A new interface capturing model with explicit constraint on mass conservation

Literature Report

Daniël Pols - 4284569

17 May 2018

Summary

In solving two-phase flows, the location of the interface between the phases is necessary to handle interface jump conditions when solving the Navier-Stokes equation. Current interface capturing and advection methods, however, suffer from various issues. The level set method uses the signed distance to the interface and the interface being the zero level set of this function allows the evolution of the level set field to be described with a simple advection equation. This means that no additional steps are required, but solving the advection equation generally does not conserve mass. On the other hand, Volume of Fluid methods utilise the local fluid fractions to represent the phase interface. For these methods, a mass conserving advection algorithm exists [1], but the absence of an explicit interface requires expensive reconstruction methods to be used instead. Additionally, this Volume of Fluid advection method is subject to a restrictive CFL condition on the time-step and, being dimensionally split, requires a structured grid to be used. Other Volume of Fluid or Moment of Fluid advection methods that do not have these conditions are not mass conserving. Dual interface methods that combine information from the level set and volume fractions are able to achieve higher accuracy, but these method still use Volume of Fluid advection to remain mass conserving and are thus subject to the same conditions. These methods use the level set field to avoid expensive reconstruction method, which negates one drawback of VoF methods.

It is preferable to allow solving the advection equation on unstructured grids. Finite volume methods become difficult to implement and finite element methods may induce instabilities in the solution, so the discontinuous Galerkin discretisation method is considered. The discontinuous Galerkin method is analogous to the Galerkin finite element method, but with piecewise continuous solution instead. While in the finite element method, the residual of the PDE must be orthogonal to the space of global basis functions, in a discontinuous Galerkin discretisation it is only required to be orthogonal to the basis functions defined on one element. This also has the result that the solution is multiply-defined on element boundaries, and adjacent elements are only coupled by the 'numerical flux', which is generally a combination of the multiply-defined flux values on the element boundaries. An added benefit of this discretisation is that any mass correction steps can be applied locally due to the level of independence of the control volumes.

The optimisation-based mass conservation method as described in [14, 15] is similar to techniques that are expected to be necessary for this research. This method introduces a control variable to supply a nonlinear system of equations that must be satisfied to obtain mass conservation. The exact form of this system depends on the discretisation and time stepping method used. The optimisation problem aims to minimise a given functional while satisfying this system of equations, and from the associated Lagrangian first-order optimality conditions can be derived. The resulting conditions can then be solved using fixed-point iteration to obtain a new conservative level set field. The difference between the scalar and vector control approaches mentioned is that where the scalar approach utilises the gradient of a scalar field as control variable, the vector approach requires a vector field instead. This results in a larger, but possibly more nicely structured system for the vector approach compared to the scalar approach. With the optimality conditions mentioned, the obtained solution is in general not unique. This can be resolved by applying additional conditions or adding extra terms to the functional.

The idea for further research is now to use the momentary fluxes instead of the donating regions for advection of the volume fractions, and apply the described methods to keep the algorithm mass conserving.

Contents

Summary i

1 Motivation for an implicit interface time-integration method 1

2 Incompressible immiscible two-phase flow model 2

2.1 Interface models 2

2.2 Level set method 3

2.3 Volume of fluid method 3

2.3.1 Interface Reconstruction 4

2.3.2 Interface advection 6

2.3.3 Convergence analysis on VoF interface models 8

2.4 Dual interface methods 9

2.5 Concluding remarks 11

3 Discontinuous Galerkin 12

3.1 Theory behind DG discretisation 12

3.2 DG on general polygons 13

3.3 Concluding remarks 16

4 Optimisation-based mass conservation 17

4.1 Scalar control approach 17

4.2 Vector control approach 19

4.3 Concluding remarks 20

5 Research Questions 21

5.1 Time schedule 22

6 Conclusion 23

References 24

1 Motivation for an implicit interface time-integration method

Solving two-phase flows is important for many applications in science and industry. The modelling of chemical reactors are an example of this, but fluid jetting [8] and complex gas-fluid mixtures in pipelines [10] are also instances where accurate solving of two-phase flows is needed. Solving the Navier-Stokes equations for two-phase flows, however, requires knowledge of the location of the phase interface to deal with jumps in pressure and velocity, and to handle the effect of surface tension. While methods exist to find the location of said interface [1, 7, 9, 11, 12], these methods suffer from some drawbacks that mainly occur through the use of 'donating regions'. This results in restrictive requirements on the time step size and necessitates structured grids, while methods that attempt to avoid using donating regions or that use unstructured grids do not conserve mass while becoming extremely complicated [2]. The requirements of rectangular grids complicates the use of two-phase flow solvers on complicated domains, whereas the time step restriction in some cases cannot be met (e.g. cylindrical coordinates) [10].

To avoid these problems, this project aims to formulate a description of a dual interface capturing method with inherent mass conservation that does not rely on donating regions for interface advection. In section 2, existing models and their drawbacks will be described in more detail, and the step to dual interface methods will be highlighted. Then, the discontinuous Galerkin method [13, 3] will be described in section 3, which allows accurate discretisation on unstructured grids and allows local application of mass conserving corrections. Lastly, optimisation-based mass correction steps for the level set method [14, 15] will be described in section 4, as it is expected that similar techniques will be required in this research.

2 Incompressible immiscible two-phase flow model

In two-phase flows, a domain $\Omega \in \mathbb{R}^2$ is occupied by two different materials, for example water and air, that are separated by an interface. If the flow is assumed to be incompressible, so

$$\nabla \cdot \boldsymbol{u} = 0 \tag{2.1}$$

for the velocity vector $\boldsymbol{u}$, then the time evolution of the momentum field is governed by the incompressible Navier-Stokes equations

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u}\nabla \cdot \boldsymbol{u} = -\frac{1}{\rho}\nabla p + \frac{1}{\rho}\nabla \cdot \mu(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^t) + \boldsymbol{g}, \tag{2.2}$$

where ρ is the density, μ the viscosity, p the pressure, and $\boldsymbol{g}$ the outside forces, which usually includes gravity. Since the materials are incompressible, the density and viscosity are constant within each material. Due to cohesive forces, at the interface between the two materials, surface tension must also be considered. It can be shown that for a surface element A the surface tension force equals

$$\boldsymbol{f}_s = -\int_A \sigma\kappa\boldsymbol{n}\,\mathrm{d}A, \tag{2.3}$$

where $\boldsymbol{n}$ is the outward normal of A with unit length, σ a constant indicating the strength of the surface tension, and $\kappa = \nabla \cdot \boldsymbol{n}$ the curvature of the surface. Obviously, the surface tension acts as a force normal to the interface with magnitude $\sigma\kappa$. Additionally, the flow velocity must be continuous at the interface, which results in the interface conditions

$$[\boldsymbol{u}] = 0 \tag{2.4}$$

$$[p\boldsymbol{n} + \mu(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^t) \cdot \boldsymbol{n}] = \sigma\kappa\boldsymbol{n}, \tag{2.5}$$

where the brackets denote jumps across the interface. Consider also a vector $\boldsymbol{s}$ parallel to the interface. To avoid giving the interface infinite acceleration (since the interface has no mass), both sides of the interface must experience equal tangential stresses. As such,

$$\left[\mu\frac{\partial u_s}{\partial n}\right] = 0. \tag{2.6}$$

Since the location of the interface must be known in order to apply the interface jump conditions (2.4) to (2.6), an interface model is required to numerically integrate the flow velocity in time. In this section, some background on interface models is given, and different interface model types are discussed.

2.1 Interface models

Most methods that find the location of an interface are either *interface tracking* methods or *interface capturing* methods. Interface tracking methods attempt to explicitly represent the interface, for example (in 2D) by chains of line segments, and move this representation every time step with the local flow velocity. While interface tracking methods can work fine in many applications, trouble occurs when dealing with changing topologies. When two surfaces intersect, or when a surface folds over itself, some parts of the chain must be reworked to merge or split interfaces. However, if such changes are not anticipated, expensive checks must be done to see whether or not two segments intersect. Additionally, the reordering of segments in such a case is also not trivial.

Interface capturing methods do not explicitly represent the interface in the way tracking methods do. Instead, an indicator function is used which allows distinction between the fluids, and the method tracks the change of the indicator function instead. This change implicitly defines the interface, and topology changes should be automatically accounted for. Since interface capturing methods should be applicable in situations with changing topologies, only this kind of method will be described hereafter.

When using indicator functions that are not smooth around the interface, special care must be taken. Note that in two-phase flows, since the interface moves according to the local flow velocity, for these indicators the interface represents a linearly degenerate wave. When linearly degenerate waves are present, it is shown in Banks [4] that for a finite volume non-compressive discretisation of order p, the solution is only approximated with $O\left(\frac{p}{p+1}\right)$. This means that for non-smooth indicators, methods of this type that directly discretise the advection equation will have at most linear convergence. As such, when higher order accuracy is needed, a different type of method is necessary to numerically approximate the solution of the interface advection problem.

2.2 Level set method

The *level set* (LS) interface capturing method [9] uses a marker function ϕ defined on the domain Ω that changes sign at the interface. The interface is implicitly defined as the zero level set of ϕ:

$$\Gamma(t) = \{\boldsymbol{x} \in \Omega : \ \phi(\boldsymbol{x}, t) = 0\}. \tag{2.7}$$

The interface is evolved by applying the advection scheme to the level set function in the divergence-free flow field $\boldsymbol{u}$,

$$\frac{\partial \phi}{\partial t} + \boldsymbol{u} \cdot \nabla \phi = 0. \tag{2.8}$$

Since the level set function is typically smooth near the interface, straightforward application of an advection scheme is allowed. Since there is no jump discontinuity at the interface, the level set method also does not suffer from the lower order of approximation that was mentioned in section 2.1. The level set function is usually chosen to be the signed distance to the nearest interface surface. This allows for easy extracting of certain geometrical properties like the unit normal $\boldsymbol{n}$ to the interface and the curvature κ of the interface:

$$\boldsymbol{n} = \nabla \phi, \qquad \kappa = \nabla \cdot \boldsymbol{n}. \tag{2.9}$$

The signed distance property of the level set function generally does not hold under advection through a non-uniform flow, so to restore this property, the function ϕ must be reinitialised. This can be done, for example, through the use of a partial differential equation

$$\frac{\partial \phi}{\partial t'} = \operatorname{sign}(\phi_0)(1 - \nabla \phi), \tag{2.10}$$

$$\phi_0 = \phi|_{t'=0}, \tag{2.11}$$

with t' an artificial time. This PDE leaves the zero level set unchanged, and reaches equilibrium when $|\nabla \phi| = 1$, so when ϕ is again a distance function.

The major drawback of the level set method is that while the level set function itself is conserved, the area enclosed by the interface does not have to be. While higher order discretisation schemes improve mass conservation, the loss of mass is not strictly a result of numerical errors, but rather a property of the advection equation itself.

2.3 Volume of fluid method

The *volume of fluid* (VoF) method [1, 5] is an interface capturing method that is based on the marker particles method, but with much lower storage requirements. The marker particle method involves filling a fluid region with particles that move with the local flow velocity. This, like interface capturing methods, automatically deals with topology changes of the interface, but the amount of points needed grows immensely, especially in three dimensions. The volume of fluid method aims to use the upsides of this method, while only requiring storage for one value per cell. The volume of fluid method requires only the *fractional volume of fluid* inside a cell, which implicitly defines the location of the interface.

