Mixed finite element level set method for numerical simulation of immiscible fluids

S. Hysing*

Abstract

A new realization of a finite element level set method for simulation of immiscible fluid flows is introduced and validated on numerical benchmarks. The new method involves a mixed discretization of the dependent variables, discretizing the flow variables with non-conforming Rannacher-Turek finite elements while keeping a simple first order conforming discretization of the level set field. A three step segregated solution approach is employed, first using a discrete projection method to decouple and compute the velocity and pressure separately, after which the level set field can be computed independently.

The developed method is tested and validated on a static bubble test case and on a numerical rising bubble test case for which a very accurate benchmark solution has been established. The new approach is also compared against two commercial simulation codes, Ansys Fluent and Comsol Multiphysics, which shows that the developed method is a magnitude or more accurate and at the same time significantly faster than state of the art commercial codes.

KEY WORDS: numerical simulation; multiphase flow; level set method; finite element method

*Department of Mathematics, Shanghai Jiaotong University, 800 Dongchuan Road, Shanghai 200240, China (shuren.hysing@sjtu.edu.cn, shuren.hysing@math.tu-dortmund.de).

1 INTRODUCTION

Modern computer hardware together with improved simulation algorithms has made computer simulations a commonly used tool by engineers today. Simulations of industrially relevant flows can now seemingly be made both quickly and easily. However, although problems involving complex physical phenomena or large geometries can be simulated, there is still room for improvement with regard to accuracy and computational efficiency. Typical applications involving two-phase flows and free interfaces, which is the main concern of this paper, can for example be to understand droplet generation in inkjet printing and spray processes, to study wave generation and force impact on ships and offshore structures, and design microfluidic lab-on-chip devices for medical analysis.

Numerical simulation of immiscible fluid flows has come a long way since the early Marker-and-Cell method of Harlow and Welch [6] which eventually evolved into the Volume of Fluid (VOF) method by Hirt and Nichols [7]. In these approaches a scalar function for the fluid volume fraction is tracked throughout the simulation from which the interfaces are reconstructed. The level set method by Osher and Sethian [20] was alternatively designed to implicitly track interfaces by representing and embedding them as an iso-contour level of a higher dimensional function. In these Eulerian immersed interface approaches the free boundaries are allowed to move arbitrarily through a fixed computational grid and do not have to be resolved sharply. This simplifies the implementation and can lead to significant performance gains in contrast to moving mesh methods where the interfaces always are aligned with the edges of the grid cells. An approach to combine the advantages of a sharp Lagrangian interface representation with an Eulerian representation of the other flow variables is the Front Tracking method by Unverdi and Tryggvason [27]. All of these methods have their respective strengths and weaknesses and much work has been done to improve them in an effort to yield more accurate and faster algorithms.

With all this in mind a new methodology is introduced for simulation of immiscible fluid flows which essentially consists of combining a non-conforming finite element flow solver with a conforming level set interface tracking method. This technique, although somewhat unconventional, has resulted in a simulation code which has proven to be both significantly faster and at the same time more accurate than two major commercial computational fluid dynamics (CFD) and simulation software tools.

This paper first describes the numerical algorithms and methods necessary to realize the new simulation approach which then is applied to both standard test problems and also benchmarked against the commercial simulation codes Ansys Fluent and Comsol Multiphysics. The following section first presents an efficient method to discretize and realize a solver for the fluid flow and also discusses how to efficiently incorporate surface tension effects. Section 3 focuses on interface tracking, deriving the level set method, discussing its discretization in space and time, level set reinitialization, and computation of normals and curvature. The flow and interface tracking algorithms are combined in Section 4 to establish a complete solution approach. Section 5 presents results from testing and validation of the developed code on both a static bubble test case, as well as a rising bubble benchmark problem for which very accurate reference solutions have previously been obtained. Lastly, a comparison with current commercial codes is presented in Section 6 together with a summary and conclusions.

2 FLOW SOLVER

This section discusses discretization and solution techniques for the Navier-Stokes equations which are the mathematical model equations describing flow of incompressible fluids. The task is to solve the following saddle point system

$$\begin{aligned} \rho(\mathbf{x})\left(\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u}\right) &= -\nabla p + \nabla\cdot\left(\mu(\mathbf{x})(\nabla\mathbf{u} + \nabla\mathbf{u}^T)\right) + \rho(\mathbf{x})\mathbf{g} \quad (1)\\ \nabla\cdot\mathbf{u} &= 0 \end{aligned}$$

for the unknown velocity, $\mathbf{u}$, and pressure, p, in a given domain $\Omega \subset \mathbb{R}^d$. For immiscible fluid flows, where one has a mixture of two or more fluids, the density, ρ, and viscosity, μ, fields will be discontinuous at the $d-1$ dimensional interface, $\Gamma \subset \Omega$, separating the different fluids. A single field representation of them, $\rho = \rho(\mathbf{x})$ and $\mu = \mu(\mathbf{x})$, will only be piecewise constant and thus vary with the spatial coordinates $\mathbf{x} \in \Omega$. The right hand side vector $\mathbf{g}$ represents an external force field such as gravity.

The presented methodology shares concepts from and continues to build on the FeatFlow software suite (short for Finite Element Analysis Tools for Flow) which has been developed to be able to accurately and efficiently simulate single phase laminar fluid flows [1, 26].

2.1 Surface tension effects

When surface tension or capillary effects are present and cannot be considered negligible the following boundary conditions apply

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

at the interface Γ. Here $\hat{\mathbf{n}}$ denote the interface normal and $[\mathsf{A}]|_\Gamma$ is the jump of property A across the interface. These conditions imply continuity of the velocity in the normal direction across the interface and also a jump in the normal stress proportional to the coefficient of surface tension, σ, and the curvature of the interface, κ. The interface conditions can also conveniently be rewritten as volumetric forces and then take the form

$$\mathbf{f}_{st} = \sigma\kappa\hat{\mathbf{n}}\delta(\Gamma,\mathbf{x}) \quad (2)$$

where $\delta(\Gamma,\mathbf{x})$ is a Dirac delta function localizing the surface tension force to the interface between the different fluids. The force term $\mathbf{f}_{st}$ will be added to the right hand side of the momentum equations (1) similar to the gravity force. This means that the term will be evaluated explicitly in time, that is the normal and curvature contributions will be calculated from the interface position at the previous time step. This is called an immersed interface (or alternatively an immersed boundary) approach and has its roots in the early work on blood flow by Peskin [21], and its extension to volume of fluid (VOF) calculations with surface tension forces by Brackbill, Kothe, and Zemach who dubbed it the continuum surface force (CSF) method [2].

The surface tension force can also be rewritten semi-implicitly in time by first expressing the curvature as a function of the interface coordinates $\mathbf{x}|_\Gamma$. After applying partial integration and using the fact that the interface position at the

new unknown time step can be rewritten as the old position plus an update, $\mathbf{x}|_\Gamma^{n+1} = \mathbf{x}|_\Gamma^n + \Delta t\, \mathbf{u}$, one will arrive at the following weak form of the surface tension force

$$\mathbf{f}_{st} = \int_\Omega \sigma\ \delta(\Gamma, \mathbf{x})\ \underline{\nabla}(\mathbf{x}|_\Gamma^n) \cdot \underline{\nabla}\mathbf{v}\ d\mathbf{x} \tag{3}$$

$$+ \Delta t \int_\Omega \sigma\ \delta(\Gamma, \mathbf{x})\ \underline{\nabla}\mathbf{u} \cdot \underline{\nabla}\mathbf{v}\ d\mathbf{x} \tag{4}$$

where $\mathbf{v}$ represents suitably chosen test functions and $\underline{\nabla} = \nabla - \hat{\mathbf{n}}(\hat{\mathbf{n}} \cdot \nabla)$ denotes the gradient in the tangential direction of the interface. The first term (3) is simply another form of the explicit CSF force (2), while the additional second term (4) is implicit in the velocity $\mathbf{u}$. The advantage this semi-implicit discretization has over the purely explicit one is that the additional new term represents diffusion in the tangential direction of the interface. This stabilizing diffusion is scaled by both the time step and the coefficient of surface tension which results in a more physical implementation of capillary effects. An increased coefficient of surface tension or a larger time step will now generate more interface diffusion which balances the destabilizing source term. In this way the effects of the capillary time step restriction, $\Delta t_{num}^{(ca)} < \sqrt{\langle\rho\rangle\, h^3/2\pi\sigma}$, can be reduced allowing for larger time steps and an overall more efficient method [11].

