Volume-Preserving Continuous Galerkin Level-Set Approach for Linear Finite Elements

den Ouden-van der Horst, Dennis; Möller, Matthias

Volume-Preserving Continuous Galerkin Level-Set Approach for Linear Finite Elements

D. den Ouden-van der Horst[a,*], M. Möller[a]

[a] *Delft Institute of Applied Mathematics, Delft University of Technology, Van Mourik Broekmanweg 6, 2628 XE Delft, The Netherlands*

1. Introduction

Many physical phenomena can be described as the evolution of two phases coexisting within the same domain. Examples of such phenomena are the transport of gas and oil, solidification and phase transformations. Each of these phenomena require a description of the dynamics under which the phases change and a technique for tracking the interface between the relevant phases.

Currently several techniques exist for describing the evolution of an interface between two phases. We can distinguish these techniques as either interface-tracking or interface-capturing. Interface-tracking techniques commonly describe the interface exactly, for example by representing the interface explicitly in a mesh [15], by assuming a parametric shape of the interface (See for example [19]) or by introducing markers indicating one of the phases [7] or the interface [4], and tracking explicitly the evolution of this interface. The class of interface-capturing techniques describe the interface implicitly by a function, such as the level-set method [12], the volume-of-fluid method [9] and the moment-of-fluid method [5], and track the evolution of this function explicitly.

Recently, both methods have been combined to exploit the ease of capturing the location of the interface from the level-set method and the volume-preserving capacities of the volume-of-fluid method. So far, this coupling has only been performed on regular quadrilateral meshes adopting finite-volume discretisations [17] and on triangular meshes in the context of discontinuous-Galerkin finite-elements.

In this article we develop a volume-preserving level-set method by coupling a Galerkin level-set formulation based on linear triangles with the volume-of-fluid method on star-shaped polygonal finite-volume meshes.

This article will first introduce introduce the level-set method and the volume-of-fluid method and our choice of discretisation for each of the methods. Subsequently we will define the coupling between the two methods and the novel volume-correction algorithm which will ensure volume preservation during advection of the level-set and volume-of-fluid functions. Finally we will investigate the numerical and practical aspects of the volume-correction algorithm by several examples.

2. The numerical framework

2.1. The Level-Set Function on the Primal Mesh

The level-set method, as introduced by [12], considers the evolution of a sharp interface between two phases tracked by the zero level-set of a signed-distance function. Given the (closed) interface $\Gamma(t)$Interface between Ω^+ and Ω^-.Time. between the two phases $\Omega^+(t)$Domain where $\phi(\vec{x},t) > 0$. and $\Omega^-(t)$Domain where $\phi(\vec{x},t) < 0$., the level-set function $\phi(\vec{x},t)$Level-set function. is defined as:

$$\phi(\vec{x},t) = \begin{cases} \min\limits_{\vec{y}\in\Gamma(t)} \|\vec{x}-\vec{y}\| & \text{if } \vec{x}\in\Omega^+, \\ 0 & \text{if } \vec{x}\in\Gamma(t), \\ -\min\limits_{\vec{y}\in\Gamma(t)} \|\vec{x}-\vec{y}\| & \text{if } \vec{x}\in\Omega^-. \end{cases} \tag{1}$$

Coordinate vector.Coordinate vector.

Commonly the level-set method is used on regular meshes in combination with finite-volume techniques, similar to the approach of [12]. In recent works by [10] and [16] the level-set method has been applied within the framework of linear finite-elements. This approach has beneficial properties, such as the possibility to split any triangular mesh into two disjoint meshes, one for each phase, which are fitted exactly to the boundary $\Gamma(t)$, without the necessity of generating new meshes.

*Corresponding author
Email address: d.denouden-vanderhorst@tudelft.nl (D. den Ouden-van der Horst)

In this article we will focus on the level-set method on triangular primal meshes that are generated from an initial base triangulation by using hierarchical local mesh-refinement/-coarsening driven by the level-set function $\phi(\vec{x}, t)$. The hierarchy-preserving mesh-coarsening strategy was first suggested by [8] and extended to an efficient dynamic mesh adaptation procedure for mixed tri-/quad-meshes in [11].

Let $\mathcal{T}_0 = (E_0, V_0)$ denote an initial conformal triangulation of the computational domain Ω_h consisting of a set of triangular elements E_0 and vertices V_0. Each vertex $\mathbf{v} \in V_0$ and element $K \in E_0$ belongs to generation zero.