Define a colour function χ which separates regions occupied by different materials,

$$\chi(\boldsymbol{x}) = \begin{cases} 1, & \boldsymbol{x} \text{ occupied by fluid} \\ 0, & \text{otherwise.} \end{cases} \tag{2.12}$$

The cell-averaged value of χ then represents the fraction of the cells volume occupied by fluid, which defines the volume of fluid function ψ,

$$\psi = \frac{1}{|\Omega|} \int_{\Omega} \chi(\boldsymbol{x}) \, d\boldsymbol{x}, \qquad |\Omega| = \int_{\Omega} d\boldsymbol{x}, \tag{2.13}$$

where Ω now denotes the space inside an arbitrary cell. If the cell-averaged colour function ψ has unit value inside the cell, it is completely filled with fluid, and likewise the cell contains no fluid if its ψ value is zero. Naturally, cells for which the value of ψ is between zero and one is then filled with both fluid and gas, and as such will contain an interface. Note that the VoF method is essentially the cell-averaged version of the marker particle method, while VoF has much lower storage requirements. The only information that is currently missing in the volume of fluid method is the location of fluid inside a boundary cell, which can also be obtained.

The direction normal to the interface inside a boundary cell lies in the direction where the colour function χ changes most rapidly. Once the interface orientation is known, the location of the interface can be constructed so that the amount of fluid inside the cell agrees with ψ. The colour function could then be advected using

$$\frac{\partial \chi}{\partial t} + \boldsymbol{u} \cdot \nabla \chi = 0. \tag{2.14}$$

Note, however, that χ is a step function, and that the discontinuity in χ cannot be represented in the discrete case. As such, approximations for the gradient of χ will not converge. This means that different methods must be used for advection and for finding the interface normal. Additionally, the advection equation cannot be applied directly to the volume fraction ψ. Doing so would again disregard the step-like behaviour caused by the interface and instead create boundary cells in multiple layers around the actual location of the interface. The fluxes can be approximated geometrically, but first an *interface reconstruction* method is necessary before an *interface advection* scheme can be applied.

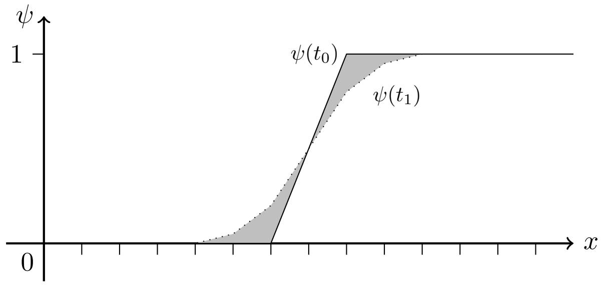

Figure 1: Effect of advection equation on fractional volume ψ; loss of interface definition after one step. Desired behaviour: filling/emptying of central cell.

2.3.1 Interface Reconstruction

For the interface reconstruction step, two methods will be described here. The first method is described by Weymouth and Yue [1] which is similar to the original reconstruction method proposed by Hirt and Nicols [5]. The second method is part of the moment of fluid (MoF) method [7, 8] which also uses the location of fluid centroids to improve the reconstruction algorithm.

In the reconstruction step, the problem lies in estimating the orientation of the phase interface. Once the interface normal $\boldsymbol{m}$ and volume fraction ψ are both known, calculating the exact location of the interface is a simple process. Thus, it is important to uncover an exact relationship between the normal and the volume fraction field. The length of the normal vector is arbitrary, so in two dimensions the general reconstruction of the interface can be written as

$$y = \alpha - mx, \tag{2.15}$$

where the y-component of the interface normal has been set to 1. If the y-values of the interface are known, the normal is calculated simply using

$$m = -\frac{\partial y}{\partial x}. \tag{2.16}$$

Since the volume fractions are the only information available, these y-values are unknown. However, the cell-average heights can be found. Define the mean value of the interface height as

$$\bar{y} = \frac{\int_a^b y(x)\; dx}{\int_a^b dx}, \tag{2.17}$$

and note that when the interface does not cross the top or bottom boundary of a cell, this can be rewritten to

$$\bar{y} = \frac{\int_\Omega \chi\; dv}{\int_{\partial\Omega} dx} = \psi \Delta y. \tag{2.18}$$

With this formula and an understanding of its validity, a reconstruction scheme can be made. Important is that the interface does not pass through the top or bottom boundary of the cells on which (2.18) is applied. First, an estimation must be made to approximate the orientation of the interface. Using central differences in ψ, it can be determined whether the interface lies more horizontally or more vertically, that is, if it should be described as $y(x)$ or as $x(y)$. If the y-direction has the largest difference in ψ, then considering this direction to be 'up', it is most likely that the interface does not cross the top or bottom boundary, so (2.18) is applicable.

Additionally, the volume fractions in a neighbourhood of 3×3 cells are summed in the 'up' direction that was just determined. This essentially creates three cells with triple the height, which again ensures that the top and bottom boundaries do not intersect the interface. Then, (2.16) can be applied to the cell-averaged height $\psi\Delta y$, using forward or backward differences as necessary,

$$m_x = \frac{\partial \bar{y}}{\partial x},\ m_y = 1. \tag{2.19}$$

Here, the last precaution is made to ensure the validity of (2.18). If the central cell is more than half full, then the cell 'downwind' of the estimated normal is used in this difference. If the central cell is less than half full, the 'upwind' value is used instead. This is because when using the original approximation of the interface and a central cell that is more than half full, it is more likely that the interface crosses the top boundary of the fuller cell than that it crosses the bottom boundary of the emptier cell, as a larger volume difference between the cells is required to make this happen. Since this method exactly reproduces linear interfaces, it is assured to be second-order.

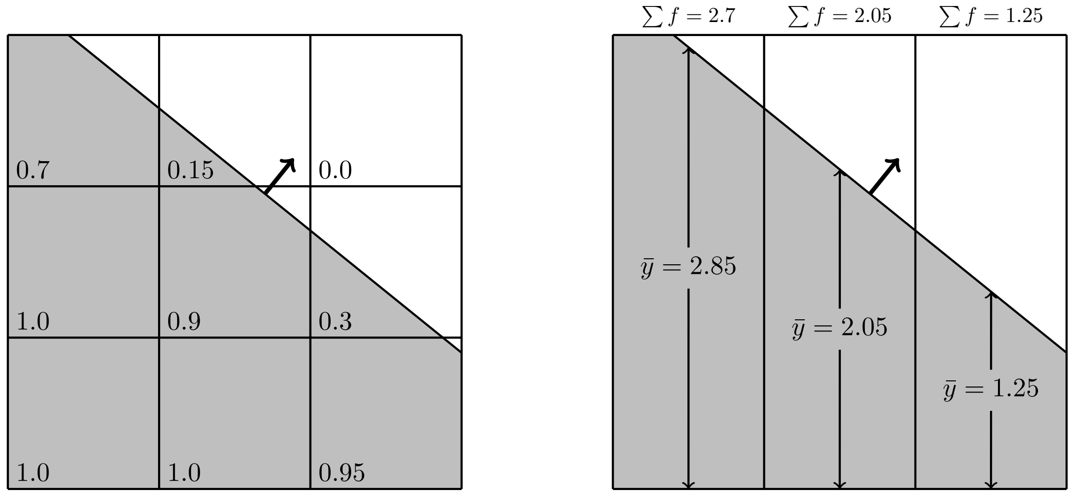

Figure 2: Illustration of the cell concatenation process for unit square cells. The largest difference in volume fractions lies in the y-direction (using central differences), so the cells are summed in this direction. Equation (2.18) holds for the centre and right cells and the calculation for m_x uses only these two cells. [1]

This results in an interface reconstruction method for two- and three-dimensional problems. This method approximates the interface normal to second order accuracy, and when comparing to other second order methods shows a significant reduction in computation time.

While the method described above is able to reconstruct an interface to second order accuracy, it still leaves some things to be desired. Since the method needs information from the neighbourhood of a cell, it only exactly reproduces linear interfaces that are linear in the entire 3×3 neighbourhood. Since the interface is assumed to be linear in this region, this also means that in parts of the interface with high curvature the method does not perform very well. The justification behind this method also appears weak, as the method seems to require (approximations to) the non-defined derivatives of the colour function χ. Because of this, it seems that a method which only uses local data might provide better solutions.

The moment of fluid (MoF) interface reconstruction method uses the information given by the fluid centroids to reconstruct the interface without requiring data from other cells. To do so, first note that when constructing an interface that exactly reproduces the volume fraction ψ, the location of the interface is uniquely defined by its orientation or phase angle θ with respect to some reference angle. Thus, a continuous objective function can be defined that is minimised or maximised with the desired θ. The MoF reconstruction method aims to minimise the distance between the given centroid $\boldsymbol{x}^*$ and the centroid resulting from the linear approximation $\boldsymbol{x}_l(\theta)$, so

$$f(\theta) = ||\boldsymbol{x}_l(\theta) - \boldsymbol{x}^*||^2, \tag{2.20}$$

where the θ that minimises f must be found. This can be done using an iterative process. For any phase angle, the height of the fluid and thus the location of the interface can be determined cheaply for convex cells using, for example, the flood algorithm [7]. Then, the fluid occupies a polygon inside the cell so for convex cells the fluid centroid is easy to calculate. The distance between the fluid centroid and the advected centroid from the previous time step then gives the value of the objective function for this phase angle. Now, note that the derivative of the objective function is given by

$$f'(\theta) = 2((\boldsymbol{x}_l(\theta) - \boldsymbol{x}^*) \cdot \boldsymbol{x}_l'(\theta)), \tag{2.21}$$

where for convex cells,

$$\boldsymbol{x}_l'(\theta) = -\frac{1}{12}\frac{|\Gamma(\theta)|^3}{\psi|\Omega|}\boldsymbol{t}_\theta, \tag{2.22}$$

with $\boldsymbol{t}_\theta$ the tangent to the interface $\Gamma(\theta)$, $|\Gamma(\theta)|$ the length of the interface, and $|\Omega|$ the volume of the entire cell. With the derivative of f found, it is possible to minimise the objective function using iterative processes such as Newton's method. This does require a good initial guess, but taking the interface normal in the direction from the reference centroid to the cell centre proves a good enough guess for this method.

The MoF interface reconstruction method has the nice property that all linear interfaces are exactly reproduced, even if it is not linear in a neighbourhood around the cell. Moreover, if the distance between the two centroids reaches zero, it can be proven that the reconstructed interface best approximates the actual interface and thus should always be used [12]. It can also be proven that for smooth interfaces, the maximum deviation in MoF reconstruction scales quadratically with the diameter of the cell [7]. In regions with high curvature the MoF reconstruction has lower deviation from the true interface than VoF methods. Lastly, MoF reconstruction has no need for a structured grid, unlike VoF interface reconstruction. All methods can be applied to general polygons, but although no condition is required, it is preferable for these polygons to be convex to allow easier and cheaper algorithms to be used.