2.2 Temporal discretization

The first step in the numerical discretization process is to select an appropriate time stepping scheme. A good compromise between accuracy, robustness, computational cost, and ease of implementation is the θ-scheme approach. This method allows one to simultaneously implement and easily switch between the first order Backward Euler, second order Crank-Nicolson, and multi-step schemes such as the strongly A-stable Fractional-step-θ-scheme. The θ-scheme applied to the Navier-Stokes equations results in the following general semi-discrete system for each time step

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

$$\begin{aligned} \rho\frac{\mathbf{u} - \mathbf{u}^n}{\Delta t} + \theta\left[\rho(\mathbf{u}\cdot\nabla)\mathbf{u} - \nabla\cdot\left(\mu(\nabla\mathbf{u} + (\nabla\mathbf{u})^T)\right)\right] + \nabla p &= \mathbf{b}^{n+1} \\ \nabla\cdot\mathbf{u} &= 0 \end{aligned} \tag{5}$$

with right hand side

$$\begin{aligned} \mathbf{b}^{n+1} &= \theta\mathbf{f}^{n+1} + (1-\theta)\mathbf{f}^n \\ &\quad - (1-\theta)\left[\rho^n(\mathbf{u}^n\cdot\nabla)\mathbf{u}^n - \nabla\cdot\left(\mu^n(\nabla\mathbf{u}^n + (\nabla\mathbf{u}^n)^T)\right)\right]. \end{aligned} \tag{6}$$

The parameter θ is chosen according to the time stepping scheme, $\theta = 1$ for the Backward Euler scheme and $\theta = 1/2$ for the Crank-Nicolson scheme. The Fractional-step-θ-scheme is a multi-step scheme involving the following time stepping parameters [3]

$$\theta = 1 - \frac{\sqrt{2}}{2}, \qquad \theta' = 1 - 2\theta, \qquad \alpha = \frac{1-2\theta}{1-\theta}, \qquad \beta = 1 - \alpha$$

where the whole time interval Δt is split into the following three sub time steps

$$\rho \frac{\mathbf{u}^{n+\theta} - \mathbf{u}^{n}}{\Delta t/3} + \alpha\theta N(\mathbf{u}^{n+\theta})\mathbf{u}^{n+\theta} + \theta \nabla p^{n+\theta} = \\ = \quad \alpha\theta \mathbf{f}^{n+\theta} + \beta\theta \mathbf{f}^{n} - \beta\theta N(\mathbf{u}^{n})\mathbf{u}^{n} \\ \nabla \cdot \mathbf{u}^{n+\theta} = 0, \quad (7)$$

$$\rho \frac{\mathbf{u}^{n+1-\theta} - \mathbf{u}^{n+\theta}}{\Delta t/3} + \beta\theta' N(\mathbf{u}^{n+1-\theta})\mathbf{u}^{n+1-\theta} + \theta' \nabla p^{n+1-\theta} = \\ = \quad \beta\theta' \mathbf{f}^{n+1-\theta} + \alpha\theta' \mathbf{f}^{n+\theta} - \alpha\theta' N(\mathbf{u}^{n+\theta})\mathbf{u}^{n+\theta} \\ \nabla \cdot \mathbf{u}^{n+1-\theta} = 0, \quad (8)$$

$$\rho \frac{\mathbf{u}^{n+1} - \mathbf{u}^{n+1-\theta}}{\Delta t/3} + \alpha\theta N(\mathbf{u}^{n+1})\mathbf{u}^{n+1} + \theta \nabla p^{n+1} = \\ = \quad \alpha\theta \mathbf{f}^{n+1} + \beta\theta \mathbf{f}^{n+1-\theta} - \beta\theta N(\mathbf{u}^{n+1-\theta})\mathbf{u}^{n+1-\theta} \\ \nabla \cdot \mathbf{u}^{n+1} = 0. \quad (9)$$

Here $N(\mathbf{w})\mathbf{u} = \left[\rho(\mathbf{w} \cdot \nabla)\mathbf{u} - \nabla \cdot \left(\mu(\nabla \mathbf{u} + (\nabla \mathbf{u})^T)\right)\right]$ represents the in compact form of the nonlinear convective and diffusive terms.

2.3 Spatial discretization

The next step is to discretize the unknown dependent variables $\mathbf{u}$ and p in space. Commonly employed discretization schemes are finite volume (FVM), finite difference (FDM), and finite element methods (FEM). A suitable but not always obvious choice is the non-conforming Rannacher-Turek $\tilde{Q}_1 Q_0$ Stokes finite element pair which has been shown to be very robust and computationally efficient for solving the incompressible Navier-Stokes equations on arbitrary grids [24, 26]. In the following, this methodology is used and extended to be able to efficiently treat two-phase flows involving immiscible fluids.

The first step in deriving the finite element formulation of the Navier-Stokes equations follows the usual procedure of multiplying the equations (5) with arbitrary test functions $\mathbf{v}$ and integrating over the whole domain Ω. Partial integration by using using the Gaussian theorem on the second order viscous diffusion term is also applied, which reduces the smoothness requirements of the involved variables.

The discretization step itself consists of subdividing or triangulating the domain Ω into smaller cells. This triangulation is denoted by $\mathcal{T}_h$ where $h(K)$ is the diameter or width of cell K. The employed Rannacher-Turek elements requires that $\mathcal{T}_h$ is built up of quadrilateral cells in two dimensions or hexahedral cells in three dimensions. The approximation of the velocity components, u_i,

in each cell is then represented as multilinear polynomial shape functions, $\tilde{Q}_1$, defined as

$$\tilde{Q}_1(K) := \left\{ q \circ \psi_K^{-1} : \quad q \in span\left\langle 1, x_i, x_i^2 - x_{i+1}^2, i = 1, \ldots, d \right\rangle \right\}$$

where $\psi_K : \hat{R} \to K$ is the multilinear (bilinear in two dimensions and trilinear in three dimensions) transformation from the reference element $\hat{R} = [-1, 1]^d$ to the cell K. The corresponding degrees of freedom are determined by either of the functionals

$$F_E^{(a)} := |E|^{-1} \oint_E v \; ds \qquad \text{or} \qquad F_E^{(b)} := v(m_E)$$

where $E \subset \partial \mathcal{T}_h$ denotes the cell edges (or faces in three dimensions). Using the functional $F_E^{(a)}$ will result in degrees of freedom corresponding to the mean values over the edges, and using $F_E^{(b)}$ will similarly correspond to pointwise values on the edge midpoints m_E. Alternatively, one can employ the non-parametric counterparts which, although computationally expensive, are better suited on grids with high anisotropies [26].

The pressure field, p, is approximated by piecewise constant values on each cell, the so called Q_0 element, defined by the space

$$L_h = \{ q_h \in L_0^2 : q_{h|K} \equiv \text{const}, \quad \forall \; K \in \mathcal{T}_h \}.$$

Finally, the test function space $\mathbf{v}$ is normally taken as the same function space which approximates the dependent variables, the standard Galerkin formulation, but can also be modified to include certain desirable properties, for example stabilization of convective terms in the form of streamline diffusion or streamline upwind Petrov-Galerkin (SUPG) stabilization [4, 9].

Discretization in both time and space now yields the following saddle point system which must be solved to advance the solution:

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

$$\begin{cases} S\mathbf{u} + \Delta t B p = \mathbf{b} \\ B^T \mathbf{u} = 0 \end{cases} \tag{10}$$

where the system or iteration matrix S is defined as

$$S = \left[M_\rho + \theta \Delta t N_{\rho,\mu}(\mathbf{u}) \right].$$

Here M_ρ denotes the density weighted mass matrix

$$M_\rho = \int_\Omega \rho(\mathbf{x}) \mathbf{u} \cdot \mathbf{v} \, d\Omega$$

arising from the discretization of the time derivative. The transport operator $N(\cdot)$ is given by

$$N_{\rho,\mu}(\mathbf{w}) = \int_\Omega \rho(\mathbf{x}) \left((\mathbf{w} \cdot \nabla) \mathbf{u} \right) \cdot \mathbf{v} + \mu(\mathbf{x}) \left(\nabla \mathbf{u} + (\nabla \mathbf{u})^T \right) \cdot \nabla \mathbf{v} \, d\Omega$$

and contains the nonlinear convective and diffusive contributions which, similar to the mass matrix, depend on the variable density and viscosity fields. B and B^T are simply discrete analogs of the gradient and divergence operators, and are used to include the pressure gradient and to enforce the incompressibility constraint. The discrete right hand side is finally given by

$$\mathbf{b} = \int_\Omega \mathbf{b}_h^{n+1} \cdot \mathbf{v}\, d\Omega,$$

where $\mathbf{b}_h^{n+1}$ is the discrete analog to equation (6). A suitable and efficient method to solve this system, with proper treatment of the pressure and continuity equation, will be discussed next.

2.4 Discrete projection method

Due to the incompressibility constraint special care must be taken when solving the saddle point system (10). Standard direct and iterative solvers which are capable of dealing with zeros on the diagonal can be applied to invert the coupled system directly. However, as the mesh size decreases and the number of unknowns increases direct solvers will require huge amounts of memory while iterative solvers will require more and more iterations to converge.

Two-phase flow applications most often are time dependent in nature and involve phenomena that occur on very small time scales, for example break up and coalescence, which poses natural restrictions on the maximum allowable time step size. Projection method approaches have previously been shown to be very efficient for solving time dependent incompressible flow problems [25]. The essence of this methodology is to split the momentum equations and the continuity equation, leading to smaller Poisson type problems which very efficiently can be solved independently from each other. A correction or projection step must also be added to account for the incompressibility constraint.

The projection method, in this case the discrete pressure Schur complement approach, applied to (10) involves the following steps to obtain the new velocity field $\mathbf{u}^{n+1}$ and pressure field p^{n+1}:

1. Solve the momentum equations for the approximate velocity field $\tilde{\mathbf{u}}$ (while ignoring the incompressibility constraint):

$$S\tilde{\mathbf{u}} \;=\; \mathbf{b}^{n+1} - \Delta t B p^n$$

2. Construct the pressure right hand side:

$$f_p \;=\; \frac{1}{\Delta t} B^T \tilde{\mathbf{u}}$$

3. Approximate the exact pressure matrix $P^* = B^T S^{-1} B$ with $P = B^T M_{\rho,l}^{-1} B$ and solve the pressure Poisson problem:

$$Pq \;=\; f_p$$

4. Update the pressure field:

$$p^{n+1} \;=\; p^n + \alpha_R q + \alpha_D M_{p,l}^{-1} f_p$$

where α_R and α_D are appropriately chosen damping parameters, and $M_{p,l}^{-1}$ is the discrete lumped pressure mass matrix.

