@gpusph.org

GPUSPH was the first implementation of SPH to run entirely on GPU with CUDA and aims to provide a basis for cutting edge SPH simulations

GPUSPH 2.0

tar file: tar -xvf filename in a directory——it will create a series of subdirectories

build	all the object files
dist	executable file (but also in top level directory)
options	saved input variables: compute_capability, problem, etc
scripts	shell scripts to look at output
src	source files and headers
tests	output from GPUSPH runs

makefile in top level directory

GPUSPH Source Files

Problem files: DamBreak3D.cc, DamBreakObjects.cc, OpenChannel.cc,
SolitaryWave.cc, TestTopo.cc, WaveTank.cc

Actual Objects: Object.cc, Cone.cc, Cube.cc, Cylinder.cc, Disk.cc, Rect.cc,
Sphere.cc, Point.cc, Vector.cc

Methods: TextWriter.cc, CustomTextWriter.cc,
VTKwriter.cc, VTKLegacyWriter.cc, Writer.cc

Data storage and descriptions: EulerParameters.cc, InputProblem.cc,
ParticleSystem.cc

Kernels: all .cu files (all lower case)
forces, euler, buildneibs

Makefile

Build GPUSPH

make compute=30 problem=DamBreakObjects

Main in GPUSPH.cc

```
int  main( int argc, char** argv)
{
	if (sizeof(uint) != 2*sizeof(short)) {
		fprintf(stderr, "Fatal: this architecture does not have uint = 2 short\n");
		exit(1);
	}
	signal(SIGINT, quit);
	signal(SIGUSR1, show_timing);

	parse_options(argc, argv);
	init(clOptions.problem.c_str());

	// do an initial write
	get_arrays(true);
	do_write();

	if (clOptions.console) {
		console_loop();
	} else {
		glutInit(&argc, argv);
		glutInitDisplayMode(GLUT_RGB | GLUT_DEPTH | GLUT_DOUBLE);
		glutInitWindowSize(800, 600);
		viewport[0] = 0;
		viewport[1] = 0;
		viewport[2] = 800;
		viewport[3] = 600;
		glutCreateWindow("GPUSPH:  Hit Space Bar to start!");

#ifdef TIMING_LOG
		timing_log = fopen("timing.txt","w");
#endif

		initGL();
		initMenus();

		glutDisplayFunc(display);
		glutReshapeFunc(reshape);
		glutMouseFunc(mouse);
		glutMotionFunc(motion);
		glutKeyboardFunc(key);
		glutIdleFunc(idle);

		glutMainLoop();
	}

	quit(0);

	return 0;
}
```

Make another problem

make problem=DamBreakGate
WaveTank
SolitaryWave