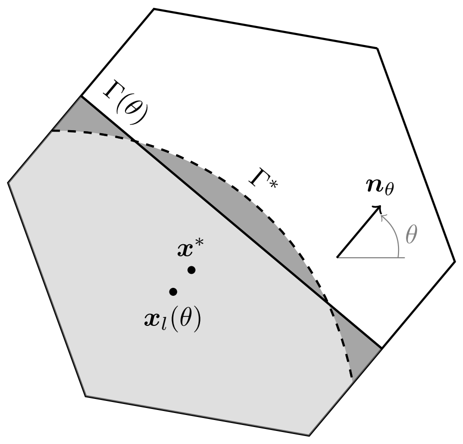

Figure 3: Illustration of the moment of fluid interface reconstruction. The true interface and fluid centroid are denoted by Γ^* and $\boldsymbol{x}^*$ respectively, with the angle-dependent approximations $\Gamma(\theta)$ and $\boldsymbol{x}_l(\theta)$. The error in interface reconstruction (maximum deviation) can be proven to be second order. [7]

2.3.2 Interface advection

Now that the location of the reconstructed interface is known, interface advection methods can be applied. As mentioned, the advection equation cannot be directly implemented for the volume fractions as this would ignore the discontinuity in χ that is the interface. Instead, special care must be taken during advection. Currently, the only mass conserving advection method for the volume fractions is given in [1]. For this method, note that the integral form of the colour function conservation equation (2.14) is

$$\frac{\partial \psi}{\partial t}|\Omega| + F_{net} = \oint_{\Omega} \chi \nabla \cdot \boldsymbol{u} \; dv. \tag{2.23}$$

The integral conservation equation must be used since the gradient of χ is undefined at the interface. This problem can be split into sequential updates of the volume fraction in each spatial dimension, where

$$\Delta\psi \frac{|\Omega|}{\Delta t} = \Delta_d F_d + \int_{\Omega} \chi \frac{\partial u_d}{\partial x_d} \; dv, \quad \text{for } d = 1, ..., \mathcal{N}, \tag{2.24}$$

with $\mathcal{N}$ the amount of spacial dimensions, $\Delta\psi$ the change in volume fraction over one time step, and $\Delta_d F_d$ the net flux entering the cell in dimension d. The dilatation term in (2.24) is necessary since the different one-dimensional flows that are considered one at a time are generally *not* divergence-free, even though the total flow is.

To ensure mass conservation, a short list of requirements can be made. If for a given algorithm:

1. the flux terms are conservative, *and*
2. the dilatation terms sum to zero, *and*
3. no clipping or filling of a cell is needed due to violation of $0 \leq \psi \leq 1$ at any stage,

then the algorithm must preserve ψ to machine precision. Combining the integral term in (2.24) with the cell-centre value of the colour function

$$\chi_c = \begin{cases} 1 & \text{if } \psi > \frac{1}{2}, \\ 0 & \text{otherwise}, \end{cases} \tag{2.25}$$

produces the advection method

$$\Delta\psi = \frac{\Delta t}{|\Omega|} \left(\Delta_d F_d + \chi_c \frac{\partial u_d}{\partial x_d} \right), \tag{2.26}$$

which satisfies the first two conditions. Additionally, it can be shown that a Courant restriction

$$\Delta t \sum_{d=1}^{\mathcal{N}} \left| \frac{u_d}{\Delta x_d} \right| < \frac{1}{2} \tag{2.27}$$

is required to fulfil the last requirement. This advection method is second order accurate, and for incompressible flows is fully conservative. The use of donating regions does, however, introduce some complications. First, it means that a structured (usually rectangular) grid is required as donating regions for general polygonal cells are difficult to use. Second, the condition (2.27) is unwanted since for some problems, it is impossible to meet [10], and in general increases computational requirements for higher accuracy. Additionally, volume of fluid advection methods require expensive methods for both interface reconstruction and for determining the amount of fluid in donating regions.

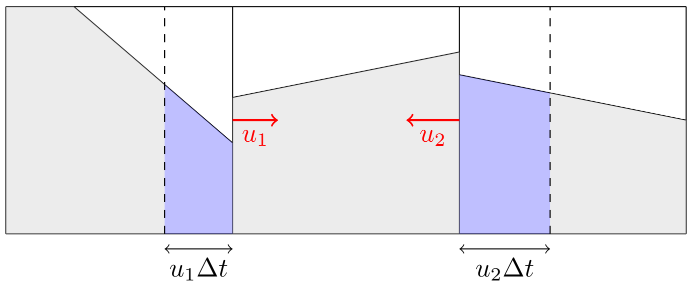

Figure 4: Illustration of a donating regions method. The blue regions are added to the middle cell before advection in the y-direction happens.

While this method is currently the only VoF advection method that preserves mass to machine precision, other methods also exist. Methods based on donating regions that are not dimensionally split or that are built for unstructured grids are, however, not mass conserving and generally do not extend well to three dimensions. Another type of method is the Lagrangian method where the pre-image or post-image of a cell is computed by moving the vertices of a cell with the flow velocity in time. This method avoids all the drawbacks of this mass conserving donating region method, but the edges of a cell do not remain straight lines after advection in nonlinear flows, thus causing inherent mass errors that must be corrected with (usually arbitrary) mass correction steps.

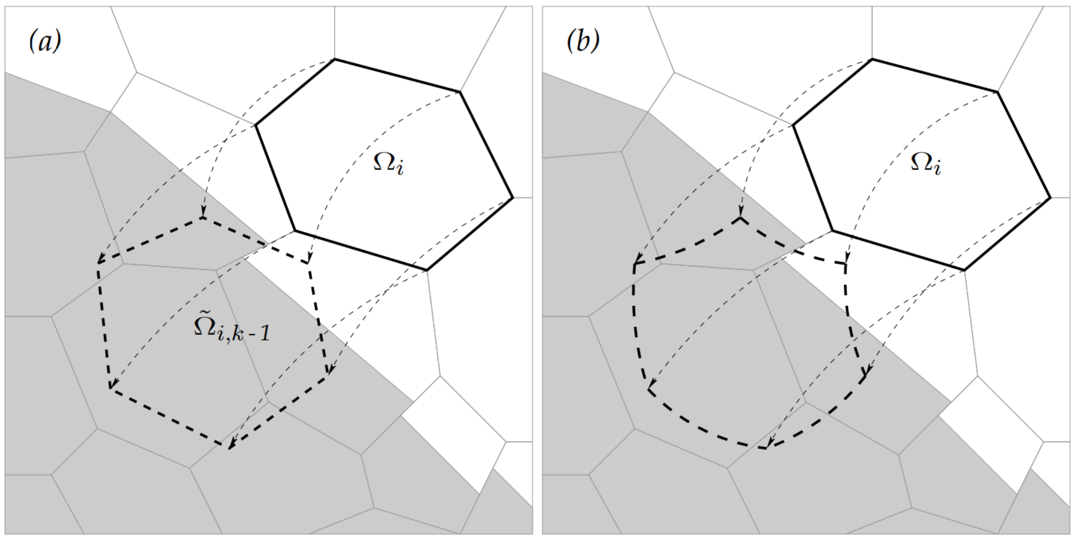

Figure 5: The discrete (a) and true (b) Lagrangian pre-images of the cell Ω_i. [7]
The difference in volume between the original cell and the discrete pre-image causes mass errors.

2.3.3 Convergence analysis on VoF interface models

While the method described in [1] is numerically shown to be second order accurate and induces no loss of matter or volume, more analysis can be done on Volume-of-Fluid methods. The interface reconstruction method and interface advection method separately have been proven to be second order accurate, but this does not necessarily mean that the combination is as well. Zhang and Fogelson [6] provides further investigation in the convergence of VoF methods, and contains some analysis towards higher order methods. In [6], a framework is introduced for interface tracking methods (MARS), and VoF is a special class of the MARS method. Here, $\mathcal{M}_\mathcal{C}$ is the region occupied by the tracked material (fluid) inside a cell $\mathcal{C}$, and as such the volume fraction of material $\mathcal{M}$ in cell $\mathcal{C}$ is

$$\langle \psi(t) \rangle_\mathcal{C} = \frac{||\mathcal{M}_\mathcal{C}||}{||\mathcal{C}||}, \tag{2.28}$$

where the volume of a region is defined to be

$$||\mathcal{S}|| = \left| \int_\mathcal{S} \mathrm{d}\boldsymbol{x} \right|.$$

In most VoF methods, for example in [1], the error is measured by the geometric error

$$E_g(t_n) = \sum_{\mathcal{C}\subset\Omega} ||\mathcal{C}||\, |\langle \psi(t_n) \rangle_\mathcal{C} - \langle \psi \rangle_\mathcal{C}^n|, \tag{2.29}$$

where $\langle \psi(t_n) \rangle_\mathcal{C}$ and $\langle \psi \rangle_\mathcal{C}^n$ are respectively the exact and computed volume fraction in cell $\mathcal{C}$. For analysis, instead, a different measure is used

$$E_1(t_n) = ||\mathcal{M}(t_n) \oplus \mathcal{M}^n|| = \sum_{\mathcal{C}\subset\Omega} ||\mathcal{M}_\mathcal{C}(t_n) \oplus \mathcal{M}_\mathcal{C}^n||, \tag{2.30}$$

where $\mathcal{M}(t_n)$ denotes the exact region occupied by material $\mathcal{M}$ at time t_n, $\mathcal{M}^n$ its approximation and $\oplus$ the symmetric difference between two sets. The measure (2.30) is more rigorous than E_1. The measure E_g is simply the difference between calculated and exact volume fractions in each cell, while E_1 represents the volume of wrongly classified regions. As such, an error $E_1 = 0$ does not necessarily imply perfect reconstruction of the interface, while for E_g this must hold. This means that the measure E_g tends to hide reconstruction errors. Moreover, the error E_1 does not depend on the partition of Ω, while E_g does. While, in practice, the calculation of E_1 is ill-conditioned, this measure is used for theoretical analysis. Since, generally, $E_g \to E_1$ as cell volume and time step approach zero, E_g is used for numerical experiments.

The theoretical analysis then shows that a VoF method is second order accurate in the 1-norm (2.30) if

(a) its advection algorithm is second-order accurate,

(b) its reconstruction scheme is second-order accurate,

(c) its reconstruction scheme is idempotent.

Since piecewise-linear interface reconstruction algorithms are always idempotent as the volume fraction is preserved, this shows that the method of [1] is second-order accurate in the more rigorous measure (2.30).

Analysis was also done on developing a fourth-order VoF method. This would involve using a fourth-order advection algorithm. A cubic-spline representation of the initial condition reduces the representation error to fourth order. The difficulty with this VoF is finding an algorithm that reconstructs the interface from volume fraction with fourth-order accuracy at every time step. This if a difficult task if the Lagrangian length scale is proportional to the Eulerian grid size, and constructing cubic splines from volume fraction is sensitive to numerical noise. It is possible to use adaptive mesh refinement to construct the interface on a grid with mesh-width $h_F = O(h^2)$, where h is the mesh-width of the coarsest grid. However, this method may be extremely inefficient compared to alternatives.

2.4 Dual interface methods

So far, the level set and volume of fluid interface methods have been described. The level set method has a straightforward advection step but does not innately conserve mass, while the volume of fluid method can be made mass conserving, but requires expensive interface reconstruction steps and needs donating regions for the advection step. Dual interface methods aim to combine these two methods to solve as many drawbacks as possible and become more efficient than the two types of methods separately. The dual interface method that will be used as baseline for the rest of this research is the mass conserving level set (MCLS) method [9, 10], but the coupled level set-volume of fluid (CLSVOF) method [11] and its moment of fluid equivalent CLSMOF [12] are also examples of dual interface methods. These dual interface methods are very similar, with only slight differences in the interface reconstruction and level set correction steps.

