FEM techniques for interfacial flows

Stefan Turek, Shu-Ren Hysing

(ture@featflow.de)

Institute for Applied Mathematics and Numerics

University of Dortmund

Workshop on two-phase incompressible flows
Aachen – 25-27/6/2007

Modeling of Interfacial Flows

Numerical simulation of interfacial or two-phase flows
with immiscible fluids poses some challenging problems

Issues

- Accurate interface tracking
- Mass conservation
- Resolution of discontinuous fluid properties
- Treatment of interfacial boundary conditions
- Overall numerical efficiency?

Modeling of Interfacial Flows

Numerical simulation of interfacial or two-phase flows
with immiscible fluids poses some challenging problems

Issues

How can we devise an efficient solution algorithm
which addresses these issues?

Equations governing the flow of incompressible fluids

Modeling of Interfacial Flows

- The Navier-Stokes equations govern incompressible fluid flow

$$
\begin{aligned}
\rho(\mathbf{x})\left(\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u}\right) &= -\nabla p + \nabla\cdot\left(\mu(\mathbf{x})(\nabla\mathbf{u} + (\nabla\mathbf{u})^T)\right) + \rho(\mathbf{x})\mathbf{g} \\
\nabla\cdot\mathbf{u} &= 0
\end{aligned}
$$

with *varying* density $\rho(\mathbf{x})$ and viscosity $\mu(\mathbf{x})$ fields

Interfacial Boundary Conditions

- Direct interface conditions

$$
[\mathbf{u}]\big|_\Gamma = 0, \qquad -\left[-p\mathbf{I} + \mu(\nabla\mathbf{u} + (\nabla\mathbf{u})^T)\right]\big|_\Gamma \cdot \hat{\mathbf{n}} = \sigma\kappa\hat{\mathbf{n}}
$$

- Implicit conditions by weighted volume forces

$$
\mathbf{f}_{st}\big|_\Gamma = \sigma\kappa\hat{\mathbf{n}}
$$

Modeling of Interfacial Flows

- The Navier-Stokes equations govern incompressible fluid flow

$$\begin{aligned}\rho(\mathbf{x})\left(\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u}\right) &= -\nabla p + \nabla\cdot\left(\mu(\mathbf{x})(\nabla\mathbf{u} + (\nabla\mathbf{u})^T)\right) + \rho(\mathbf{x})\mathbf{g} \\ \nabla\cdot\mathbf{u} &= 0\end{aligned}$$

with *varying* density $\rho(\mathbf{x})$ and viscosity $\mu(\mathbf{x})$ fields

Interfacial Boundary Conditions

- Direct interface conditions

$$[\mathbf{u}]|_\Gamma = 0, \quad -\left[-p\mathbf{I} + \mu(\nabla\mathbf{u} + (\nabla\mathbf{u})^T)\right]\Big|_\Gamma \cdot \hat{\mathbf{n}} = \sigma\kappa\hat{\mathbf{n}}$$

- Implicit conditions by weighted volume forces

$$\mathbf{f}_{st}|_\Gamma = \sigma\kappa\hat{\mathbf{n}}$$

Time Discretization

Given $\mathbf{u}^n$ and time step $k = \Delta t$, then solve for $\mathbf{u} = \mathbf{u}^{n+1}$ and $p = p^{n+1}$

$$\begin{aligned} \rho(\mathbf{x})\frac{\mathbf{u}-\mathbf{u}^n}{k} + \theta\left[-\nabla\cdot\left(\mu(\mathbf{x})(\nabla\mathbf{u}+(\nabla\mathbf{u})^T)\right)+\rho(\mathbf{x})(\mathbf{u}\cdot\nabla)\mathbf{u}\right]+\nabla p &= \mathbf{f}^{n+1} \\ \nabla\cdot\mathbf{u} &= 0 \end{aligned}$$

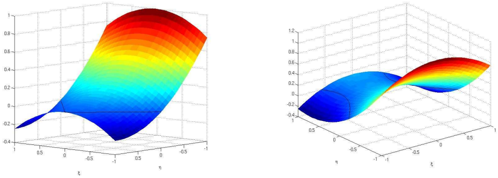
with right hand side

$$\begin{aligned} \mathbf{f}^{n+1} &= \theta\left(\rho(\mathbf{x})^n\mathbf{g}^{n+1}+\mathbf{f}_{st}^{n+1}\right)+(1-\theta)\left(\rho(\mathbf{x})^{n-1}\mathbf{g}^n+\mathbf{f}_{st}^n\right) \\ &- (1-\theta)\left[-\nabla\cdot\left(\mu(\mathbf{x})^n(\nabla\mathbf{u}^n+(\nabla\mathbf{u}^n)^T)\right)+\rho(\mathbf{x})^n(\mathbf{u}^n\cdot\nabla)\mathbf{u}^n\right] \end{aligned}$$

The parameter θ is chosen according to the time stepping scheme, $\theta = 1$ for backward Euler, $\theta = 1/2$ for the Crank-Nicolson scheme, or varying θ according to a Fractional-step-θ-scheme

Space Discretization

- Any method can in general be used to discretize the equations in space (FDM, FEM, FVM)
- We currently prefer to use the efficient nonconforming rotated $\tilde{\mathbb{Q}}_1\mathbb{Q}_0$ finite element spaces



- In the future we plan to move to the highly accurate $\mathbb{Q}_2\mathbb{P}_1$ basis

Space Discretization

Given $\mathbf{u}^n$ and time step $k = \Delta t$, then solve for $\mathbf{u} = \mathbf{u}^{n+1}$ and $p = p^{n+1}$

$$\begin{cases} S\mathbf{u} + kBp &= \mathbf{f}^{n+1} \\ B^T\mathbf{u} &= 0 \end{cases}$$

with

$$S\mathbf{u} = [M_\rho + \theta k N(\mathbf{u})]\mathbf{u}$$

$$N(\mathbf{v}) = -\nabla \cdot \left(\mu(\mathbf{x})(\nabla\mathbf{u} + (\nabla\mathbf{u})^T)\right) + \rho(\mathbf{x})(\mathbf{u} \cdot \nabla)\mathbf{v}$$

B and B^T are discrete counterparts to the *grad* and *div* operators

Discrete Projection Method

Solve the system
$$\begin{cases} S\mathbf{u} + kBp &= \mathbf{f}^{n+1} \\ B^T\mathbf{u} &= 0 \end{cases}$$

with the discrete projection method

- $S\tilde{\mathbf{u}} = \mathbf{f}^{n+1} - kBp^n$
- $f_p = \frac{1}{k}B^T\tilde{\mathbf{u}}$
- $B^T S^{-1} Bq \approx B^T M_{\rho,l}^{-1} Bq = Pq = f_p$
- $p^{n+1} = p^n + \alpha_R q + \alpha_D M_p^{-1} f_p$
- $\mathbf{u}^{n+1} = \tilde{\mathbf{u}} - kM_{\rho,l}^{-1} Bq$

Solver Structure

We prefer to use an operator splitting approach

Given $\mathbf{u}^n$ and p^n do for time level $n+1$:

1. Solve the Navier-Stokes equations, with the interface given at Γ^n, to obtain $\mathbf{u}^{n+1}$ and p^{n+1}
2. (Optionally perform grid adaptation)
3. Move the interface according to the given physics
4. Perform post-processing; e.g. computation of normals and curvature
5. (Optionally perform grid adaptation)
6. Reiterate time loop if deemed necessary (Goto 1)

Interface Tracking

Interface Tracking

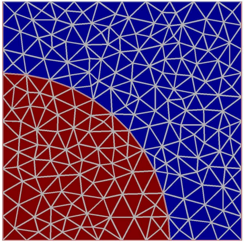
The tasks of an interface tracking algorithm are:

- Correctly propagate the interface given a velocity field
- Minimize additional mass loss due to the algorithm itself
- Enable easy and quick identification of both the interface and the different phases
- Enable easy reconstruction of interface normal vectors and curvature
- Be relatively easy to implement and efficient to solve
- Ideally treat coalescence and break-up automatically

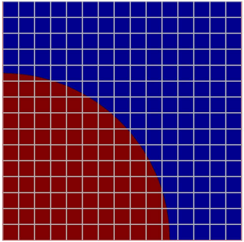
Interface Tracking

Two main schools of thought...

Lagrangian



Eulerian



Interface Tracking

Which method should we use then?

- Pure Lagrangian Approach
- Front Tracking Method
- Segment Projection Method
- Marker and Cell (MAC)
- Volume of Fluid (VOF)
- Phase Field Method
- Level Set Method (LS)
- Particle Level Set Method (PLS)
- Combination (LS-VOF)

Interface Tracking

Our selection criteria

- Eulerian, simplifies implementation
- Allow for discretization and solution with fast and efficient PDE solver techniques
- Handle coalescence and break-up automatically
- Allow for global reconstruction of normals and curvature

Interface Tracking

Our selection criteria

Level Set Method

Level Set Method

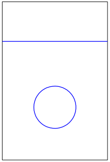
Interface Tracking

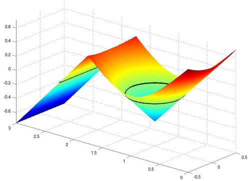
The Level Set Method

The general idea is to embed an interface in a higher dimensional function ϕ. The interface movement is then governed by a standard transport equation

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

with initial condition $\phi(\mathbf{x} = \Gamma, t = 0) = 0$





Properties

The Level Set Method - Properties

- A smoothness constraint on the level set field relaxes the requirements on the solver
- The natural choice is to restrict the level set field to be a signed distance function

$$|\nabla \phi| = 1$$

- At any given point the magnitude of the level set function will thus represent the shortest distance to the interface

Properties

The Level Set Method - Advantages

- The smooth LS function allows for higher order discretization techniques to be employed
- The governing transport equation can be solved with efficient standard solvers
- The interface is implicitly but also exactly defined
- Break up and coalescence is treated automatically
- Geometrical quantities such as normals and curvature can be reconstructed globally

Problems

The Level Set Method - Problems

- The velocity field $\mathbf{v}$ convecting the level set function can not in general be expected to maintain ϕ as a distance function

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

- The distance function property is only preserved if $\nabla \mathbf{v} \cdot \nabla \phi = 0$
- Stretching and folding of the level set function can cause eventual solver failure

Problems

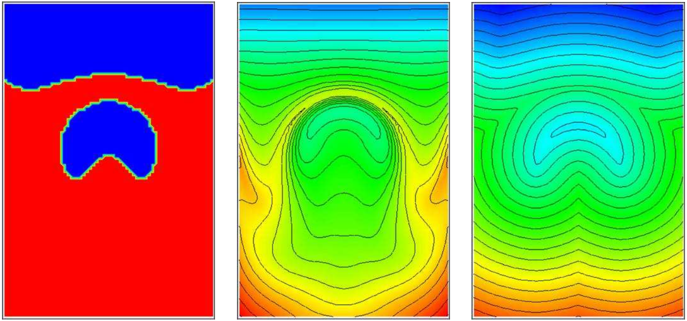
The Level Set Method - Problems

Periodic Reinitialization

Reinitialization

Reinitialization Methods

A stretched and folded level set function can be periodically reinitialized in order to maintain the distance function property