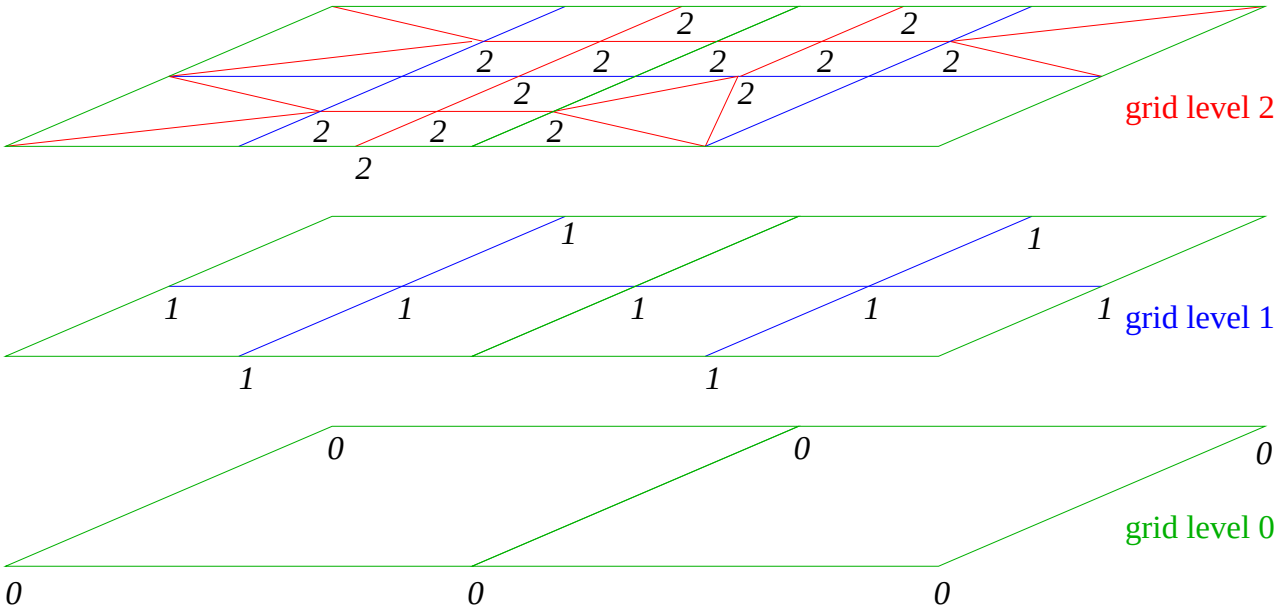
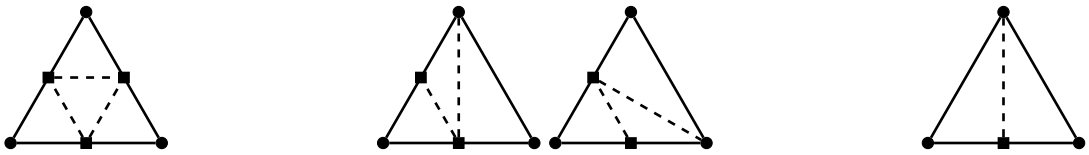


Figure 1: Hierarchical adaptive refinement of the primal mesh.



(a) Red. (b) Blue. (c) Green.

Figure 2: Definition of the local mesh refinements.

The element indicator for refinement $\zeta(K)$Element indicator for refinement. for an element $K \in \mathcal{T}$Triangulation.Linear element. and the corresponding refinement and coarsen levels ζ^+Refinement level. and ζ^-Coarsen level. are in this article based on the current age $\mathcal{G}^c(\vec{v})$Current vertex age. and the desired age $\mathcal{G}^d(\vec{v})$Desired vertex age. of a vertex $\vec{v}$Vertex of the linear mesh.. We define the current age $\mathcal{A}^c(K)$ of any element $K \in \mathcal{T}$ as the maximum age of its vertices if the element is the result of a red refinement, Figure 2(a), and one age less if the element has been refined using the green refinement, Figure 2(c), so

$$\mathcal{A}^c(K) = \begin{cases} \max\limits_{\vec{v} \in K} \mathcal{G}^c(\vec{v}) & \text{if } K \text{ is red refined,} \\ \max\limits_{\vec{v} \in K} \mathcal{G}^c(\vec{v}) - 1 & \text{otherwise.} \end{cases} \tag{2}$$

If an element K has not been refined, we define for consistency that the element is the result of a red refinement and the above definition will give $\mathcal{A}^c(K) = 0$.

Based on the level-set function $\phi(\vec{x}, t)$ we define the desired age $\mathcal{G}^d(\vec{v})$ of a vertex $\vec{v}$ by

$$\mathcal{G}^d(\vec{v}) = \begin{cases} n_{\max} & \text{if } |\phi(\vec{v}, t)| \leq 2h_{\min}, \\ n_{\min} & \text{otherwise.} \end{cases} \tag{3}$$

Here $n_{\max}$Maximum number of refinements., $n_{\min}$Minimum number of refinements. and $h_{\min}$Minimum edge length. represent, respectively, the maximum number of refinements, the minimum number of refinements

and the minimum edge length of the current triangulation $\mathcal{T}$. The desired average age $\bar{\mathcal{A}}^d(K)$Desired average element age. for an element $K \in \mathcal{T}$ is

$$\bar{\mathcal{A}}^d(K) = \frac{1}{|K|} \sum_{\vec{v} \in K} \mathcal{G}^d(\vec{v}), \tag{4}$$

and subsequently the desired age $\mathcal{A}^d(K)$Desired element age. is

$$\mathcal{A}^d(K) = \begin{cases} \bar{\mathcal{A}}^d(K) & \text{if } \max\limits_{\vec{v} \in K} \mathcal{G}^d(\vec{v}) = \bar{\mathcal{A}}^d(K), \\ \max\left(\max\limits_{\vec{v} \in K} \mathcal{G}^d(\vec{v}) - 1, n_{\min} \right) & \text{otherwise.} \end{cases} \tag{5}$$

These definitions will cause all elements located fully within a band of $2h_{\min}$ on either side of the interface $\Gamma(t)$ to have a desired age of $n_{\max}$. All elements neighbouring this narrow band will have a desired age of $\max(n_{\max} - 1, n_{\min})$, while all remaining elements will have a desired age of $n_{\min}$. This choice will force the number of final refinements in elements away from the narrow band to be as low as possible, and the elements around the narrow band to be as refined as defined.