5. Project the approximate velocity field to a divergence free space:

$$\mathbf{u}^{n+1} = \tilde{\mathbf{u}} - \Delta t M_{\rho,l}^{-1} B q$$

The key to efficiency lies in optimizing the two computationally intensive parts, the sparse matrix inversions in steps 1 and 3, as well as the finite element matrix assembly.

An efficient method to solve the discretized momentum equations in step 1 of the discrete projection method is to use a geometric multigrid approach. The idea behind multigrid is to assemble and iteratively solve the linear systems on a sequence of grids. This allows for the slowly converging low frequency errors on the finest grid to quickly be filtered out on the coarser grids (the low frequency error is seen as having a higher frequency on the coarser grids). A near linear efficiency can in this way be achieved in the optimal case (with linear meaning that the cost of solving the systems increase linearly with the number of degrees of freedom) [5]. This is in contrast to standard iterative solvers which require an increasing number of iterations to converge as the computational grids are refined.

For solving the momentum equations and establishing the approximate velocity fields an F-cycle multigrid solver with two or three grid levels is used. A stabilized biconjugate gradient method (BiCGStab) is chosen as a smoother applying one smoothing step between grid level changes, and successive over-relaxation (SOR) is used as a solver on the coarsest grids. More details on implementing this multigrid approach to solving the momentum equations is described in references [12, 26].

The solution of the pressure Poisson problem, step 3 above, is the second computationally expensive part of the discrete projection method. To handle this efficiently a similar approach to solving the momentum equations is used. An important difference between single and multiphase flow is in the choice and construction of the lumped mass matrix, which is the main constituent of the approximate projection matrix $P = B^T M_{\rho,l}^{-1} B$. For multiphase flow applications the lumped mass matrix needs to be weighted with the discontinuous density field and there are a number of ways of accomplishing this leading to quite different results.

The two main approaches is either to first lump the standard mass matrix and then scale it with the density, or first assemble the full density weighted mass matrix exactly and lump it afterwards. The first approach is easy and cheap, the standard row-sum lumped mass matrix is computed and stored once, after which it is multiplied with a corresponding density field vector. The density field can either be evaluated pointwise in the nodes (or midpoints) or alternatively averaging on the cells first, after which a mean density for each cell can be evaluated.

The second alternative is to construct the mass matrix exactly to resolve the density jumps on the elements accurately, after which a lumping procedure can be applied. For a mass matrix constructed from the $\tilde{Q}_1$ basis functions one can in general not use the standard row-sum lumping procedure since negative diagonal

entries can be created. This is due to the combination of the density jump and non-positiveness of parts of the basis functions which can create dominating off-diagonal entries. However, there are two other approaches available, diagonal lumping where only the positive diagonal entries are kept, and HRZ-lumping (from Hinton, Rock, and Zienkiewicz [8]) where the diagonal mass matrix is scaled locally so that the total mass of each element is preserved.

The different approaches were evaluated by simulating a standard rising bubble test problem and recording the average number of multigrid iterations required to reduce the residual norm in the pressure Poisson equation to 10^{-10}. The results are shown in Table 1. It is clear that the exact assembly plus HRZ-mass lumping is by far the best choice, requiring on average around 10 iterations. For the variants where the standard lumped mass matrix is scaled by the density it is preferable to calculate an average density for each cell. This approach did however take 2-3 times more iterations to converge compared to using the HRZ-lumped mass matrix. The most costly method is to scale with the density evaluated directly in the nodes, requiring over 100 iterations to achieve convergence on the coarser grid levels.

Table 1: Average number of multigrid iterations required to solve the pressure Poisson problem. (Pw.) denotes pointwise evaluation of the density, (Avg.+Pw.) density averaging before pointwise evaluation, and (Exact+HRZ) exact assembly with HRZ-lumping.

Grid level	Pw.	Avg.+Pw.	Exact+HRZ
3	111	31	14
4	109	33	12
5	40	23	8
6	35	23	7
7	26	27	5

2.5 Convective stabilization

High Re-number flows, where convective terms dominate over diffusive often require special numerical treatment. This is mainly due to the limited resolution (the number of cells) that can be afforded in the simulations. The unresolved subgrid effects will eventually cause oscillations and unphysical solution behaviors if not suppressed. A mechanism to handle this consists of locally adding small amounts of numerical diffusion to counterbalance the dominating convective terms.

An elegant approach to add artificial stabilization in the finite element context consists of introducing the following modification to the test function space $\mathbf{v}_{PG} := \mathbf{v} + \delta\nabla\mathbf{v}$, where $\delta = \delta(Re_h, h)$ is locally tuned to apply the correct amount of stabilization. This classical Petrov-Galerkin (PG) approach only introduces artificial diffusion in the flow or streamline directions, and is thus called streamline diffusion (SD) or streamline upwind Petrov-Galerkin (SUPG) [4, 9]. The method is consistent, meaning that the stabilization disappears for the exact solution, linear with respect to the solution variables, and is also easy

to implement. However, a major drawback of the streamline diffusion approach is that it is not monotonicity preserving thus allowing unphysical under- and over-shoots to appear. Depending on the nature of the phenomena under study it can be very critical to limit or even completely suppress this behavior (for example when solving for turbulence variables or chemical reactions).

An alternative is to use total variation diminishing (TVD) and flux-corrected transport (FCT) schemes. These schemes allow for very accurate solutions at the cost of introducing an additional nonlinearity. However, if treated properly, or if the problem is already nonlinear, the additional cost will be quite low. A very elegant and general approach used here to introduce TVD, in particular with respect to the FEM context, is to use the FEM-TVD method developed by Kuzmin [15, 16]. In this type of approach one modifies the standard Galerkin discretization by adding diffusive and antidiffusive fluxes on the matrix level. The fluxes are determined by comparing the physical diffusive and convective contributions, computing the required stabilizing flux terms, and applying an appropriate flux limiter. The reader is referred to the works of Kuzmin et al. for further details [17, 18].

2.6 Nonlinear iteration techniques

The convective term in the momentum equations contains a non-linearity which must be treated appropriately. An efficient way to solve the nonlinear model problem $A(u)u = f$ is to use the following iterative defect correction approach

$$u_j = u_{j-1} + \omega\ C^{-1} r_{j-1}, \qquad j = 0, 1, 2, \ldots$$

where $r_l = f - A^{-1}(u_l)u_l$ is the defect vector in iteration l, and ω is a damping parameter. The preconditioning matrix C should ideally be chosen so that it is cheap to construct and easy to invert while still allowing for fast convergence. Typically one will use $C = A$ for the standard fixed point approach while full Newton schemes require more elaborate constructions. The iterative procedure is stopped when a predefined convergence criterion has been reached, for example when the norm of the residual error $||r_j||$ has decreased sufficiently or the solution difference between iterates $||u_j - u_{j-1}||$ is small enough.

The iterative defect correction scheme applied to the Navier-Stokes equations leads to the following linearization of the convective term

$$(\mathbf{u}_j \cdot \nabla)\mathbf{u}_j \approx (\mathbf{u}_{j-1} \cdot \nabla)\mathbf{u}_j$$

where $\mathbf{u}_{j-1}$ is the solution from the previous step in the defect loop. An alternative for time dependent problems is to use extrapolation backwards in time by replacing

$$(\mathbf{u}^{n+1} \cdot \nabla)\mathbf{u}^{n+1} \quad \text{by either} \quad (\mathbf{u}^{n} \cdot \nabla)\mathbf{u}_j \quad \text{or} \quad ((2\mathbf{u}^{n} - \mathbf{u}^{n-1}) \cdot \nabla)\mathbf{u}_j.$$

In certain cases it is even possible to treat the convective term completely explicit transferring it to the right hand side. Although these extrapolation techniques remove the nonlinearities, and thus are computationally favorable, they should be used with caution since the time step size may have to be reduced dramatically if the nonlinearity is strong.

