Computational Methods for Modeling Multiphase Flow and Fluid-Structure Interaction in Coastal and Hydraulic Engineering Applications

Chris Kees

Coastal and Hydraulics Laboratory
US Army ERDC, Vicksburg, MS, USA

February 7, 2020
Monash Workshop on Numerical Differential Equations and Applications

COASTAL & HYDRAULICS LABORATORY

Outline

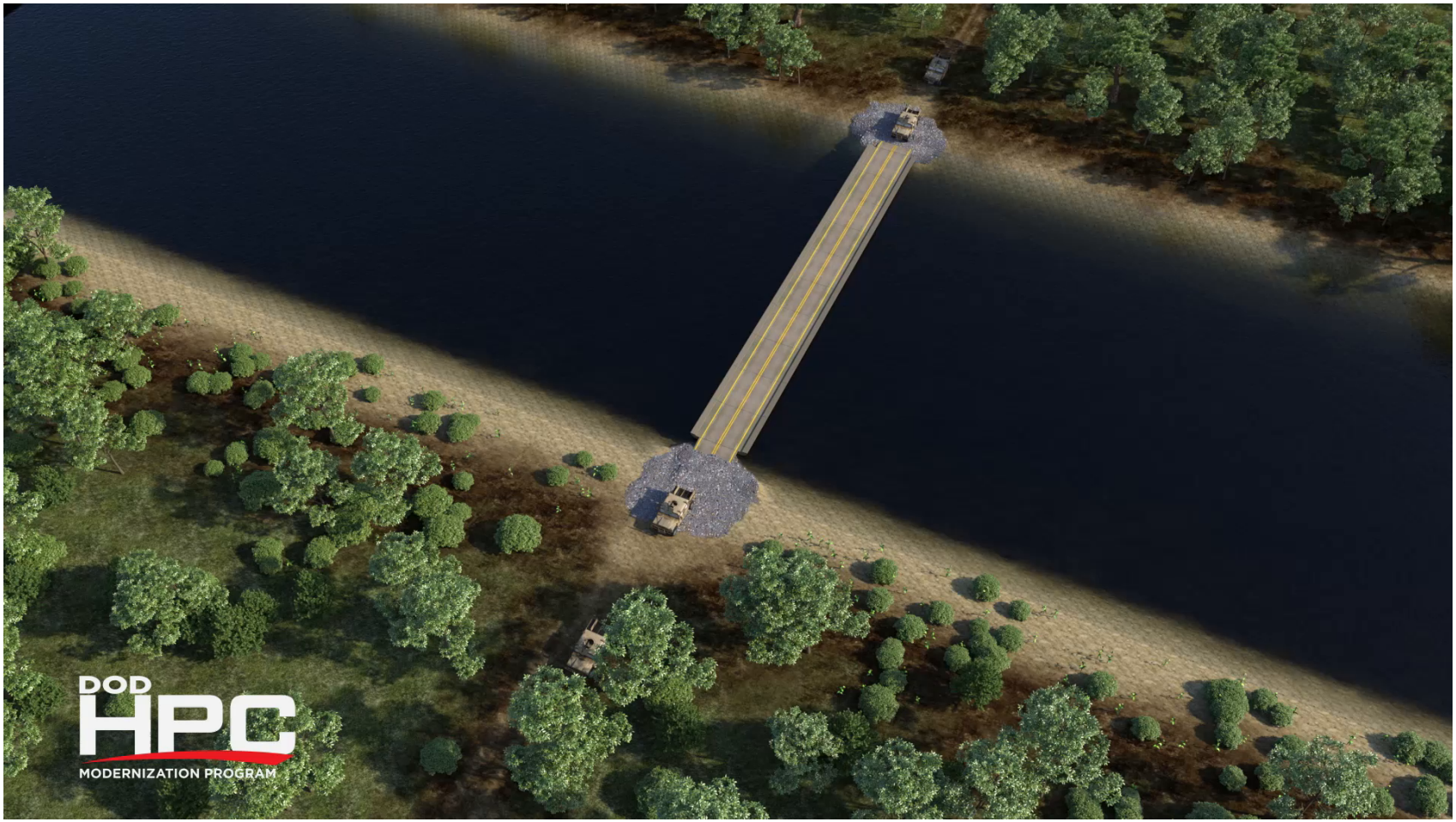
- Introduction and Motivation
- Some recent work on CutFEM and IFEM
- The Proteus Tookit for Computational Methods and Simulation

Acknowledgements

- ERDC/USACE: Ahmadia, Bryant, Collins, de Lataillade, Farthing, Howington, Pratt, Loney, Murphy, Sharp, Shepherd, Tovar, Quezada de Luna, Yang
- HR Wallingford: Dimakopoulos, Cozzuto, Sklia, Zve
- UNC: Miller, Gray
- NCSU: Kelley
- UT: Dawson, Mattis (TUM), Povich (NATO)
- Oxford: Bootland, Fletcher, Wong, Wathen
- Texas A&M: Guermond, Popov
- TU Dortmund: Kuzmin
- Clemson: Bentley, Chrispell, Jenkins, Irvin
- RPI: Sahni, Shepherd, Zhang
- MIT: Patera
- Wisconsin: Negrut, Rakhsha

COASTAL & HYDRAULICS LABORATORY

Improved Ribbon Bridge



COASTAL & HYDRAULICS LABORATORY

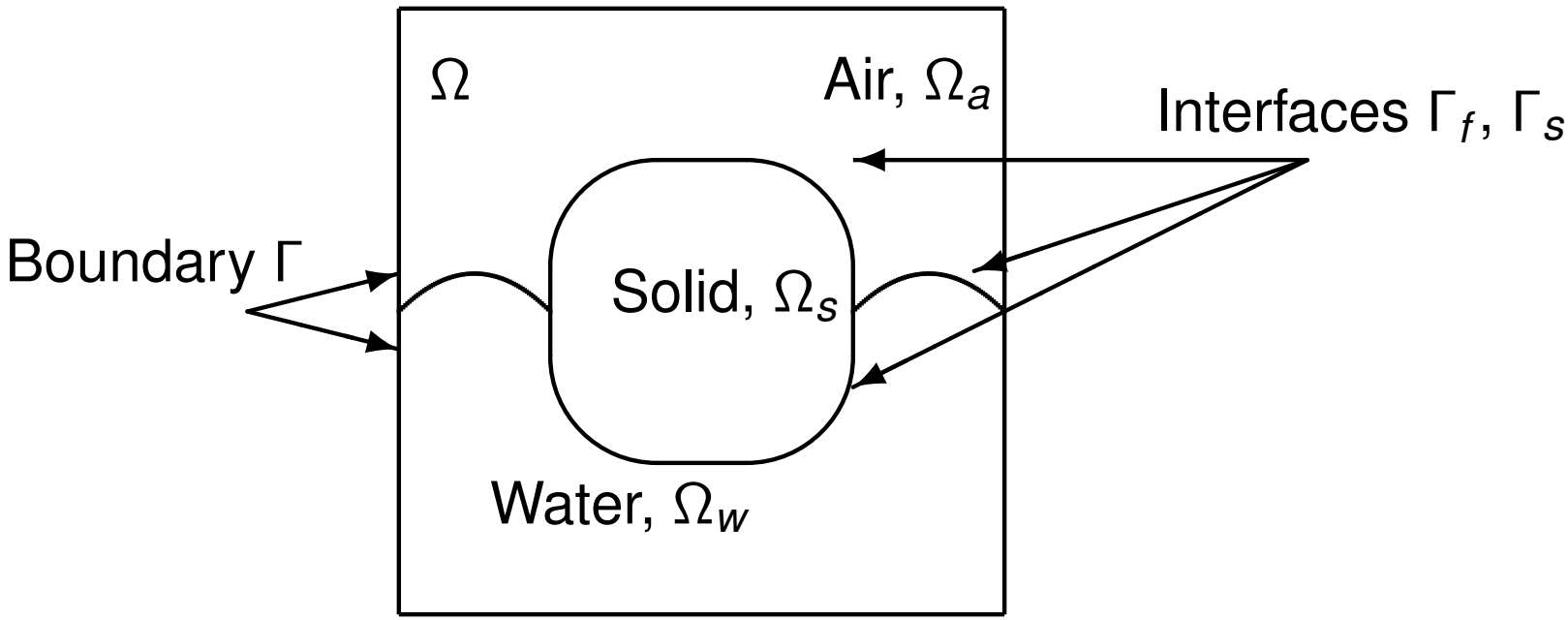
Wet Gap Crossing

- Provide assault and tactical bridge crossing capability
- Support loads up to MLC 80T/110W
- Helmand Example: 100 to 400 Meter Feet Wide, 8-18fps
- Is it stable?
- What are the loads on the overhead cable?

Beach Erosion and Deposition

COASTAL & HYDRAULICS LABORATORY

Simplified Domain



The fixed computational domain Ω and boundary Γ, with three dynamic subdomains Ω_w, Ω_a, and Ω_s and phase boundaries Γ_f and Γ_s

Numerical challenges and level set methods

- Dynamic interfaces: the fluid-fluid (air-water) interface, Γ_f, and the fluid-solid interface, Γ_s are moving in time.
- Changing topology: inter-granular contacts and bubble/droplet formation will occur, so the topology of $\Omega_s, \Omega_w, \Omega_a$ will change.
- In some important cases, ALE (conforming mesh deformation) is viable for the solid phase.
- Define "level set" functions ϕ_s and ϕ_f with the following properties:

$$\Omega_s(t) = \{\mathbf{x} \in \Omega | \phi_s(x,t) < 0\} \tag{1.1a}$$
$$\Omega_w(t) = \{\mathbf{x} \in \Omega | \phi_f(x,t) < 0\} \tag{1.1b}$$
$$\Omega_a(t) = \{\mathbf{x} \in \Omega | \phi_f(x,t) > 0\} \tag{1.1c}$$
$$\Gamma_s(t) = \{\mathbf{x} \in \Omega | \phi_s(x,t) = 0\} \tag{1.1d}$$
$$\Gamma_f(t) = \{\mathbf{x} \in \Omega | \phi_f(x,t) = 0\} \tag{1.1e}$$

Key ideas

- Represent the free boundary as an implicit surface, $\phi(\mathbf{x}, t) = 0$ (Osher and Sethian).
- Construct an equation for ϕ that is physically correct on the free boundary and reasonable in the rest of the domain:

$$\frac{\partial \phi}{\partial t} + \nabla \phi \cdot \mathbf{v} = 0 \tag{1.2}$$

- Used immersed boundary approximations to enforce boundary conditions along $\phi = 0$.
- Calculate the signed distance to $\phi = 0$:

$$\|\nabla \phi^d\| = 1 \tag{1.3}$$

- Do in a moving frame that allows the mesh to conform to a moving solid.

Two-Phase Flow, Part 1

1. Variable coefficient Navier-Stokes:

$$\begin{aligned} \nabla \cdot \mathbf{v}_+ &= 0 \\ \rho\left(\frac{\mathbf{v}_+ - \mathbf{v}_-}{\Delta t} + \mathbf{v}_+ \cdot \nabla \mathbf{v}_+\right) + \nabla \cdot [-2\mu\varepsilon(\mathbf{v}_+)] + \nabla p_+ &= \rho \mathbf{g} \end{aligned}$$

where

$$\rho = H_\epsilon(\phi_-)\rho_a + [1 - H_\epsilon(\phi_{-1})]\rho_w; \quad \mu = H_\epsilon(\phi_-)\mu_a + [1 - H_\epsilon(\phi_{-1})]\mu_w$$

2. Interface (level-set) transport:

$$\frac{\phi_+^* - \phi_-}{\Delta t} + \mathbf{v}_+ \cdot \nabla \phi_+^* = 0$$

Two-Phase Flow, Part 2

3. Fluid volume fraction advection

$$\frac{H_+^* - H_-}{\Delta t} + \nabla \cdot \left(H_+^* \mathbf{v}_+ \right) = 0$$

4. Redistance level set

$$\begin{aligned} \|\nabla \phi_+^{**}\| &= 1 \\ \phi_+^{**} &= 0 \text{ on } \phi_+^* = 0 \end{aligned}$$

5. Correct level set

$$\begin{aligned} H_\epsilon(\phi' + \phi_+^{**}) - H_+^* &= \epsilon_1 \Delta \phi' \\ \nabla \phi' \cdot \mathbf{n} &= 0 \text{ on } \partial\Omega \end{aligned}$$

$$\phi_+ = \phi' + \phi_+^{**} \quad H_+ = H_\epsilon(\phi' + \phi_+^{**})$$

See Quezada, Kuzmin, and K. 2019 for monolithic variant

COASTAL & HYDRAULICS LABORATORY

Three-Phase Level Set Formulation

$$\nabla \cdot \mathbf{v} = 0 \text{ in } \Omega \tag{1.4}$$

$$H_\epsilon(\phi_s)\left\{\rho\left(\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v}\cdot\nabla\mathbf{v}\right) + \nabla\cdot[-2\mu_f\varepsilon(\mathbf{v})] + \nabla p - \rho\mathbf{g}\right\} + \text{Nitsche}(\phi_s, \mathbf{v}) + [1 - H_\epsilon(\phi_s)](\mathbf{u} - \mathbf{u}_s) = 0 \text{ in } \Omega \tag{1.5}$$