Using these definitions, we set the element indicator for refinement $\zeta(K)$ for element K as

$$\zeta(K) = \begin{cases} 1 & \text{if } \mathcal{A}^d(K) > \mathcal{A}^c(K) \text{ and } \mathcal{A}^d(K) = n_{\max}, \\ \text{-1} & \text{if } \mathcal{A}^d(K) < \mathcal{A}^c(K) \text{ and } \mathcal{A}^d(K) = n_{\min}, \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

The refinement and coarsen levels ζ^+ and ζ^- are set at $\zeta^+ = \frac{1}{2}$ and $\zeta^- = \text{-}\frac{1}{2}$. To obtain an adequate refinement based on $\phi(\vec{x}, t)$, the local refinements are performed until no elements have been coarsened or refinement, using at each step the updated value of $\phi(\vec{x}, t)$, which can be based on the exact distance to the interface $\Gamma(t)$, if known, or some approximation of this distance.

2.2. The Volume-of-Fluid Function on the Dual Mesh

The volume-of-fluid method, as introduced by [9], defines a function $\psi(\vec{x}, t)$Volume-of-fluid function. on a two-phase domain $\Omega = \Omega^+ \cup \Omega^-$Union of Ω^+ and Ω^-. with definition

$$\psi(\vec{x}, t) = \begin{cases} 1 & \text{if } \vec{x} \in \Omega^+, \\ 0 & \text{if } \vec{x} \in \Omega^-. \end{cases} \tag{7}$$

Within the original article by [9] and all subsequent applications on the volume-of-fluid-method, such as [17], the finite-volume method is used to approximate $\psi(\vec{x}, t)$ on quasi-regular meshes with control volumes LFinite-volume control-volume. in the shape of, in example, quadrilaterals or cylindrical boxes. For each control volume L the average value of $\psi(\vec{x}, t)$ is monitored, where an average of one indicates that the control volume is occupied fully by phase Ω^+, an average of zero indicates occupation fully by phase Ω^- and an average between zero and one indicates that the control volume is cut by the interface $\Gamma(t)$.

In this article we will not focus on the evolution of the volume-of-fluid function $\psi(\vec{x}, t)$, but rather on the representation of $\psi(\vec{x}, t)$ on star-shaped polytopal control-volumes. We define a star-shaped polytopal control-volume L as a set of $N(L)$Number of boundary vertices of control volume L. boundary vertices $\vec{w}_i, i = 1, \ldots, N(L)i$-th boundary vertex of a control volume. with collocation point $\bar{\vec{w}}$Collocation point of a control volume.. We furthermore demand that any straight line-segment from the collocation point $\bar{\vec{w}}$ to any point within the control volume L does not intersect with the boundary of L. This definition of control volumes allows us to obtain a straight-forward coupling between the level-set function $\phi(\vec{x}, t)$ on linear meshes and the volume-of-fluid function $\psi(\vec{x}, t)$ on star-shaped polytopal meshes.

2.3. Coupling the Methods

Given the level-set function $\phi(\vec{x}, t)$ on a linear mesh $\mathcal{T}$, in the following referred to as the primal mesh, we define the volume-of-fluid function on a related dual mesh $\mathcal{U}$The dual mesh. In principle any dual mesh $\mathcal{U}$ based on the primal mesh $\mathcal{T}$ can be used in our method which is discussed below. We have chosen for the median dual mesh, which can be easily constructed from the primal mesh $\mathcal{T}$ by following the next steps if $\Omega \subseteq \mathbb{R}^2$:

1. For every element $K \in \mathcal{T}$:
   (a) For each edge e of K define the dual edge $\mathcal{E}_{e,K}$ as an edge between the barycentre of e and the barycentre of K.Edge of the dual mesh between mesh objects i and j of the primal mesh.
2. For every vertex $\vec{v} \in \mathcal{T}$ with connected edges $e_1, \ldots, e_n$ define $\mathcal{K}_{\vec{v}}$ as the control volume bounded by all dual edges $\mathcal{E}_{e_i,\cdot}$ connected to edge $e_i, i = 1, \ldots, n$.Control volume associated to $\vec{v}$.