Reinitialization Methods

Several methods exist for constructing a distance function from given interface data. The simplest ones use interface approximations for the reconstruction

- Redistancing via "Brute Force"
- Algebraic Newton approach

The more complex algorithms deal with solving the Eikonal equation $|\nabla \phi| = F$ on a fixed mesh

- The Fast Marching Method
- The Fast Sweeping Method
- PDE Based redistancing
- Branch and Bound approach to redistancing

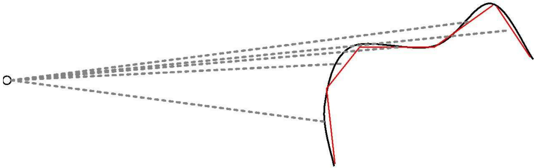
Reinitialization Methods

The different methods naturally have their respective strengths and weaknesses

- Brute force method
  - + Very robust
  - – Scales quite badly, $\mathcal{O}(N * M)$
- Algebraic Newton Approach
  - + Potentially very fast, $\mathcal{O}(N)$
  - – Convergence dependent upon approximate distance function
- The Fast Marching Method
  - + Easy to implement in an unstructured context
  - – Needs a heap structure to sort marching order, $+\mathcal{O}(log N)$
- The Fast Sweeping Method
  - + Potentially very fast, $\mathcal{O}(N)$
  - – Difficult to generalize to fully unstructured grids

Brute Force Algorithm

- Approximate the interface curve with line segments
- Calculate the minimum distance to all segments for each point of interest
- Algorithmic complexity $\mathcal{O}(N * M)$

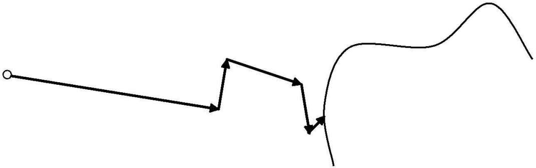


Algebraic Newton Approach

- Solve the following equation for each grid point $\mathbf{x}_0$

$$L(\mathbf{x}) = \begin{bmatrix} \Psi(\mathbf{x}) \\ \nabla\Psi(\mathbf{x}) \times (\mathbf{x} - \mathbf{x}_0) \end{bmatrix} = 0$$

- where Ψ is a given approximate distance field



- Algorithmic complexity is $\mathcal{O}(N)$

Newton System

$$L(\mathbf{x}) = \left[\begin{array}{c} \Psi(x,y) \\ (x-x_0)\Psi_y - (y-y_0)\Psi_x \end{array} \right]$$

$$J(\mathbf{x}) = \frac{\partial L}{\partial \mathbf{x}} = \left[\begin{array}{cc} \Psi_x & \Psi_y + (x-x_0)\Psi_{xy} - (y-y_0)\Psi_{xx} \\ \Psi_y & -\Psi_x + (y-y_0)\Psi_{xy} + (x-x_0)\Psi_{yy} \end{array} \right]^T$$

- Typical Newton iteration

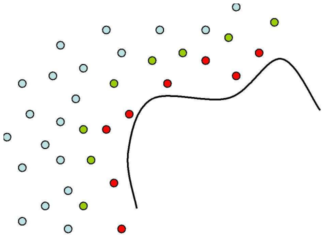
$$\mathbf{x}^{k+1} = \mathbf{x}^k - \delta J^{-1}(\mathbf{x}^k) L(\mathbf{x}^k)$$

- New distance is given by

$$\phi(\mathbf{x}_0) = |\mathbf{x} - \mathbf{x}_0|$$

Fast Marching Method

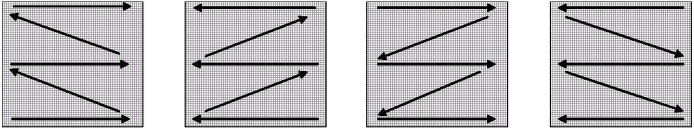
- Update grid points in order of increasing distances with the help of a difference formula
- The solution will thus correspond to an upwind solution



- Algorithmic complexity $\mathcal{O}(N log N)$

Fast Sweeping Method

- Update grid points in predetermined characteristic directions to capture the propagation of information



- Uses the same difference formula as the fast marching method
- Algorithmic complexity $\mathcal{O}(N)$

Difference Update

A difference approximation to the Eikonal equation can be calculated as

$$\begin{cases} \mathbf{P}\nabla\phi(x) = v(x) \\ |\nabla\phi(x)| = 1 \end{cases} \quad \Rightarrow \quad v(x)^T(\mathbf{P}\mathbf{P}^T)v(x) = 1,$$

given N_{dim} known distance values. Using the identities

$$v_i(x) = \phi(x)a_i + b_i, \quad Q = (\mathbf{P}\mathbf{P}^T)^{-1}$$

gives the following relation for the unknown distance value $\phi(x)$

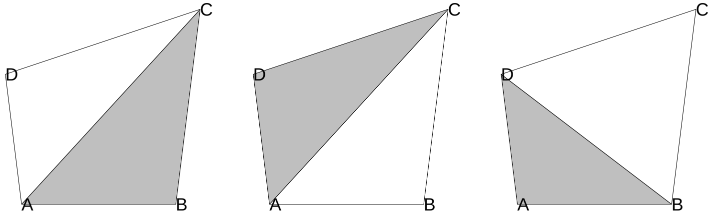
$$(a^TQa)\phi(x)^2 + (2a^TQb)\phi(x) + (b^TQb - 1) = 0$$

where the coefficients in the differentiation formulas are

$$\left[\begin{array}{ll} a_i^{\mathcal{O}(1)} = \frac{1}{|x-x_i|}, & b_i^{\mathcal{O}(1)} = -\frac{\phi(x_i)}{|x-x_i|} \\ a_i^{\mathcal{O}(2)} = \frac{2}{|x-x_i|}, & b_i^{\mathcal{O}(2)} = -\frac{2\phi(x_i)}{|x-x_i|} - \mathbf{P}_i \cdot \nabla\phi(x_i) \end{array}\right]$$

Quadrilateral Treatment

- Since the difference update requires simplexes to work with each quadrilateral is subdivided into triangles



- This subdivision lessens the upwinding restriction

PDE Based Redistancing

- The stationary limit of the following PDE can also be used to apply reinitialization to a given approximate distance field ϕ_0

$$\frac{\partial \phi^*}{\partial t} + \mathbf{q} \cdot \nabla \phi^* = S(\phi_0), \qquad \mathbf{q} = S(\phi_0) \frac{\nabla \phi^*}{|\nabla \phi^*|}$$

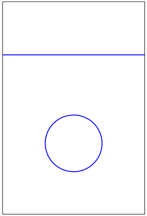
- where $S(\phi_0)$ is an appropriately chosen sign function

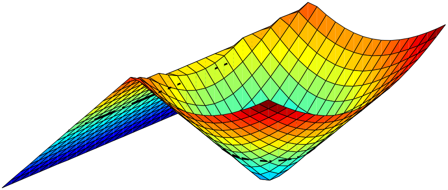
Test Case

A non-trivial test case was chosen to represent a typical level set computation with the following properties

- Include both smooth regions and shocks
- Exact solution available:

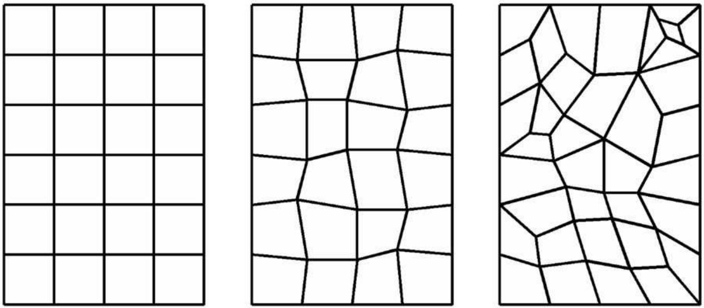
$$\phi(\mathbf{x}) = min(2.25 - y, \sqrt{x^2 + (y-1)^2} - 0.4)$$



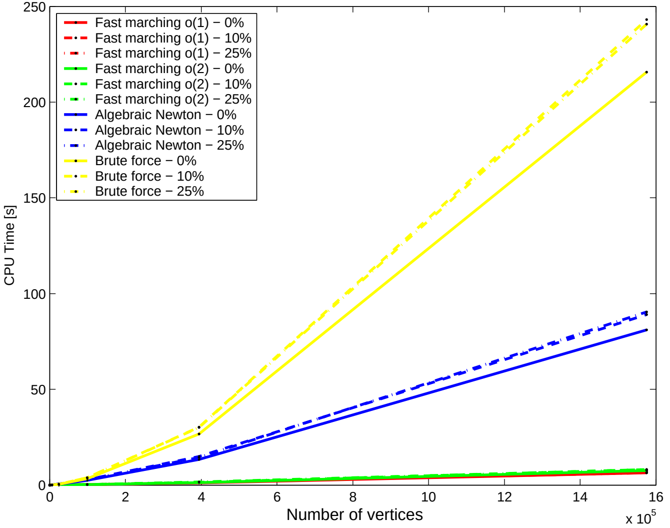


Mesh Cases

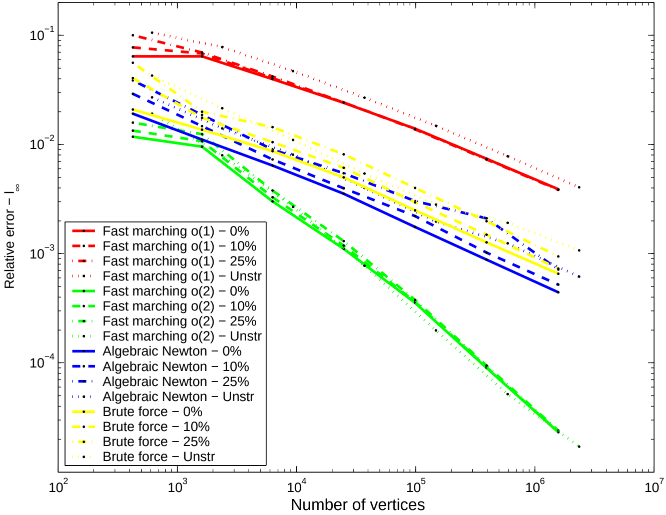
- Different degrees of grid distortion was imposed to test the unstructured capabilities of the algorithms
- The number of grid points was varied between $400 - 2.4 \cdot 10^6$



Results - CPU Time



Results - Accuracy



Reinitialization

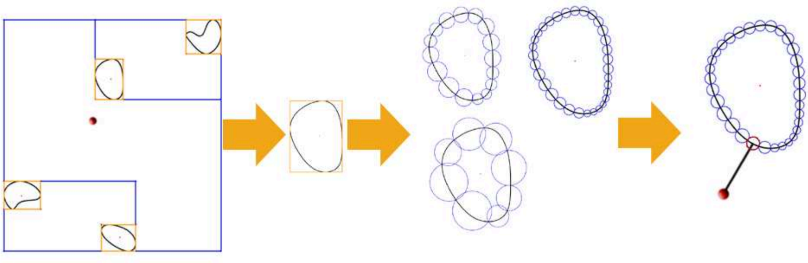
- The fast marching method scales very well with increasing grid density
- Fast marching is very fast even for very dense grids, $\mathcal{O}(10)$ seconds for $2.4 \cdot 10^6$ grid points
- The second order update only costs marginally more than the first order version
- The fast marching method is therefore our preferred algorithm!