Nitsche

- Integrate by parts on fluid momentum with no-slip only on Γ_s:

$$\int_{\Omega_f} [\mu \nabla \mathbf{v} \cdot \nabla \mathbf{w}] \, dV - \int_{\Gamma_s} \mu \nabla \mathbf{v} \cdot \mathbf{n} \mathbf{w} dS = 0$$

- Nitsche 71 showed how to enforce Dirichlet conditions consistently via penalty:

$$\int_{\Gamma_s} \left[-\mu(\phi) \frac{\partial \mathbf{v}}{\partial \mathbf{n}} \mathbf{w} - \mu(\phi) \frac{\partial \mathbf{w}}{\partial \mathbf{n}} \mathbf{v} + C(h_e)(\mathbf{v} - \mathbf{v}_s)\mathbf{w} \right] dS$$

- We avoid explicitly meshing Γ_s by using the level set tools:

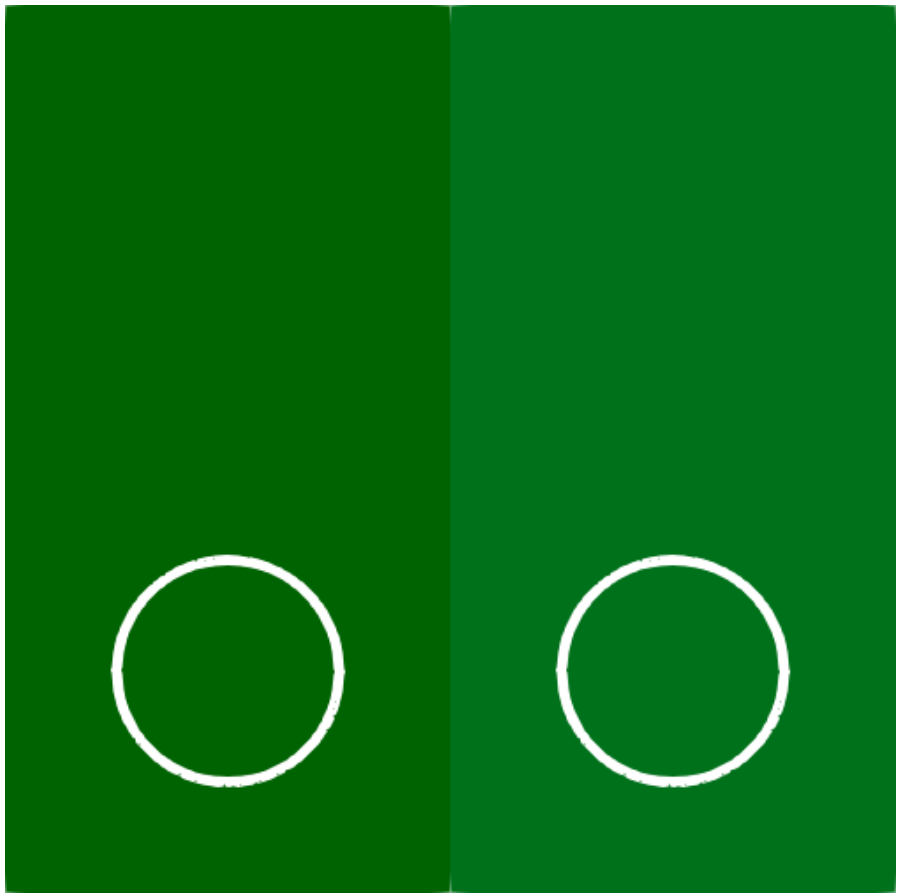
$$\int_{\Gamma_s} f dS = \int_{\Omega} \delta(\phi_s) f dV \approx \int_{\Omega} \delta_\epsilon(\phi_s) f dV$$

See [Osher and Fedkiw 2006]

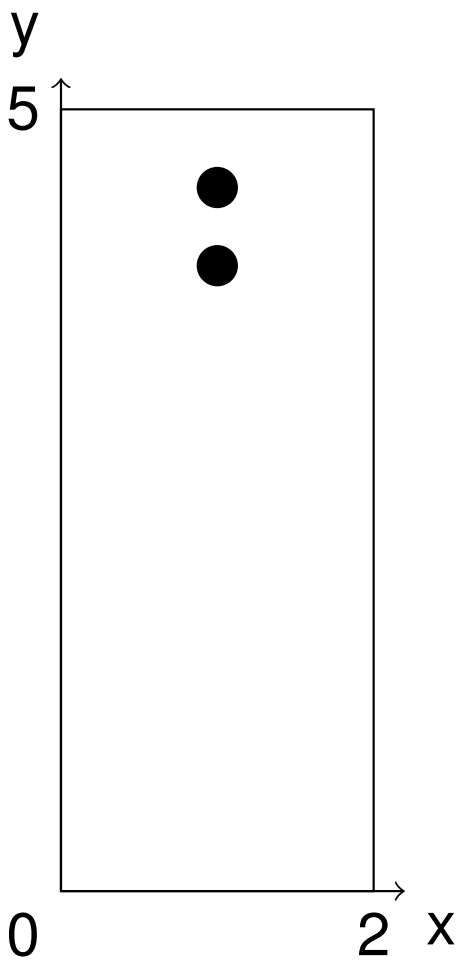
Prior Work in Proteus and Chrono

- We have developed a numerical laboratory combining Finite Element Methods (FEMs) from Proteus (https://proteustoolkit.org), and Discrete Element Methods (DEMs) from Chrono (https://projectchrono.org).
- The approach is based generally on level set methods, while using regularized Heaviside and Dirac functions to approximate volume and surface integrals on cells cut by the interface.
- These methods can produce qualitatively correct results for problems involving granular contacts and surface tension.

Surface Tension (Laplace-Beltrami Operator)