3 LEVEL SET SOLVER

The level set method introduced by Osher and Sethian [20] has proved to be applicable in many diverse fields such as image processing, crystal growth, inverse problems, and multiphase flow [19]. The main idea of the level set method is to embed an interface $\Gamma(t)$ (represented by a curve in two dimensions or surface in three dimensions) in a higher dimensional function ϕ, that is

$$\Gamma(t) = \{\mathbf{x} \in \mathbb{R}^d \mid \phi(\mathbf{x}, t) = v_{ls}\},$$

where v_{ls} is the contour level or isosurface value implicitly representing the interface. The choice of v_{ls} is arbitrary but is usually taken as zero, since it is then possible to identify the different phases by simply using the sign of the level set function. It is appropriate to initialize ϕ as a signed distance function

$$\phi(\mathbf{x}, 0) = d(\Gamma, \mathbf{x}) = \begin{cases} dist(\Gamma, \mathbf{x}), & \mathbf{x} \in \Omega_1, \\ v_{ls}, & \mathbf{x} \in \Gamma, \\ -dist(\Gamma, \mathbf{x}), & \mathbf{x} \in \Omega_2, \end{cases}$$

where Ω_1 and Ω_2 denote the two regions that the fluids occupy. Advantages of using a distance function as a level set field is that it for the most part is smooth and simplifies construction and regularization of Heaviside and delta functions. Methodologies using local grid adaptation and grid deformation can also make use of the distance function to quickly and easily identify where to refine the grid. Additionally, normals, $\hat{\mathbf{n}}$, and curvature, κ, can globally be defined as

$$\hat{\mathbf{n}} = \frac{\nabla\phi}{|\nabla\phi|}, \qquad \kappa = -\nabla \cdot \hat{\mathbf{n}}. \tag{11}$$

The starting point for deriving an evolution equation for the level set function is to recognize that the following must hold for the moving interface (from here on taken as the zero level set, that is $v_{ls} = 0$)

$$\phi(\mathbf{x}(t), t) = 0.$$

Direct differentiation with the chain rule yields

$$\frac{\partial\phi}{\partial t} + \nabla\phi \cdot \frac{\partial \mathbf{x}(t)}{\partial t} = 0.$$

The speed with which the front propagates in the normal direction is given by $F = \hat{\mathbf{n}} \cdot \frac{\partial \mathbf{x}(t)}{\partial t}$. Using this and the definition of the normal vector in (11) results in the following evolution equation

$$\frac{\partial\phi}{\partial t} + F|\nabla\phi| = 0$$

which must hold globally for all values of ϕ. The speed function F can depend on many variables such as mean curvature, external forces, but when coupled with the Navier-Stokes equations will only depend on the fluid velocity, that is $F = \hat{\mathbf{n}} \cdot \mathbf{u}$, which gives

$$\frac{\partial\phi}{\partial t} + (\mathbf{u} \cdot \nabla)\phi = 0, \tag{12}$$

where the definition of the normal (11) has been used once more.

What essentially has been achieved is a fully implicit formulation of the interface and its evolution in space at the cost of operating in a higher dimension. Additionally, geometrical properties such as interface normals and curvature are also implicitly defined and are possible to reconstruct globally.

3.1 Numerical treatment

The level set equation (12) is a pure hyperbolic transport equation and thus allows for application of standard solution tools, such as those used to treat the momentum equations in Section 2. As for the Navier-Stokes equations time discretization can also be implemented here by applying the single step θ-scheme, which results in the following problem formulation

Given ϕ^n *and the time step* $\Delta t = t^{n+1} - t^n$*, then solve for* $\phi = \phi^{n+1}$

$$\frac{\phi - \phi^n}{\Delta t} + \theta(\mathbf{u} \cdot \nabla)\phi = b$$

with the right hand side

$$b = (\theta - 1)(\mathbf{u} \cdot \nabla)\phi^n.$$

Alternatively, one can as before apply the A-stable Fractional-step-θ-scheme which leads to a three step method analogous to equations (7)-(9).

It is natural to choose the same discretization scheme in space as for the Navier-Stokes equations, in this case the finite element method. However, a continuous representation is appropriate in the level set context since the interfaces and the level set function for the most part should be smooth. By using the same approximation order as for the velocities the choice falls upon first order continuous basis functions, the Q_1 space, which is defined by

$$Q_1(K) = \left\{ q \circ F_K^{-1} : \quad q \in span\left\langle 1, x_i, x_i x_{\mathrm{mod}(i,d)+1}, i = 1, \ldots, d \right\rangle \right\}.$$

After discretization in space, the discrete form of the level set equation will read

$$\begin{aligned} [M_{(l)} + \Delta t \theta A]\phi^{n+1} &= b^n, \\ b^n &= [M_{(l)} - \Delta t(1-\theta)A]\phi^n, \end{aligned} \tag{13}$$

where $M_{(l)} = \int_\Omega v_1 v_2 \, d\Omega$ is the mass matrix which can be lumped if appropriate. A is the transport matrix, responsible for convecting the level set function and thus also implicitly the interface, and is given by

$$A = \int_\Omega (\mathbf{u} \cdot \nabla) v_1 v_2 \, d\Omega. \tag{14}$$

Here v_1 and v_2 denote the Q_1 basis and test function spaces, respectively. A is a trilinear form involving the explicit velocity field $\mathbf{u}$ which must be evaluated appropriately in the assembly step. Since $\mathbf{u}$ is continuously changing A needs to be re-assembled in each time step in contrast to the mass matrix which only has to be computed once.

Since equation (12) is a hyperbolic transport equation without including any diffusion, some form of artificial stabilization is necessary. At first it might seem that a linear scheme, such as the SUPG streamline diffusion scheme described previously, is more appropriate than FEM-TVD which renders the problem nonlinear. In practice this is not the case since the cost of solving the level set equation for the most part can be rendered almost negligible [12]. Furthermore FEM-TVD has the additional benefits of generally being more accurate than linear stabilization schemes, monotonicity preserving, and also completely free of artificial tuning parameters [16, 17, 18].

For the numerical solution and inversion of the linear system in equation (13) one can apply standard iterative techniques, such as BiCGStab and multigrid schemes. When using FEM-TVD an iterative defect approach, like the one described in Section 2.6, is necessary to treat the added nonlinearity. To achieve fast convergence the preconditioning matrix, C, in the defect loop should as closely as possible resemble the iteration matrix

$$C \approx [M_{(l)} + \Delta t\, \theta A].$$

It does not need to be exact, and can in fact be approximated with just the mass matrix, M, if the time steps are small enough. If the mass matrix also is lumped, M_l, the inversions become trivial and will cost next to nothing. For larger time steps this approach will usually also work but will require more iterations in the defect correction loop to converge.

3.2 Reinitialization

Even if the level set function is initialized as a perfect distance function it will generally distort significantly with time causing convergence difficulties and reducing accuracy for normal and curvature computations. The velocity with which the level set field is transported will preserve the distance function property if $\nabla\phi \cdot \nabla F = 0$ is fulfilled, which means that the normal velocity has to be constant along the characteristics normal to the interface. This is unfortunately not the case for the velocity fields arising from fluid flow simulations. To remedy this it is common practice to periodically apply what is known as redistancing or reinitialization of the distorted level set field to recover the distance function property. Reinitialization is equivalent to solving the Eikonal equation

$$|\nabla\phi(\mathbf{x})| = F(\mathbf{x}) \tag{15}$$

with boundary condition $\phi(\mathbf{x}) = 0,\ \mathbf{x} \in \Gamma$. The speed function is here taken as unity, $F(\mathbf{x}) = 1$, in order to recover the standard distance function.

A reinitialization method must fulfill a number of requirements to be of practical interest . Firstly, the chosen method should ideally not move the zero level set interface, which would cause unphysical mass loss. Secondly, it should be as accurate as possible, since an accurate level set field will result in better normal and curvature computations. Thirdly, the computational overhead can not be so significant that the computations are dominated by the redistancing step. After carefully examining and comparing five different reinitialization algorithms in [10] the one method that proved to posses the best combination of speed and accuracy, especially on very dense meshes, was the fast marching method by Sethian [23].

The fast marching method takes into account the characteristics of the solution, knowing that information only propagates outward from the zero level set. Starting from there, each grid point is updated node by node in order of increasing distance in an upwind fashion. The key to efficiency of fast marching is the realization of a heap structure for the uncorrected nodes and thus the algorithmic complexity of the fast marching method can be estimated to be order $O(N\ log\ N)$ where N is the number of nodes [23]. The fast marching reinitializtion must first be initialized by computing and correcting the distance of the nodes belonging to cells intersected by the interface. This is accomplished by computing and choosing the smallest Euclidean distance to the closest interface segments.

3.3 Normal and curvature computation

Geometric properties such as normal vectors, $\hat{\mathbf{n}}$, and curvature, κ, can easily be recovered globally from the level set function. Direct differentiation of the level set function ϕ again gives

$$\hat{\mathbf{n}} = \frac{\nabla\phi}{|\nabla\phi|}, \quad \kappa = -\nabla \cdot \hat{\mathbf{n}}.$$

Although the finite element method allows for this approach, it is not the best alternative since accuracy is lost when differentiating ϕ. The recovered normals will also be discontinuous at the cell edges if C^0-functions are used (for example with the Q_1 elements used in this approach). This will also cause ambiguous evaluations necessitating some form of averaging procedure. Furthermore, first order basis functions are not two times differentiable. It is therefore preferable to find another method to reconstruct the curvature.