Branch and Bound Algorithm

- Employs methods from computer graphics (Ray Tracing, Collision Detection)
- Interfaces are described by NURBS curves or approximations of NURBS curves (point sampling)
- Hierarchical data structures (bounding volume hierarchies) supply lower and upper bounds for the minimum distance
- By repeated refinement the bounds converge against the solution

Branch and Bound Algorithm

Traversal of the bounding volume hierarchy to find the solution.



Branch and Bound Algorithm

- Algorithmic complexity $\mathcal{O}(N log M)$ (N = #grid points, M = #interfaces)
- Accuracy is dependent on the quality of approximation (#point samples)
- Accuracy can be improved by Newton-Iteration $\rightarrow$ increased runtime
- Hardware acceleration (parallelization, SIMD optimization, ...)

Comparison - FMM/BAB

Grid Points	CPU FMM [s]	*e*	CPU BAB [s]	*e*
24576	0.05	0.247E-03	0.02	0.704E-04
98304	0.24	0.752E-04	0.07	0.816E-04
393216	1.18	0.221E-04	0.28	0.612E-03
1572864	6.47	0.638E-05	1.09	0.158E-02

Table: Fast Marching Method vs Branch and Bound

Grid Points	CPU FMM [s]	*e*	CPU BABN [s]	*e*
24576	0.05	0.247E-03	0.04	0.388E-06
98304	0.24	0.752E-04	0.17	0.308E-05
393216	1.18	0.221E-04	0.60	0.159E-05
1572864	6.47	0.638E-05	4.52	0.194E-05

Table: Fast Marching Method vs Branch and Bound with Newton-Iteration

Treatment of Surface Tension

Physical time scales

Different time scales are important in interfacial flows

$$Fr = \frac{U^{(g)}}{\sqrt{gL}} = \frac{L}{\Delta t^{(g)}_{phys}\sqrt{gL}} \approx 1 \qquad \Rightarrow \qquad \Delta t^{(g)}_{phys} \approx \sqrt{\frac{L}{g}}$$

$$Ca = \frac{\mu U^{(ca)}}{\sigma} = \frac{\mu L}{\Delta t^{(ca)}_{phys}\sigma} \approx 1 \qquad \Rightarrow \qquad \Delta t^{(ca)}_{phys} \approx \frac{\mu L}{\sigma}$$

$$St = \frac{\rho L^2}{\mu \Delta t^{(v)}_{phys}} \approx 1 \qquad \Rightarrow \qquad \Delta t^{(v)}_{phys} \approx \frac{\rho L^2}{\mu}$$

Numerical time scales

Gravitational time step restriction

$$\left.\begin{aligned} \frac{v_g \Delta t_{num}^{(g)}}{h} &\leq 1 \\ v_g &= g \Delta t_{num}^{(g)} \end{aligned}\right\} \quad \Rightarrow \quad \Delta t_{num}^{(g)} = \sqrt{\frac{h}{g}}$$

Capillary time step restriction

$$\left.\begin{aligned} \frac{v_{ca} \Delta t_{num}^{(ca)}}{h} &\leq 1 \\ a_{ca} = \frac{\delta_h \sigma \kappa_h}{\rho} &\approx \frac{\sigma}{h^2 \rho} \\ v_{ca} \approx a_{ca} \Delta t_{num}^{(ca)} &= \frac{\Delta t_{num}^{(ca)} \sigma}{h^2 \rho} \end{aligned}\right\} \quad \Rightarrow \quad \Delta t_{num}^{(ca)} = \sqrt{\frac{\rho}{\sigma}} h^{3/2}$$

Modeling of Interfacial Flows

- Interfacial or two-phase flow where capillary surface tension forces are dominant poses some challenging problems
- Surface tension effects are generally modeled both explicitly in time and space leading to the capillary time step restriction

$$\Delta t_{num}^{(ca)} < \sqrt{\frac{\langle \rho \rangle \, h^3}{2\pi\sigma}}$$

Goal

Remove the capillary time step constraint
while retaining a fully Eulerian interface description

Modeling of Interfacial Flows

- Interfacial or two-phase flow where capillary surface tension forces are dominant poses some challenging problems

- Surface tension effects are generally modeled both explicitly in time and space leading to the capillary time step restriction

$$\Delta t_{num}^{(ca)} < \sqrt{\frac{\langle \rho \rangle \, h^3}{2\pi\sigma}}$$

Goal

Remove the capillary time step constraint
while retaining a fully Eulerian interface description

Computation of Surface Tension

Surface tension effects can essentially be included in the Navier-Stokes equations in two different ways

- Explicit interface reconstruction and direct evaluation

$$\mathbf{f}_{st} = \int_{\Gamma} \sigma \kappa \hat{\mathbf{n}} \; d\Gamma$$

- Implicit incorporation via the continuum surface force (CSF) model

$$\mathbf{f}_{st} = \int_{\Omega} \sigma \kappa \hat{\mathbf{n}} \delta(\Gamma) \; d\Omega$$

Definitions

Definition (Tangential gradient)

The tangential gradient *of a function f, which is differentiable in an open neighborhood of Γ, is defined by*

$$\underline{\nabla} f(x) = \nabla f(x) - (\hat{\mathbf{n}}(x) \cdot \nabla f(x))\hat{\mathbf{n}}(x), \quad x \in \Gamma$$

where ∇ denotes the usual gradient in $\mathbb{R}^d$

Definition (Laplace-Beltrami operator)

If f is two times differentiable in a neighborhood of Γ, then we define the Laplace-Beltrami operator *of f as*

$$\underline{\Delta} f(x) = \underline{\nabla} \cdot (\underline{\nabla} f(x)), \quad x \in \Gamma$$

Definitions and Derivation

Theorem

A theorem of differential geometry states that

$$\underline{\Delta}\mathrm{id}_\Gamma = \kappa \hat{\mathbf{n}}$$

where κ is the mean curvature and id_Γ is the identity mapping on Γ

Derivation

First take surface tension force source term, multiply it with the test function space $\mathbf{v}$, and apply partial integration

$$\begin{aligned} \mathbf{f}_{st} &= \int_\Gamma \sigma\kappa\hat{\mathbf{n}} \cdot \mathbf{v}\ d\Gamma = \int_\Gamma \sigma(\underline{\Delta}\mathrm{id}_\Gamma) \cdot \mathbf{v}\ d\Gamma = \\ &= -\int_\Gamma \sigma\underline{\nabla}\mathrm{id}_\Gamma \cdot \underline{\nabla}\mathbf{v}\ d\Gamma + \int_\gamma \sigma\partial_\gamma \mathrm{id}_\Gamma \cdot \mathbf{v}\ d\gamma \end{aligned}$$

Fully Implicit Evaluation in Space

- Boundary integrals can be transformed to volume integrals with the help of a Dirac delta function $\delta(\Gamma, \mathbf{x})$

$$
\begin{aligned}
\mathbf{f}_{st} &= \int_\Omega \sigma\kappa\hat{\mathbf{n}}\cdot\mathbf{v}\ \delta(\Gamma,\mathbf{x})\ d\mathbf{x} &&= \int_\Omega \sigma(\underline{\Delta}\mathrm{id}_\Gamma)\cdot(\mathbf{v}\delta(\Gamma,\mathbf{x}))\ d\mathbf{x} \\
&= -\int_\Omega \sigma\underline{\nabla}\mathrm{id}_\Gamma\cdot\underline{\nabla}(\mathbf{v}\delta(\Gamma,\mathbf{x}))\ d\mathbf{x} &&= -\int_\Omega \sigma\underline{\nabla}\mathrm{id}_\Gamma\cdot\underline{\nabla}\mathbf{v}\ \delta(\Gamma,\mathbf{x})\ d\mathbf{x}
\end{aligned}
$$

- Application of the semi-implicit time integration

$$
\Gamma^{n+1} = \Gamma^n + \Delta t\ \mathbf{u}^{n+1}
$$

yields

$$
\begin{aligned}
\mathbf{f}_{st} = &-\int_\Omega \sigma\underline{\nabla}(\mathrm{id}_\Gamma)^n\cdot\underline{\nabla}\mathbf{v}\ \delta(\Gamma^n,\mathbf{x})\ d\mathbf{x} \\
&-\Delta t^{n+1}\int_\Omega \sigma\underline{\nabla}\mathbf{u}^{n+1}\cdot\underline{\nabla}\mathbf{v}\ \delta(\Gamma^n,\mathbf{x})\ d\mathbf{x}
\end{aligned}
$$

Regularization

- Regularization of $\delta(\Gamma, \mathbf{x})$ can easily be accomplished with the help of a distance function

$$\delta_{\boldsymbol{\epsilon}}(\Gamma, \mathbf{x}) = \delta_\epsilon(dist(\Gamma, \mathbf{x})),$$

where $dist(\Gamma, \mathbf{x})$ gives the minimum distance from $\mathbf{x}$ to Γ

- The regularized continuous delta function δ_ϵ is defined as

$$\delta_\epsilon(x) = \begin{cases} \frac{1}{\epsilon}\varphi(x/\epsilon) & |x| \le \epsilon \quad = \quad mh, \\ 0 & |x| > \epsilon \quad = \quad mh, \end{cases}$$

where h is the mesh spacing which together with the constant m defines the support ϵ of the regularized delta function, φ is a characteristic function determining the kernel shape

Implicit Surface Tension Force Expression

The surface tension forces are finally given by ...

Implicit Surface Tension Force Expression

$$\begin{aligned}\mathbf{f}_{st} \quad = \quad & -\int_\Omega \sigma \; \delta_\epsilon(dist(\Gamma^n, \mathbf{x})) \; \underline{\nabla}(\widetilde{\mathrm{id}}_\Gamma)^n \cdot \underline{\nabla}\mathbf{v} \; d\mathbf{x} \\ & -\Delta t^{n+1} \int_\Omega \sigma \; \delta_\epsilon(dist(\Gamma^n, \mathbf{x})) \; \underline{\nabla}\mathbf{u}^{n+1} \cdot \underline{\nabla}\mathbf{v} \; d\mathbf{x}\end{aligned}$$

Implicit Surface Tension Force Expression

The surface tension forces are finally given by ...