Sedimentation of two particles in a 2D channel



$$D = 0.25cm$$

$$\Omega = [0, 2] \times [0, 6]$$

$$c_1 = [1, 5]$$

$$c_2 = [1, 4.5]$$

$$\rho_s = 1.5$$

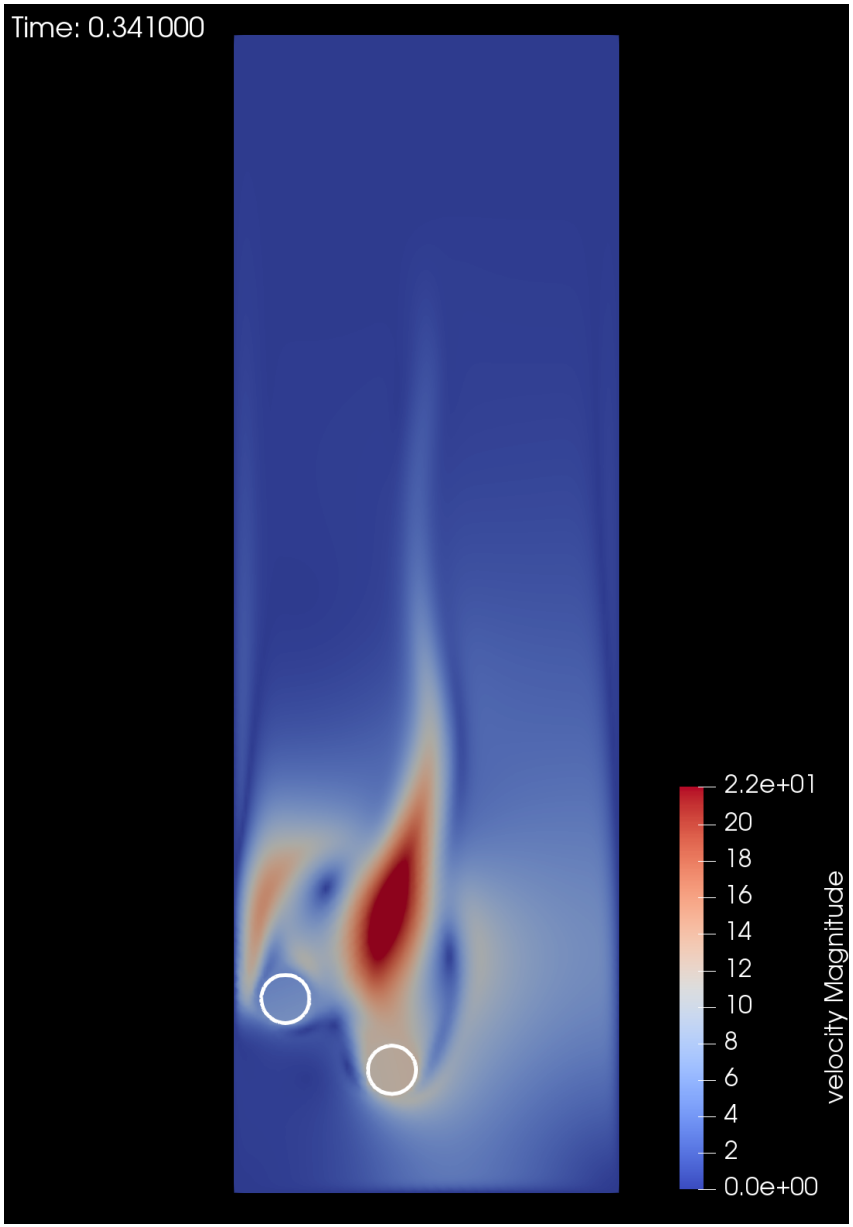
$$\rho_f = 1.0$$

$$\nu_f = 0.01$$

$$g = [0, -981, 0]$$

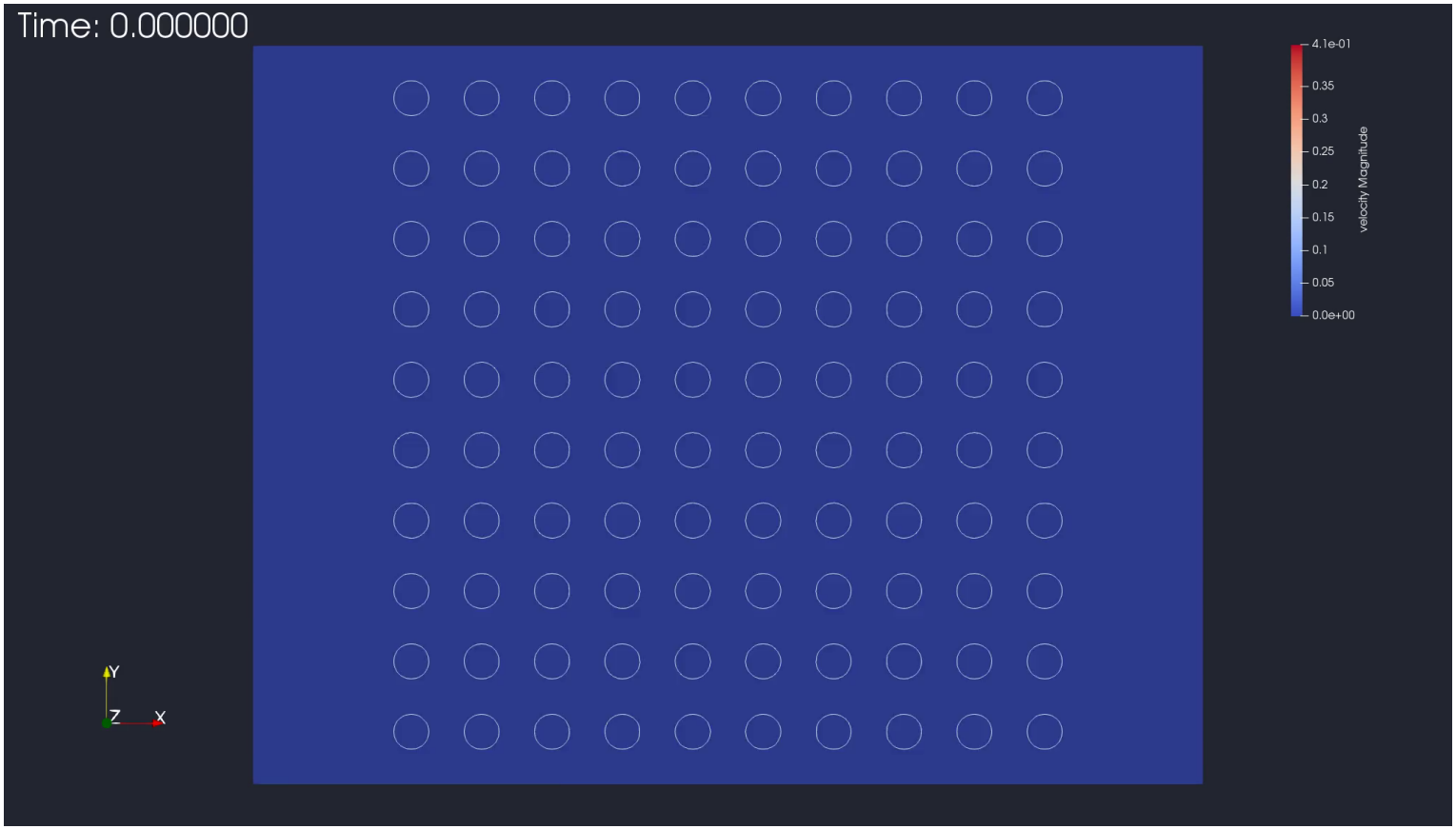
BC: no-slip + open top

Result

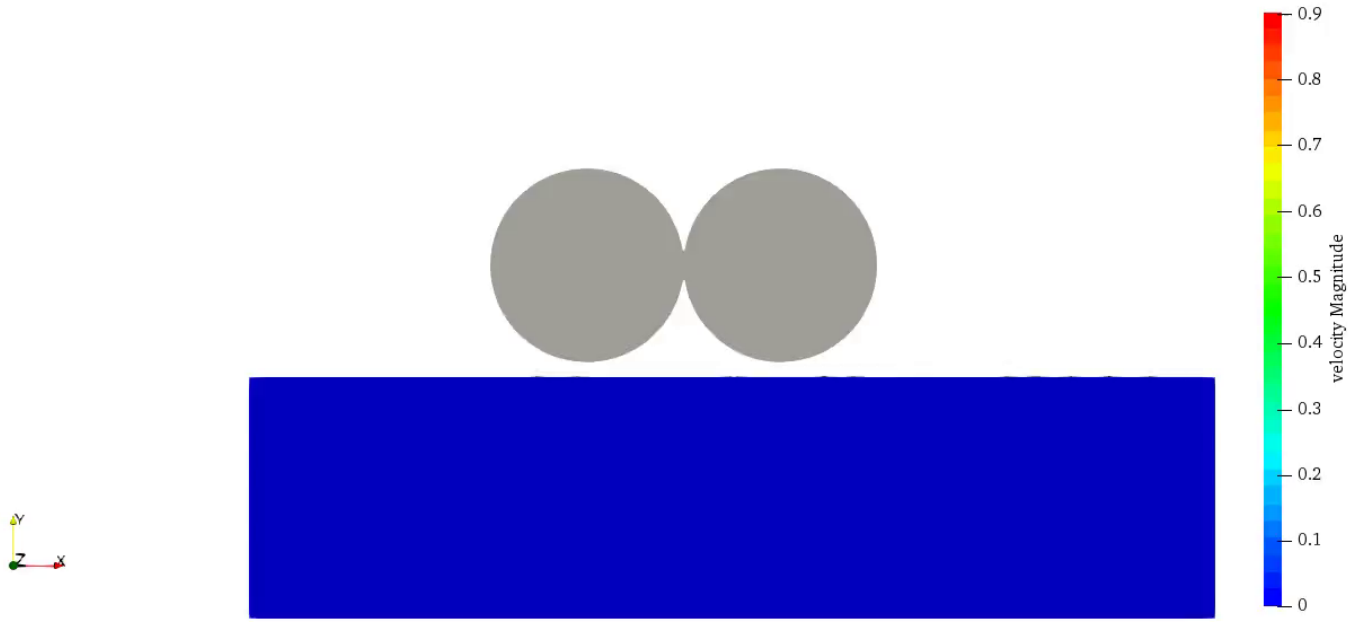


- $h = 1/32$, structured
- $\Delta t = 0.0005$
- $\mathbb{P}_2 - \mathbb{P}_1$

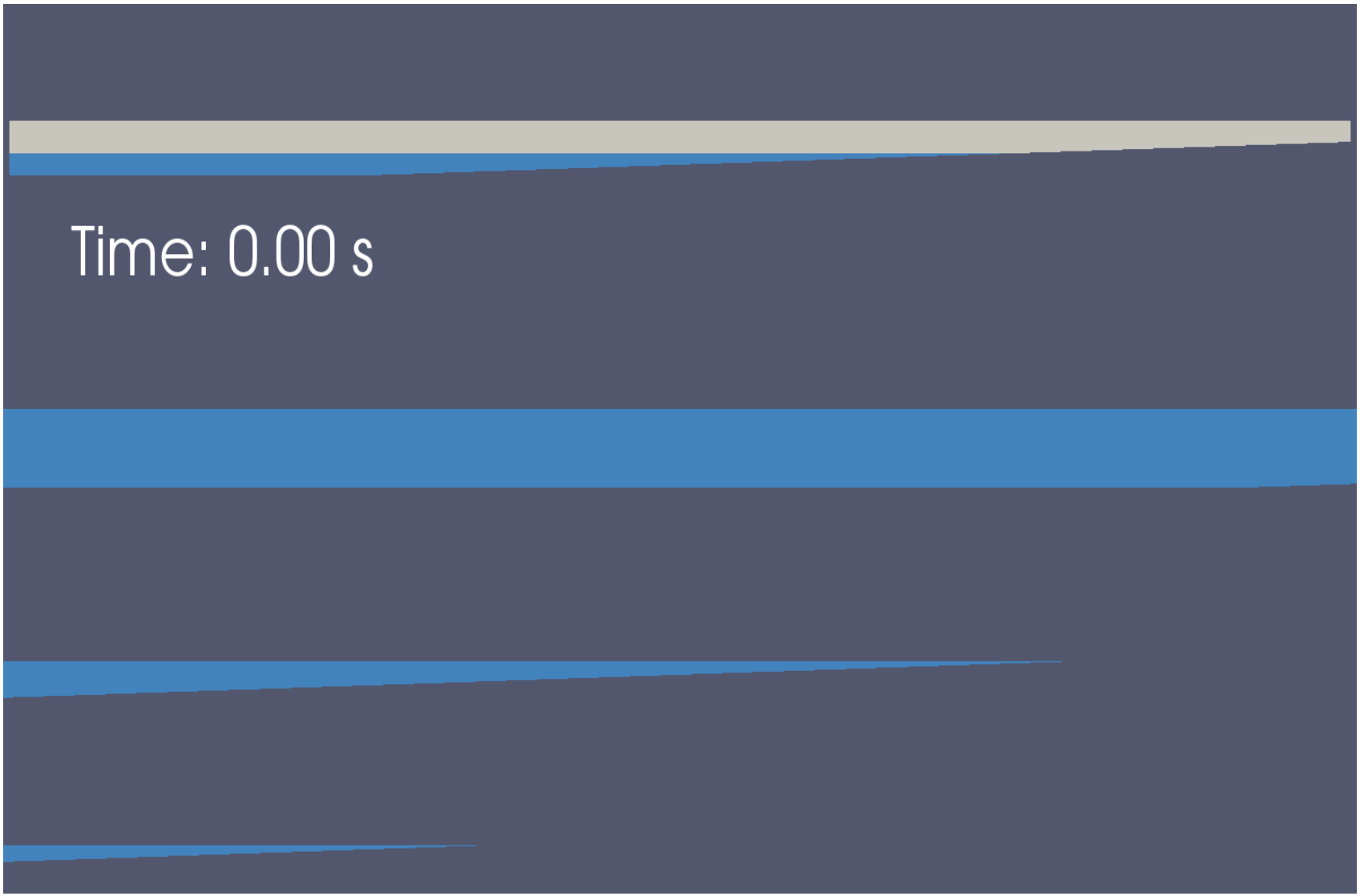
Settling


Time: 0.000000
4.1e-01
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0.0e+00
velocity Magnitude
Y
Z
X