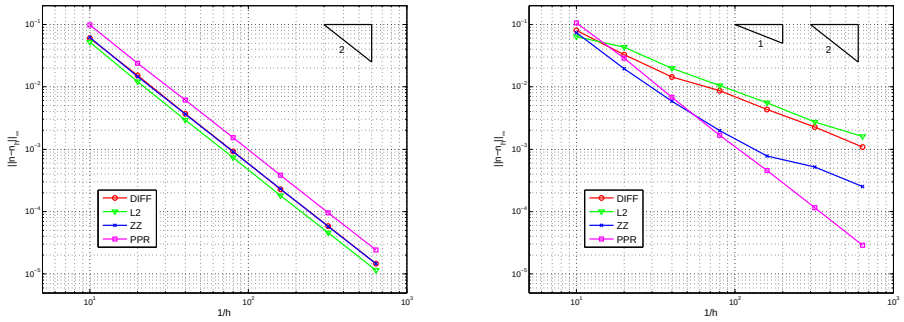
More accurate approaches include reconstruction via L^2-projection and patch recovery techniques. The L^2-projection approach minimizes the following variational forms

$$M_{(l)} n_{i,h} = B_i \phi_h, \quad M_{(l)} \kappa_h = -\sum_i (B_i n_{i,h}), \qquad i = 1, \ldots, d$$

where $M_{(l)}$ is the (possibly lumped) mass matrix, $n_{i,h}$ the recovered normal vector in direction i, and B_i the gradient matrix $B_i = \int \frac{\partial v_1}{\partial x_i} v_2 \, d\Omega$. An underscore h indicates the discrete counterpart to the continuous variable. Once recovered, the discrete normals and curvature now exist in the corresponding finite element space and can be evaluated accordingly. If the mass matrix is lumped the inversion is trivial, otherwise appropriate linear solvers have to be applied.

The main idea behind patch based gradient recovery techniques is to use information available in the surrounding cells to construct a more accurate representation of the gradients. In the following the ZZ-patch recovery technique (from Zienkiewicz and Zhu [29, 30]) and the PPR-technique (short for polynomial preserving recovery [28]) will be explored. The idea of the ZZ-technique is to evaluate the gradients in superconvergent points of the cells neighboring the node of interest, after which a local least-squares fitting is applied. This construction has been shown to achieve better convergence on regular grids than otherwise possible [29]. The PPR-technique alternatively constructs a high order polynomial around the evaluation point from which the gradients can be recovered in a robust way.

The described gradient recovery techniques have been applied to a test problem in order to assess the accuracy of the resulting normal and curvature fields. The test configuration considered a circle with radius 0.25 centered in a 2×2 domain. Errors could easily be calculated since the analytic normal and curvature fields around a circle are known. Figure 1(a) shows the maximum norm of the error of the normal reconstruction, which was evaluated pointwise along the circumference of the circle. The computations were performed for regular refinements of a perfect tensor product grid, and it is clear that all methods yielded second order convergence. However, when the grid nodes were stochastically perturbed by 20% some clear differences became apparent (Figure 1(b)). The direct differentiation (DIFF) and L^2-projection (L2) methods now only converged with first order accuracy. The ZZ-technique still converged with second order on the coarser grids but decreased to first order as the grids were refined. The PPR approach was as robust as before and achieved full second order convergence.

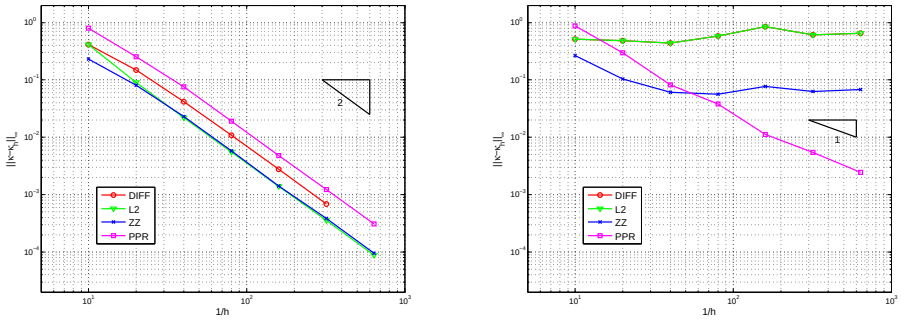


(a) No perturbation

(b) 20% perturbation

Figure 1: Normal reconstruction errors on tensor product grids with and without perturbation of the nodes.

The curvature must be recovered in a second step using the previously computed normals. In this way it is essential that the normals are recovered as exact as possible in order to be able to compute an accurate representation of the curvature. Figure 2(a) shows the results for the tensor product grids which, as for the normals, achieved full second order convergence for all methods. This is not surprising since the recovery methods are the same, and the employed grids are computationally favorable. The absolute values of the curvature errors were a magnitude higher than for the normals, but this is also to be expected since some accuracy must be lost in the additional reconstruction step. When the grids were perturbed on the other hand the results were completely different (Figure 2(b)). Now the direct differentiation (DIFF) and L^2-projection (L2) methods completely stopped converging (note that their respective curves overlap each other). The ZZ-patch recovery shows a quite low initial error but also stopped converging as the grid was refined. The only method that still converged was the PPR-technique. Clearly, if one works with anything but perfectly regular grids, the PPR-technique, although expensive, is the safest choice giving at least first order convergence in the curvature reconstruction.



(a) No perturbation

(b) 20% perturbation

Figure 2: Curvature reconstruction errors on tensor product grids with and without perturbation of the nodes.

4 SOLUTION APPROACH

The two previous sections have discussed the different components needed to numerically simulate two-phase flows. The interface is always implicitly coupled to the flow variables through the density, viscosity, and surface tension. The flow variables are in return coupled to the interface through the velocity field. In the solution process the order and way in which these variables are solved for needs to be considered carefully. The following section will cover the construction of an appropriate solver sequence or structure, and discuss various efficiency aspects relating to it.

4.1 Solution procedure

Both the solution of the Navier-Stokes equations, including the interface tracking algorithms, can in essence be treated in a fully coupled monolithic way, that is to set up and simultaneously solve the following large system for all unknowns

$$\begin{cases} \rho(\phi)\left(\dfrac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u}\right) = -\nabla p + \nabla\cdot\left(\mu(\phi)(\nabla\mathbf{u} + \nabla\mathbf{u}^T)\right) + \mathbf{f}, \\ \mathbf{f} = \rho(\phi)\mathbf{g} + \sigma\kappa(\phi)\hat{\mathbf{n}}(\phi)\delta(\phi), \\ \nabla\cdot\mathbf{u} = 0, \\ \dfrac{\partial \phi}{\partial t} + (\mathbf{u}\cdot\nabla)\phi = 0. \end{cases} \tag{16}$$

A monolithic approach may seem convenient and robust, but is not very efficient from a computational point of view. It is generally always less costly to invert and solve a series of smaller matrix systems than a single large one. For example, the discrete projection method presented in Section 2 allows for this by separating the velocity computations from the pressure.

The interface tracking algorithm, in this case the level set equation, should theoretically also be included implicitly so that all interfaces, which move with the velocity field, are updated continuously. This is usually not very practical since the density and viscosity fields, gravitational force, and surface tension in (16) all depend on the position of the interface. In practice it is more convenient to treat the interface explicitly, that is to regard it as fixed, for example using the interface position at the old time step, while the other quantities are computed.

In an analogous way it is natural to also add the interface tracking algorithm explicitly, that is updating the positions of the interfaces after the new velocity and pressure fields have been computed. In this case it is the velocity field in the trilinear form (14) convecting the interfaces that is assumed to be known and is treated explicitly. All together this will lead to a segregated solver structure with the following form:

Segregated solver structure

- Initialization
- Time stepping loop:

  For $t^{n+1} \in [0, T]$, and time step $\Delta t = t^{n+1} - t^n$ do

  1. Flow solver (Discrete projection method):
     Given $\Gamma = \Gamma^n$, and implicitly $\rho = \rho(\Gamma^n)$ and $\mu = \mu(\Gamma^n)$, update the velocity field $\mathbf{u}^{n+1}$ and pressure p^{n+1}
  2. Interface tracking solver (Level set method):
     Given $\mathbf{u} = \mathbf{u}^{n+1}$ update the interface Γ^{n+1}
  3. Postprocessing

The initialization step includes input of parameters, grid generation, and allocation of memory blocks for arrays, vectors, and matrices. It is advantageous to incorporate the discrete projection method and interface tracking, steps 1 and 2, modularly so that the code easily can be extended if one for example wants to include chemical reactions or heat transfer and solidification effects. This also allows one to easily and conveniently activate and deactivate modules, for example to only use the flow solver if one is studying single phase flow. Steps 1-3 can also be repeated iteratively in each time step until convergence in a suitable norm has been reached. The postprocessing step contains routines for graphical and data output, and time stepping control. Local grid adaption and grid deformation techniques can either be added in this step, or between steps 1 and 2, depending on which variables are least sensitive to interpolation.

4.2 Flow solver

The solution of the flow variables, even when using the discrete projection method, is the most costly component and hence should be given most attention. A corresponding flow solver module (treating step 1 above) will have the following structure:

Flow solver: Discrete projection method module

1. Given the interface $\Gamma = \Gamma^n$ construct the variable density, $\rho = \rho(\Gamma^n)$,
   and viscosity, $\mu = \mu(\Gamma^n)$, fields
2. Assemble mass and diffusion matrices
3. Generate projection matrix
4. Assemble right hand side vector (gravity and surface tension effects)
5. Assemble convective terms and calculate initial defect
6. Defect correction loop for the momentum equations:
   (a) Assemble convective terms and generate iteration matrices
   (b) Solve the linearized momentum equations
   (c) Calculate new defect
   (d) Check for convergence (go to step 6a if necessary)
7. Calculate the right hand side for the pressure Poisson problem
8. Solve the pressure Poisson problem
9. Update pressure and velocity fields

In this algorithmic structure, the mass and diffusion matrices are assembled once outside the defect loop. This saves computational effort in direct proportion to the number of required defect iterations compared to assembling everything in the defect loop. Appropriate boundary conditions must also be applied to the momentum equations at flow inlets, outlets and wall boundaries.