Implicit Surface Tension Force Expression

$$\begin{aligned}\mathbf{f}_{st} &= -\int_\Omega \sigma\ \delta_\epsilon(dist(\Gamma^n,\mathbf{x}))\ \underline{\nabla}(\widetilde{\mathrm{id}}_\Gamma)^n \cdot \underline{\nabla}\mathbf{v}\ d\mathbf{x} \\ &\quad -\Delta t^{n+1}\int_\Omega \sigma\ \delta_\epsilon(dist(\Gamma^n,\mathbf{x}))\ \underline{\nabla}\mathbf{u}^{n+1} \cdot \underline{\nabla}\mathbf{v}\ d\mathbf{x}\end{aligned}$$

Level Set Method

A number of key points makes the *level set method* an ideal candidate for interface tracking algorithm when implementing the proposed surface tension force expressions

- Distance functions are in general readily available allowing for simple construction of the regularized Dirac delta functions
- Geometrical quantities such as normal and tangent vectors can be reconstructed globally, eliminating the need to extend these quantities from the interface separately
- The level set method can be coupled with the finite element method giving access to the variational form of the equations

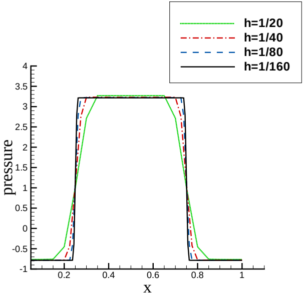
Method Validation

Numerical Examples

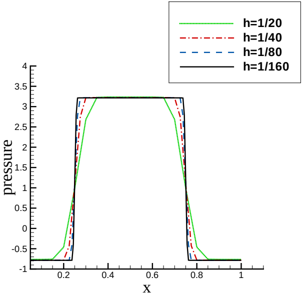
Validation, Laplace-Young Law

A perfect circular static bubble should follow the Laplace-Young law

$$p_{inside} = p_{outside} + \frac{\sigma}{r}$$



(a) CSF



(b) CSF-LBI

Figure: Pressure cut-line for four different mesh sizes.

Example, Oscillating Bubble (CSF)

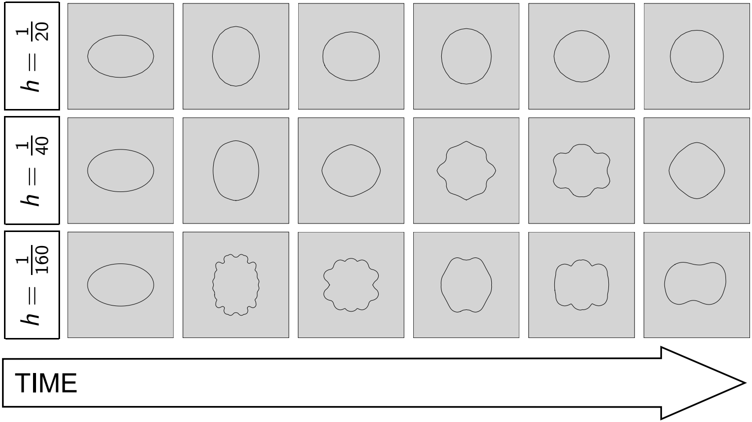


Figure: Evolution of an oscillating bubble; standard explicit CSF method.

Example, Oscillating Bubble (CSF-LBI)

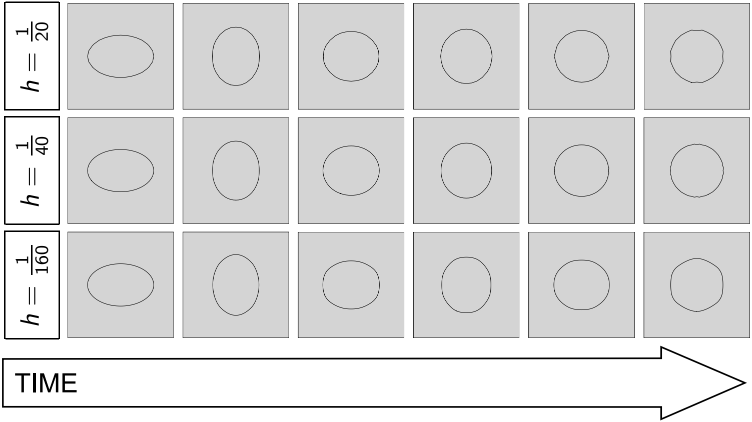


Figure: Evolution of an oscillating bubble; semi-implicit CSF-LBI method.

Example, Rising Bubble (CSF)

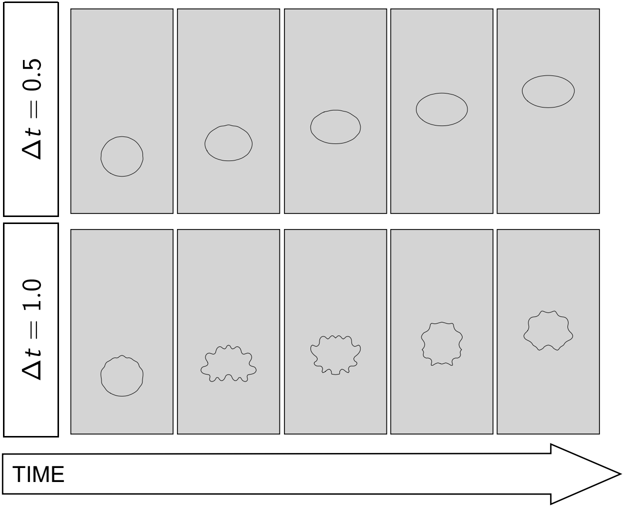


Figure: Evolution of a rising bubble with the standard explicit CSF method.

Example, Rising Bubble (CSF-LBI)

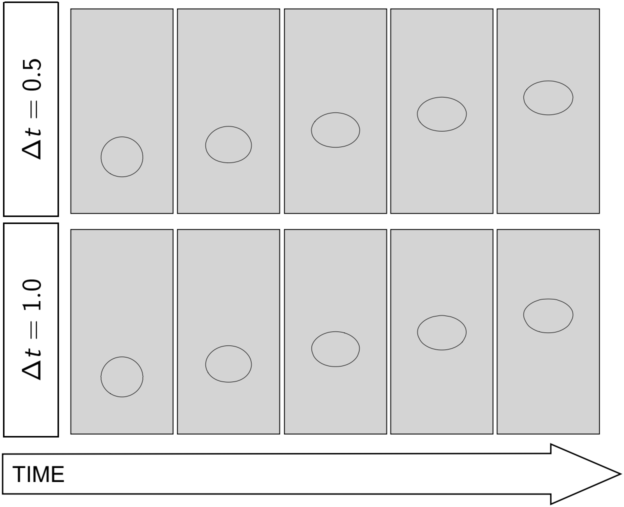


Figure: Evolution of a rising bubble with the semi-implicit CSF-LBI method.

A new implicit surface tension variant has been proposed which relaxes the capillary time step restriction imposed on explicit implementations

Additional advantages

- Fully implicit in space
- Is easily implemented when using the level set method together with finite elements
- Explicit computation of curvature not necessary
- Conceptually identical algorithm in 3D

Grid Deformation Techniques

Grid Deformation

The grid deformation process involves constructing a transformation ϕ, from the computational space ξ to the physical space $x = \phi(\xi)$

There are two basic types of grid deformation methods

- *local based*, generally computing x by minimizing a variational form
- *velocity based*, computing the mesh velocity $v = x_t$ (Lagrangian)

The latter method has several advantages:

- only linear Poisson problems on fixed meshes are needed to be solved
- a monitor function may be obtained directly from error distributions
- mesh tangling can be prevented quite easily
- the data structure is always the same as that for the starting mesh

Grid Deformation

Given the area distribution of the undeformed mesh $g(x)$, and a monitor function/size distribution $f(x)$ for the target grid, then the transformation ϕ can be computed via the following four steps:

1. Compute the scale factors c_f and c_g for the given monitor function $f(x) > 0$ and the area distribution g using

$$c_f \int_\Omega \frac{1}{f(x)} dx = c_g \int_\Omega \frac{1}{g(x)} dx = |\Omega|,$$

where, $\Omega \subset \mathbb{R}^n$ is a computational domain. Let $\tilde{f}$ and $\tilde{g}$ denote the reciprocals of the scaled functions f and g, that is,

$$\tilde{f} = \frac{c_f}{f}, \qquad \tilde{g} = \frac{c_g}{g}$$

Grid Deformation

2. Compute a grid-velocity vector field $v : \Omega \to \mathbb{R}^n$ by satisfying the following linear Poisson equation

$$-\mathrm{div}(v(x)) = \tilde{f}(x) - \tilde{g}(x), \quad x \in \Omega, \quad \text{and} \quad v(x) \cdot \mathfrak{n} = 0, \quad x \in \partial\Omega,$$

   where $\mathfrak{n}$ being the outer normal vector of the domain boundary $\partial\Omega$, which may consist of several boundary components

3. For each grid point x, solve the following ODE system

$$\frac{\partial \varphi(x,t)}{\partial t} = \eta(\varphi(x,t),t), \quad 0 \leq t \leq 1, \quad \varphi(x,0) = x,$$

   with

$$\eta(y,s) := \frac{v(y)}{s\tilde{f}(y) + (1-s)\tilde{g}(y)}, \quad y \in \Omega, \quad s \in [0,1]$$

4. Get the deformed grid points via $\phi(x) := \varphi(x,1)$

Grid Deformation

Monitor function

- In the case of interfacial flow, then the monitor function f can be constructed from a distance function, giving the shortest distance to the interface, and possibly also weighing in the interface curvature κ

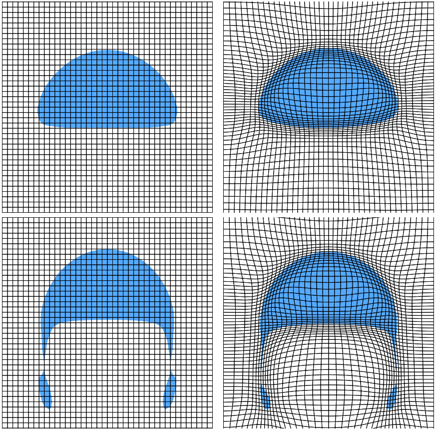
$$f = f\left(|\phi(x)|, \kappa(x)\right)$$

- This makes the *level set method* ideally suited to use as interface tracking algorithm, since both the distance function and curvature are defined globally

Grid Deformation

Question

Does the increase in accuracy justify the increase in CPU time?