Falling Spheres

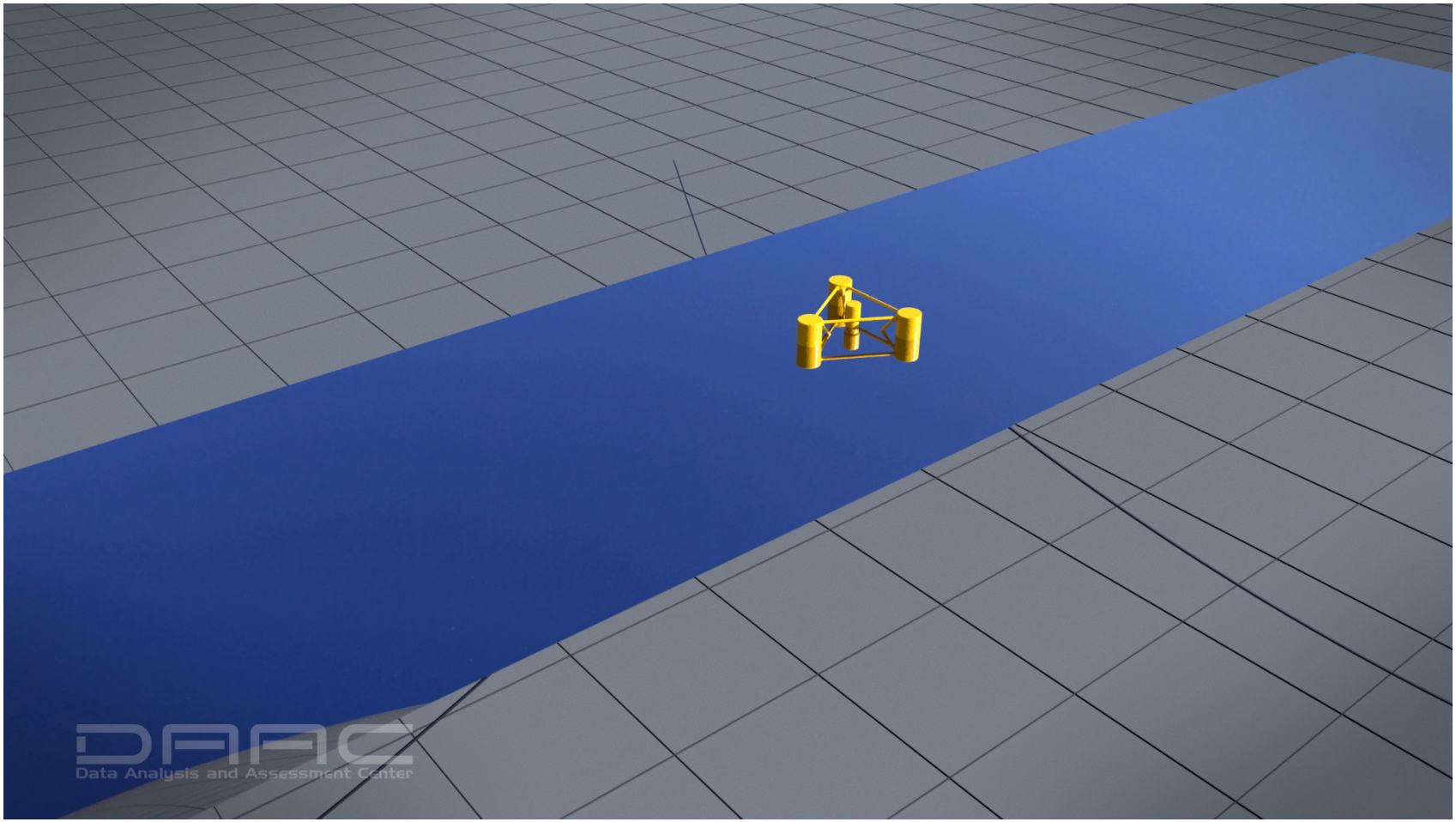


Ting and Kirby Experiment



COASTAL & HYDRAULICS LABORATORY

Computational Experiment: Offshore Platform



COASTAL & HYDRAULICS LABORATORY

Comparisons: Offshore Platform

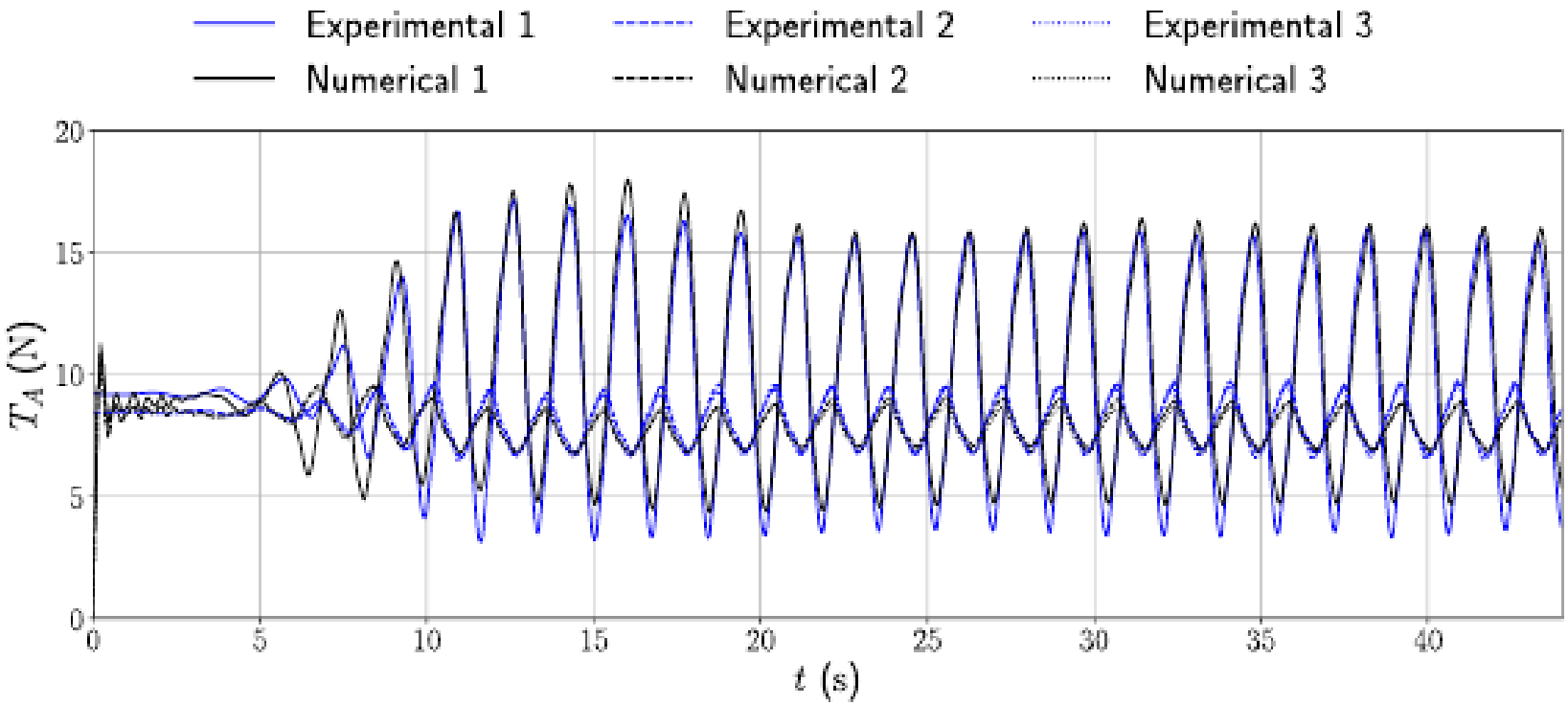


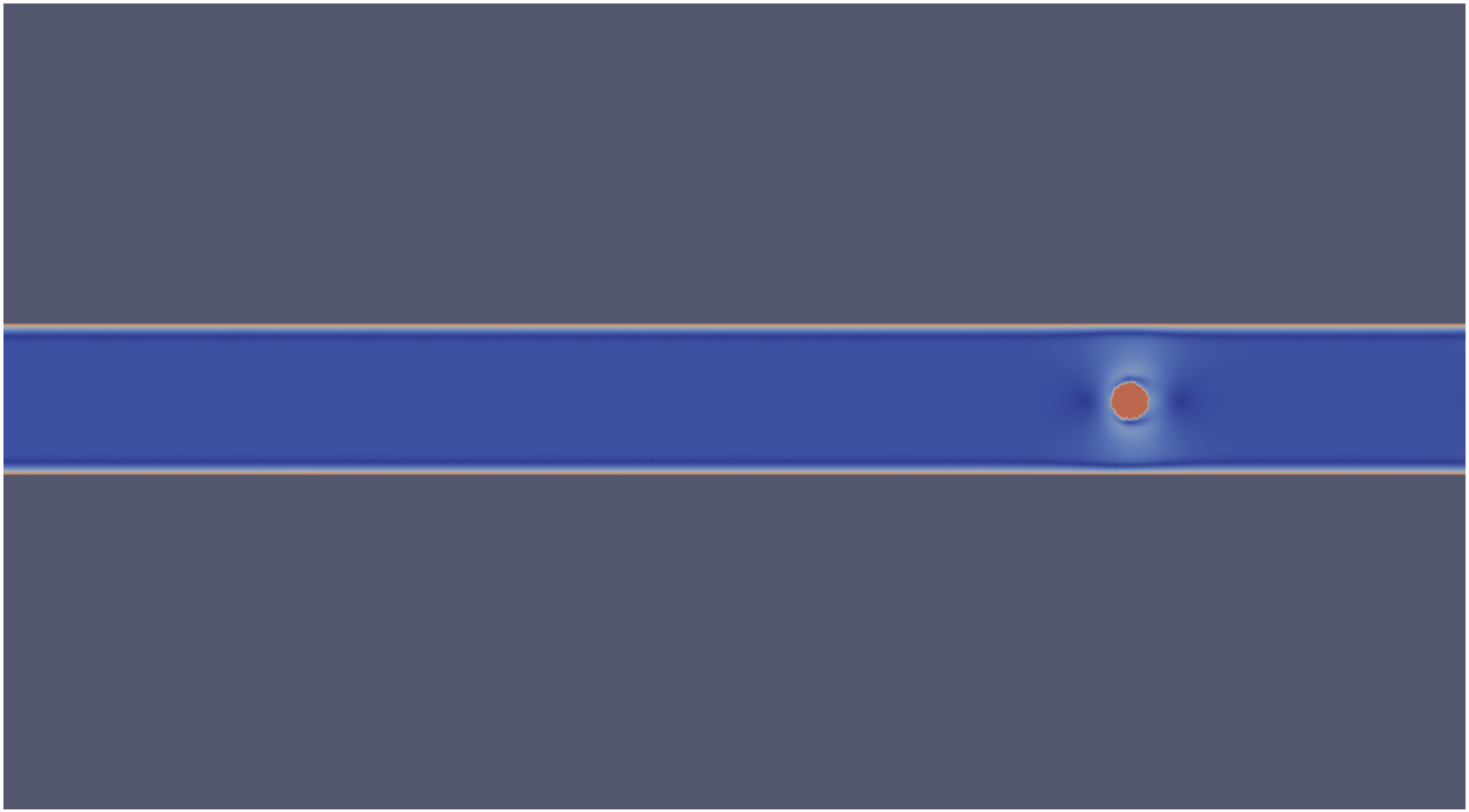
Figure V.39: Tensions at fairleads for coupled simulation of DeepCwind semi-submersible

Issues

- Strong mesh sensitivity of critical integral quantities, such as the drag and torque on particles.
- Difficulty resolving thin jets and related shear layers.
- Inability to preserve certain equilibrium states, such as flat water surface at hydrostatic pressure and zero velocity (lake at rest).

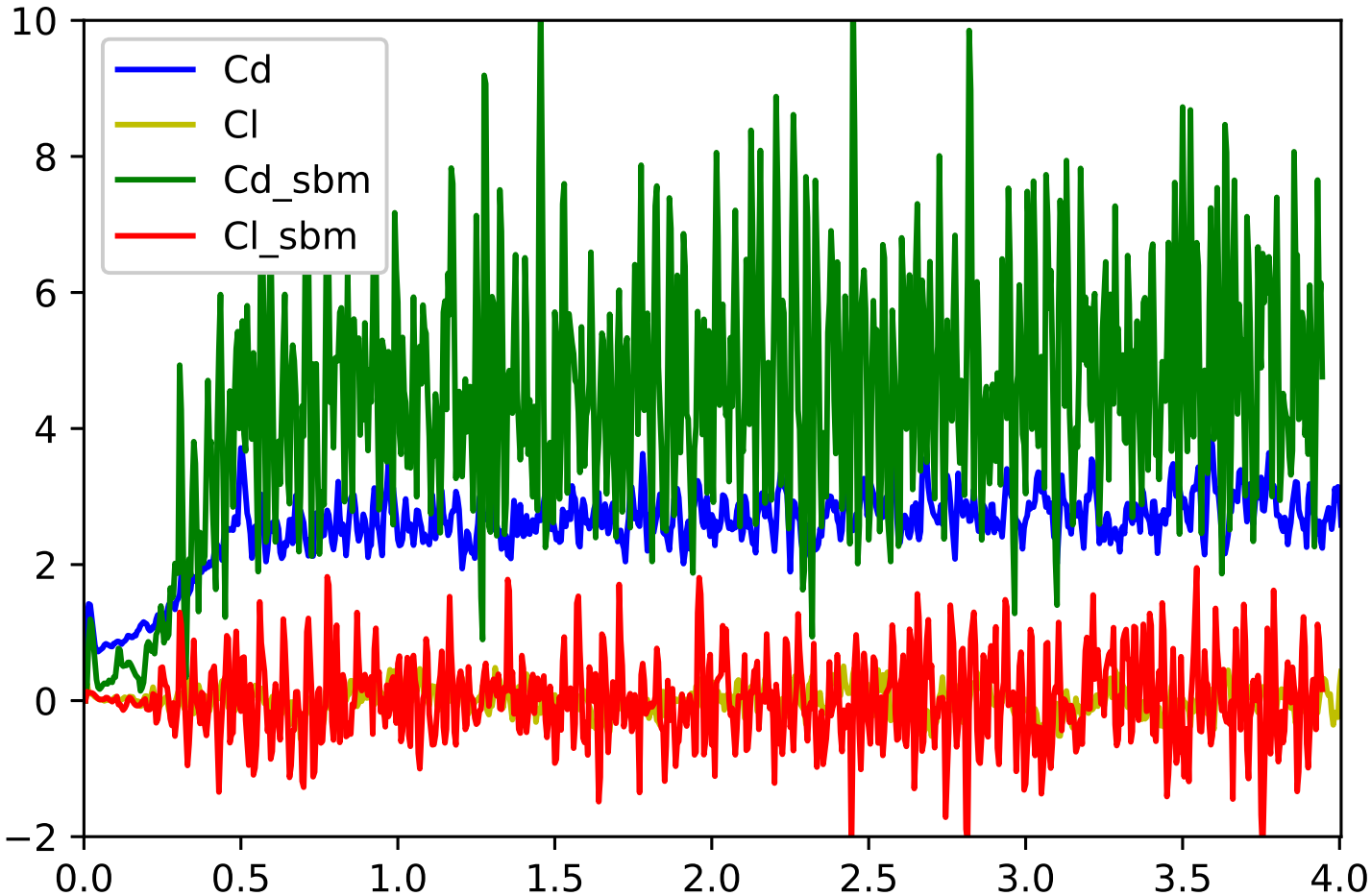
Hypothesis: These issues arise from the use of regularized Heaviside and Dirac functions to approximate integrals in cells cut by interfacial boundaries, leading to poor enforcement of interfacial jump conditions.
Approach: Integrate cut cells exactly and enforce proper jump conditions at interfaces.