4.3 Interface tracking solver

An interface tracking module for treating the level set equation is in theory quite easy to construct. The governing equation is a standard convection diffusion equation transporting a relatively smooth scalar property for which many techniques have been developed. The most costly component will again be the assembly step even though only one convective transport operator needs to be reassembled in each time step. The velocity field in the trilinear operator which in this case comes from another finite element space, the non-conforming $\tilde{Q}_1$ space, should be evaluated directly in the cubature points to achieve full accuracy. This is will increase the cost since it will require a higher order quadrature rule than normal (up to a fourth order rule in the two dimensional case).

Routines for periodical reinitialization, and normal and curvature computations are added in a postprocessing step after the level set field has been computed. The level set interface tracking module is broken down into the following steps:

Interface tracking: Level set module

1. Assemble convection matrix
2. Generate right hand side vector
3. Calculate initial defect
4. Defect correction loop for the level set equation:
   (a) Assemble contributions for artificial convective stabilization
   (b) Solve the level set equation
   (c) Calculate new defect
   (d) Check for convergence (go to step 4a if necessary)
5. Apply reinitialization of the level set field if necessary
6. Calculate normal and curvature fields

Note that the mass matrix is assembled once and then reused (under the assumption that the grid is fixed and not moving). If the applied convective stabilization is linear the defect correction loop 4(a)-(d) will converge in one step and can then be omitted.

5 NUMERICAL VALIDATION

This section describes the testing undertaken to validate the simulation code implemented according to the described methodology. The modular design makes it easier to debug and validate the code by testing the solvers independently before applying them to coupled and more complex problems. The flow solver was first successfully validated on the DFG flow around cylinder benchmarks for which very accurate reference solutions are available (for example from the FeatFlow CFD benchmark database [31]). The level set module was then by itself tested on stretching and tearing interface tracking test cases [22]. The following section describes how the implemented approach was able to treat more complex two-phase flow problems.

5.1 Static bubble

This test case models a perfectly stationary two dimensional bubble at equilibrium. According to the Laplace-Young law the pressure inside the bubble should be equal to $p_{in} = p_{out} + \sigma/r$, where r is the radius of the bubble. Since everything is stationary the velocity should ideally be zero everywhere, however, due to certain imbalances in the numerical method unphysical spurious velocity currents will be generated.

The test configuration considers a bubble with radius $r = 0.25$ positioned in the center of a unit square. The coefficient of surface tension, and the viscosity inside and also outside the bubble were all set to unity while the densities were given a magnitude of 10^4. This corresponds to a Laplace number of $La = (2r)\sigma\rho\mu^{-2} = 5 \cdot 10^3$. A fixed time step of $\Delta t = 0.01$ was used in the simulations which were run until $t = 125$.

In Table 2 the errors in the computed solution compared with the analytical solution are given for four different mesh widths ($h = 1/[20, 40, 80, 160]$). The middle column shows the error of the dimensionless velocity in the l^1 norm, $\frac{1}{N}\sum_{i=1}^{N} |\mathbf{u}_i\mu/\sigma|$, where N is the number of nodes. As with all Eulerian interface tracking methods there were spurious velocity currents present, however, the errors quickly decreased and converged with 2nd order. The right column in Table 2 additionally shows how well the pressure field fulfilled the Laplace-Young law, that is $|p_{in} - p_{out} - \sigma/r|$. This error will be nonzero due to the combination of the constant Q_0 pressure approximation on each cell and the Eulerian nature of the level set method which allows the interface to intersect cells arbitrarily. These errors also decreased very rapidly and showed a full 2nd order convergence rate.

1/h	$\mathbf{u}\mu/\sigma$	$\lvert p_{in} - p_{out} - \sigma/r \rvert$
20	$6.7 \cdot 10^{-4}$	$4.8 \cdot 10^{-2}$
40	$1.9 \cdot 10^{-4}$	$1.2 \cdot 10^{-2}$
80	$5.2 \cdot 10^{-5}$	$3.1 \cdot 10^{-3}$
160	$1.4 \cdot 10^{-5}$	$7.8 \cdot 10^{-4}$
ROC $\approx$	1.9	2.0

Table 2: Errors and convergence rates (ROC) for the non-dimensional velocity and pressure.

Figure 3 shows pressure cut-lines for various levels of grid refinement (using mesh widths $h = 1/20$ to $h = 1/160$). It can be seen that the pressure approximation is quite sharp and non-oscillating even on fairly coarse grids. This simple test case has shown that the two-phase flow module and surface tension calculations were implemented correctly and produced accurate results.

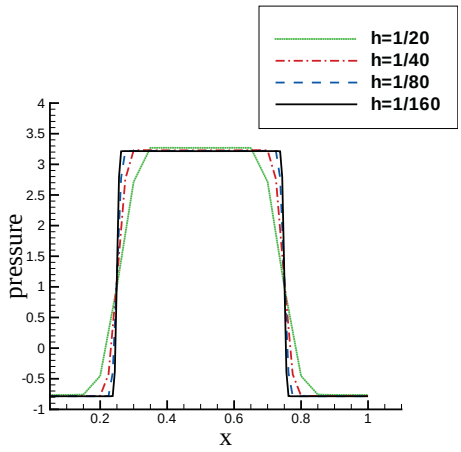


Figure 3: Pressure cut-line ($y = 0.5$) for different mesh sizes.

5.2 Rising bubble

This complex benchmark test case has been throughly studied by three independent research groups who managed to determine accurate and quantitative reference solutions for the circularity, rise velocity, and center of mass of a bubble in a liquid column [13]. The test case begins with a circular bubble with radius $r = 0.25$ centered in the lower half of a 1×2 rectangular domain. The buoyancy force will cause the bubble to rise and also deform. Aside from tracking the bubble shape the following quantities are also measured:

Center of Mass. The center of mass of the bubble is defined by

$$\mathbf{X}_c = (x_c, y_c) = \frac{\int_{\Omega_2} \mathbf{x}\, dx}{\int_{\Omega_2} 1\, dx}$$

where Ω_2 denotes the region that the bubble occupies.

Circularity. The degree of circularity is in two dimensions defined as

$$\not{\epsilon} = \frac{P_a}{P_b} = \frac{\text{perimeter of area-equivalent circle}}{\text{perimeter of bubble}} = \frac{\pi d_a}{P_b}\,.$$

P_a denotes the perimeter or circumference of a circle with diameter d_a which has an area equal to that of a bubble with perimeter P_b. For a perfect circular bubble the circularity will be equal to unity and decrease as the bubble is deformed.

Rise Velocity. The mean velocity with which a bubble is rising and moving is defined as

$$\mathbf{U}_c = \frac{\int_{\Omega_2} \mathbf{u}\, dx}{\int_{\Omega_2} 1\, dx}$$

where Ω_2 again denotes the region that the bubble occupies. The rise velocity is simply the corresponding velocity component in the direction the bubble is moving.

The physical parameters of the two fluids and dimensionless numbers defining the test case are given in Table 3. A subscript 1 is assigned to the heavier surrounding fluid and 2 the lighter fluid of the bubble.

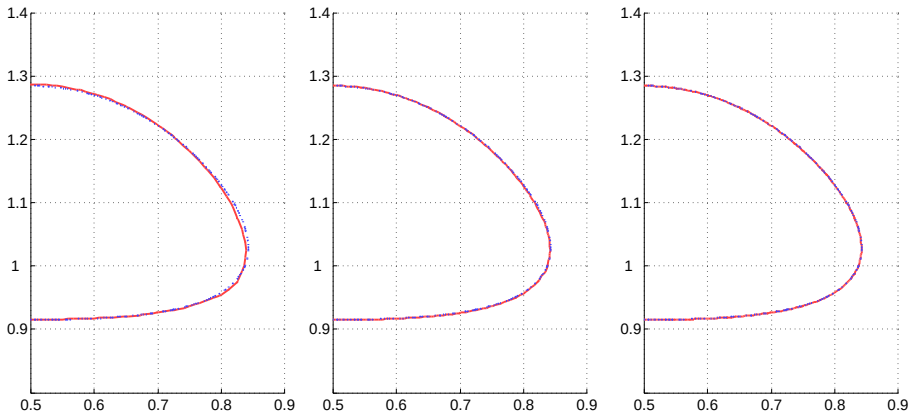
Table 3: Physical parameters and dimensionless numbers defining the rising bubble test case.

ρ_1	ρ_2	μ_1	μ_2	g	σ	Re	Eo	ρ_1/ρ_2	μ_1/μ_2
1000	100	10	1	0.98	24.5	35	10	10	10

Computations were performed on rectangular tensor product grids with mesh widths $h = 1/[40, 80, 160, 320]$. The time step was scaled with the mesh size as $\Delta t = h/16$. Table 4 shows the simulation statistics for the different grid levels where the number of elements is denoted by NEL, the total number of degrees of freedom by NDOF, and the total number of time steps by NTS.

Table 4: Simulation statistics for the rising bubble test case.

$1/h$	NEL	NDOF	NTS
40	3200	19561	1920
80	12800	77521	3840
160	51200	308641	7680
320	204800	1231681	15360



(a) $h = 1/40$ (b) $h = 1/80$ (c) $h = 1/160$

Figure 4: Computed bubble shapes at time t=3 on successively refined grids (solid red) compared to the reference solution (dashed blue).