Static Bubble Test

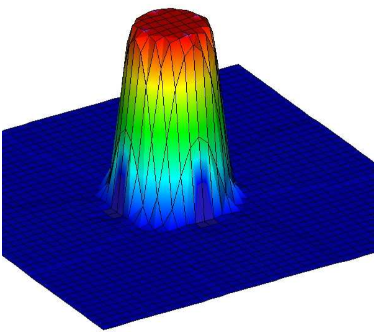


Figure: Pressure field for a static bubble with and without grid deformation

Static Bubble Test

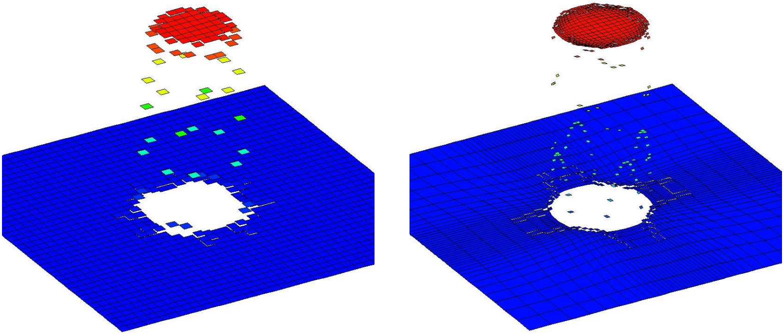


Figure: Pressure field for a static bubble with and without grid deformation

Static Bubble Test

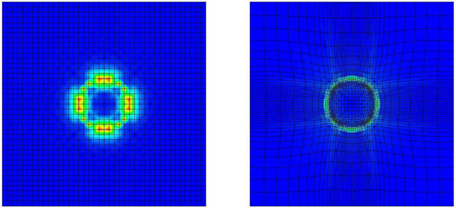


Figure: Velocity error for a static bubble with and without grid deformation

GridLevel	3	4	5	6
Tensor product grid				
U error	$3.7 \cdot 10^{-2}$	$1.1 \cdot 10^{-2}$	$1.0 \cdot 10^{-2}$	$5.6 \cdot 10^{-3}$
P error	5.583%	1.433%	0.212%	0.037%
Adapted grid				
U error	$2.0 \cdot 10^{-2}$	$7.3 \cdot 10^{-3}$	$3.5 \cdot 10^{-3}$	$1.8 \cdot 10^{-3}$
P error	2.969%	0.261%	0.042%	0.013%

Pressure Separation Algorithms

Pressure separation Algorithms

The pressure separation algorithm is designed for flow situations which are dominated by the pressure gradient or higher order pressure derivatives.

- A priori error estimate for Navier-Stokes problem

$$h|\mathbf{u}-\mathbf{u}_h|_{1,\Omega}+\|\mathbf{u}-\mathbf{u}_h\|_{0,\Omega}\leq \quad Ch^{k+1}\left\{|\mathbf{u}|_{k+1,\Omega}+\frac{1}{\nu}|p|_{k,\Omega}\right\}$$

- Modified problem with pressure separation (**Ganesan & John**)

$$\frac{\partial\mathbf{u}}{\partial t}+\mathbf{u}\cdot\nabla\mathbf{u}-\nu\Delta\mathbf{u}+\nabla\tilde{p}=\mathbf{f}-\nabla p_{sep}\quad(\tilde{p}=p-p_{sep})$$

- New a priori error estimate

$$h|\mathbf{u}-\mathbf{u}_h|_{1,\Omega}+\|\mathbf{u}-\mathbf{u}_h\|_{0,\Omega}\leq \quad Ch^{k+1}\left\{|\mathbf{u}|_{k+1,\Omega}+\frac{1}{\nu}|p-p_{sep}|_{k,\Omega}\right\}$$

- Improvement if $\frac{1}{\nu}|p|_{k,\Omega}$ dominant and $|p-p_{sep}|_{k,\Omega}\ll|p|_{k,\Omega}$

How relevant for real CFD simulations?

Pressure Separation Algorithms

- **Stationary case**

1. Compute $(\mathbf{u}_h, p_h)$, as finite element solution for the original Navier-stokes equations
2. Compute $p_{sep,h} = I(p_h)$, interpolation of p_h, such that $|p_h - p_{sep,h}|_k \ll |p_h|_k$
3. Compute $(\mathbf{u}_{sep,h}, \tilde{p}_h)$, the finite element solution of the modified Navier-Stokes equations with $\nabla p_{sep,h}$ as right hand side:

$$(\tilde{\mathbf{u}}_h, \tilde{p}_h) := \mathrm{NS}^{-1}(\mathbf{f} - \nabla p_{sep,h})$$

4. Set $\mathbf{u}_h = \mathbf{u}_{sep,h}$ and $p_h = p_h + \tilde{p}_h$

- **Remarks**

◊ This algorithm requires almost double CPU times w.r.t solving the original problem.
◊ $p_{sep,h}$ can be deduced from p_{2h} in multigrid.
◊ If p_h was approximated by piecewise constant function, $p_{sep,h}$ in the second step can be taken as its linear interpolation (see Ganesan & John (2005)).

Pressure Separation Algorithms

- **Nonstationary case**

1. Compute $p^n_{sep,h} := I(p^{n-1}_h)$, interpolation of p^{n-1}_h
2. Compute $(\mathbf{u}^n_{sep,h}, \tilde{p}^n_h)$, the finite element solution of the modified Navier-Stokes equations with $\nabla p^n_{sep,h}$ as right hand side

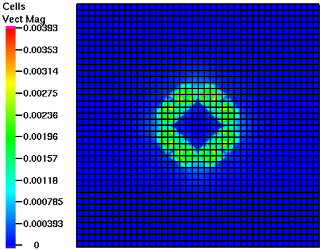
$$(\tilde{\mathbf{u}}^n_h, \tilde{p}^n_h) := \mathrm{NS}^{-1}(\mathbf{f^n} - \nabla p^n_{sep,h})$$

3. Set $\mathbf{u}^n_h = \mathbf{u}^n_{sep,h}$ and $p^n_h = p^{n-1}_h + \tilde{p}^n_h$

- **Remarks**

$\Diamond$ This algorithm is simple.
$\Diamond$ $p^n_{sep,h}$ in the first step can be taken as high order exrapolation, as for instance $p^n_{sep,h} = I(2p^{n-1}_h - p^{n-2}_h)$.

Static Bubble Test



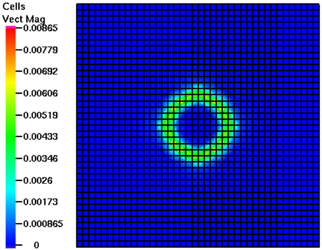


Figure: Velocity field for a static bubble with and without PSepA

Edge-oriented FEM Stabilization

Edge-oriented Stabilization FEM

- **Based only on the "smoothness" of the discrete solution** we add the following jump term

$$\sum_{\text{edge E}} \max(\gamma \nu h_E, \gamma^* h_E^2) \int_E [\nabla \mathbf{u}][\nabla \mathbf{v}] d\sigma \quad \text{with } \gamma, \gamma^* \in [0.0001, 0.1]$$

- only one generic stabilization takes care of all instabilities

  1. insatisfaction of Korn's inequality $(\gamma \nu h_E)$

  2. convection dominated flow for medium and high Reynolds number, even for pure transport $(\gamma^* h_E^2)$

- independent of the local Reynolds number and finite element space

Can EO-FEM solve the spurious velocity problem ?

If so, how to generalize the mesh-dependent penalty parameter ?

Static Bubble Test: Errors on an equidistant mesh

Level	$\lvert p_i - p_o\rvert/(\frac{\sigma}{r})$	$\lVert \mathbf{u}-\mathbf{u}_h\rVert_0$	$\lvert \mathbf{u}-\mathbf{u}_h\rvert_{1,h}$	N/MG	$\lvert p_{in} - p_{out}\rvert/(\frac{\sigma}{r})$	$\lVert \mathbf{u}-\mathbf{u}_h\rVert_0$	$\lvert \mathbf{u}-\mathbf{u}_h\rvert_{1,h}$	N/MG
	without pressure separation				with pressure separation			
	without edge-oriented FEM							
4	0.954349	0.00260818914	0.207652409	5/1	0.9923660	0.00155247296	0.118638289	5/1
5	0.979682	0.00097177495	0.153784641	5/1	0.9975795	0.00060710946	0.0917261177	5/1
6	0.992961	0.00036200902	0.112884238	4/1	1.0012544	0.00023717308	0.0694932176	4/1
7	0.997166	0.00013827272	0.082118238	4/1	1.0010944	9.7099099E-05	0.0514852452	4/1
	with edge-oriented FEM with global constant penalty parameter $\gamma = 1d1$							
4	0.951601	3.340218E-05	0.0025474881	6/1	0.9520738	3.330621E-05	0.00256013778	6/1
5	0.979383	1.086225E-05	0.0016437028	5/1	0.9792187	1.214066E-05	0.00181935191	5/1
6	0.992989	4.221967E-06	0.0012491933	5/1	0.9926422	4.712407E-06	0.00140673313	5/1
7	0.997110	1.624452E-06	0.0009130933	4/1	0.9966825	1.725864E-06	0.00103094094	4/1
	with edge-oriented FEM with global constant penalty parameter $\gamma = 1d3$							
4	0.951998	3.385404E-07	2.600357E-05	6/1	0.988809	2.201445E-07	1.644319E-05	6/1
5	0.979198	1.233316E-07	1.846353E-05	5/1	0.997101	8.065997E-08	1.169144E-05	5/1
6	0.992635	4.789259E-08	1.428115E-05	5/1	1.001279	3.218998E-08	9.075670E-06	5/1
7	0.996678	1.753280E-08	1.046342E-05	4/1	1.000998	1.258395E-08	6.466150E-06	4/1
	with edge-oriented FEM with local penalty parameter γ as a function of the monitor/distance function							
4	0.949683	3.617674E-07	2.686804E-05	6/1	0.986366	2.402242E-07	1.756815E-05	6/1
5	0.978834	1.191665E-07	1.730090E-05	5/1	0.996440	9.042803E-08	1.250519E-05	5/1
6	0.992673	4.752538E-08	1.313859E-05	5/1	1.000876	3.748013E-08	9.682061E-06	5/1
7	0.996931	2.010113E-08	9.567174E-06	4/1	1.000757	1.671886E-08	6.858230E-06	4/1

- **Pressure Separation:** Good results for the pressure
- **Edge-oriented FEM:** Excellent results for the velocity with any desired error & no degradation in the performance of the iterative solver