In the MCLS method, first both the level set function $\phi^{(n-\frac{1}{2})}$ and the VoF function $\psi^{(n-\frac{1}{2})}$ are advected according to the local flow velocity $\boldsymbol{u}$:

$$\frac{\partial \phi}{\partial t} + \boldsymbol{u} \cdot \nabla \phi = 0, \quad \frac{\partial \psi}{\partial t} + \boldsymbol{u} \cdot \nabla \psi = 0. \tag{2.31}$$

For both these processes, the second order split method (2.26) is used in cylindrical coordinates. Note that (2.26) is volume-conservative for VoF as long as the CFL number for the flow field remains less than 0.5. This advection process yields both the VoF function at the next time step $\psi^{(n+\frac{1}{2})}$ and a temporary level set function $\widetilde{\phi}$. Knowing that the step $\phi^{(n-\frac{1}{2})} \to \widetilde{\phi}$ did not conserve mass, the obtained level set function $\widetilde{\phi}$ is altered such that locally, mass is conserved up to a certain tolerance.

To couple the level set and Volume of Fluid functions, the first derivative of $\widetilde{\phi}$ is approximated using central differences. Using these derivatives, the level set values at the cell vertices can be approximated. Now, it is easily determined if an interface exists in this cell by comparing the sign of the minimum and maximum level set values. Additionally, there will be cell edges along which the sign of the level set function changes. The zero points along these edges are obtained through linear interpolation, and a linear interface is constructed using these points. Now, an interface is given, so the volume occupied by the fluid is easily determined by calculating the area of a polygon. This defines a cheap function f such that the VoF function $\psi = f(\phi, \nabla\phi)$. Since f is cheap, its inverse g such that

$$g(f(\phi, \nabla\phi), \nabla\phi) = \phi \tag{2.32}$$

can be approximated numerically (e.g. by using a root finder) when correcting the local level set function.

Now that the functions f and g are known, the level set function $\phi^{(n+\frac{1}{2})}$ can be determined. First, construct from the temporary level set function $\widetilde{\phi}$ a VoF function $\widetilde{\psi} = f(\widetilde{\phi}, \nabla\widetilde{\phi})$. Then, in every computational cell, compute the error

$$\Delta\psi = |\widetilde{\psi} - \psi^{(n+\frac{1}{2})}|. \tag{2.33}$$

If the error is too large, i.e. $\Delta\psi \geq \epsilon$ in some cell, update the level set using

$$\widetilde{\phi} = g(\psi^{(n+\frac{1}{2})}, \nabla\widetilde{\phi}) \tag{2.34}$$

where the derivative of the old $\widetilde{\phi}$ is used. Now $\nabla\widetilde{\phi}$ can be recalculated and the process is repeated. This continues until $\Delta\psi < \varepsilon$ holds in every computational cell. Then, $\phi^{(n+\frac{1}{2})} = \widetilde{\phi}$. For the inital advection step for the Volume of Fluid field $\psi^{(n-\frac{1}{2})} \to \psi^{(n+\frac{1}{2})}$, note that the level set function $\phi^{(n-\frac{1}{2})}$ is also necessary to provide information about the interface. Due to the adjustments that happen after advection, $\psi^{(n-\frac{1}{2})} = f(\phi^{(n-\frac{1}{2})}, \nabla\phi^{(n-\frac{1}{2})})$, so the level set function contains the correct information for the Volume of Fluid function to use.

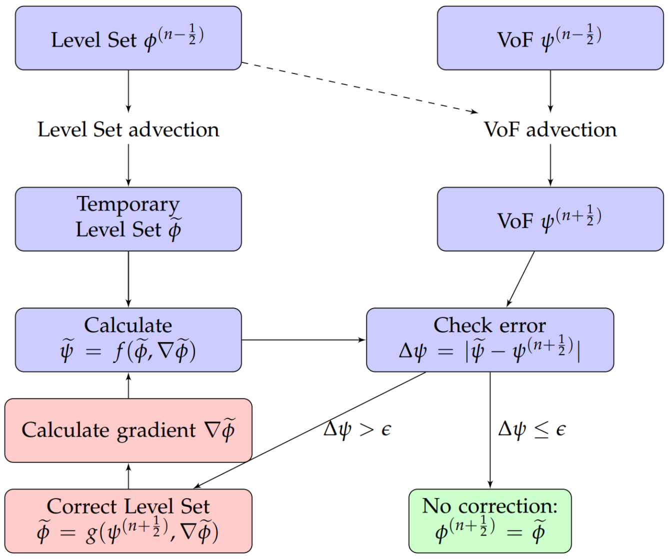
Level Set $\phi^{(n-\frac{1}{2})}$
VoF $\psi^{(n-\frac{1}{2})}$
Level Set advection
VoF advection
Temporary Level Set $\widetilde{\phi}$
VoF $\psi^{(n+\frac{1}{2})}$
Calculate $\widetilde{\psi} = f(\widetilde{\phi}, \nabla\widetilde{\phi})$
Check error $\Delta\psi = |\widetilde{\psi} - \psi^{(n+\frac{1}{2})}|$
Calculate gradient $\nabla\widetilde{\phi}$
$\Delta\psi > \epsilon$
$\Delta\psi \leq \epsilon$
Correct Level Set $\widetilde{\phi} = g(\psi^{(n+\frac{1}{2})}, \nabla\widetilde{\phi})$
No correction: $\phi^{(n+\frac{1}{2})} = \widetilde{\phi}$

Figure 6: Schematic overview of the MCLS algorithm. [10]

During the advection step, the level set function quickly loses its signed distance property $||\nabla\phi|| = 1$. Having a level set function double as a signed distance is nice when, for example, approximating the gradient or determining regularized fluid viscosity. Thus, every few iterations the level set function is re-initialised to a distance function. This is done by solving the equation

$$\frac{\partial\phi}{\partial\tau} = \mathrm{sign}(\phi_n)(1 - ||\nabla\phi||_2), \quad \phi(0) = \phi_n \tag{2.35}$$

where τ is a pseudo time. The equation is integrated over this pseudo time until a steady-state solution has been obtained. Additional measures are applied to reduce movement of the interface, and an amount of time steps is chosen such that the level set function is reinitialised in a neighbourhood of about 5 cells around the interface to reduce computational effort, as outside this region the level set field is not used.

This algorithm has some differences with the CLSVOF method as outlined in [11]. In the CLSVOF method, the interface reconstruction for VoF advection creates a level set function for a linear interface, where coefficients are determined to minimise the error in the level set function in a 3×3 grid around the interface cell. This line is then shifted to agree with the local value of the VoF field. Additionally, the coupling of level set and VoF is done in such a way that preserves the signed distance property of the level set function. In an area of K cells around an interface cell, the level set function in cell (i', j') is set to a value based on the sign of the level set value in cell centre

(i', j') and the value in the closest cell vertex or edge centroid of the interface cell. If these values have opposite sign, this indicates an interface must be present between these two points, so the level set function is set to the distance between these points. If they have equal sign instead, no interface separates these two points, so the shortest distance from cell centre (i', j') to the reconstructed interface in the interface cell is used instead.

2.5 Concluding remarks

In this section, the baseline level set and volume of fluid interface methods have been described and analysed. The benefits and drawbacks of each method have been highlighted to illustrate the motivation behind working towards a new method. While dual interface methods nullify some of the drawbacks involved with both of these methods, mass conservation only holds while donating regions are being used for the advection step of the volume of fluid field. The methods described in this section do allow for better understanding of further steps in this research. For example, a discretisation method is required for dealing with unstructured grids. Normal finite volume methods are difficult to implement on non-regular grids, and finite element methods cannot be used to create equivalent methods. Instead, the discontinuous Galerkin method will be used, which is described in more detail in section 3. Additionally, it is likely that an VoF advection scheme must be used that does not inherently conserve mass to avoid using donating regions. Instead, other methods must be sought to guarantee conservation. This situation seems very similar to the optimisation mass correction step for the level set method as presented in [14, 15], especially when combined with the discontinuous Galerkin discretisation. Thus, it may be worthwhile to explore these methods.

For the interface reconstruction step, different procedures are used in the described methods. The method in [1] uses volume fractions inside a neighbourhood to approximate the interface normal, [7, 8] use the fluid centroids to instead use only local data, and dual methods use the slope of the level set function either locally [9] or again in a neighbourhood [11]. The method using neighbouring cells to approximate the interface normal direction are likely unusable for the purposes of this research, as unstructured grids make implementation difficult. So then, the problem becomes deciding between the MCLS implementation of interface reconstruction or using the moment of fluid method instead. While the MCLS method is much cheaper, as it only requires two linear interpolations per cell, the moment of fluid reconstruction method can be shown to produce optimal results in certain circumstances. However, it is not clear if the added accuracy in interface reconstruction outweighs the extra computation time that is required to compute the optimal interface, so more research should be done for this comparison.

The added time complexity of any different discretisations or correction steps should also be researched. The comparison in runtime of the created method with existing interface methods is a step that lacks in other research like [9, 8, 11]. This comparison, however, is extremely important since having an unconstrained time step is not relevant when one iteration takes many times longer than for methods with time step restrictions. For the MCLS method, it is then also not known how the added complexity of the level set correction step compares to the time saved in the interface reconstruction step. Since methods like discontinuous Galerkin discretisation and optimisation based mass correction steps are being considered for this research, which have a large amount of variables or need an unknown amount of iterations, these choices should be justified using numerical results.

3 Discontinuous Galerkin

As mentioned in section 2, a method that can be applied to general unstructured grids is preferred. Since finite volume methods are difficult to implement on such grids, another discretisation method must be found. For general differential equation on this type of grids, a finite element discretisation might be used, but for conservation laws (such as the advection equation), the symmetry in basis functions might cause stability problems [13]. As such, the discontinuous Galerkin (DG) discretisation [13, 3] is considered instead. DG discretisation can be concisely described as a combination of the finite element and finite volume methods. Instead of using global basis functions, the basis differs on each element, and the solution is allowed to be multi-valued on element boundaries to grant each element some independence from the rest. Adjacent elements are then only coupled through a 'numerical flux' which depends on the multiple flux values on element boundaries, which can be used to recreate finite volume fluxes (e.g. using 'upwind' fluxes). In this section, first the Galerkin FEM is described before the DG method is explicitly stated to highlight the similarities between the two.