Scripted Cylinder Test



COASTAL & HYDRAULICS LABORATORY

Force Oscillations for Moving Bodies



Solid Surface Area Oscillations

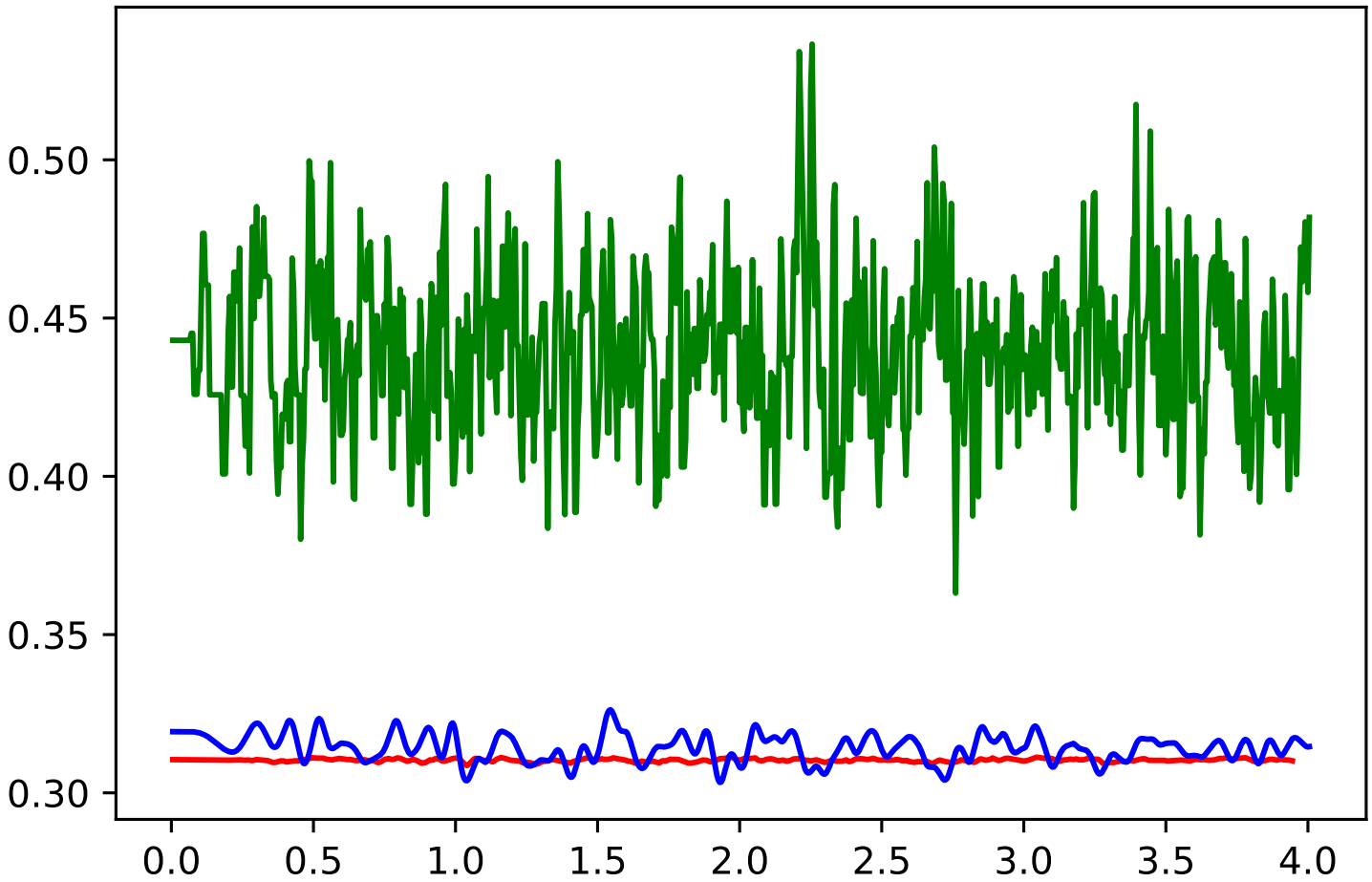
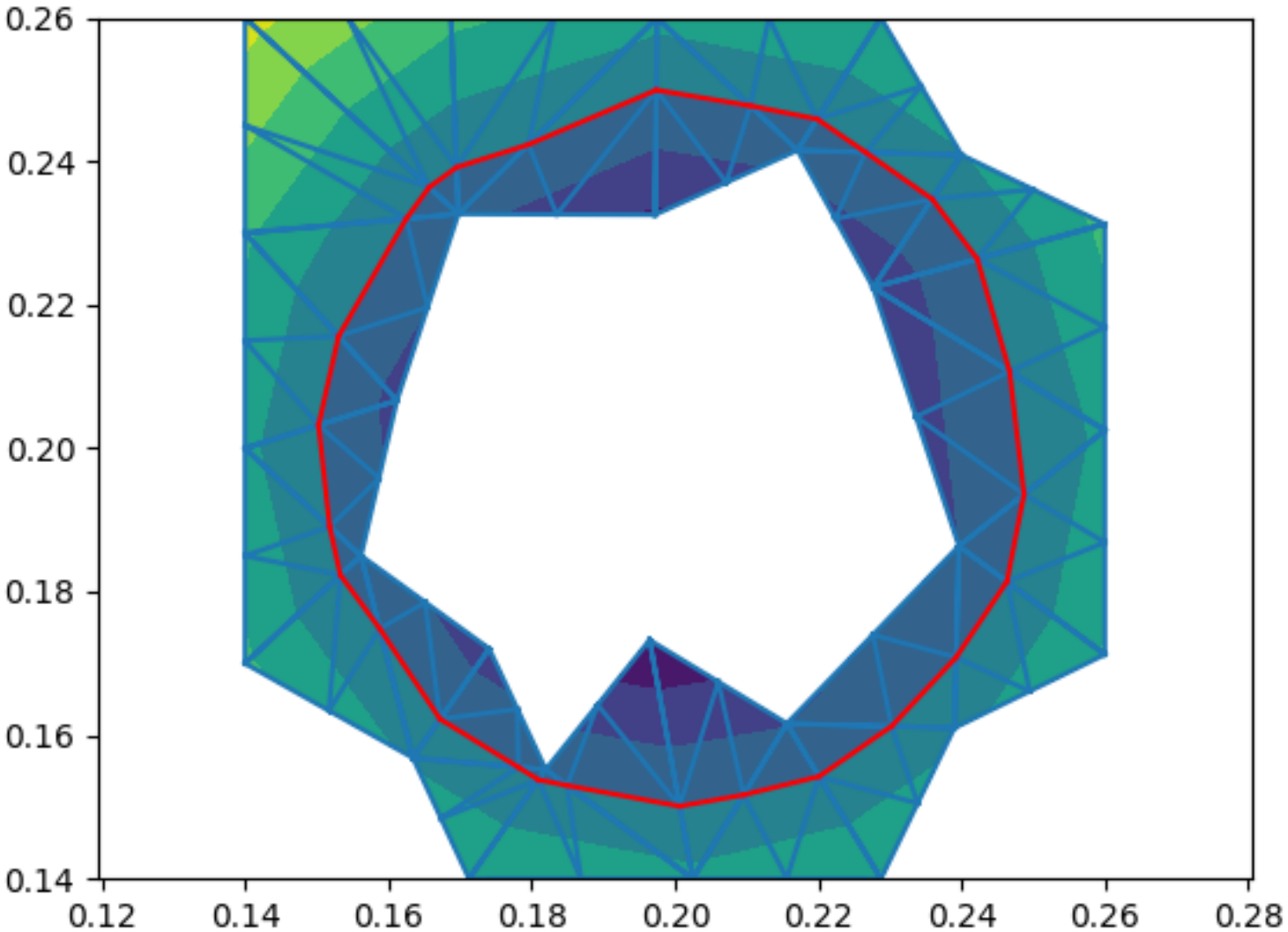


Figure: Green=SBM, Blue=IBM, Red=Cut

COASTAL & HYDRAULICS LABORATORY

Cut Cell Integration



COASTAL & HYDRAULICS LABORATORY

Three Phase Formulation with Jump Conditions

$$\nabla \cdot \mathbf{u}_a = 0, \quad \text{in } \Omega_a \tag{1.6a}$$

$$\rho_a(\partial_t \mathbf{u}_a + (\mathbf{u}_a \cdot \nabla)\mathbf{u}_a) - \nabla \cdot \sigma_a = \rho_a \mathbf{g}, \quad \text{in } \Omega_a \tag{1.6b}$$

$$\nabla \cdot \mathbf{u}_w = 0, \quad \text{in } \Omega_w \tag{1.6c}$$

$$\rho_w(\partial_t \mathbf{u}_w + (\mathbf{u}_w \cdot \nabla)\mathbf{u}_w) - \nabla \cdot \sigma_w = \rho_w \mathbf{g}, \quad \text{in } \Omega_w \tag{1.6d}$$

$$\mathbf{u}_a - \mathbf{u}_w = 0 \quad \text{on } \Gamma_f \tag{1.6e}$$

$$\sigma_a \cdot \mathbf{n}_f - \sigma_w \cdot \mathbf{n}_f = \gamma \kappa \mathbf{n}_f \quad \text{on } \Gamma_f \tag{1.6f}$$

$$\sigma_a \cdot \mathbf{t}_{1,f} - \sigma_w \cdot \mathbf{t}_{1,f} = 0 \quad \text{on } \Gamma_f \tag{1.6g}$$

$$\sigma_a \cdot \mathbf{t}_{2,f} - \sigma_w \cdot \mathbf{t}_{2,f} = 0 \quad \text{on } \Gamma_f \tag{1.6h}$$

$$\mathbf{u} - \mathbf{u}_s = \mathbf{u}_c \quad \text{on } \Gamma_s \tag{1.6i}$$

$$\mathbf{u} \cdot \mathbf{n} = 0 \quad \text{on } \Gamma \tag{1.6j}$$

where

$$\sigma = -pI + 2\mu\varepsilon(\mathbf{u}) \tag{1.7}$$

COASTAL & HYDRAULICS LABORATORY

Three-Phase Level Set Formulation

$$\int_{\Omega} H(\phi_s)\nabla\mathbf{v}\cdot\nabla\mathbf{w}dV = 0 \tag{1.8}$$

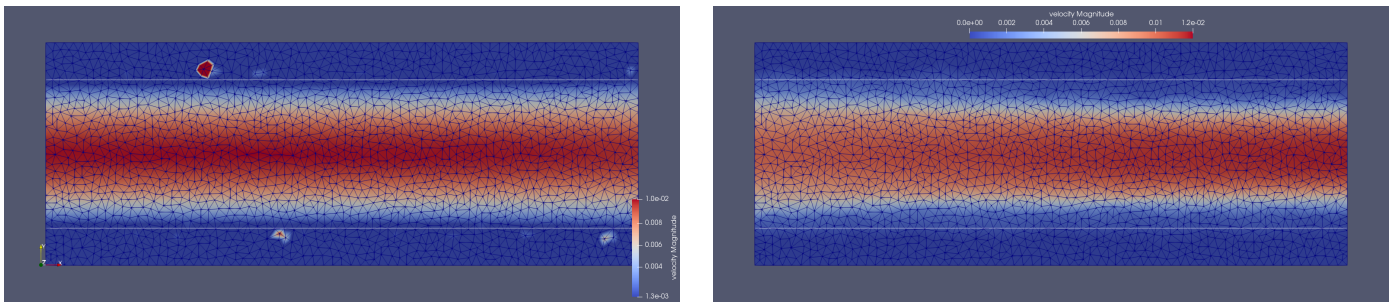
$$\int_{\Omega} H(\phi_s)\left\{\left[\rho\left(\frac{\partial\mathbf{v}}{\partial t}+\mathbf{v}\cdot\nabla\mathbf{v}\right)+\nabla p-\rho\mathbf{g}\right]\mathbf{w}+2\mu_f\varepsilon(\mathbf{v})\cdot\mathbf{w}\right\}dV$$
$$+\int_{\Omega}\delta(\phi_s)\left[-\mu(\phi)\frac{\partial\mathbf{v}}{\partial\mathbf{n}}\mathbf{w}-\mu(\phi)\frac{\partial\mathbf{w}}{\partial\mathbf{n}}\mathbf{v}+C(h_e)(\mathbf{v}-\mathbf{v}_s)\mathbf{w}\right]dV = 0 \tag{1.9}$$

CutFEM and Ghost Penalty

The coercivity of the problem needs to be extended to the computational domain. One approach is a an edge-based "Ghost Penalty"(Burman 2010).

$$j(u_h, v_h) := \sum_{F \in \mathcal{F}_G} \int_F [\gamma(\nabla u^+ - \nabla u^-) \cdot \mathbf{n}][(\nabla w^+ - \nabla w^-) \cdot \mathbf{n})] dS \tag{1.10}$$

where $\mathcal{F}_G$ is the set of internal element boundaries of cut cells.



CutFEM without ghost penalty (left) and with ghost penalty (right). See Massing, Schott, and Wall 2017 for Oseen/CIP.

Immersed Interface Finite Element Method (IIFEM)

- For two-fluid immersed interface problems, Leveque and Li developed a second order accurate finite difference scheme for by enforcing jump conditions for the solution and normal derivatives across the interface (LeVeque and Li 1994,1997)
- This approach was extended to FEM in the Immersed Interface Finite Element Method (IIFEM), by constructing a local basis with the desired jump conditions (Li et al. 2003, Ji et al. 2014)
- A key insight is that the algebraic system that results from doubling the number of degrees of freedom (as in CutFEM) leads to a solvable algebraic system for exactly half the duplicated degrees of freedom.