If $\Omega \subseteq \mathbb{R}^3$ then perform the next steps:

1. For every element $K \in \mathcal{T}$:
   (a) For each face f of K define the dual edge $\mathcal{E}_{f,K}$ as an edge between the barycentre of K and the barycentre of f.Face of the primal mesh.
   (b) For each edge e of K define the dual edge $\mathcal{E}_{e,f}$ for each face f of K connected to e as an edge between the barycentre of e and the barycentre of f.Edge of the primal mesh.
   (c) For each edge e of K with connected faces f_1, f_2 of K define the dual face $\mathcal{F}_{e,K}$ as the quadrilateral with boundary edges $\mathcal{E}_{e,f_1}, \mathcal{E}_{e,f_2}, \mathcal{E}_{f_1,K}$ and $\mathcal{E}_{f_2,K}$.Face of the dual mesh associated to mesh objects i and j of the primal mesh.
2. For every vertex $\vec{v} \in \mathcal{T}$ with connected edges $e_1, \ldots, e_n$ define $\mathcal{K}_{\vec{v}}$ as the control volume bounded by all dual faces $\mathcal{F}_{e_i,\cdot}$ connected to edge $e_i, i = 1, \ldots, n$.

Two examples of such dual meshes can be seen in Figure 3 and Figure 4. For each control volume $\mathcal{K}_{\vec{v}}$ we set the vertex $\vec{v}$ as the collocation point $\vec{\bar{w}}$. The median dual mesh $\mathcal{U}$ is now defined as the collection of all control volumes $\mathcal{K}_{\vec{v}}$. Each of the control volumes $\mathcal{K}_{\vec{v}}$ are star-shaped polytopes as a result of their definition.

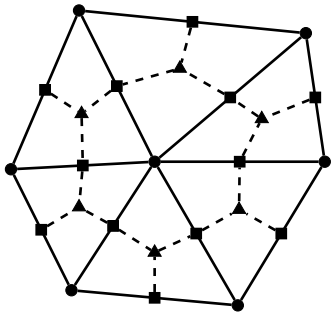


Figure 3: Definition of the dual mesh for a triangular primal mesh. Dashed lines represent edges of the dual mesh, solid lines the edges of the primal mesh.

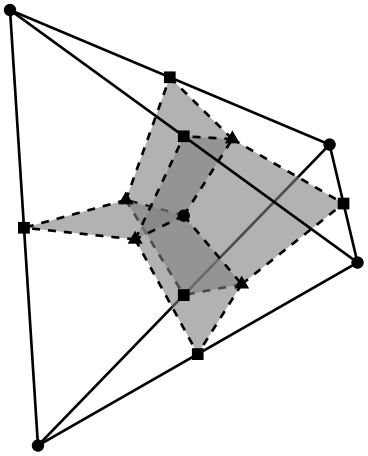


Figure 4: Definition of the dual mesh for a single tetrahedron. Dashed lines represent edges of the dual mesh, solid lines the edges of the primal mesh, grey planes the faces of the dual mesh.

We now define the volume-of-fluid function $\psi_\phi(\vec{v}, t)$Volume-of-fluid function derived from $\phi(\vec{x}, t)$. within the control volume $\mathcal{K}_{\vec{v}}$ based on the level-set function $\phi(\vec{x}, t)$ as

$$\psi_\phi(\vec{v}, t) = \frac{1}{\mathbb{A}_{\vec{v}}} \int_{\mathcal{K}_{\vec{v}}} H\left(\phi(\vec{x}, t)\right) \, \mathrm{d}\vec{x}, \tag{8}$$

with $\mathbb{A}_{\vec{v}}$ the area/volume of $\mathcal{K}_{\vec{v}}$ and $H(\cdot)$ the Heaviside function.Area/volume of control volume $\mathcal{K}_{\vec{v}}$.Heaviside function. It is clear that for a control volume where $\phi(\vec{x}, t) \geq 0$ for all $\vec{x} \in \mathcal{K}_{\vec{v}}$, or equivalently $\mathcal{K}_{\vec{v}} \in \Omega^+$, we have $\psi_\phi(\vec{v}, t) = 1$, whereas $\phi(\vec{x}, t) \leq 0$ for all $\vec{x} \in \mathcal{K}_{\vec{v}}$, or equivalently $\mathcal{K}_{\vec{v}} \in \Omega^-$, will give $\psi_\phi(\vec{v}, t) = 0$.

However, if $\phi(\vec{x}, t)$ changes sign on $\mathcal{K}_{\vec{v}}$, we have $0 < \psi_\phi(\vec{v}, t) < 1$. Obtaining this value is using the definitions we have chosen straightforward. Consider an arbitrary control volume $\mathcal{K}_{\vec{v}}$, see Figure 5 for a two-dimensional

illustration, and the zero level of $\phi(\vec{x},t)$ on this control volume. Within each primal element connected to $\vec{v}$ the level set function is assumed linear, so within $\mathcal{K}_{\vec{v}}$ the zero level of $\phi(\vec{x},t)$ consists of piecewise linear segments, each located in a part of the generating elements of the primal mesh. Using simple geometric tools, one can compute the corresponding area/volume of the region occupied by Ω^+ within each generating element and subsequently add those values to obtain $\psi(\vec{v},t)$.