3.1 Theory behind DG discretisation

For the advection equation

$$\frac{\partial \phi}{\partial t} + \boldsymbol{u} \cdot \nabla \phi = 0 \tag{3.1}$$

assume that the numerical solution ϕ_h is in some space V_h spanned by basis functions $N^1(\boldsymbol{x}), ..., N^n(\boldsymbol{x})$, so

$$\phi_h(\boldsymbol{x}) = \sum_{k=1}^{n} \phi(\boldsymbol{x}^k) N^k(\boldsymbol{x}) \tag{3.2}$$

where the basis functions are chosen such that ∇N^j is integrable over Ω. The optimal solution $\phi_h \in V_h$ has been found when the residual

$$\mathcal{R}_h(\boldsymbol{x}, t) = \frac{\partial \phi_h}{\partial x} + \boldsymbol{u} \cdot \nabla \phi_h \tag{3.3}$$

is orthogonal to the space V_h. Thus, we want to find ϕ_h such that

$$\int_\Omega \mathcal{R}_h N^i d\Omega = 0 \quad \forall i = 1, ..., n \tag{3.4}$$

This leads to the Galerkin FEM scheme

$$\mathcal{M} \frac{\mathrm{d}\boldsymbol{\phi}_h}{\mathrm{d}t} + \mathcal{S} \boldsymbol{\phi}_h = 0 \tag{3.5}$$

where

$$\mathcal{M}_{ij} = \int_\Omega N^i N^j \, d\Omega, \qquad \mathcal{S}_{ij} = \int_\Omega N^i \; \boldsymbol{u} \cdot \nabla N^j \, d\Omega \tag{3.6}$$

The Galerkin FEM scheme is a valid discretisation for many differential equations, but might induce instabilities when applied to the advection equation. Instead, elements from finite volume methods are incorporated into this scheme, leading to the discontinuous Galerkin method [13]. Instead of using global basis functions, the space is separated into K elements $D^1, ..., D^k$ such that

$$\Omega \simeq \Omega_h = \bigcup_{k=1}^{K} D^k \tag{3.7}$$

Then, the numerical approximation is approximated at each element

$$\phi_h(\boldsymbol{x}) = \bigoplus_{k=1}^{K} \phi_h^k(\boldsymbol{x}), \qquad \phi_h^k(\boldsymbol{x}) = \sum_{i=1}^{n_p} \phi_h^k(\boldsymbol{x}_i) N_i^k(\boldsymbol{x}), \;\; \boldsymbol{x} \in D^k \tag{3.8}$$

The solution inside each element is thus a linear combination of local basis functions, but the solution need not be continuous across element boundaries. Again, the residual (3.3) is required to be orthogonal to the basis functions, but now this orthogonality only needs to hold on the element D^k,

$$\int_{D^k} \mathcal{R}_h N_i^k d\Omega = 0 \quad \forall i = 1, ..., n_p \tag{3.9}$$

Additionally, a numerical flux $\boldsymbol{f}^*$ is introduced, so

$$\int_{D^k} \mathcal{R}_h N_i^k d\Omega = \int_{\partial D^k} \hat{\boldsymbol{n}} \cdot \left[\boldsymbol{u}\phi_h^k - \boldsymbol{f}^*\right] N_i^k d\Gamma \tag{3.10}$$

Writing this in a similar form to (3.5) results in

$$\mathcal{M}^k \frac{\mathrm{d}\phi_h^k}{\mathrm{d}t} + \mathcal{S}^k \phi_h^k = \int_{\partial D^k} \hat{\boldsymbol{n}} \cdot \left[\boldsymbol{u}\phi_h^k - \boldsymbol{f}^*\right] N_i^k d\Gamma \tag{3.11}$$

where

$$\mathcal{M}_{ij}^k = \int_{D^k} N_i^k N_j^k \, d\Omega, \quad \mathcal{S}_{ij}^k = \int_{D^k} N_i^k \; \boldsymbol{u} \cdot \nabla N_j^k \; d\Omega \tag{3.12}$$

The numerical flux $\boldsymbol{f}^*$ is a combination of the (unequal) fluxes on either side of an element boundary. Adjacent elements are now only coupled by their flux $\boldsymbol{u} \cdot \nabla\phi$, so DG allows the level set value inside an element to be altered without changing nearby elements.

The evaluation of integrals (3.12) depends heavily on the shape of the elements in the grid. For triangular elements, as described in [13], the element can be mapped to a ‘standard’ triangle. This requires only one set of basis functions to be defined, and the mapping to the reference element is easily constructed. For square or rectangular quadrilaterals, this method can be used as well. When moving to elements with more vertices or general quadrilaterals, this becomes a lot more difficult as the transformation Jacobian is no longer constant. A method to calculate these integrals and use DG on polygons is presented in [3], which will be described hereafter.

3.2 DG on general polygons

The value of ϕ is now approximated using a *modal* expansion

$$\phi_h^k \approx \sum_{m=1}^{n_p} \widetilde{\phi}_m^k(t) \widetilde{N}_m^k(\boldsymbol{x}), \;\; \boldsymbol{x} \in D^k \tag{3.13}$$

with modal basis functions $\widetilde{N}_m$, while the flux terms are approximated using a *nodal* basis associated with the interpolation points $\{\boldsymbol{x}_m \in D^k\}_{m=1,\dots,m_p}$ by

$$f_x(\phi_h^k, \boldsymbol{x}) \approx \sum_{m=1}^{m_p} f_{x,m}^k N_m^k(\boldsymbol{x}) \tag{3.14}$$

where the basis functions approximately have the interpolation property

$$N_m^k(\boldsymbol{x}_n) \approx \delta_{m,n} \tag{3.15}$$

Using these bases, we can derive a system of ODE similar to (3.11) on D^k (dropping the superscript):

$$\sum_{n=1}^{n_p} \mathcal{M}_{mn} \frac{\mathrm{d}\widetilde{\phi}_n}{\mathrm{d}t} + \sum_{n=1}^{m_p} (\mathcal{S}_{x,mn} f_{x,n} + \mathcal{S}_{y,mn} f_{y,n}) = \int_{\partial D^k} \hat{\boldsymbol{n}} \cdot [\boldsymbol{f} - \boldsymbol{f}^*] \, \widetilde{N}_m \; d\Gamma \tag{3.16}$$

for all $m = 1, ..., n_p$. Here,

$$\mathcal{M}_{mn} = \int_{D^k} \widetilde{N}_m(\boldsymbol{x}) \widetilde{N}_n(\boldsymbol{x}) \; d\Omega, \quad \mathcal{S}_{x,mn} = \int_{D^k} \frac{\partial \widetilde{N}_m}{\partial x} N_n \; d\Omega, \quad \mathcal{S}_{y,mn} = \int_{D^k} \frac{\partial \widetilde{N}_m}{\partial y} N_n \; d\Omega \tag{3.17}$$

and $\boldsymbol{f} = \{\boldsymbol{f}_x, \boldsymbol{f}_y\} = \boldsymbol{u}\phi_h$, $\boldsymbol{f}^*$ the regular and numerical flux respectively. The evaluation of these integrals is the core issue when defining a DG scheme.

The element D^k is mapped unto a reference element $\mathcal{K}$, which is obtained by the mapping

$$\boldsymbol{\xi}(\boldsymbol{x}) = \frac{\boldsymbol{x} - \boldsymbol{x}_c}{\Delta X} \tag{3.18}$$

with $\boldsymbol{x}_c$ the centroid of D^k, and $\Delta X = \max(x_{\max} - x_{\min}, y_{\max} - y_{\min})$. Then, the monomial basis $\{\pi_m\}_{m=1,...,n_p}$ of $P^p(\mathcal{K})$, the space of polynomials with degree at most p, is constructed by

$$\pi_m(\boldsymbol{\xi}) = \xi^i \eta^j, \quad i, j \geq 0,\ i + j \leq p \tag{3.19}$$

$$m = \frac{1}{2}(i + j + 1)(i + j + 2) - i \tag{3.20}$$

Note that the number of basis functions $n_p = (p + 1)(p + 2)/2$. Using a modified Gram-Schmidt process, the orthonormal basis $\{\widetilde{N}_m(\boldsymbol{\xi})\}_{m=1,...,n_p}$ is constructed. For this process, note that on a square reference element

$$\int_{[-1,1]^2} \xi^i \eta^j \, d\xi d\eta = \begin{cases} \frac{4}{(i+1)(j+1)} & \text{if } i, j \text{ even} \\ 0 & \text{otherwise} \end{cases} \tag{3.21}$$

The basis $\{\widetilde{N}_m(\boldsymbol{x})\}_{m=1,...,n_p}$ is then defined by remapping the basis functions to the physical element D^k:

$$\widetilde{N}_m(\boldsymbol{x}) = (\widetilde{N}_m(\boldsymbol{\xi}) \circ \boldsymbol{\xi}(\boldsymbol{x})) \tag{3.22}$$

Note that the basis functions $\widetilde{N}_m(\boldsymbol{x})$ span the space $P^p(D^k)$ and are orthogonal in the L_2 norm, so

$$\int_{D^k} \widetilde{N}_m \widetilde{N}_n \, d\Omega = 0 \text{ if } n \neq m$$

which implies that $\mathcal{M}$ is a diagonal matrix. The transformed set $\widetilde{N}_m(\boldsymbol{\xi})$ are even orthonormal. Given a nodal set $\{\boldsymbol{x}_m\}_{m=1,...,m_p}$, the nodal basis functions $N_m(\boldsymbol{x})$ can be constructed from the conditions

$$N_m(\boldsymbol{x}_n) = \delta_{m,n}, \quad \phi(\boldsymbol{x}) = \sum_{m=1}^{n_p} \widetilde{\phi}_m \widetilde{N}_m(\boldsymbol{x}) \doteq \sum_{m=1}^{m_p} \phi_m N_m(\boldsymbol{x}) \tag{3.23}$$

This results in the transformation

$$\boldsymbol{\phi} = \mathbf{V}\widetilde{\boldsymbol{\phi}}, \quad \widetilde{\boldsymbol{N}} = \mathbf{V}^T \boldsymbol{N} \tag{3.24}$$

where $\boldsymbol{\phi} = \{\phi_1, ..., \phi_{m_p}\}^T$, $\widetilde{\boldsymbol{\phi}} = \{\widetilde{\phi}_1, ..., \widetilde{\phi}_{n_p}\}^T$, $\boldsymbol{N} = \{N_1, ..., N_{m_p}\}^T$ and $\widetilde{\boldsymbol{N}} = \{\widetilde{N}_1, ..., \widetilde{N}_{n_p}\}^T$. $\mathbf{V}$ is a generalized Vandermonde matrix given by

$$V_{m,n} = \widetilde{N}_n(\boldsymbol{x}_m), \quad m = 1, ..., m_p,\ n = 1, ..., n_p \tag{3.25}$$

The only remaining task is to find a nodal set $\{\boldsymbol{x}_m\}_{m=1,...,m_p}$. For general polygons, this process is not explicitly described in [3], but for a square reference element, the nodal points are given by a Legendre-Gauss-Lobatto disribution, i.e.

$$\boldsymbol{\xi}_{(p+1)j+i+1} = (x_i, x_i) \quad 0 \leq i, j \leq p \tag{3.26}$$

where the set $\{x_i\}_{i=0,...,p}$ are the zeros of the function $(1 - x^2)\frac{\mathrm{d}P_p(x)}{\mathrm{d}x}$. The nodal points can then be remapped to the physical element. Note that in this instance, $m_p = (p + 1)^2 \geq n_p$. Note that since $\mathbf{V}$ is a $n_p \times m_p$ matrix, when $m_p > n_p$ the inverse is not defined. Thus, a pseudoinverse

$$\mathbf{V}^{-1} = \left(\mathbf{V}^T \mathbf{V}\right)^{-1} \mathbf{V}^T \tag{3.27}$$

is defined to be used in the inverse transformations $\boldsymbol{\phi} \to \widetilde{\boldsymbol{\phi}}$ and $\widetilde{\boldsymbol{N}} \to \boldsymbol{N}$.

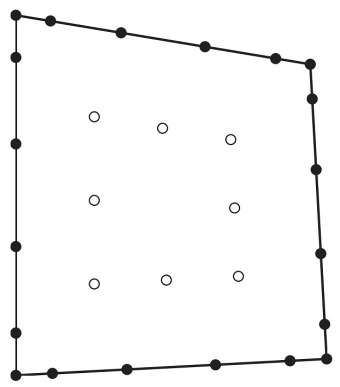

Figure 7: Example of a nodal point distribution on a quadrilateral element. [3]

If the matrix

$$\mathcal{D}_{x,ij} = \left.\frac{\partial \widetilde{N}_j}{\partial x}\right|_{\boldsymbol{x}_i} \tag{3.28}$$

is defined, then all tools necessary for calculation of the mass and stiffness matrices are present.