Static Bubble Test: Errors on an aligned mesh

Level	$\|p_{in} - p_{out}\|/(\frac{\sigma}{r})$	$\\|\mathbf{u} - \mathbf{u}_h\\|_0$	$\|\mathbf{u} - \mathbf{u}_h\|_{1,h}$	N/MG	$\|p_{in} - p_{out}\|/(\frac{\sigma}{r})$	$\\|\mathbf{u} - \mathbf{u}_h\\|_0$	$\|\mathbf{u} - \mathbf{u}_h\|_{1,h}$	N/MG
	without pressure separation				with pressure separation			
				without edge-oriented FEM				
4	1.000669	0.0001899205	0.09765440	6/1	1.0019009	0.0001749634	0.04170730	6/1
5	1.000135	3.503739E-05	0.05796067	5/1	1.0009837	5.679786E-05	0.03268579	5/1
6	1.000032	6.628077E-06	0.03782558	4/1	1.0003227	1.897943E-05	0.02315339	3/1
7	1.000000	2.257852E-06	0.02894883	4/1	1.0001409	6.480485E-06	0.01641194	4/1
				with edge-oriented FEM with global constant penalty parameter $\gamma = 1d1$				
4	1.000719	1.872302E-05	0.004474451	5/1	1.0008292	1.559681E-05	0.00334168	5/1
5	1.000336	4.214648E-06	0.002285405	4/2	1.0005137	5.341385E-06	0.00252941	4/2
6	1.000109	1.665666E-06	0.001819288	4/2	1.0001367	2.051831E-06	0.00201336	4/2
7	1.000040	5.368627E-07	0.001158316	4/2	1.0000440	6.515407E-07	0.00128245	4/2
				with edge-oriented FEM with global constant penalty parameter $\gamma = 1d3$				
4	1.000712	2.186715E-07	5.113188E-05	5/1	1.0006502	1.810321E-07	3.810090E-05	5/1
5	1.000347	5.257776E-08	2.710133E-05	4/2	1.0004652	6.212258E-08	2.860246E-05	4/2
6	1.000113	2.139767E-08	2.195185E-05	4/2	1.0001190	2.437015E-08	2.309864E-05	4/2
7	1.000043	6.806535E-09	1.378594E-05	4/2	1.0000350	7.652504E-09	1.453748E-05	4/2
				with edge-oriented FEM with local penalty parameter γ as a function of the monitor/distance function				
4	1.000599	5.221021E-07	0.0001083159	6/1	1.0008286	4.957030E-07	8.887125E-05	5/1
5	1.000277	1.994104E-07	8.527144E-05	4/2	1.0000613	1.726385E-07	6.962604E-05	4/2
6	0.999927	7.079676E-08	6.249968E-05	5/2	0.9997869	7.318976E-08	5.877441E-05	5/2
7	1.000017	2.608346E-08	4.290585E-05	4/2	0.9999271	2.460148E-08	3.811101E-05	4/2

- **Grid deformation:** Good results for the pressure & amelioration in the velocity
- **Edge-oriented FEM:** Excellent results for the velocity with any desired error & no degradation in the performance of the iterative solver

Flow with Interface

- **Pressure Separation:** Good results for the pressure mainly seen on non adapted mesh
- **Grid deformation:** Good results for the pressure as well as significant amelioration in the velocity
- **Edge-oriented FEM:**

  1. Excellent results for the velocity with any desired error without any degradation in the performance of the iterative solver for both equidistant and aligned mesh

  2. The penalty mesh-dependent parameter can be applied as global constant as well as a function of the interface or location of the spurious velocity;

  $$\sum_{\text{edge E}} \max[\gamma\nu h_E, \gamma^* h_E^2, \gamma_{\text{dist}} f(dist(\Gamma); h_E)) h_E] \int_E [\nabla \mathbf{u}][\nabla \mathbf{v}] d\sigma \quad \text{with ,}$$

  $\gamma_{\text{dist}} \gg 0$ (big enough), $dist(\Gamma)$ a distance function to the interface, and

  f any variant of dirac function

Static Bubble Test

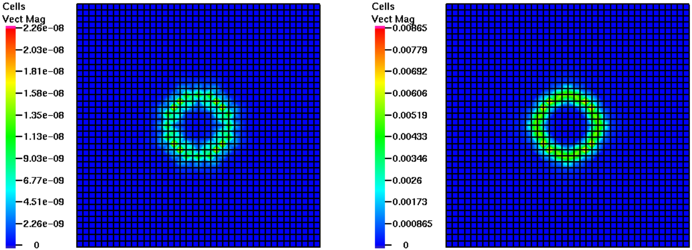


Figure: Velocity field for a static bubble with and without EO-FEM

Future Research Directions

In development...

- ALE techniques coupled with time dependent grid deformation
- Inclusion of *pressure separation* techniques to improve the velocity and pressure
- Linear high order *edge stabilization* for convection of the level set field
- $\mathbb{Q}_2\mathbb{P}_1$ finite element approximation for the NS-equations and $\mathbb{Q}_n$ for the level set equation
- 3D $+$ benchmarking
- Contact angle, heat transfer, and solidification effects

Benchmarking

Why benchmark

Why spend valuable time and effort to establish benchmark test cases?

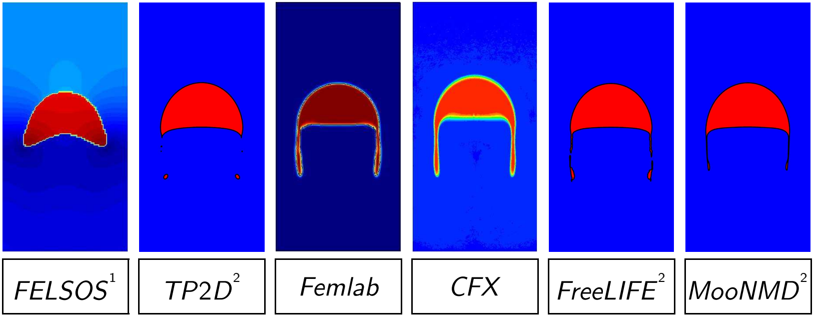
- Validation
- Comparison
- Evaluation

Why benchmark

Why spend valuable time and effort to establish benchmark test cases?

What is the CPU cost for achieving a certain accuracy?

Validation?



1) A. Smolianski; *Finite-element/level-set/operator-splitting (FELSOS) approach for computing two-fluid unsteady flows with free moving interfaces*, Int. J. Numer. Meth. Fluids 2005; 48:231-269.

2) S. Hysing, S. Turek, D. Kuzmin, N. Parolini, E. Burman, S. Ganesan, and L. Tobiska; *Proposal for quantitative benchmark computations of bubble dynamics*, Submitted to Int. J. Numer. Meth. Fluids.

Benchmarks

Development of quantitative two-phase flow benchmarks for critical evaluation of new and existing methods

Bubble benchmark quantities

- Center of mass
- Circularity
- Rise velocity

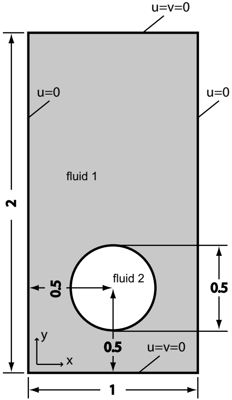


Figure: Initial configuration and boundary conditions for the test cases

Benchmark Quantities

- Center of mass:

$$\mathbf{x}_c = \int_{\Omega_2} \mathbf{x} \; d\mathbf{x} / \int_{\Omega_2} 1 \; d\mathbf{x}$$

- Circularity:

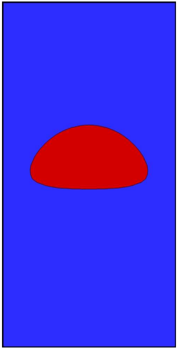
$$\not{c} = \frac{\text{perimeter of area-equivalent circle}}{\text{perimeter of } \Omega_2}$$

- Rise velocity:

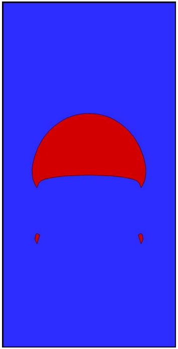
$$\mathbf{U} = \int_{\Omega_2} \mathbf{u} \; d\mathbf{x} / \int_{\Omega_2} 1 \; d\mathbf{x}$$

Benchmark Test Cases

Test Case	1	2
ρ_1 (liquid)	1000	1000
ρ_2 (gas)	1	100
μ_1 (liquid)	10	10
μ_2 (gas)	0.1	1
g_y	-0.98	-0.98
σ	1.96	24.5
Re	35	35
Eo	125	10
ρ_1/ρ_2	1000	10
μ_1/μ_2	100	10



Test Case 1



Test Case 2

Benchmark Test Cases

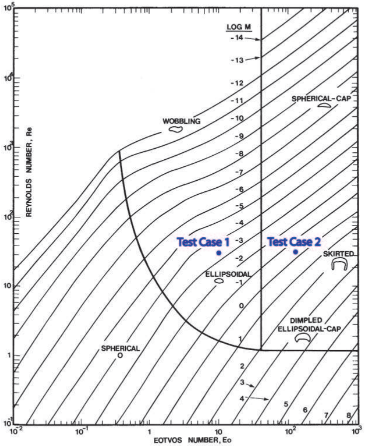


Figure: Shape regimes for bubbles and drops in unhindered gravitational motion through liquids [Clift *et al.*, Bubbles, Drops and Particles (1978)]

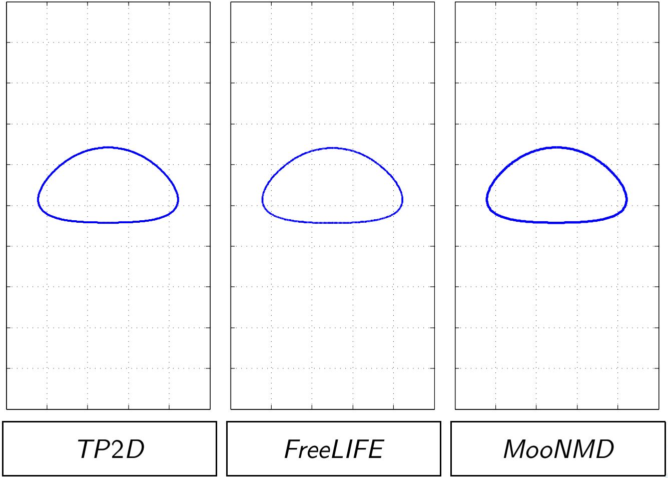
Preliminary computations

Participating Groups

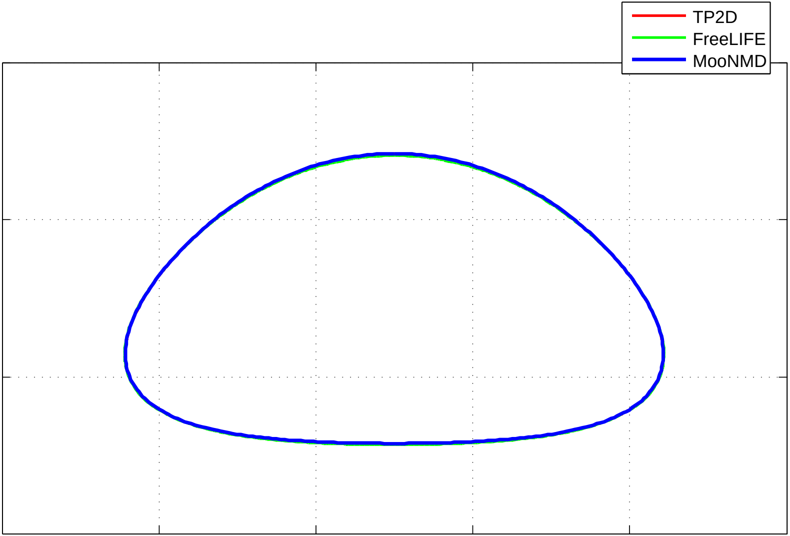
Group and Affiliation		Code/Method
1	Uni. Dortmund, Inst. of Applied Math. *S. Turek, D. Kuzmin, S. Hysing*	**TP2D** *FEM-Level Set*
2	EPFL Lausanne, Inst. of Analysis and Sci. Comp. *E. Burman, N. Parolini*	**FreeLIFE** *FEM-Level Set*
3	Uni. Magdeburg, Inst. of Analysis and Num. Math. *L. Tobiska, S. Ganesan*	**MooNMD** *FEM-ALE*