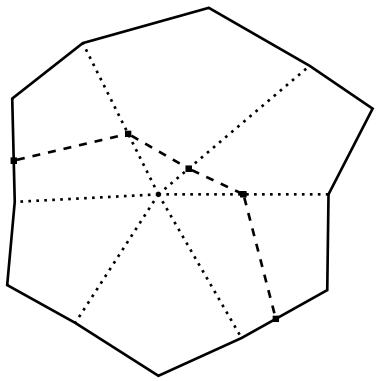


Figure 5: Illustration of a control volume $\mathcal{K}_{\vec{v}}$ cut by the zero contour of $\phi(\vec{x},t)$. Solid lines are edges of $\mathcal{K}_{\vec{v}}$, dotted lines are partial edges of the primal mesh and dashed lines represent $\phi(\vec{x},t)=0$ on $\mathcal{K}_{\vec{v}}$.

3. Volume Correction

During the (discrete) advection of both the level-set function $\phi(\vec{x},t)$ and the volume-of-fluid function $\psi(\vec{x},t)$, discrepancies between these functions may be introduced, due to which the volume-of-fluid function $\psi_\phi(\vec{x},t)$ (Equation 8) based on $\phi(\vec{x},t)$ and the given volume-of-fluid function $\psi(\vec{x},t)$ may not be equal. These discrepancies are commonly the result of volume loss during the advection or the reinitialisation of the level-set function $\phi(\vec{x},t)$, whereas the volume-of-fluid function $\psi(\vec{x},t)$ commonly preserves volume between two consecutive time steps or can be obtained exact. In this section we discuss the approach taken to resolve the differences between the two volume-of-fluid functions $\psi_\phi(\vec{x},t)$ and $\psi(\vec{x},t)$, the numerical solution of the resulting equations and how this approach can be applied in two different ways.

3.1. The Approach

We start by making the assumption that for both the level-set function $\phi(\vec{x},t)$ and the volume-of-fluid function $\psi(\vec{x},t)$ we have their discrete counterparts $\phi_h(\vec{x},t)$ and $\psi_h(\vec{x},t)$ at some time t, respectively, on the primal mesh $\mathcal{T}$ and the dual mesh $\mathcal{U}$.Discrete counterpart of $\phi(vecx,t)$.Discrete counterpart of $\psi(vecx,t)$. We further assume that the volume-of-fluid function conserves volume exactly.

Using $\phi_h(\vec{x},t)$, we can construct the discrete volume-of-fluid function $\psi_{\phi,h}(\vec{x},t)$ on the dual mesh $\mathcal{U}$ by Equation 8.Discrete volume-of-fluid function derived from $\phi_h(\vec{x},t)$. If this volume-of-fluid function $\psi_{\phi,h}(\vec{x},t)$ does not equal the exact volume-of-fluid function $\psi_h(\vec{x},t)$, we seek an update $\tilde{\phi}_h(\vec{x},t)$ of $\phi_h(\vec{x},t)$ such that

$$\left| \frac{1}{\mathbb{A}_{\vec{v}}} \int_{\mathcal{K}_{\vec{v}}} H\left(\tilde{\phi}_h(\vec{x},t)\right) \,\mathrm{d}\vec{x} - \psi_h(\vec{v},t) \right| < \epsilon, \tag{9}$$

Discrete update of $\phi_h(\vec{x},t)$. for each vertex $\vec{v}$ of the primal mesh $\mathcal{T}$, where $\tilde{\phi}_h(\vec{x},t)$ is assumed to be linear on the primal mesh $\mathcal{T}$, similar to $\phi_h(\vec{x},t)$, and where $\epsilon > 0$ is a fixed tolerance.

Finding the update $\tilde{\phi}_h(\vec{x},t)$ is done similar to the procedure in [17] by first computing an update $\Delta\phi_{\vec{v}}$ per control volume $\mathcal{K}_{\vec{v}}$ such that

$$\left| \frac{1}{\mathbb{A}_{\vec{v}}} \int_{\mathcal{K}_{\vec{v}}} H\left(\phi_h(\vec{x},t) + \Delta\phi_{\vec{v}}\right) \,\mathrm{d}\vec{x} - \psi_h(\vec{v},t) \right| < \epsilon. \tag{10}$$

Tolerance for volume conservation.Local discrete update of $\phi_h(\vec{x},t)$.

After a solution has been found of Equation 10, we set the nodal values of $\tilde{\phi}_h(\vec{x},t)$ in each vertex $\vec{v}$ of the primal mesh to

$$\tilde{\phi}_h(\vec{v},t) = \phi_h(\vec{v},t) + \Delta\phi_{\vec{v}}. \tag{11}$$

After validating Equation 9 and setting $\phi_h(\vec{x},t) = \tilde{\phi}_h(\vec{x},t)$, we repeat this procedure until Equation 9 is true.