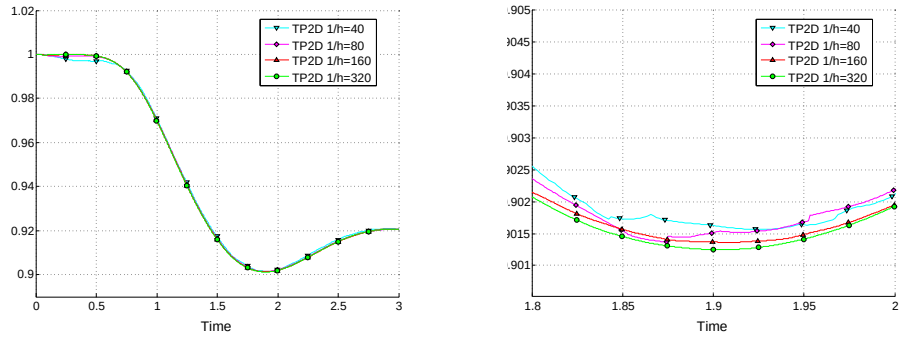
In Figure 4 the symmetric right half of the bubble shapes on various grids at the final time t=3 are compared with the reference solution established in the benchmarks [14]. It is apparent that the solution on the coarsest grid $h = 1/40$ (Figure 4(a)) is already very good but differs somewhat from the reference solution. The computation on a one level finer grid ($h = 1/80$ shown in Figure 4(b)) is clearly better and further refinements yield bubble shapes which are visually indistinguishable from the reference shape. Qualitatively looking at the bubble shapes is clearly not sufficient to say anything about the accuracy on the finer grids. One therefore has to switch to a quantitative analysis instead and study the defined benchmark quantities.

The relative error norms for the circularity, center of mass, and rise velocity are shown in Table 5 together with the estimated convergence rates (ROC). The reference solution is as before taken from the benchmarks. It is evident that all quantities converge with a more than linear convergence order, approaching quadratic convergence in the l_1 and l_2 norms. In the maximum norm the convergence order decreased to 1.16 for the circularity and 1.39 for the rise velocity.

Table 5: Relative error norms and convergence orders for the three benchmark quantities.

$1/h$	$\|\|e\|\|_1$	ROC_1	$\|\|e\|\|_2$	ROC_2	$\|\|e\|\|_\infty$	ROC_∞
			Circularity			
40	1.00e-03		1.22e-03		2.89e-03	
80	3.01e-04	1.74	3.63e-04	1.75	9.67e-04	1.58
160	8.83e-05	1.77	1.10e-04	1.72	4.32e-04	1.16
			Center of mass			
40	2.65e-03		2.99e-03		3.56e-03	
80	9.64e-04	1.46	1.02e-03	1.55	1.14e-03	1.64
160	2.62e-04	1.88	2.71e-04	1.91	2.96e-04	1.95
			Rise velocity			
40	1.19e-02		1.29e-02		1.49e-02	
80	2.90e-03	2.04	3.07e-03	2.07	5.08e-03	1.55
160	7.73e-04	1.91	7.85e-04	1.97	1.94e-03	1.39

The following figures depict the time evolution of the benchmark quantities. From Figure 5(a), which shows the circularity, it is quite hard to discern any significant differences between the different grids. Only for the coarsest grid ($h = 1/40$) is it possible to see some deviations; the circularity drops too quickly up until $t = 0.7$, after which the correct solution behavior is recovered. A close up around the point of minimum circularity is shown in Figure 5(b) from where it is possible to see the convergence behavior. Most notable is that there are irregularities or small jumps in the curves for the two coarsest grids which is due to the reinitialization procedure which was applied every 20 time steps.



(a) Circularity

(b) Close-up of the circularity

Figure 5: Time evolution of the circularity.

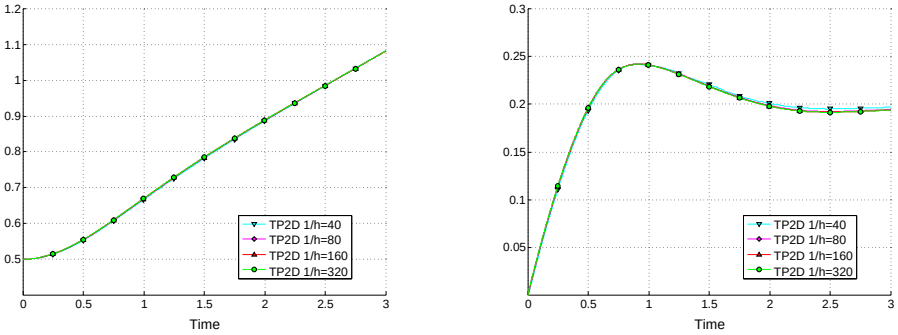
Table 6 shows that the inflection point of minimum circularity converges towards a value of 0.9013 around $t = 1.90$. This compares very well with the suggested reference minimum at $t = 1.9$ with value 0.9012±0.0001[13].

Table 6: Minimum circularity (ϕ_{min}) and maximum rise velocity ($V_{c,max}$), with corresponding incidence times, and the final position of the center of mass (y_c).

$1/h$	40	80	160	320
ϕ_{min}	0.9016	0.9014	0.9014	0.9013
$t\|_{\phi=\phi_{min}}$	1.9234	1.8734	1.9070	1.9041
$V_{c,max}$	0.2418	0.2418	0.2419	0.2417
$t\|_{V_c=V_{c,max}}$	0.9141	0.9375	0.9281	0.9213
$y_c(t=3)$	1.0818	1.0810	1.0812	1.0813

Both the center of mass, shown in Figure 6(a), and the mean rise velocity of the bubble, shown in Figure 6(b), converge very nicely. From Table 6 one can see that the maximum rise velocity of $V_{c,max} = 0.2417$ is attained quite early at time $t = 0.92$. This can be compared to the target range 0.2419±0.0002 at time 0.932. The center of mass of the bubble can asymptotically be described as a linear function of time and approaches $y_c = 1.0813$ towards the end of the simulation which lies in the reference value range 1.081±0.001.

By having used the rising bubble benchmark in the validation process one can be very confident that the developed code can simulate complex two-phase flow problems accurately. This is in contrast to comparing with experimental measurements and data for which there can be large uncertainties.



(a) Center of mass

(b) Rise velocity

Figure 6: Center of mass and rise velocity of the bubble.

6 SUMMARY AND CONCLUSIONS

This paper has presented a novel mixed finite element level set method for simulation of flows with immersed interfaces. The new method combines an efficient finite element flow solver based on the non-conforming Rannacher-Turek $\tilde{Q}_1Q_0$ elements with a conforming Q_1 level set solver to track the interfaces. The two solvers are coupled in a segregated and modular way sequentially solving for each of the dependent variables. This modular approach is computationally efficient and also allows each solver module to be tested and used completely independent from each other.

The governing equations are discretized in time with a θ-scheme approach resulting in robust and accurate single and multi-step schemes. In space the $\tilde{Q}_1Q_0$ discretization allows the construction of an efficient discrete projection method which decouples the velocity and pressure variables from each other. One important key to efficiency here is to use HRZ-mass lumping when assembling the density weighted mass matrix used in the projection operator. The resulting sparse linear equation systems can then be inverted and solved inexpensively using a multi-grid approach. Furthermore, FEM-TVD stabilization is applied when necessary to treat the case of dominating convection terms in both the flow and level set solvers. Surface tension effects are included with an implicit approach which circumvents the capillary time step restriction allowing for larger time steps.

The level set method is used as interface tracking method since the governing equation is a standard hyperbolic transport equation and the level set field for the most part is smooth. This allows for standard PDE solution techniques to be applied. Moreover, both the interface, and normal and curvature fields can be recovered globally if needed, as for instance in the implicit surface tension model. It was shown that in order to achieve convergence for the normal and curvature fields on anisotropic or distorted grids it was necessary to use a higher gradient recovery method such as the PPR method. Although the level set field will distort with time previous studies have determined that the most suitable method to accurately and quickly recover the distance function property was to periodically reinitialize it with the fast marching method.

The simulation code that has been developed according to the described approach has been tested on several numerical benchmarks to assert its accuracy. It was first tested on a flow around cylinder problem, for which very accurate reference solutions are available, which showed that the flow solver produced reliable results for single phase flows. Furthermore, an equilibrium static bubble problem confirmed that surface tension effects were implemented correctly. A rising bubble problem finally validated the overall coupled approach and verified that the method could produce correct results for complex two-phase flow problems.

Commercial software tools are widely used by industrial engineers today to simulate various physical processes. Except for monetary cost, they offer many benefits over academic tools; commercial codes are reasonably easy to use, are often documented extensively, have user support, and usually include tried and tested algorithms which produce qualitatively good results. However, what is usually not known is how accurate these codes really are, on an absolute level that is, and what performance can be expected for a specific problem. This was, within the context of two-phase flows, examined by simulating one of the previously described benchmark test cases with two different commercial codes, the general and flexible PDE simulation package Comsol Multiphysics and the dedicated CFD flow solver Ansys Fluent. The results from these codes was also compared with those computed with the newly developed code.