Test Case 1

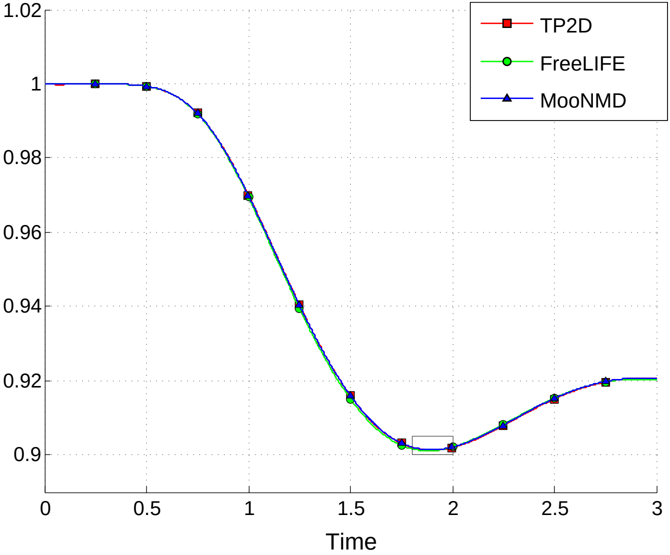
Visual comparison



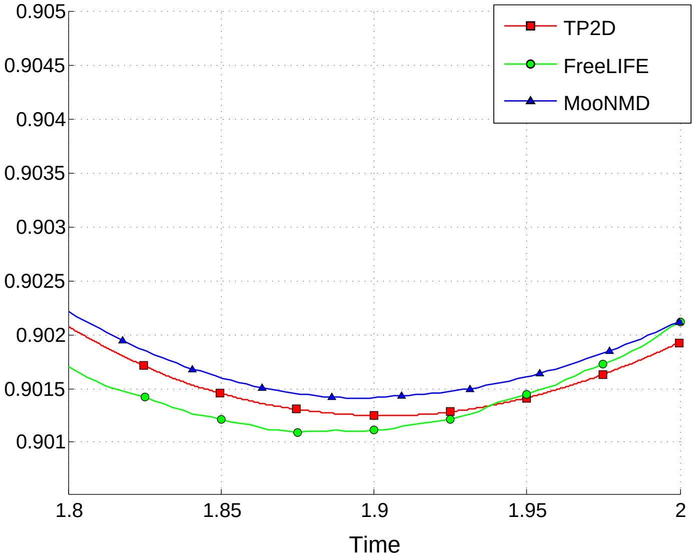
Visual comparison



Benchmark quantities - circularity



Benchmark quantities - circularity



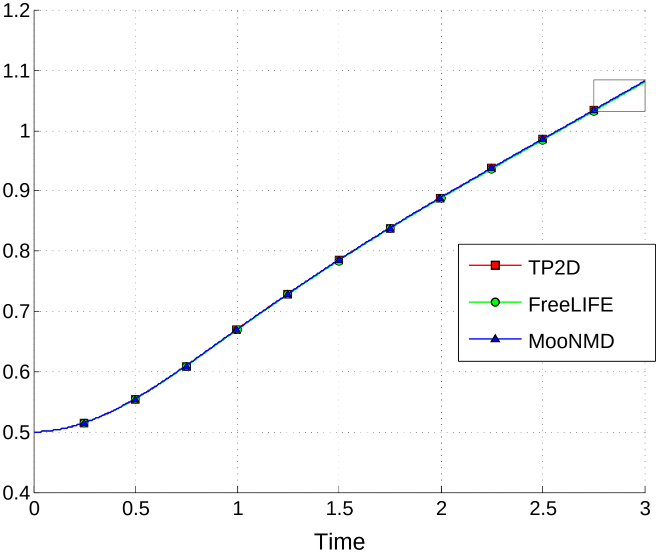
Benchmark quantities - circularity

GridLevel	1	2	3	4
Minimum circularity, $\not{c}_{min}$				
TP2D	0.9016	0.9014	0.9014	0.9013
FreeLIFE		0.9060	0.9021	0.9011
MooNMD	0.9022	0.9018	0.9013	0.9014
Incidence time, $t\|_{\not{c}=\not{c}_{min}}$				
TP2D	1.9234	1.8734	1.9070	1.9041
FreeLIFE		1.8375	1.9125	1.8750
MooNMD	1.8630	1.8883	1.9023	1.8987

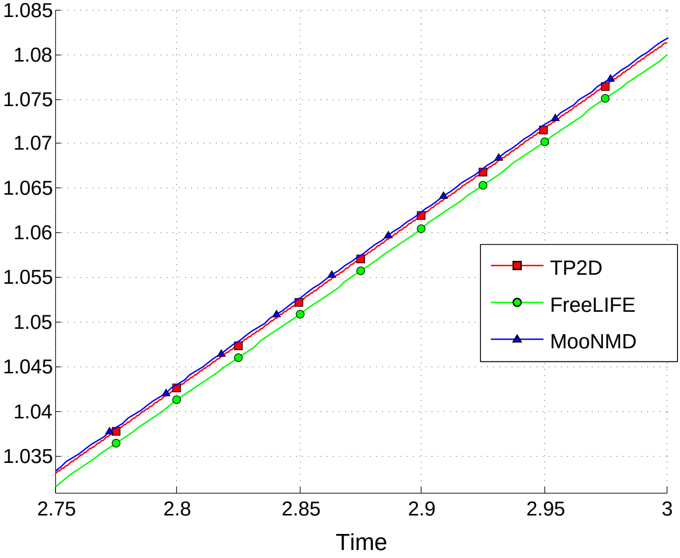
Reference target range

$$\not{c}_{min} = 0.9012 \pm 0.0002, \qquad t|_{\not{c}=\not{c}_{min}} = 1.89 \pm 0.01$$

Benchmark quantities - center of mass



Benchmark quantities - center of mass



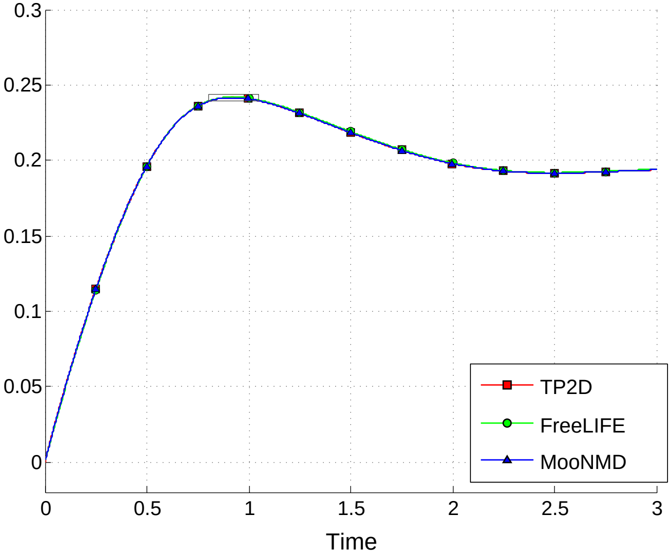
Benchmark quantities - center of mass

GridLevel	1	2	3	4
Center of mass, $y_c\|_{t=3}$				
TP2D	1.0818	1.0810	1.0812	1.0813
FreeLIFE		1.0715	1.0817	1.0799
MooNMD	1.0833	1.0823	1.0815	1.0817

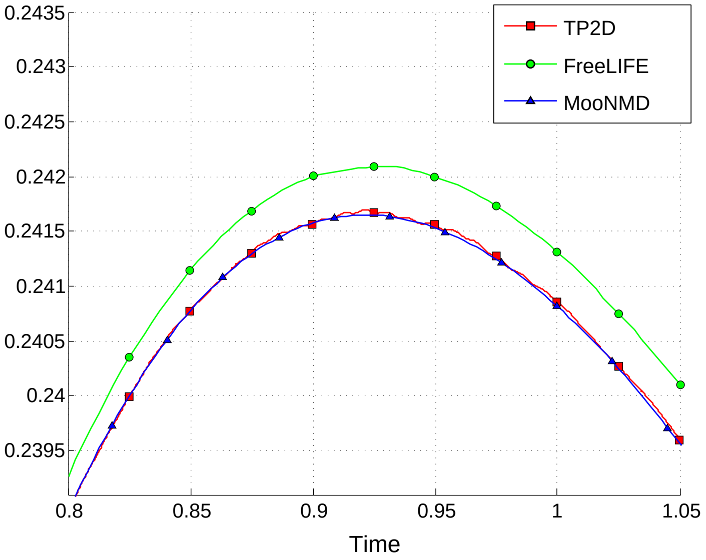
Reference target range

$$y_c|_{t=3} = 1.081 \pm 0.001$$

Benchmark quantities - rise velocity



Benchmark quantities - rise velocity



Benchmark quantities - rise velocity

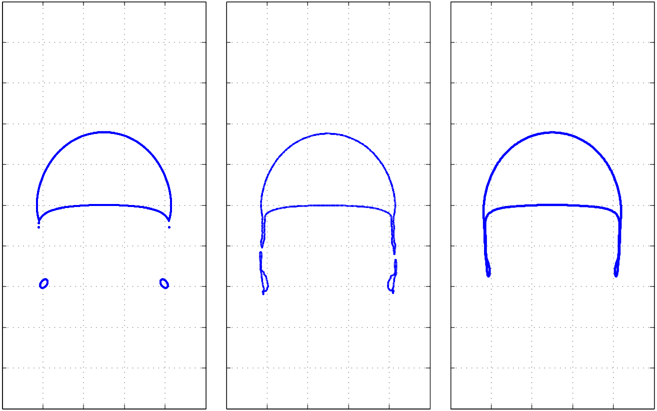
GridLevel	1	2	3	4
Maximum rise velocity, $V_{c,max}$				
TP2D	0.2418	0.2418	0.2419	0.2417
FreeLIFE		0.2427	0.2410	0.2421
MooNMD	0.2418	0.2417	0.2417	0.2417
Incidence time, $t\|_{V_c=V_{c,max}}$				
MooNMD	0.9236	0.9236	0.9249	0.9214
FreeLIFE		0.9000	0.9375	0.9313
TP2D	0.9141	0.9375	0.9281	0.9213