3.2. Choice of Iterative Methods

The procedure described above which allows the recovery of a volume-conserving level-set function gives a solution to the vector-valued fixed-point problem

$$\vec{\phi}_h = \vec{g}(\vec{\phi}_h) \tag{12}$$

where the function $\vec{g}(\vec{\phi}_h)$ is defined as

$$\vec{g}(\vec{\phi}_h) = \vec{\phi}_h + \Delta\vec{\phi}_{\vec{v}}(\vec{\phi}_h), \tag{13}$$

and $\vec{\phi}_h$ and $\Delta\vec{\phi}_{\vec{v}}$ are, respectively, the list of nodal values of $\phi_h(\vec{x}, t)$ and the list of vertex values $\Delta\phi_{\vec{v}}$.

For obtaining the solution of the fixed-point problem in Equation 12 several methods exist, amongst the most common are the Picard method and the Newton-Raphson method. The latter however requires the transformation of Equation 12 into the vector-valued root-finding problem

$$\vec{f}(\vec{\phi}_h) = \vec{0}, \tag{14}$$

where the function $\vec{f}(\vec{\phi}_h)$ is defined as

$$\vec{f}(\vec{\phi}_h) = \vec{\phi}_h - \vec{g}(\vec{\phi}_h) = -\Delta\vec{\phi}_{\vec{v}}(\vec{\phi}_h). \tag{15}$$

A common drawback of the Picard method for Equation 12 is the theoretical first order convergence of the iterates, whereas the Newton-Raphson method for Equation 14 exhibits theoretical second order convergence. On the other hand, the Picard method requires no information on the Jacobian of the function, whereas the Newton-Raphson method does. For both methods several techniques exist that resolve the problems mentioned above partially and will be discussed below.

By [1] the currently-called Anderson acceleration for fixed-point problems was introduced, which improves the order of convergence for the Picard method. This technique has since been extended by [6] and a good review of this method and the improvements can be found in [20]. For Equation 12, we choose some value $m \geq 0$ which represents the maximum extra number of previous iterates that are used in obtaining a new iterate for the solution of Equation 12. For a given iteration number $k \geq 0$, define $m_k = \min(m, k)$ and construct the matrices F_k and $\mathcal{F}_k$ as, respectively,

$$F_k = \left[\vec{f}(\vec{\phi}_h^{(k-m_k+1)}), \vec{f}(\vec{\phi}_h^{(k-m_k+2)}), \cdots, \vec{f}(\vec{\phi}_h^{(k-1)}), \vec{f}(\vec{\phi}_h^{(k)})\right], \tag{16}$$

and

$$\mathcal{F}_k = F_k - F_{k-1}. \tag{17}$$

These matrices require the $m_k + 1$ iterates $\vec{\phi}_h^{(i)}, i = k - m_k, \ldots, k$ and their function values. Next we find the solution $\vec{\gamma}^{(k)} = [\gamma_0^{(k)}, \gamma_1^{(k)}, \cdots, \gamma_{m_k-2}^{(k)}, \gamma_{m_k-1}^{(k)}]^T$ of the minimisation problem

$$\min_{\vec{\gamma}=[\gamma_0,\gamma_1,\cdots,\gamma_{m_k-2},\gamma_{m_k-1}]^T} \left\|\vec{f}(\vec{\phi}_h^{(k)}) - \mathcal{F}_k\vec{\gamma}\right\|_2. \tag{18}$$

and set the next approximation of the solution to Equation 12 to

$$\vec{\phi}_h^{(k+1)} = \vec{g}(\vec{\phi}_h^{(k)}) - \sum_{i=0}^{m_k-1} \gamma_i^{(k)} \left(\vec{g}(\vec{\phi}_h^{(k-m_k+i+1)}) - \vec{g}(\vec{\phi}_h^{(k-m_k+i)})\right). \tag{19}$$

For short-hand we introduce the notation m-Picard for the Picard method accelerated with the use of m previous iterates. If $m = 0$ Equation 19 reduces to the regular non-accelerated Picard method.

[3] introduced the currently-called Broyden method for solving vector-valued root-finding problems for which the Jacobian is difficult to obtain. The regular Newton-Raphson iteration

$$\vec{\phi}_h^{(k+1)} = \vec{\phi}_h^{(k+1)} - J^{-1}(\vec{\phi}_h^{(k)})\vec{f}(\vec{\phi}_h^{(k)}), \tag{20}$$

is replaced by

$$\vec{\phi}_h^{(k+1)} = \vec{\phi}_h^{(k+1)} - B_{(k)}^{-1}\vec{f}(\vec{\phi}_h^{(k)}), \tag{21}$$

with $B_{(k)}$ an approximation of the Jacobian $J(\vec{\phi}_h^{(k)})$ or

$$\vec{\phi}_h^{(k+1)} = \vec{\phi}_h^{(k+1)} + H_{(k)}\vec{f}(\vec{\phi}_h^{(k)}), \tag{22}$$

with $H_{(k)}$ an approximation of the negative inverse Jacobian $J^{-1}(\vec{\phi}_h^{(k)})$. Both $B_{(k)}$ and $H_{(k)}$ are obtained with a rank-1 updates given by

$$B_{(k+1)} = B_{(k)} + \frac{1}{\vec{p}_{(k)}^T \vec{p}_{(k)}} \vec{f}(\vec{\phi}_h^{(k+1)}) \vec{p}_{(k)}^T, \tag{23}$$

$$H_{(k+1)} = H_{(k)} - \frac{1}{\vec{p}_{(k)}^T H_{(k)} \vec{y}_{(k)}} H_{(k)} \vec{f}(\vec{\phi}_h^{(k+1)}) \vec{p}_{(k)}^T H_{(k)}, \tag{24}$$

where

$$\vec{p}_{(k)} = \vec{\phi}_h^{(k+1)} - \vec{\phi}_h^{(k)}, \qquad \vec{y}_{(k)} = \vec{f}(\vec{\phi}_h^{(k+1)}) - \vec{f}(\vec{\phi}_h^{(k)}). \tag{25}$$