Firstly, for the mass matrix, note that

$$M_{ij} = \int_{D^k} \widetilde{\boldsymbol{\phi}}\widetilde{\boldsymbol{\phi}}\ d\Omega = (\Delta X)^2 \int_{\mathcal{K}} \widetilde{\boldsymbol{\phi}}\widetilde{\boldsymbol{\phi}}\ d\boldsymbol{\xi} = (\Delta X)^2 \delta_{i,j} \tag{3.29}$$

where $(\Delta X)^2$ is the Jacobian of transformation (3.18). Calculation of the stiffness matrix is done with respect to the nodal basis instead. Since $\widetilde{\boldsymbol{N}} = \mathbf{V}^T \boldsymbol{N}$, it holds that $\frac{\partial \widetilde{\boldsymbol{N}}}{\partial x} = \mathbf{V}^T \frac{\partial \boldsymbol{N}}{\partial x}$. Thus,

$$\mathcal{S}_x = \mathbf{V}^T \mathscr{S}_x, \quad \text{with } \mathscr{S}_{x,ij} = \int_{D^k} \frac{\partial N_i}{\partial x} N_j\ d\Omega \tag{3.30}$$

The derivative can be expanded into the nodal approximation similar to (3.14), i.e.

$$\frac{\partial N_i}{\partial x}(\boldsymbol{x}) = \sum_{n=1}^{m_p} D_{x,ni} N_n(\boldsymbol{x}), \quad \text{with } D_{x,ij} = \left.\frac{\partial N_j}{\partial x}\right|_{\boldsymbol{x}_i} \tag{3.31}$$

Using the derivative matrix $\mathbf{D}_x$, manipulating the result gives

$$\mathscr{S}_x = \mathbf{D}_x^T \mathscr{M} \tag{3.32}$$

where $\mathscr{M}$ is a mass matrix with respect to the nodal basis functions, defined as

$$\mathscr{M} = \int_{D^k} \boldsymbol{N}\boldsymbol{N}^T\ d\Omega = (\Delta X)^2 \int_{\mathcal{K}} \boldsymbol{N}\boldsymbol{N}^T\ d\boldsymbol{\xi} = (\Delta X)^2 \int_{\mathcal{K}} (\mathbf{V}^{-1})^T \widetilde{\boldsymbol{N}}\widetilde{\boldsymbol{N}}^T \mathbf{V}^{-1}\ d\boldsymbol{\xi} = (\Delta X)^2 (\mathbf{V}^{-1})^T \mathbf{V}^{-1} \tag{3.33}$$

due to the orthonormal property of the modal basis functions on the reference element $\mathcal{K}$. Now, only the derivative matrix must be evaluated. Since $\frac{\partial \widetilde{\boldsymbol{N}}}{\partial x} = \mathbf{V}^T \frac{\partial \boldsymbol{N}}{\partial x}$, this matrix can be determined by

$$\mathbf{D}_x \mathbf{V} = \mathcal{D}_x \tag{3.34}$$

where the matrix $\mathcal{D}_x$ has entries

$$\mathcal{D}_{x,ij} = \left.\frac{\partial \widetilde{N}_j}{\partial x}\right|_{\boldsymbol{x}_i} \tag{3.35}$$

which can be computed from the derivatives of the monomials (3.19). Computation of $\mathcal{S}_y$ is analogous to $\mathcal{S}_x$.

Finally, for computation of the edge integral, the element boundary is divided into multiple straight edge segments

$$\partial D^k = \bigcup_{j=1}^{n_e} \partial D_j^k \tag{3.36}$$

with n_e the total number of edge elements needed. Then, the edge integral is calculated using

$$\int_{\partial D_j^k} \hat{\boldsymbol{n}} \cdot (\boldsymbol{f} - \boldsymbol{f}^*) \widetilde{N}_i \; d\Gamma \approx \frac{|(\partial D_j^k)|}{2} \sum_{m=1}^{p+1} \left[\int_{-1}^{1} \widetilde{N}_i \bar{N}_m \; d\varsigma \right] ((\boldsymbol{f} - \boldsymbol{f}^*) \cdot \hat{\boldsymbol{n}})(\boldsymbol{x}(\varsigma_m)) \tag{3.37}$$

Here, the flux is interpolated using one-dimensional Lagrange basis functions $\bar{N}_m$ of order p defined on the set of interpolation nodes $\{\varsigma_m\}$ with the Gauss-Lobatto distribution. The value of $\boldsymbol{x}(\varsigma_m)$ is the result of the linear mapping $[-1,1] \to \partial D_j^k$ and can be determined easily from the edge points of ∂D_j^k.

Note that the described method is very expensive in terms of computation time or memory usage. Since, in general, every control volume is unique, each cell will have different basis functions and nodal points. This requires the Vandermonde matrix $\mathbf{V}$ of each cell to be stored, or to be calculated at every time step. Alternatively, for quadrilateral cells it is possible to calculate the relevant matrices on a square reference element $I_q = [-1,1]^2$ and map the results to the physical element D^k. The basis functions are then defined as $N_m(\boldsymbol{x}) = N_m(\boldsymbol{\xi}(\boldsymbol{x}))$, where the bi-linear mapping $I_q \to D^k$ is given by

$$\boldsymbol{x}(\boldsymbol{\xi}) = \sum_{i=1}^{4} \boldsymbol{x}_i^k L_{q,i}(\boldsymbol{\xi}) \tag{3.38}$$

where $\boldsymbol{x}_i^k$ denote the ith vertex of D^k, labelled in a counter-clockwise manner, and

$$\begin{aligned} L_{q,1} &= \frac{(1-\xi)(1-\eta)}{4} & \qquad L_{q,2} &= \frac{(1+\xi)(1-\eta)}{4} \\ L_{q,3} &= \frac{(1+\xi)(1+\eta)}{4} & \qquad L_{q,4} &= \frac{(1-\xi)(1+\eta)}{4} \end{aligned} \tag{3.39}$$

Note that for this transformation, it is wanted that $L_{q,i}(\boldsymbol{\xi}(\boldsymbol{x}_i^k)) = \delta_{i,j}$. However, only for quadrilaterals with two parallel edges is the mapping (3.38) constant. For general quadrilaterals the mass and stiffness matrices thus can no longer be obtained by scaling the element matrices of the reference element. However, by approximating the inverse transformation $\boldsymbol{\xi}(\boldsymbol{x})$, a less accurate but more memory-friendly approach is constructed.

3.3 Concluding remarks

This section describes the discontinuous Galerkin method, which aims to be a combination of the finite element and finite volume discretisation methods. This is achieved by creating a 'numerical flux', whose main justification is its purpose of connecting adjacent elements, which seems to be necessary to avoid instabilities when handling conservation laws. The method allows advection of the level set field on a non-structured grid, which works towards the goals set for this research. This method still has some properties that should be taken into careful consideration when using, however. The method is purely local, but it can only achieve this by allowing multi-valued solutions at element boundaries. This means that the method has a large number of unknowns, though only the interface cells need a high amount of nodal points. The effect of this on computation time and memory usage should be tested.

Additionally, the grid that is used with this method must be chosen carefully. While a method is described to allow the use of DG on any grid, a method using 'reference' elements is only described for triangular and quadrilateral elements, with the latter requiring an approximate inverse transformation to avoid taking expensive steps. Since only a small amount of elements can be considered separately, it is questionable whether a grid consisting of many pentagonal or larger polygonal cells is worth using. The approximate inverse transformation should induce more errors for larger polygons, so at some point is it no longer feasible to use a regular polygon as reference element. Whether this cut-off is at pentagonal cells or not should be researched.

4 Optimisation-based mass conservation

In subsection 2.5, it is noted that the volume of fluid advection method will most likely not be mass conserving if donating regions are to be avoided. As such, a different method for mass conservation must be found. The optimisation-based approach described in [14, 15] might be a solution to this problem. While the method described is applied to level set fields, with some alteration a similar method should be able to work for volume of fluid fields as well. Because of this, the *scalar control approach* and the *vector control approach* are described.

4.1 Scalar control approach

Suppose that the level set function ϕ satisfies the equation

$$\frac{\partial H(\phi)}{\partial t} + \nabla \cdot (\boldsymbol{u} H(\widetilde{\phi}) - \nabla v) = 0 \tag{4.1}$$

where v is a scalar-valued control variable, $\widetilde{\phi}$ indicates the advected level set field using a nonconservative method, and

$$H(\phi) = \begin{cases} 1 & \phi \geq 0 \\ 0 & \phi < 0 \end{cases} \tag{4.2}$$

a step function. Assuming that the interface does not cross the boundary $\partial\Omega$, so $H(\widetilde{\phi}) = 0$, the integral conservation law holds under the Neumann boundary condition

$$\hat{\boldsymbol{n}} \cdot \nabla v = 0 \quad \text{on } \partial\Omega \tag{4.3}$$

The goal of this optimisation-based correction is to find a solution (ϕ, v) to the transport equation (4.1) such that ϕ and $\widetilde{\phi}$ are as close together as possible. The cost functional used in [14] is

$$J(\phi, v, \widetilde{\phi}) = \frac{1}{2}||\phi - \widetilde{\phi}||^2 + \frac{\beta}{2}||v||^2 \tag{4.4}$$

with $\beta > 0$ a regularisation constant and $|| \cdot ||$ the L^2 norm on Ω. In general, this optimisation problem aims to minimise the cost functional with respect to variables ϕ and v such that these variables conform to the PDE constraint (4.1) with relevant natural boundary condition (4.3).

To apply this optimisation problem, first a discretisation must be applied. First, equations (3.1) and (4.1) are discretised in time. For example, using a two-level θ scheme,

$$\widetilde{\phi}^{n+1} + \theta \Delta t \boldsymbol{u} \cdot \nabla \widetilde{\phi}^{n+1} = \phi^n - (1-\theta)\Delta t \boldsymbol{u} \cdot \nabla \phi^n \tag{4.5}$$

and

$$H(\phi^{n+1}) - \nabla^2 v^{n+1} = H(\phi^n) - \theta \Delta t \nabla \cdot (H(\widetilde{\phi}^{n+1})\boldsymbol{u}) - (1-\theta)\Delta t \nabla \cdot (H(\phi^n)\boldsymbol{u}) \tag{4.6}$$

For this, note that (4.1) can be written as

$$\frac{\partial H(\phi)}{\partial t} + \nabla \cdot (\boldsymbol{u} H(\widetilde{\phi})) = \nabla^2 v \tag{4.7}$$

and that the time step Δt can be absorbed into the control variable v, thus leading to the discretisation (4.6). Discretisation in space is performed using a finite element method, which means that ϕ and u are discretised using basis functions

$$\phi_h(\boldsymbol{x}) = \sum_{k=1}^{K} \phi_k N^k(\boldsymbol{x}), \quad v(\boldsymbol{x}) = \sum_{k=1}^{K} v_k N^k(\boldsymbol{x}) \tag{4.8}$$

Substituting these approximations into (4.6) and integrating over the domain yields a nonlinear system of equations

$$g(\phi^{n+1}) + S v^{n+1} = g(\phi^n) + \theta \Delta t f(\widetilde{\phi}^{n+1}) + (1-\theta)\Delta t f(\phi^n) \tag{4.9}$$

where

$$\begin{aligned} s_{ij} &= \int_\Omega \nabla N^i \cdot \nabla N^j \; d\Omega \\ f_i(\phi) &= \int_\Omega \nabla N^i \cdot \boldsymbol{u} H(\phi) \; d\Omega \\ g_i(\phi) &= \int_\Omega N^i H(\phi) \; d\Omega \end{aligned} \tag{4.10}$$

It can be verified that a solution to (4.9) inherently satisfies a discrete mass conservation law for any choice for the control variable v. The approximations can also be substituted into the cost functional (4.4), giving

$$J(\phi_h, v_h, \widetilde{\phi}_h) = \frac{1}{2}\boldsymbol{\phi}^T M \boldsymbol{\phi} - \boldsymbol{\phi}^T M \widetilde{\boldsymbol{\phi}} + \frac{1}{2}\widetilde{\boldsymbol{\phi}}^T M \widetilde{\boldsymbol{\phi}} + \frac{\beta}{2}\boldsymbol{v}^T M \boldsymbol{v} \tag{4.11}$$

where $\boldsymbol{\phi}, \boldsymbol{v}, \widetilde{\boldsymbol{\phi}}$ denote the global vectors of nodal values and M is the mass matrix with entries

$$M_{ij} = \int_\Omega N^i N^j \; d\Omega \tag{4.12}$$

In numerical implementations, the Heaviside step function H is replaced with a smoothed approximation

$$H_\epsilon(\phi) = \frac{1}{2}\left(\frac{\phi}{\sqrt{\phi^2 + \epsilon^2}} + 1\right) \tag{4.13}$$

with regularisation parameter $\epsilon > 0$ smaller than the mesh size h to avoid mass conservation errors.