IIFEM Basis

Consider and elliptic immersed boundary problem with homogeneous jump conditions.

$$[u] = 0 \qquad [\mu \nabla u \cdot \mathbf{n}_f] = 0 \qquad \text{on } \Gamma_f \tag{1.11}$$

Consider one dimension where the cut element K is $(x_0, x_1) = (0, 1)$, $\Gamma_f = x_c \in (0, 1)$, and $\mathbf{n_f} = 1$. Define a new basis, ψ_0, ψ_1 on K by

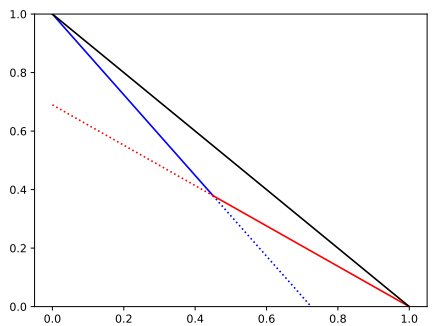
$$\psi_i = \begin{cases} \psi_i^a = a_1 + a_2 x & x \leq x_c \\ \psi_i^w = b_1 + b_2 x & x > x_c \end{cases} \tag{1.12}$$

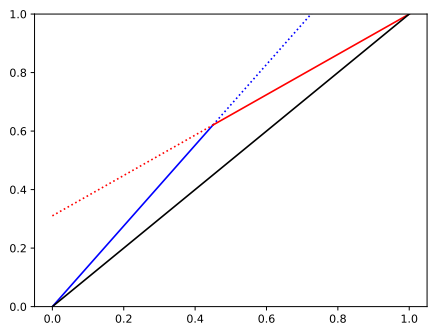
$$\psi_i^a(x_0) = \delta_{i0} \tag{1.13}$$

$$\psi_i^b(x_1) = \delta_{i1} \tag{1.14}$$

$$\psi_i^a(x_c) = \psi_i^b(x_c) \tag{1.15}$$

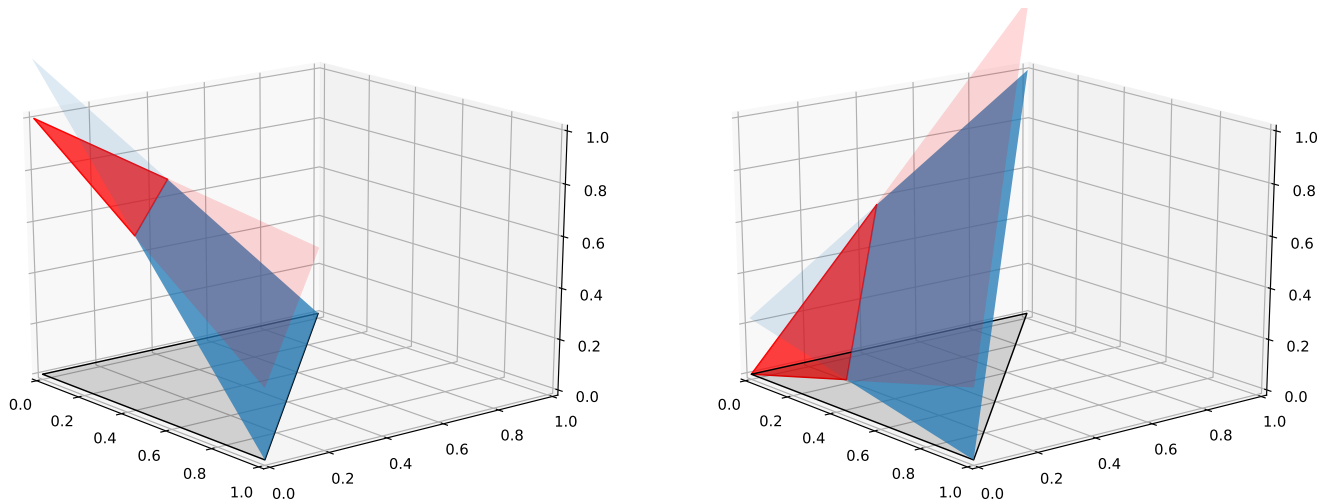
$$\mu_a \nabla \psi_i^b \cdot \mathbf{n}_f = \mu_w \nabla \psi_i^b \cdot \mathbf{n}_f \tag{1.16}$$





COASTAL & HYDRAULICS LABORATORY

2D Basis functions



IIFEM basis functions (solid red/blue) for $\mu_w/\mu_a = 2$.

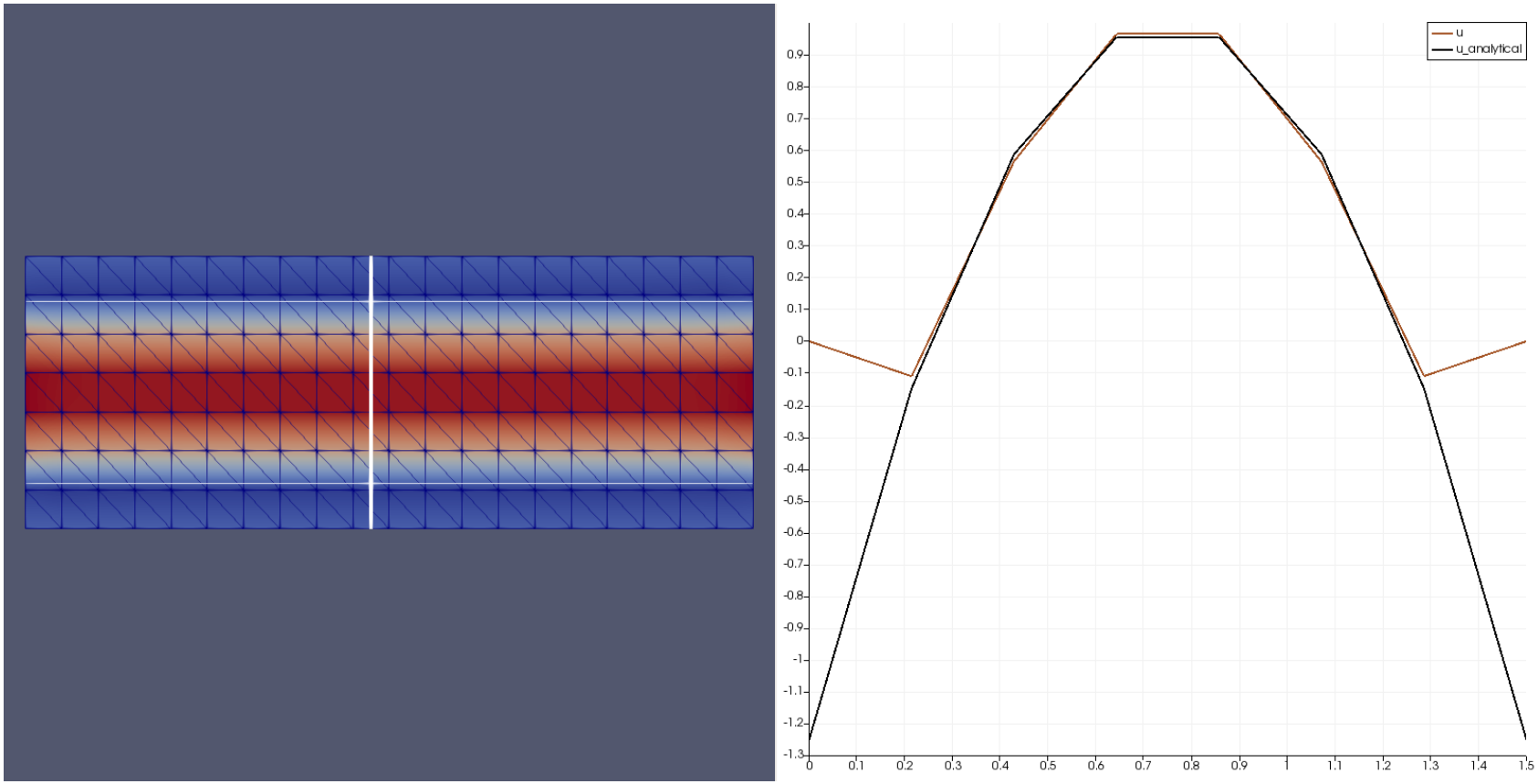
Cut Cell Integration by Equivalent Polynomials

- CutFEM and IIFEM introduce cut cells, which are challenge to integrate.
- Ventura and Bienvenuti 2015 introduced and approach based on generating “equivalent polynomial” Heaviside functions $\hat{H}$:

$$\int_K \hat{H} v_i = \int_K H v_i \quad i = 1, \ldots, \dim(P^m(K)) \tag{1.17}$$

  where $\hat{H}$ is also a polynomial of degree m.
- The FEM assembly then can proceed as if there were no cut cells as long as quadrature is sufficiently accurate.

CutFEM+Equivalent Polynomials: Duct



CutFEM + Equivalent Polynomials: Duct

Boundary Fitted Mesh

$\|(p-p^*)\|_1$	rate	$\|(u-u^*)\|_1$	rate
2.227590e-01	-	1.363730e-01	-
6.441190e-02	1.79	3.134070e-02	2.12
1.763730e-02	1.87	7.534600e-03	2.06
4.577860e-03	1.95	1.846190e-03	2.03
1.166740e-03	1.97	4.568420e-04	2.01
$\|(p-p^*)\|_2$	rate	$\|(u-u^*)\|_2$	rate
3.653670e-01	-	7.759540e-02	-
1.414790e-01	1.37	1.734230e-02	2.16
4.869300e-02	1.54	4.120990e-03	2.07
1.673540e-02	1.54	1.004600e-03	2.04
5.808630e-03	1.53	2.479960e-04	2.02
$\|(p-p^*)\|_\infty$	rate	$\|(u-u^*)\|_\infty$	rate
1.265980e+00	-	5.876160e-02	-
6.661940e-01	0.93	1.242080e-02	2.24
3.307870e-01	1.01	2.927460e-03	2.09
1.639120e-01	1.01	7.119360e-04	2.04
8.155440e-02	1.01	1.755150e-04	2.02

Unfitted Mesh

$\|(p-p^*)\|_1$	rate	$\|(u-u^*)\|_1$	rate
4.889190e-01	-	1.185600e-01	-
1.302270e-01	1.91	3.219830e-02	1.88
2.783510e-02	2.23	7.328950e-03	2.14
7.202650e-03	1.95	1.811790e-03	2.02
1.840000e-03	1.97	4.503920e-04	2.01
$\|(p-p^*)\|_2$	rate	$\|(u-u^*)\|_2$	rate
4.405330e-01	-	6.659730e-02	-
2.406210e-01	0.87	1.774190e-02	1.91
6.604950e-02	1.87	4.018790e-03	2.14
2.304690e-02	1.52	9.873070e-04	2.03
8.097610e-03	1.51	2.447100e-04	2.01
$\|(p-p^*)\|_\infty$	rate	$\|(u-u^*)\|_\infty$	rate
1.120820e+00	-	5.373470e-02	-
1.353560e+00	-0.27	1.586260e-02	1.76
3.435390e-01	1.98	3.038940e-03	2.38
1.786160e-01	0.94	7.363010e-04	2.05
9.701790e-02	0.88	1.817440e-04	2.02

COASTAL & HYDRAULICS LABORATORY

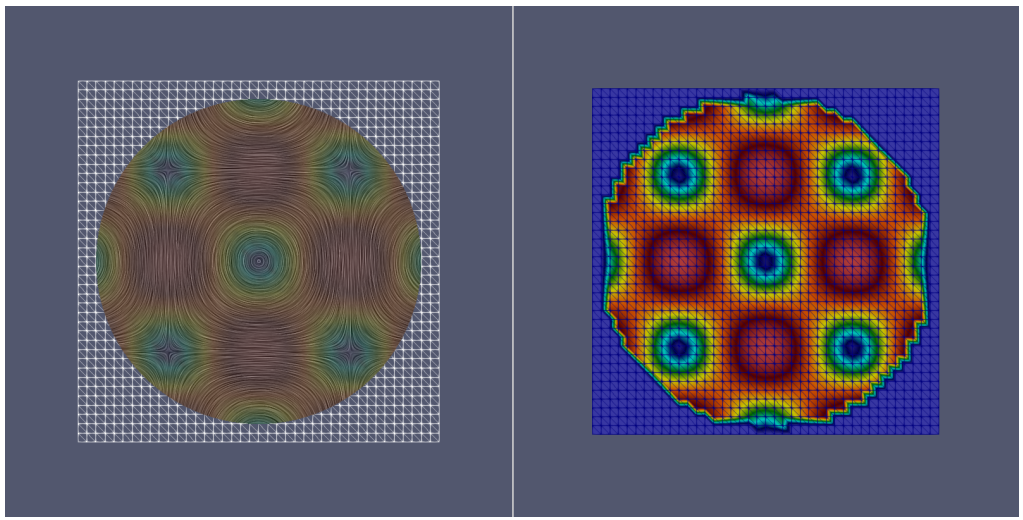
CutFEM + Equivalent Polynomials: Taylor

Boundary Fitted Mesh

$\|(p-p^*)\|_1$	rate	$\|(u-u^*)\|_1$	rate
1.629910e-01	-	4.533020e-03	-
6.031340e-02	1.43	1.693180e-03	1.42
1.619510e-02	1.90	4.778660e-04	1.83
4.442810e-03	1.87	1.332800e-04	1.84
$\|(p-p^*)\|_2$	rate	$\|(u-u^*)\|_2$	rate
5.041250e-01	-	7.002600e-03	-
2.233220e-01	1.17	2.642520e-03	1.41
7.756350e-02	1.53	7.476140e-04	1.82
2.782230e-02	1.48	2.086930e-04	1.84
$\|(p-p^*)\|_\infty$	rate	$\|(u-u^*)\|_\infty$	rate
3.546620e+00	-	2.181050e-02	-
2.054580e+00	0.79	8.358510e-03	1.38
1.010460e+00	1.02	2.662330e-03	1.65
5.080040e-01	0.99	7.365010e-04	1.85