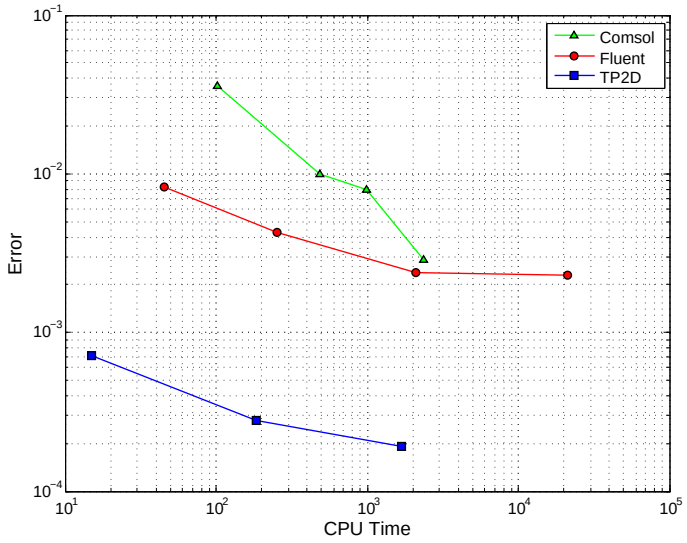


Figure 7: Averaged error in the circularity vs. CPU time.

The chosen test problem was the rising bubble benchmark previously described in Section 5.2. Computations were performed on different grid levels while measuring the required CPU time and calculating the error in the circularity. Figure 7 shows the time averaged error against the CPU time for the different codes. From the figure it is clear that the solution produced by Comsol initially had quite a large error but also converged at a high rate due to the higher order Q_2P_1 finite element discretization. Unfortunately, solutions at very fine grids were practically impossible to compute due to the strong de-

pendence on direct solvers. Fluent on the other hand started with a somewhat lower initial error but converged much slower. By the third grid level Fluent and Comsol had achieved the same level of efficiency and surprisingly further refinements yielded no improvements at all, Fluent completely stopped converging. The new approach, labeled $TP2D$ in the figure, converged with first order and showed a much better overall efficiency, requiring about ten times less effort to achieve a certain accuracy than the commercial codes would have had they been able to compute on finer grids. Note that even the error on the very coarsest grid was already lower that anything that either of the commercial codes could produce.

Altogether, a new complete approach for simulation of immiscible fluid flows with free interfaces has been described and validated on several numerical benchmark test cases. As a final test it was compared against two commercial codes showing the merit of the developed code which was able to outperform them by a magnitude or more.

ACKNOWLEDGEMENTS

The author would like to thank the German Research foundation (DFG) and the European Union for partially supporting the work under grants Paketantrag PAK178 (Tu102/27-1, Ku1530/5-1), Sonderforschungsbereich SFB708, and EU STF2/41+CRIS No: 235-833.

References

[1] Blum H, Harig J, Müller S, Turek S. *FEAT2D Finite element analysis tools.* User Manual, Release 1.3, Heidelberg, 1992.

[2] Brackbill JU, Kothe DB, Zemach C. A continuum method for modeling surface tension. *Journal of Computational Physics* 1997; **100**(2):335–354, DOI:10.1016/0021-9991(92)90240-Y.

[3] Bristeau MO, Glowinski R, Periaux J. Numerical methods for the Navier-Stokes equations. Applications to the simulation of compressible and incompressible viscous flows. *Computer Physics Reports* 1987; **6**(1-6):73–187, DOI:10.1016/0167-7977(87)90011-6.

[4] Brooks AN, Hughes TJR. Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations. *Computer Methods in Applied Mechanics and Engineering* 1982; **32**(1-3):199–259, DOI:10.1016/0045-7825(82)90071-8.

[5] Hackbusch W. *Multi-Grid Methods and Applications.* Springer Series in Computational Mathematics, Springer-Verlag, 1985, ISBN:0-387-12761-5.

[6] Harlow FH, Welch JE. Numerical Calculation of TimeDependent Viscous Incompressible Flow of Fluid with Free Surface. *Physics of Fluids* 1965; **8**(12):2182–2189, DOI:10.1063/1.1761178.

[7] Hirt CW, Nichols BD. Volume of fluid (VOF) method for the dynamics of free boundaries. *Journal of Computational Physics* 1981; **39**(1):201–225, DOI:10.1016/0021-9991(81)90145-5.

[8] Hinton E, Rock T, Zienkiewicz OC. A note on mass lumping and related processes in the finite element method.*Earthquake engineering and structural dynamics*, 1976; **4**(3):245–249, DOI:10.1002/eqe.4290040305.

[9] Hughes TJR, Brooks A. A multidimensional upwind scheme with no crosswind diffusion. *in Finite Element Methods for Convection Dominated Flows*. Eds.: T.J.R. Hughes, AMD, Vol. 34, ASME, 19–35. New York, 1979.

[10] Hysing S, Turek S. The Eikonal equation: numerical efficiency vs. algorithmic complexity on quadrilateral grids. *in Proceedings of Algoritmy*. Slovak University of Bratislava, 22–31, Bratislava, 2005, ISBN: 80-227-2192-1.

[11] Hysing S. A new implicit surface tension implementation for interfacial flows. *International Journal for Numerical Methods in Fluids* 2006; **51**(6):659–672, DOI:10.1002/fld.1147.

[12] Hysing S. *Numerical Simulation of Immiscible Fluids with FEM Level Set Techniques*. PhD Thesis, University of Dortmund, Institute of Applied Mathematics, Dortmund, 2007.

[13] Hysing S, Turek S, Kuzmin D, Parolini N, Burman E, Ganesan S, Tobiska L. Quantitative benchmark computations of two-dimensional bubble dynamics. *International Journal for Numerical Methods in Fluids* 2009; **60**(11):1259–1288, DOI:10.1002/fld.1934.

[14] Rising bubble benchmark reference data: http://www.featflow.de/en/benchmarks/cfdbenchmarking/bubble.html

[15] Kuzmin D. A high-resolution finite element scheme for convection-dominated transport. *Communications in Numerical Methods in Engineering* 2000; **16**(3):215–223, DOI: 10.1002/(SICI)1099-0887(200003)16:3¡215::AID-CNM326¿3.0.CO;2-1.

[16] Kuzmin D. On the design of general-purpose flux limiters for finite element schemes. I. Scalar convection. *Journal of Computational Physics* 2006; **219**(2):513–531, DOI:10.1016/j.jcp.2006.03.034.

[17] Kuzmin D, Turek S. High-resolution FEM-TVD schemes based on a fully multidimensional flux limiter. *Journal of Computational Physics* 2004; **198**(1):131–158, DOI:10.1016/j.jcp.2004.01.015.

[18] Kuzmin D, Löhner R, Turek S. *Flux-Corrected Transport: Principles, Algorithms, and Applications*, Scientific Computation, Springer, 2005, ISBN: 978-3-540-23730-3.

[19] Osher SJ, Fedkiw RP. *Level Set Methods and Dynamic Implicit Surfaces*, Applied Mathematical Sciences, Vol. 153, Springer, 2002, ISBN: 978-0-387-95482-0.

[20] Osher SJ, Sethian JA. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics* 1988; **79**(1):12–49, DOI:10.1016/0021-9991(88)90002-2.

[21] Peskin CS. Numerical analysis of blood flow in the heart. *Journal of Computational Physics* 1977; **25**(3):220–252, DOI:10.1016/0021-9991(77)90100-0.

[22] Rider WJ, Kothe DB. Stretching and tearing interface tracking methods. *in the proceeding of the 12th AIAA Computational Fluid Dynamics Conference*. AIAA 95-1717, 1995.

[23] Sethian JA. A fast marching level set method for monotonically advancing fronts. *Proceedings of National Academy of Sciences* 1996; **93**(4)1591–1595.

[24] Turek S, Rannacher R. A simple nonconforming quadrilateral Stokes element. *Numerical Methods for Partial Differential Equations* 1992; **8**(2):97–111, DOI:10.1002/num.1690080202.

[25] Turek S. On discrete projection methods for the incompressible Navier-Stokes equations: an algorithmical approach. *Computer Methods in Applied Mechanics and Engineering* 1997; **143**(3-4):271–288, DOI:10.1016/S0045-7825(96)01155-3.

[26] Turek S. Efficient Solvers for Incompressible Flow Problems, An Algorithmic and Computational Approach. *Lecture Notes in Computational Science and Engineering* Volume 6. Springer-Verlag, 1999, ISBN: 3-540-65433-X.

[27] Unverdi SO, Tryggvason G. A front-tracking method for viscous, incompressible, multi-fluid flows. *Journal of Computational Physics* 1992; **100**(1):25–37, DOI:10.1016/0021-9991(92)90307-K.

[28] Zhang Z. Polynomial preserving gradient recovery and a posteriori estimate for bilinear element on irregular quadrilaterals. *International Journal of Numerical Analysis and Modeling* 2004; **1**(1):1–24.

[29] Zienkiewicz OC, Zhu JZ. The superconvergent patch recovery and a posteriori error estimators. Part 1: The recovery technique. *International Journal for Numerical Methods in Engineering* 1992; **33**(7):1331–1364, DOI:10.1002/nme.1620330702.

[30] Zienkiewicz OC, Zhu JZ. The superconvergent patch recovery and a posteriori error estimates. Part 2: Error estimates and adaptivity. *International Journal for Numerical Methods in Engineering* 1992; **33**(7):1365–1382, DOI:10.1002/nme.1620330703.

[31] FeatFlow CFD benchmark repository:
http://www.featflow.de/en/benchmarks/cfdbenchmarking.html