A common choice for $B_{(0)}$ or $H_{(0)}$ is the identity matrix. The rank-1 updates used to update the (inverse) approximate Jacobian makes no use of possible knowledge on the Jacobian, such as fixed values or a sparsity pattern. In our method, the Jacobian has the same sparsity pattern as the consistent finite-element mass matrix associated to the primal mesh $\mathcal{T}$. In [14] an update for $B_{(k+1)}$ is introduced which exploits known information on the structure of the Jacobian by only applying a rank-1 update to the unknown coefficients within the approximate Jacobian. Although this method requires more effort for each update of the approximate Jacobian, the number of unknowns that need to be stored is lower and a faster convergence to the exact Jacobian is expected. If $\vec{b}_{(k)}^i$ is the i-th row of $B_{(k)}$, this row is updated according to

$$\vec{b}_{(k+1)}^i = \vec{b}_{(k+1)}^i + \frac{1}{\hat{\vec{p}}_{(i,k)}^T \hat{\vec{p}}_{(i,k)}} \vec{f}(\vec{\phi}_h^{(k+1)}) \hat{\vec{p}}_{(i,k)}^T, \tag{26}$$

where $\hat{\vec{p}}_{(i,k)}$ is the vector $\vec{p}_{(k)}$ set to zero in those position where the exact Jacobian in row i has zeros.

In the reminder we will refer to Equation 21 as the aj-Broyden method[1], to Equation 22 as the aij-Broyden method[2] and to the sparse updated Broyden method with Equation 26 as the asj-Broyden method[3]. The iterations for all methods discussed are stopped when Equation 9 is valid with $\tilde{\phi}_h(\vec{x}, t)$ based on the nodal values $\vec{\phi}_h^{(k+1)}$.

To obtain the new iterate $\vec{\phi}_h^{(k+1)}$ with the Picard, m-Picard or any of the Broyden methods, each function evaluation requires the solution of Equation 10 for each vertex $\vec{v}$ of the primal mesh. Due to the non-linearity of Equation 10, we will use the secant method in finding this solution.

3.3. Advection Problems

In the above discussion it was assumed that a level-set function $\phi_h(\vec{x}, t)$ and a volume-of-fluid function $\psi_h(\vec{x}, t)$ are known, respectively, on a given primal mesh $\mathcal{T}$ and the dual mesh $\mathcal{U}$. The above approach is further independent of the choice for advection of the level-set function and the volume-of-fluid function or the choice of reinitialisation of the level-set function after advection. A benefit combining the level-set method with the volume-of-fluid method is that no costly interface reconstruction based on the volume-of-fluid function has to be performed.

Given any choice for advection and reinitialisation, denote by $\vec{\phi}_h^n$ and $\vec{\psi}_h^n$, respectively, the nodal values of $\phi_h(\vec{x}, t^n)$ and $\psi_h(\vec{x}, t^n)$ at some discrete time t^n. We propose Algorithm 1 as a basis for advection problems, assuming the volume-of-fluid function is advected conservatively.