For the optimisation problem, the variables ϕ_h, v_h must be found that minimise (4.11) subject to equation (4.9). Thus, this optimisation problem has an associated Lagrangian

$$\mathcal{L}(\phi, v, p) = \frac{1}{2}\boldsymbol{\phi}^T M \boldsymbol{\phi} - \boldsymbol{\phi}^T M \widetilde{\boldsymbol{\phi}} + \frac{1}{2}\widetilde{\boldsymbol{\phi}}^T M \widetilde{\boldsymbol{\phi}} + \frac{\beta}{2}\boldsymbol{v}^T M \boldsymbol{v} + \boldsymbol{p}^T q(\phi, v, \widetilde{\phi}) \tag{4.14}$$

where $q(\phi, v, \widetilde{\phi})$ is the residual of (4.9) and $\boldsymbol{p}$ is a vector containing Lagrange multipliers. The derivatives of L with respect to the three variables ϕ, v, p give the first-order optimality conditions

$$\begin{aligned} M\boldsymbol{\phi}^{n+1} + K(\phi^{n+1})\boldsymbol{p}^{n+1} &= M\widetilde{\boldsymbol{\phi}}^{n+1} \\ \beta M \boldsymbol{v}^{n+1} + S\boldsymbol{p}^{n+1} &= 0 \\ q(\phi^{n+1}, v^{n+1}, \widetilde{\phi}^{n+1}) &= 0 \end{aligned} \tag{4.15}$$

The nonlinear operator $K(\phi) = \left\{\frac{\partial g_i(\phi)}{\partial \phi_j}\right\}$ requires the derivative of the smoothed Heaviside function with respect to the discretisation constants ϕ_j, so

$$K_{ij}(\phi) = \int_\Omega N^i H'_\epsilon(\phi_h) N^j \; d\Omega \tag{4.16}$$

where the derivative is given by

$$H'_\epsilon(\phi_h) = \frac{1}{2}\frac{\epsilon^2}{(\phi^2 + \epsilon^2)^{1.5}} \tag{4.17}$$

Since the system of equations (4.15) is nonlinear, the solution must be found iteratively. From the system, the residual

$$\boldsymbol{r}^{(m-1)} = \begin{bmatrix} \boldsymbol{r}_\phi^{(m-1)} \\ \boldsymbol{r}_v^{(m-1)} \\ \boldsymbol{r}_p^{(m-1)} \end{bmatrix} = \begin{bmatrix} M(\widetilde{\boldsymbol{\phi}}^{n+1} - \boldsymbol{\phi}^{(m-1)}) - K(\phi^{(m-1)})\boldsymbol{p}^{(m-1)} \\ -\beta M \boldsymbol{v}^{(m-1)} - S\boldsymbol{p}^{(m-1)} \\ q(\phi^{(m-1)}, v^{(m-1)}, \widetilde{\phi}^{n+1}) \end{bmatrix} \tag{4.18}$$

where the superscript indicates the iterative approximation to the solution of the system. Since a solution for the problem $\boldsymbol{r} = 0$ must be found, the approximate solution can be updated using Newton's method

$$\begin{bmatrix} \boldsymbol{\phi}^{(m)} \\ \boldsymbol{v}^{(m)} \\ \boldsymbol{p}^{(m)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\phi}^{(m-1)} \\ \boldsymbol{v}^{(m-1)} \\ \boldsymbol{p}^{(m-1)} \end{bmatrix} + \begin{bmatrix} M & 0 & K(\phi^{(m-1)}) \\ 0 & \beta M & S \\ \frac{1}{\tau}M & S & 0 \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{r}_\phi^{(m-1)} \\ \boldsymbol{r}_v^{(m-1)} \\ \boldsymbol{r}_p^{(m-1)} \end{bmatrix} \tag{4.19}$$

The relaxation parameter $\tau > 0$ can be interpreted as a pseudo-time step. Instead of using the nonlinear state equation $q(\phi^{n+1}, v^{n+1}, \widetilde{\phi}^{n+1}) = 0$, the equation

$$M\frac{\boldsymbol{\phi}^{(m)} - \boldsymbol{\phi}^{(m-1)}}{\tau} + S(\boldsymbol{v}^{(m)} - \boldsymbol{v}^{(m-1)}) = q(\phi^{(m-1)}, v^{(m-1)}, \widetilde{\phi}^{n+1}) \tag{4.20}$$

is used. Each iteration (4.19) involves solving a linear system of the form

$$\begin{bmatrix} M & 0 & K(\phi^{(m-1)}) \\ 0 & \beta M & S \\ \frac{1}{\tau}M & S & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\delta\phi} \\ \boldsymbol{\delta v} \\ \boldsymbol{\delta p} \end{bmatrix} = \begin{bmatrix} \boldsymbol{r}_\phi \\ \boldsymbol{r}_v \\ \boldsymbol{r}_p \end{bmatrix} \tag{4.21}$$

which can be solved using a brute force direct method.

4.2 Vector control approach

As an alternative to the scalar control approach, the *vector control approach* [15] uses the conservation law

$$\frac{\partial H(\phi)}{\partial t} + \nabla \cdot (\boldsymbol{u}H(\widetilde{\phi}) - \boldsymbol{v}) = 0 \tag{4.22}$$

where the vector control variable $\boldsymbol{v} = (v_1, v_2)^T$ is used instead of ∇v. Mass conservation is imposed using the boundary condition

$$\hat{\boldsymbol{n}} \cdot \boldsymbol{v} = 0 \quad \text{on } \partial\Omega \tag{4.23}$$

The proposed cost functional is given by

$$J(\phi, \boldsymbol{v}, \widetilde{\phi}) = \frac{1}{2}||\phi - \widetilde{\phi}||^2 + \frac{\beta_1}{2}||v_1||^2 + \frac{\beta_2}{2}||v_2||^2 \tag{4.24}$$

where again, $||\cdot||$ denotes the L^2 norm on Ω.

The same discretisation method used in the scalar control approach now yields a state equation similar to (4.9)

$$g(\phi^{n+1}) + C_1 v_1^{n+1} + C_2 v_2^{n+1} = g(\phi^n) + \theta\Delta t f(\widetilde{\phi}^{n+1}) + (1-\theta)\Delta t f(\phi^n) \tag{4.25}$$

where the matrices C_1 and C_2 represent the two components of the discrete divergence operator and have entries

$$C_{1,ij} = \int_\Omega \frac{\partial N^i}{\partial x} N^j \, d\Omega, \qquad C_{2,ij} = \int_\Omega \frac{\partial N^i}{\partial y} N^j \, d\Omega$$

and f, g are defined as (4.10).

At each time step, the approximation $\widetilde{\phi}^{n+1}$ is determined by

$$M\widetilde{\phi}^{n+1} + \theta\Delta t C(\boldsymbol{u})\widetilde{\phi}^{n+1} = M\phi^n - (1-\theta)\Delta t C(\boldsymbol{u})\phi^n \tag{4.26}$$

where the mass matrix M and the discrete transport $C(\boldsymbol{u})$ have entries

$$M_{ij} = \int_\Omega N^i N^j \, d\Omega, \qquad C_{ij}(\boldsymbol{u}) = \int_\Omega N^i \boldsymbol{u} \cdot \nabla N^j \, d\Omega \tag{4.27}$$

Then, the solution $(\phi^{n+1}, \boldsymbol{v}^{n+1})$ is found that minimised (4.24) subject to (4.25).

The residual of the discrete state equation (4.25) is given by

$$q(\phi, v_1, v_2, \widetilde{\phi}) = g(\phi) - g(\phi^n) + C_1 v_1 + C_2 v_2 - [\theta\Delta t f(\widetilde{\phi}) + (1-\theta)\Delta t f(\phi^n)] \tag{4.28}$$

and the Lagrangian associated with the optimisation problem is

$$\mathcal{L}(\phi, v_1, v_2, p) = \frac{1}{2}\boldsymbol{\phi}^T M \boldsymbol{\phi} - \boldsymbol{\phi}^T M \widetilde{\boldsymbol{\phi}} + \frac{1}{2}\widetilde{\boldsymbol{\phi}}^T M \widetilde{\boldsymbol{\phi}} + \frac{\beta}{2}(\boldsymbol{v}_1^T M \boldsymbol{v}_1 + \boldsymbol{v}_2 M \boldsymbol{v}_2) + p^T q(\phi, v_1, v_2, \widetilde{\phi}) \tag{4.29}$$

The minimisation problem can then be solved in an iterative manner using

$$\begin{bmatrix} M & 0 & 0 & K(\phi^{(m-1)}) \\ 0 & \beta M & 0 & C_1^T \\ 0 & 0 & \beta M & C_2^T \\ K(\phi^{(m-1)}) & C_1 & C_2 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\delta\phi}^{(m)} \\ \boldsymbol{\delta v}_1^{(m)} \\ \boldsymbol{\delta v}_2^{(m)} \\ \boldsymbol{\delta p}^{(m)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{r}_\phi^{(m-1)} \\ \boldsymbol{r}_{v_1}^{(m-1)} \\ \boldsymbol{r}_{v_2}^{(m-1)} \\ \boldsymbol{r}_p^{(m-1)} \end{bmatrix} \tag{4.30}$$

where

$$\begin{bmatrix} \boldsymbol{r}_\phi^{(m-1)} \\ \boldsymbol{r}_{v_1}^{(m-1)} \\ \boldsymbol{r}_{v_2}^{(m-1)} \\ \boldsymbol{r}_p^{(m-1)} \end{bmatrix} = \begin{bmatrix} M(\widetilde{\boldsymbol{\phi}}^{(n+1)} - \boldsymbol{\phi}^{(m-1)}) - K(\phi^{(m-1)})p^{(m-1)} \\ -\beta M \boldsymbol{v}_1^{(m-1)} - C_1^T \boldsymbol{p}^{(m-1)} \\ -\beta M \boldsymbol{v}_2^{(m-1)} - C_2^T \boldsymbol{p}^{(m-1)} \\ -q(\phi^{(m-1)}, v_1^{(m-1)}, v_2^{(m-1)}, \widetilde{\phi}^{(n+1)}) \end{bmatrix} \tag{4.31}$$

Again, $K(\phi)$ is a weighted mass matrix, but instead of using the smoothed Heaviside function, it can be shown that the entries reduce to an integral over the interface

$$K_{ij} = \int_{\Gamma(\phi_h)} N^i N^j \, d\Omega \tag{4.32}$$

Additionally, the residual q can be written as a matrix-vector product by using

$$g(\phi) \approx K(\phi)\phi \tag{4.33}$$

While the system size in the vector control approach is significantly larger than in the scalar control approach, the system matrix may be better suited for the design of efficient solution techniques.