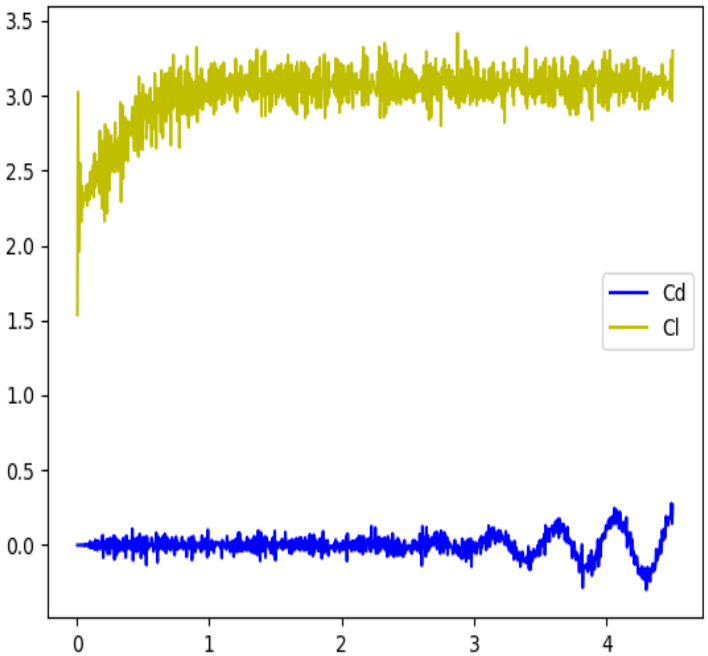
Unfitted Mesh

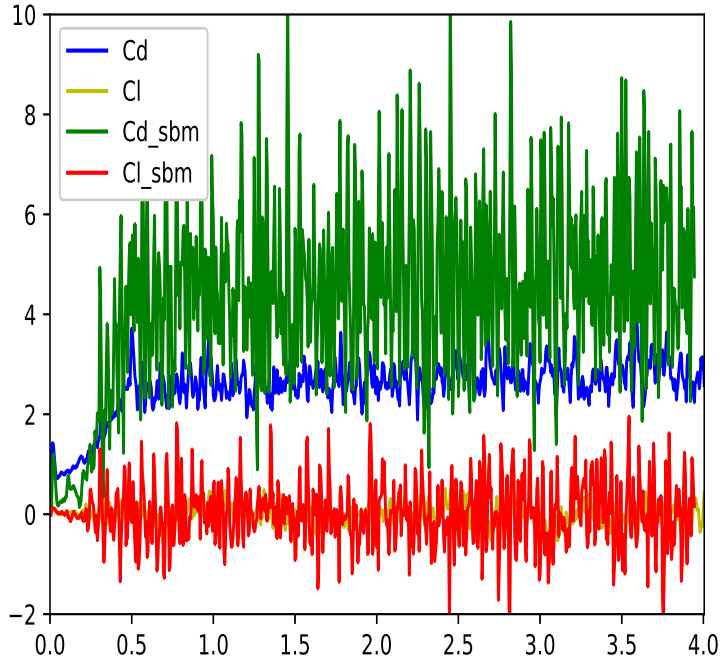
$\|(p-p^*)\|_1$	rate	$\|(u-u^*)\|_1$	rate
4.650280e-01	-	1.605080e-02	-
1.864090e-01	1.32	4.886610e-03	1.72
5.555430e-02	1.75	1.363970e-03	1.84
1.479990e-02	1.91	3.573620e-04	1.93
$\|(p-p^*)\|_2$	rate	$\|(u-u^*)\|_2$	rate
9.362260e-01	-	2.542980e-02	-
4.430540e-01	1.08	7.362900e-03	1.79
1.672410e-01	1.41	2.016140e-03	1.87
5.911910e-02	1.50	5.260030e-04	1.94
$\|(p-p^*)\|_\infty$	rate	$\|(u-u^*)\|_\infty$	rate
5.842120e+00	-	7.515420e-02	-
3.669590e+00	0.67	2.067450e-02	1.86
1.910150e+00	0.94	5.335560e-03	1.95
9.799620e-01	0.96	1.349950e-03	1.98



COASTAL & HYDRAULICS LABORATORY

CutFEM + Equivalent Polynomials: Falling Cylider





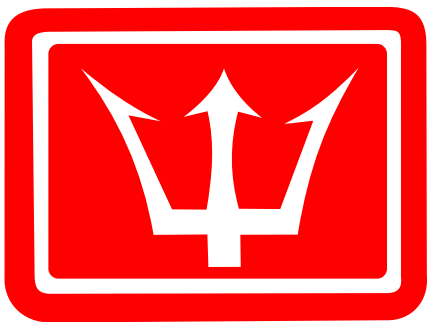
Future Work

- Use a more careful discretization of the fluid material derivative near the moving solid interface or space-time FEM.
- Compare to the aggregated unfitted approach of Badia, Verdugo, and Martin (2018) and investigate EQP-based implementation.
- Complete the IIFEM implementation for air/water and investigate whether it removes various numerical artifacts.
- Deal with three-phase cutcells and contact line boundary conditions.

Software Challenges

- Bridge “valley of death” between new advancements in model formulations/numerics and production codes/applications.
- Break out of the “model trap” in which R&D is stovepiped around a single set of equations or single method.
- Explicit multi-physics and multi-numerics support.

“Good numerics is ultimately about getting things to work; the slavish and blind devotion to one approach above all others is usually a sign of unfamiliarity with the range of troubles and challenges presented by real applications.” -James Sethian



Proteus Toolkit
Source code
https://github.com/erdc/proteus

COASTAL & HYDRAULICS LABORATORY

What is Proteus?

- Proteus is a Python package for rapidly developing computer models and numerical methods.
- The package contains a collection of modules implemented in C,C++,Fortran, and Python.
- We try to separate the physics implementation and numerical methods implementation.
- Has a layered API for model implementation. Highly optimized models for specific/detailed physics can be implemented by deriving from more generic models.
- Works with many 3rd party tools: PETSc, Trilinos, Chrono, triangle, tetgen, gmsh, scipy, sympy...

Equations Solved

- 2D & 3D incompressible Navier-Stokes (Unsteady/Steady, LES, RANS, VANS)
- 2D diffusive wave (overland flow)
- 2D shallow water and Green-Naghdi (dispersive shallow water)
- 2D & 3D two-phase incompressible, immiscible flow (hybrid VOF/level set formulation with LES, etc.)
- 2D & 3D saturated groundwater
- 2D & 3D Richards' equation (variably saturated groundwater, various constitutive models)
- 2D & 3D two-phase flow in porous media (continuum mixture formulation, incompressible or compressible)
- 2D & 3D density-dependent groundwater flow and salinity transport
- 2D & 3D eikonal equation (signed distance calculations)
- 3D elastoplastic deformation (levee stability, Mohr-Coulomb material)
- 2D & 3D 6DOF solid/air/water interaction

Verified Numerics

- Continuous linear and quadratic polynomial spaces (C^0P^1 and C^0P^2) on simplicial elements (intervals, triangles, tetrahedra) with nodal (Lagrange) basis and Bernstein basis.
- Continuous tensor product spaces (C^0Q^k) on hex/quad
- Discontinuous complete polynomial spaces ($C^{-1}P^k$) on simplicial elements with monomial basis
- P^1 non-conforming simplicial elements
- Locally discontinuous Galerkin mixed elements with static condensation
- SIPG/NIPG/IIPG primal discontinuous elements
- Residual-based variational multiscale methods (RBVMS)
- Entropy viscosity, algebraic stabilization, and FCT
- Analytical Riemann solvers (numerical fluxes) for linear advection, two-phase flow in porous media, and shallow water
- Approximate Riemann solvers: Harten-Lax-van Leer (SWE), Rusanov (two-phase flow), Cheng-Shu (Hamilton-Jacobi)
- Velocity post-processing to enforce element-wise (local) conservation

Popular Modeling Toolkit for PDE's

- "The COMSOL multiphysics simulation environment facilitates all steps in the modeling process: defining your geometry, specifying your physics, meshing, solving and then post-processing your results."
- "FEniCS is free software for automated solution of differential equations. We provide software tools for working with computational meshes, finite element variational formulations of PDEs, ODE solvers and linear algebra."
- "...OpenFOAM is a flexible set of efficient C++ modules. These are used to build a wealth of: solvers, to simulate specific problems in engineering mechanics; utilities, to perform pre- and post-processing tasks ranging from simple data manipulations to visualization and mesh processing; libraries, to create toolboxes that are accessible to the solvers/utilities, such as libraries of physical models."

Second Order Nonlinear, Heterogeneous Transport Systems

Our target problems are systems of nonlinear equations governing the transport of an abstract vector of components $u_j, j = 1, \ldots, n_c$:

$$\frac{\partial m^i}{\partial t} + \nabla \cdot \left(\mathbf{f}^i - \sum_k^{n_c} \boldsymbol{a}^{ik} \nabla \phi^k \right) + r^i + h^i(\nabla u) = 0$$

where $i = 1, \ldots n_c$. The large majority of models in hydrology, hydraulics, and coastal engineering are in this class.

A simple example

For $(t, x, y) \in [0, T] \times [0, 1] \times [0, 1]$ find u such that

$$
\begin{aligned}
(Mu)_t + \nabla \cdot [\mathbf{B}u - \boldsymbol{A}\nabla u] &= 0 \\
u(0, x, y) &= 0 \\
u(t, x, 0) &= u(t, 0, y) = 1 \\
u(t, x, 1) &= u(t, 1, x) = 0 \\
M &= 1 \\
\mathbf{B} &= (1, 1) \\
\boldsymbol{A} &= 0.001\boldsymbol{I}
\end{aligned}
$$

adr.py

```
20      def evaluate(self,t,c):
21          c[('m',0)][:]           = self.M*c[('u',0)]
22          c[('dm',0,0)][:]        = self.M
23          c[('f',0)][...,0]       = self.B[0]*c[('u',0)]
24          c[('f',0)][...,1]       = self.B[1]*c[('u',0)]
25          c[('df',0,0)][...,0]    = self.B[0]
26          c[('df',0,0)][...,1]    = self.B[1]
27          c[('a',0,0)][...,0] = self.A[0][0]
28          c[('a',0,0)][...,3] = self.A[1][1]
```

adr.py

```
3  class LAD(TC_base):
4      """
5      Coefficients for linear advection-diffusion
6      """
7      def __init__(self,M,A,B):
8          TC_base.__init__(self,
9                           nc=1, #number of components
10                          variableNames=['u'],
11                          mass      = {0:{0:'linear'}},
12                          advection = {0:{0:'linear'}},
13                          diffusion = {0:{0:{0:'constant'}}},
14                          potential = {0:{0:'u'}},
15                          reaction  = {0:{0:'linear'}})
16         self.M=M;
17         self.A=A;
18         self.B=B;
19
20     def evaluate(self,t,c):
```

ladr_2d_p.py

```
6   nd = 2; #Two dimensions
7   L=(1.0,1.0,1.0);
8   T=1.0
9
10  coefficients=LAD(M=1.0,
11                   A=[[0.001,0.0],
12                      [0.0,0.001]],
13                   B=[2.0,1.0])
14
15  def getDBC(x,flag):
16      if x[0] == 0.0 or x[1] == 0.0:
17          return lambda x,t: 1.0
18      elif x[0] == 1.0 or x[1] == 1.0:
19          return lambda x,t: 0.0
20
21  dirichletConditions = {0:getDBC}
22  advectiveFluxBoundaryConditions = {}
23  diffusiveFluxBoundaryConditions = {0:{}}
```

ladr_2d_c0p2_n.py

```
4  timeIntegration = BackwardEuler
5  femSpaces = {0:C0_AffineQuadraticOnSimplexWithNodalBasis}
6  elementQuadrature = SimplexGaussQuadrature(nd,5)
7  elementBoundaryQuadrature = SimplexGaussQuadrature(nd-1,5)
8  nnx=11; nny=11
9  nLevels=1
10 tnList=[float(i)/10.0 for i in range(11)]
11 matrix = SparseMatrix
12 multilevelLinearSolver = LU
13 subgridError = AdvectionDiffusionReaction_ASGS(coefficients,
14                                                nd)
15 shockCapturing = ResGradQuad_SC(coefficients,
16                                  nd,
17                                  shockCapturingFactor=0.9,
18                                  lag=True)
```