Reference target range

$$V_{c,max} = 0.2417, \qquad t|_{V_c=V_{c,max}} = 0.9213\text{–}0.9214$$

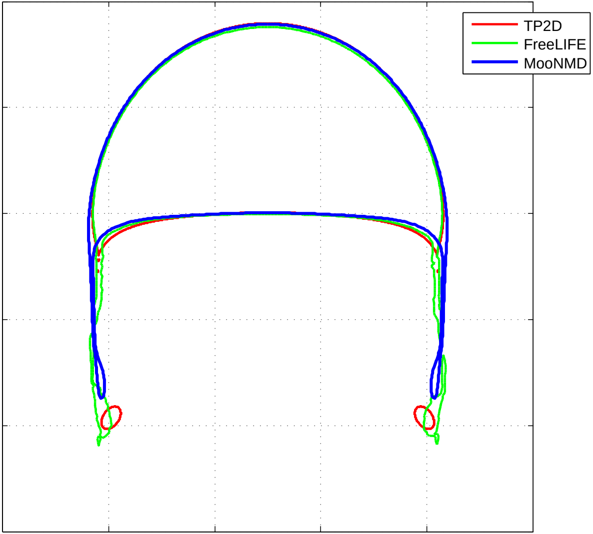
Test Case 2

Visual comparison

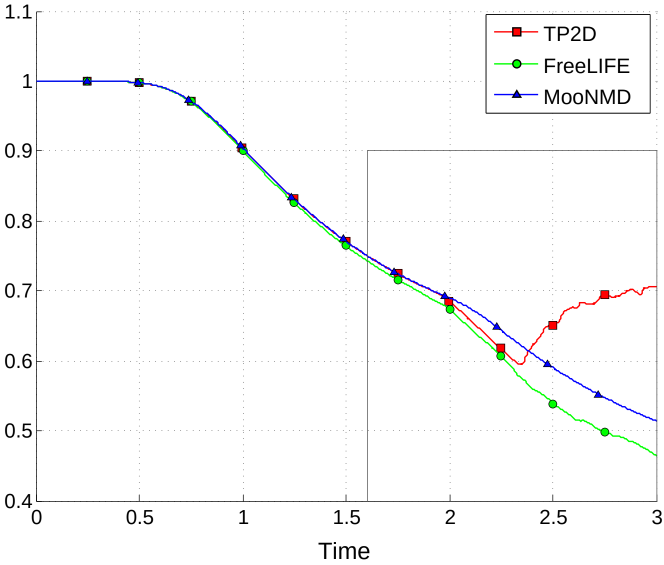


TP2D	FreeLIFE	MooNMD

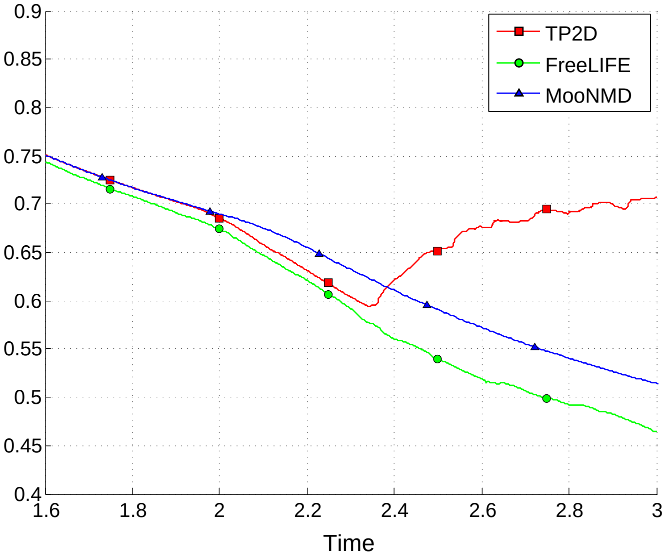
Visual comparison



Benchmark quantities - circularity



Benchmark quantities - circularity



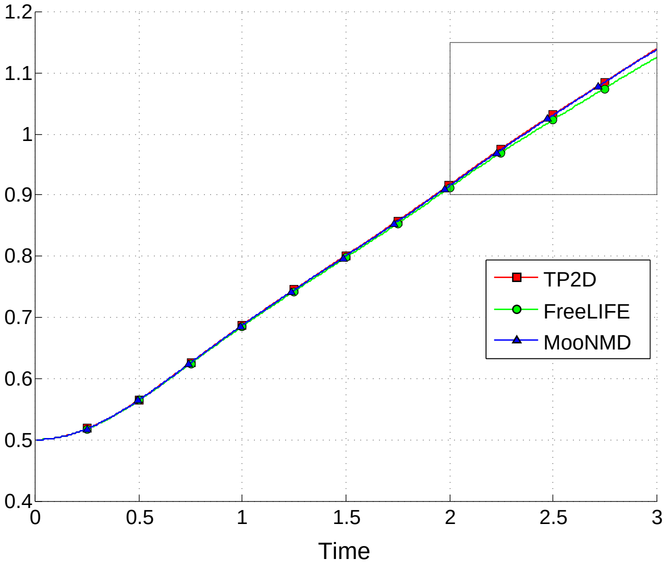
Benchmark quantities - circularity

GridLevel	1	2	3	4	5
Minimum circularity, $\not{c}_{min}$					
TP2D	0.5193	0.5717	0.5946	0.5943	0.5869
FreeLIFE			0.4868	0.5071	0.4647
MooNMD			-	0.5191	0.5144
Incidence time, $t\|_{\not{c}=\not{c}_{min}}$					
TP2D	3.0000	2.4266	2.2988	2.3439	2.4004
FreeLIFE			2.7500	2.8438	3.0000
MooNMD			-	3.0000	3.0000

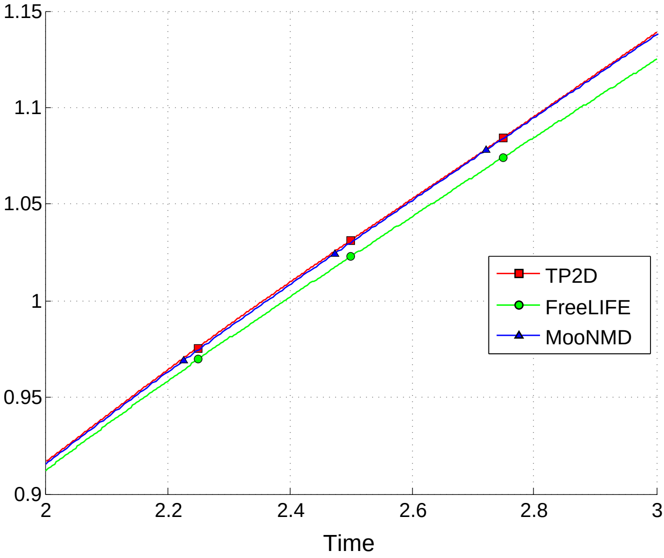
Reference target range

?

Benchmark quantities - center of mass



Benchmark quantities - center of mass



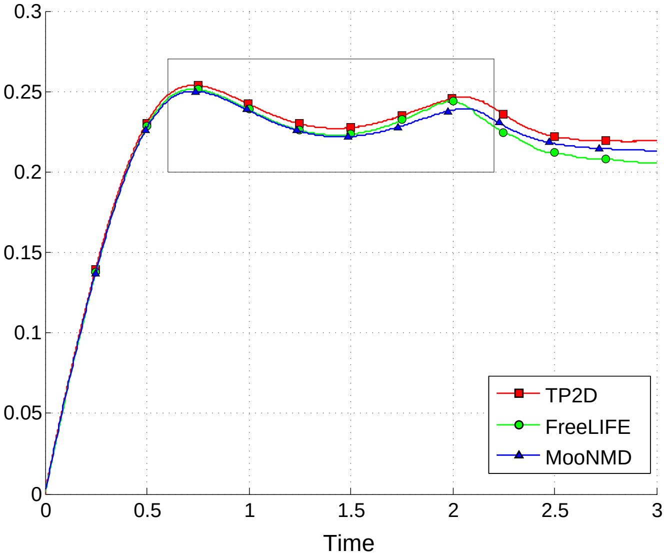
Benchmark quantities - center of mass

GridLevel	1	2	3	4	5
Center of mass, $y_c\big\|_{t=3}$					
TP2D	1.1303	1.1370	1.1377	1.1387	1.1380
FreeLIFE			1.0843	1.1099	1.1249
MooNMD			-	1.1380	1.1376

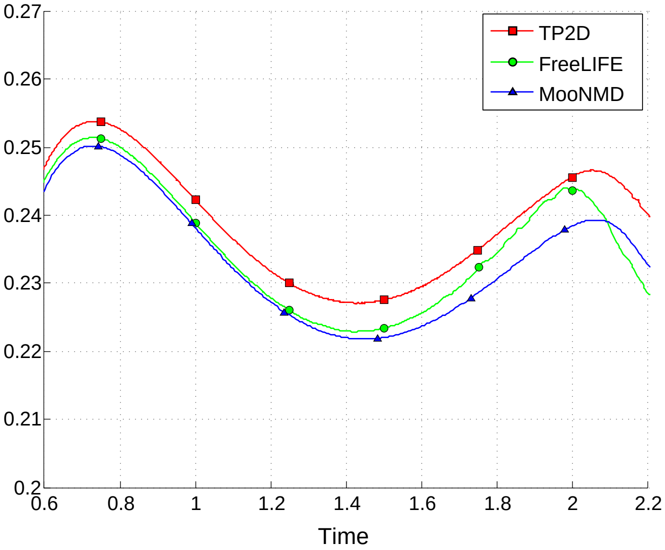
Reference target range

?

Benchmark quantities - rise velocity



Benchmark quantities - rise velocity



Benchmark quantities - rise velocity

GridLevel	1	2	3	4	5
First rise velocity maximum, $V_{c,max\ 1}$					
TP2D	0.2790	0.2638	0.2570	0.2538	0.2524
FreeLIFE			0.2563	0.2518	0.2514
MooNMD			0.2503	0.2502	0.2502
First Incidence time, $t\|_{V_c=V_{c,max\ 1}}$					
TP2D	0.7641	0.7250	0.7430	0.7340	0.7332
FreeLIFE			0.7750	0.7188	0.7281
MooNMD			0.7317	0.7317	0.7317

Reference target range

$$V_{c,max\ 1} = 0.25 \pm 0.01, \qquad t\big|_{V_c=V_{c,max\ 1}} = 0.73 \pm 0.02$$

Benchmark quantities - rise velocity

GridLevel	1	2	3	4	5
Second rise velocity maximum, $V_{c,max\ 2}$					
TP2D	0.2749	0.2597	0.2522	0.2467	0.2434
FreeLIFE			0.2397	0.2384	0.2440
MooNMD			0.2390	0.2393	0.2393
Second Incidence time, $t\|_{V_c=V_{c,max\ 2}}$					
TP2D	1.9375	1.9688	2.0234	2.0553	2.0705
FreeLIFE			1.9875	1.9062	1.9844
MooNMD			2.0650	2.0600	2.0600

Reference target range

?

Conclusions

Conclusions

- Proposed two benchmarks
- Established target reference values for the first benchmark
- Hinted at difficulties during break up in the second benchmark

Participate?

If you would like to participate in this or other numerical benchmarking projects please visit our benchmarking forum at:

www.featflow.de

or send an email to:

ture@featflow.de