4.3 Concluding remarks

In the scalar and vector control approaches for optimisation-based mass conservation, the mass conservation equation is modified to include an extra control variable. This variable will correct the convective fluxes as to enforce mass conservation while minimising deviations from a non-conservative solution. To correct any change in mass, a functional is minimised under a mass-conserving constraint. Note that in the ideal case where the approximate solution is already mass-conserving (and a signed distance function), the functional should be minimised by this solution with a zero control variable. The resulting optimisation problem is solved by studying the Lagrangian, and the resulting system of equations is solved by fixed-point iteration.

The difference in time complexity between the scalar and vector control approaches should be investigated closely. Obviously, the system in the vector control approach is several times larger that the scalar approach system. Additionally, in [15] it shows that for the problems considered, the vector control approach consistently needs more fixed-point iterations than the scalar control approach. However, the time required to perform one iteration is not mentioned in the paper, so the two method might be comparable in terms of computational work. This likely depends heavily on the solution technique used, so it might be worthwhile to investigate both methods.

It is important to note that while this optimisation-based approach results in a mass conserving level set method, when applying this same method to the volume of fluid field, or equivalently to the level set method while using DG discretisation, the resulting solution is not unique. This occurs since for the cost functional it no longer makes a difference which cell gets altered, as all interaction between the cells remain the same. Thus, when using this method there should be another term added to the cost functional to ensure uniqueness of the solution.

5 Research Questions

The idea of this research is to avoid using the donating regions, and instead use the momentary fluxes through cell interfaces

$$\frac{\mathrm{d}\Psi}{\mathrm{d}t} + F_1^i\Big| + F_2^i\Big| = 0. \tag{5.1}$$

Note that the change in volume fraction can be linked to the shift in the level set field

$$\frac{\mathrm{d}\Psi}{\mathrm{d}t} = \frac{l_{if}}{(\Delta x)^2}\frac{\mathrm{d}\Phi}{\mathrm{d}t} = \frac{l_{if}}{(\Delta x)^2}u_F, \tag{5.2}$$

with l_{if} the length of the interface and u_F its velocity. Note that using the momentary fluxes instead of the time-averaged version (donating regions) allows for advection on unstructured grids as, unlike donating regions, the momentary fluxes do not use overlapping areas.

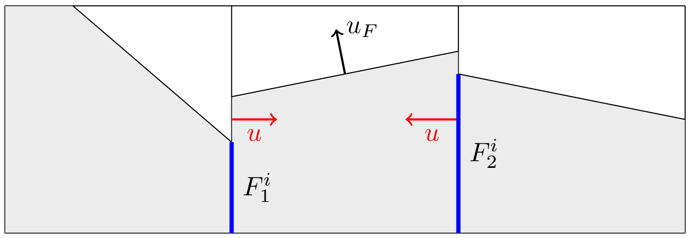

Figure 8: Illustration of the momentary fluxes. The regions defining the momentary fluxes can never overlap.

This idea forms the basis for further research. In order to provide intermediate goals for this research, some questions are posed:

(R1) *Is it worthwhile to use moment of fluid interface reconstruction alongside the level set implementation?*
As mentioned in section 2.5, it is not clear how the computation time of the level set interface reconstruction compares to that of the MoF method. Analysis of the run-times of these methods is necessary to justify either method. Since this time depends heavily on the implementation, this would be done using the average amount of iterations that are required, either for the MoF minimisation step or for the level set correction step.

(R2) *Which functional or optimality condition should be used to ensure a unique solution in the optimisation problem?*
In a discontinuous setting, the solution for the optimisation problem is generally not unique when using the functional and optimality conditions as given in section 4. Thus, additional conditions are required to ensure optimality. One possible idea is to add terms involving the total discontinuity in the interface or the curvature of the interface to the cost functional.

(R3) *Is it feasible to implement an improved method on polygonals with 5 or more vertices?*
To limit the memory use of a numerical method, the DG method cannot be performed on a large amount of unique polygons, so reference elements must be used. For quadrilaterals or polygons with more vertices, approximate transformations to the physical elements are required. It is then questionable whether it is feasible using, for example, pentagons instead of a purely quadrilateral grid.

(R4) *Is it possible for the improved method to have no stability condition on the time step size?*
A method without a Courant restriction on the time step would be the ideal end-point of this research. However, even a restriction that is several times larger than the CFL condition on the donating regions method would be a significant improvement.

5.1 Time schedule

	Week 21	Week 22	Week 23	Week 24	Week 25	Week 26	Week 27	Week 28	Week 29	Week 30	Week 31	Week 32	Week 33	Week 34	Week 35	Week 36	Week 37	Week 38	Week 39	Week 40	Week 41	Week 42	Week 43	Week 44	Week 45	Week 46	Week 47
Explicit MCLS and MoF	🟩	🟩	🟩	🟩																							
MCLS on DG grid					🟩	🟩	🟩	🟩	🟩																		
Optimisation method										🟩	🟩	🟩	🟩	🟩													
Polygonal cells															🟩	🟩	🟩	🟩									
Numerical efficiency																			🟩	🟩	🟩	🟩					
Report finishing																							🟩	🟩	🟩	🟩	

Research question (R1) can be answered during implementation of the baseline MCLS, (R2) will be investigated while working on the optimisation method. The research time for polygonal cells is also partially used as buffer time if previous parts get held up.

6 Conclusion

In this report, literature has been studied with the goal of working towards a new interface advection method to aid the solving of two-phase flows. First, the existing level set and volume of fluid interface methods have been discussed, and the steps leading to the creation of the mass conserving level set method have been highlighted. While the dual interface method solves some of the drawbacks of the individual level set and volume of fluid methods, the use of donating regions for VoF advection is still required for mass conservation. This means that a structured (rectangular) grid is necessary to allow simple dimensionally split methods, and a maximum allowed time step is imposed to avoid overfilling or over-emptying of cells. Avoiding the use of donating regions will require a discretisation method for the advection equation on polygonal cells, and likely requires a method for mass correction steps on the volume fractions. While the MCLS method is taken as a baseline for this research, it may be worthwhile to consider Moment of fluid interface reconstruction instead, where the fluid centroids are used to approximate the interface orientation.

For the advection equation, the discontinuous Galerkin method will be required on polygonal grids. This is due to the difficulty of finite volume discretisations on unstructured grids, and the possibility of instability when using finite element methods. The discontinuous Galerkin method aims to combine these two discretisation methods, which allows solving the advection equation on polygonal elements. When considering general polygons, however, the DG method becomes extremely expensive in terms of memory storage, so calculations must be done on a 'standard' polygon. The results are then mapped back to the original control volume. This mapping is generally difficult and expensive to find, so it must be approximated as well. As a result, only on triangular or rectangular grids can the DG method be applied exactly. For other grids, approximations must be used, so it may not be worthwhile to use DG on grids with a large number of unique polygons. An upside of DG is that adjacent elements are only coupled through a 'numerical flux', which means that any mass correction steps can be applied on a purely local basis.

Since the volume of fluid advection method likely cannot be inherently mass conserving, mass correction steps will be required. It appears plausible that the required steps will be similar to the described optimisation-based mass correction steps for the level set field. This can be done using either the scalar or vector control approaches, and difference in time complexity between these methods should be investigated. It should be noted that in the volume of fluid methods, the different volume fractions are independent, so solutions to the optimisation problem will not be unique. Thus, additional conditions should be added to ensure uniqueness.

The idea for further research is now to use the momentary fluxes instead of the donating regions for advection of the volume fractions, and apply the described methods to keep the algorithm mass conserving.

References

[1] Gabriel D Weymouth and Dick K-P Yue. Conservative Volume-of-Fluid method for free-surface simulations on Cartesian-grids. *Journal of Computational Physics*, 229(8):2853–2865, 2010.

[2] F Raees, DR Heul, and C Vuik. A mass-conserving level-set method for simulation of multiphase flow in geometrically complicated domains. *International Journal for Numerical Methods in Fluids*, 81(7):399–425, 2016.

[3] D Wirasaet, EJ Kubatko, CE Michoski, S Tanaka, JJ Westerink, and C Dawson. Discontinuous Galerkin methods with nodal and hybrid modal/nodal triangular, quadrilateral, and polygonal elements for nonlinear shallow water flow. *Comput. Methods Appl. Mech. Engrg*, 270:113–149, 2014.

[4] Jeffrey W Banks, T Aslam, and William J Rider. On sub-linear convergence for linearly degenerate waves in capturing schemes. *Journal of Computational Physics*, 227(14):6985–7002, 2008.

[5] Cyril W Hirt and Billy D Nichols. Volume of fluid (VOF) method for the dynamics of free boundaries. *Journal of computational physics*, 39(1):201–225, 1981.

[6] Qinghai Zhang and Aaron Fogelson. MARS: An analytic framework of interface tracking via mapping and adjusting regular semialgebraic sets. *SIAM Journal on Numerical Analysis*, 54(2):530–560, 2016.

[7] Vadim Dyadechko and Mikhail Shashkov. Moment-of-fluid interface reconstruction. *Los Alamos Report LA-UR-05-7571*, 2005.

[8] Matthew Jemison, Mark Sussman, and Marco Arienti. Compressible, multiphase semi-implicit method with moment of fluid interface representation. *Journal of Computational Physics*, 279:182–217, 2014.

[9] Sander Pieter van der Pijl. *Computation of bubbly flows with a mass-conserving level-set method*. Citeseer, 2005.

[10] GT Oud. A dual interface method in cylindrical coordinates for two-phase pipe flows. 2017.

[11] Mark Sussman and Elbridge Gerry Puckett. A coupled level set and volume-of-fluid method for computing 3D and axisymmetric incompressible two-phase flows. *Journal of computational physics*, 162(2):301–337, 2000.

[12] Matthew Jemison, Eva Loch, Mark Sussman, Mikhail Shashkov, Marco Arienti, Mitsuhiro Ohta, and Yaohong Wang. A coupled level set-moment of fluid method for incompressible two-phase flows. *Journal of Scientific Computing*, 54(2-3):454–491, 2013.

[13] Jan S Hesthaven and Tim Warburton. *Nodal discontinuous Galerkin methods: algorithms, analysis, and applications*. Springer Science & Business Media, 2007.

[14] Dmitri Kuzmin. An optimization-based approach to enforcing mass conservation in level set methods. *Journal of Computational and Applied Mathematics*, 258:78–86, 2014.

[15] Christopher Basting and Dmitri Kuzmin. Optimal control for mass conservative level set methods. *Journal of Computational and Applied Mathematics*, 270:343–352, 2014.