A multi-physics example

```
1  pnList = [("twp_navier_stokes_sloshbox_2d_p" ,
2             "twp_navier_stokes_sloshbox_2d_c0p1c0p1_n"),
3            ("ls_sloshbox_2d_p" ,
4             "ls_sloshbox_2d_c0p1_n"),
5            ("vof_sloshbox_2d_p" ,
6             "vof_sloshbox_2d_c0p1_n"),
7            ("redist_sloshbox_2d_p" ,
8             "redist_sloshbox_2d_c0p1_n"),
9            ("ls_consrv_sloshbox_2d_p" ,
10            "ls_consrv_sloshbox_2d_c0p1_n")]
```

Domain Specific Languages

- DSL's have proven useful linear algebra (FLAME), and FEM (FEniCS, Firedrake)
- Representing PDE's in continuous, strong formulation as classes allows modular backends for fully discrete formulations (FEM, FVM, FDM,...)
- With a DSL, information can be inferred from a suitable symbolic expression (and potentially optimized) and the API code can be generated.
- As a proof of concept we developed a simpled Domain Specific Language embedded in Python based on pymbolic (with A. Klöckner, Illinois and R. Kirby, Baylor)
- Earlier attempts include an embedded LaTeX DSL [Farthing, Sassen, Prins, Miller, 2004]

DSL: Burgers

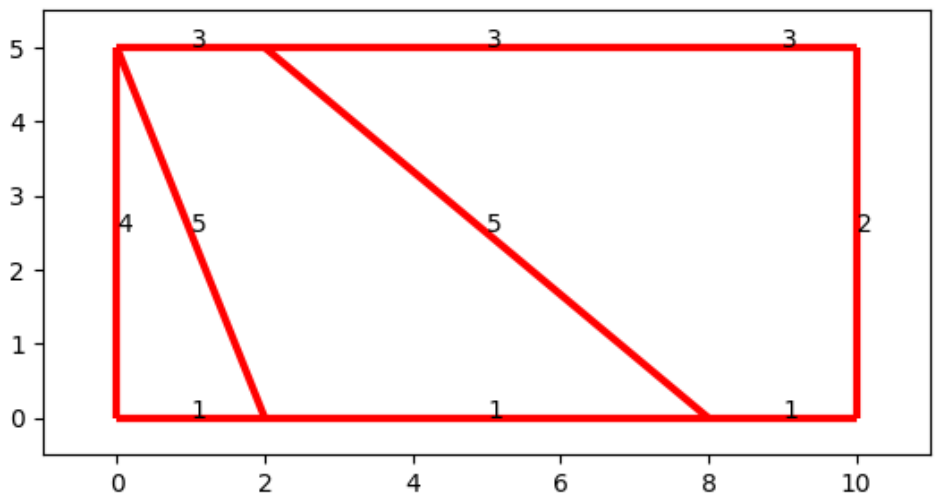
```
6
7   A = 1.0e-1
8   B = np.array([1.0, 2.0])
9   C = 1.0e-12
10
11  u = sym.Field("u")
12  eqns = sym.join(sym.d_dt(u)
13                  + sym.div(B * u**2)
14                  - sym.div(A*sym.grad(u))
15                  + C*u)
16
17  coefficients = generate_coefficients(eqns,
18                                       unknowns=[u],
19                                       dim=2,
20                                       name="Burgers")
21
```

PDE Domains

- Defining the PDE represents only a small fraction of the work for real applications.
- Precisely defining the domain geometry, material properties, boundary conditions, and obtaining good quality meshes is frequently a stumbling block–we aren't taught this in class.

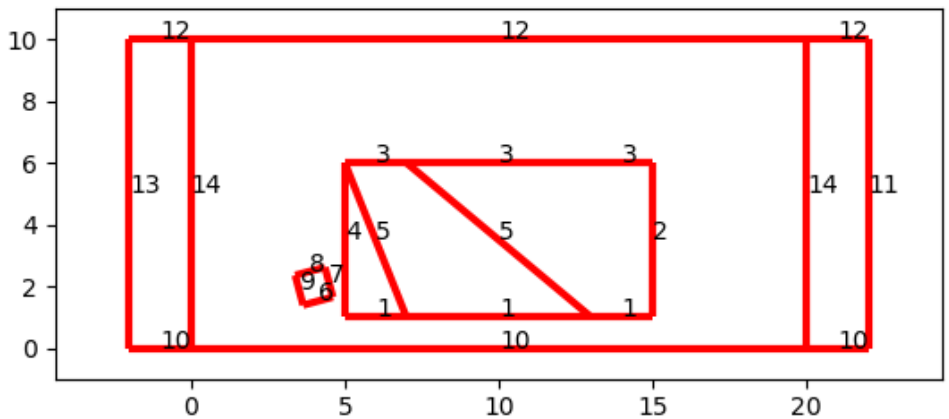
SpatialTools: Shapes

```
1   domain2D = Domain.PlanarStraightLineGraphDomain()
2
3   vertices = [[0., 0.], [2., 0.], [8., 0.], [10., 0.],
4               [10., 5.], [8., 5.], [2., 5.], [0., 5.]]
5   segments = [[0, 1], [1, 2], [2, 3], [3, 4],
6               [4, 5], [5, 6], [6, 7], [7, 0],
7               [1, 7], [2, 6]]
8
9   bt = {'bottom': 1, 'right': 2, 'top': 3, 'left': 4, 'sponge': 5}
10  vertexFlags = [bt['bottom'], bt['bottom'], bt['bottom'], bt['bottom'],
11                 bt['top'], bt['top'], bt['top'], bt['top'],
12                 bt['sponge'], bt['sponge']]
13  segmentFlags = [1, 1, 1, 2, 3, 3, 3, 4, 5, 5]
14  regions = [[1., 1.], [5., 1.], [9., 1.]]
15  regionFlags = [2, 1, 3]
16
17  custom = st.CustomShape(domain2D, boundaryTags=bt,
18                          vertices=vertices, vertexFlags=vertexFlags,
19                          segments=segments, segmentFlags=segmentFlags,
20                          regions=regions, regionFlags=regionFlags)
```



SpatialTools: Modification and Composition

```
22  custom.translate([5., 1.])
23
24  tank = st.Tank2D(domain2D, dim=[20., 10.])
25  tank.setSponge(x_n=2., x_p=2.)
26
27  rectangle = st.Rectangle(domain2D, dim=[1., 1.], coords=[2., 2.])
28  rectangle.rotate(np.pi/12., pivot=[2., 2.])
29  rectangle.setBarycenter(my_rectangle.coords)
30  rectangle.setPosition([4., 2.])
31  st.assembleDomain(domain2D)
```



Numerical Methods R&D

- Convervative level set schemes for FEM
  - Incompressible Two-phase Flow via a Monolithic Phase-Conservative Level Set Method (2019) M.Quezada de Luna, J.H. Collins, and C.E. Kees *International Journal for Numerical Methods in Fluids*, In Review, https://arxiv.org/abs/1903.06919.
  - A Monolithic Conservative Level Set Method with Built-In Redistancing (2019) M. Quezada de Luna, D. Kuzmin, C.E. Kees, *Journal of Computational Physics*, 379, 262-278, https://doi.org/10.1016/j.jcp.2018.11.044
  - A Conservative Level Set Method for Variable-Order Approximations and Unstructured Meshes (2011) C.E. Kees, I. Akkerman, Y. Bazilevs, and M.W. Farthing. *Journal of Computational Physics*, 230(12), 4536–4558, https://doi.org/10.1016/j.jcp.2011.02.030.
- Equivalent polynomials for exact Heaviside and Dirac integrals with IIFEM (In progress)
- Fluid-structure interaction with ALE, added mass stabilization, and $h-$adaptivity
  - High-Fidelity Computational Modelling of Fluid-Structure Interaction for Moored Floating Bodies (2019) T. de Lataillade *Doctoral Thesis, Edinburgh/Exeter/Strathcylde*
  - Dynamic, Anisotropic Adaptive Mesh Refinement for Fluid-Structure Interaction (2019) A. Zhang *Doctoral Thesis, RPI*.

COASTAL & HYDRAULICS LABORATORY

Numerical Methods R&D, Cont'd

- Entropy viscosity methods for subgrid/numerical viscosity
  - Robust and Explicit Relaxation Technique for Solving the Green-Naghdi Equations (2019) J.-L. Guermond, B. Popov, E. Tovar, and C.E. Kees *Journal of Computational Physics*, In Press.
  - Well-Balanced Second-Order Finite Element Approximation of the Shallow Water Equations with Friction (2018) J.L. Guermond, M.Q. de Luna, B. Popov, C.E. Kees, and M.W. Farthing *SIAM Journal on Scientific Computing* 40(6), A3873-A3901,https://doi.org/10.1137/17M1156162.
- Scalable preconditioners based on multi-phase extensions of the pressure convection-diffusions approximation of the Schur complement.
  - Preconditioners for Two-Phase Incompressible Navier-Stokes Flow (2019) N. Bootland, C.E. Kees, A. Wathen, A. Bentley *SIAM Journal on Scientific Computing*, 41(4), B843–B869, https://doi.org/10.1137/17M1153674.
- Model Reduction (POD and PR-RBC)
  - Evaluation of Galerkin and Petrov-Galerkin Model Reduction for Finite Element Approximations of the Shallow Water Equations (2017) A. Lozovsky, M. W. Farthing, and C.E. Kees *Computational Methods in Applied Mechanics and Engineering* 318, 537-571, https://doi.org/10.1016/j.cma.2017.01.027
  - POD-Based Model Reduction for Stabilized Finite Element Approximations of Shallow Water Flows (2016) A. Lozovskiy, M.W. Farthing, C.E. Kees, E. Gildin *Journal of Computational and Applied Mathematics*, 302, 50-70, https://doi.org/10.1016/j.cam.2016.01.029.

Modeling and Engineering Applications

- Waves and structures
  - Fast Random Wave Generation in Numerical Tanks (2019) A. Dimakopoulos, T. de Lataillade, C.E. Kees, *Engineering and Computational Mechanics*, *Proceedings of the Institution of Civil Engineers - Engineering and Computational Mechanics*, 1-29, https://doi.org/10.1680/jencm.17.00016.
  - Simulating Oscillatory and Sliding Displacements of Caisson Breakwaters Using a Coupled Approach (2019) G. Cozzuto, A. Dimakopoulos, T. de Lataillade, P.O. Morillas, and C.E. Kees, *Journal of Waterway, Port, Coastal, and Ocean Engineering*, 145(3), p.04019005, https://doi.org/10.1061/(ASCE)WW.1943-5460.0000504
  - Computational Model for Wave Attenuation by Flexible Vegetation (2018) S.A. Mattis, C.E. Kees, M.V. Wei, A. Dimakopoulos, and C.N. Dawson, *Journal of Waterway, Port, Coastal, and Ocean Engineering* 145(1), p.04018033, https://doi.org/10.1061/(ASCE)WW.1943-5460.0000487
  - An Immersed Structure Approach for Fluid-Vegetation Interaction (2015) S.A. Mattis, C.N. Dawson, C.E. Kees, M.W. Farthing, *Advances in Water Resources*, 80,1-16, https://doi.org/10.1016/j.advwatres.2015.02.014.
- Three-dimensional sediment processes
  - Modeling Sediment Transport in Three-Phase Surface Water Systems (2019) C.T. Miller, W.G. Gray, C.E. Kees, I.V. Rybak, B.J. Shepherd, *Journal of Hydraulic Engineering*, https://doi.org/10.1080/00221686.2019.1581673.
  - Thermodynamically Constrained Averaging Theory: Principles, Model Hierarchies, and Deviation Kinetic Energy Extensions (2018) C.T. Miller, W. G. Gray, and C. E. Kees *Entropy* 20(4), 253, https://doi.org/10.3390/e20040253
- Water resources
  - Finite Element Methods for Variable Density Flow And Solute Transport (2013) T.J. Povich, C.N. Dawson, M.W. Farthing, C.E. Kees *Computational Geosciences* 17(3), 529-549, https://doi.org/10.1007/s10596-012-9330-2.
  - Numerical Simulation of Water Resources Problems: Models, Methods, and Trends (2013) C.T. Miller, C.N. Dawson, M.W. Farthing, T.Y. Hou, J. Huang, C. E. Kees, C.T. Kelley, and H.P. Langtangen *Advances in Water Resources*, 51, 405-437, https://doi.org/10.1016/j.advwatres.2012.05.008

COASTAL & HYDRAULICS LABORATORY

Thank You!

COASTAL & HYDRAULICS LABORATORY