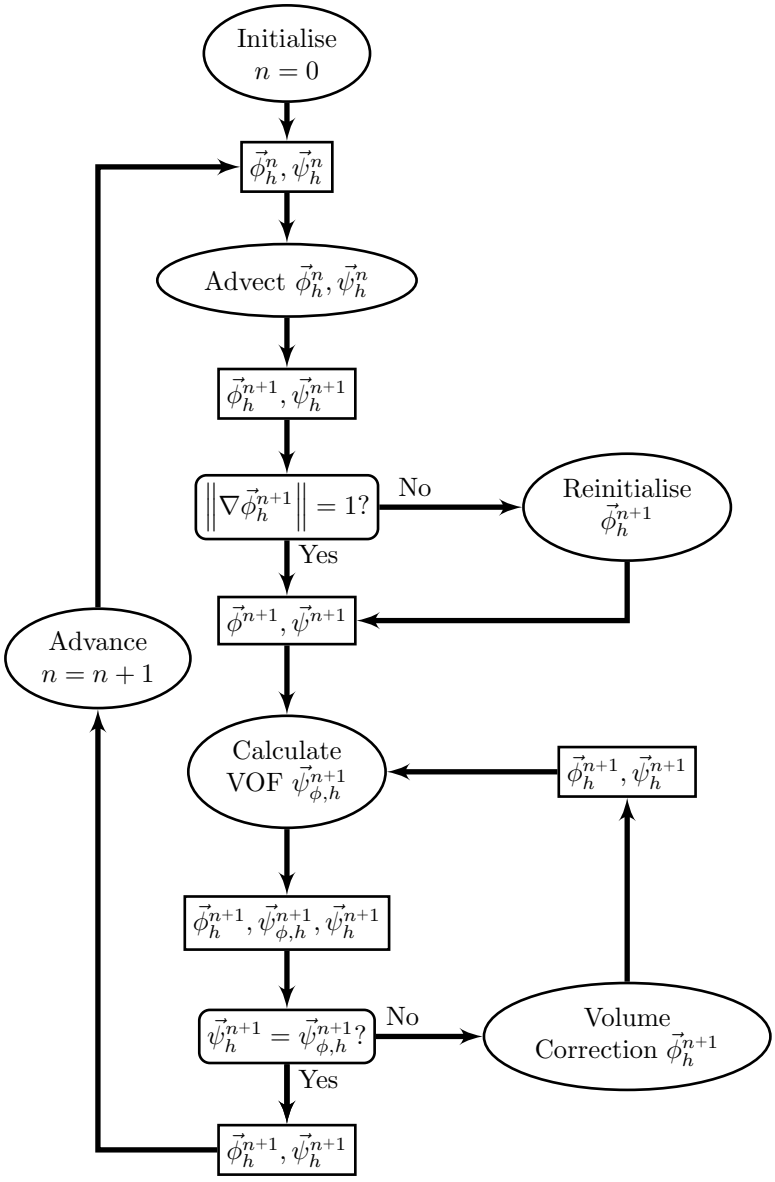
3.4. Initialisation of the Level-Set Function

Within the use of the level-set method the initial condition for the level-set function, $\phi(\vec{x}, 0)$, on a primal mesh $\mathcal{T}$ is only known for simple geometric interfaces $\Gamma(0)$ such as circles and rectangles in $\mathbb{R}^2$ and spheres and boxes in $\mathbb{R}^3$. If the geometry of the interface can be described by the union or intersection of such geometric objects, the exact initial condition can also be found with relative ease. If however the interface becomes more complex, commonly no exact value for the level-set function is known, but only an implicit function $F(\vec{x})$ of which the zero-contour captures the initial interface.

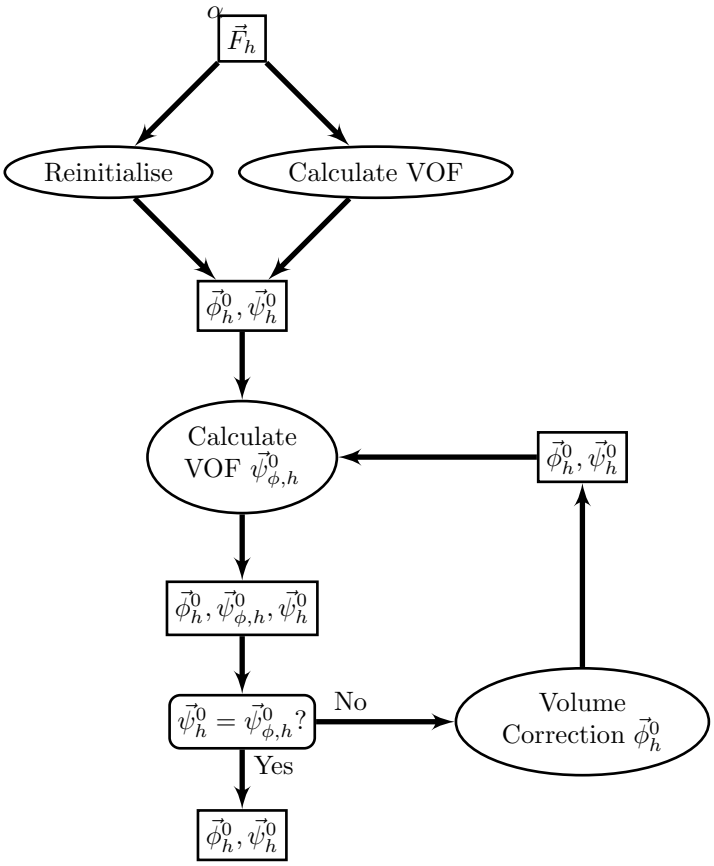
The discrete initial value for the level-set function, $\phi_h(\vec{x}, 0)$, on a primal mesh $\mathcal{T}$ can be obtained by applying any reinitialisation technique to the discrete analogue $F_h(\vec{x})$ of the implicit function $F(\vec{x})$. This however does not guarantee that the volume enclosed by the zero-contour of the level-set function $\phi_h(\vec{x}, 0)$ corresponds to the volume defined by the zero-contour of the generating function $F_h(\vec{x})$. The volume correction technique discussed in Section 3.1 can be used to remove this mismatch, for which the algorithm can be found in Algorithm 2. We note that Algorithm 2 can be used even when one is not interested in tracking the volume-of-fluid function, but merely in obtaining a good initial condition for the level-set function.

[1] aj: approximate Jacobian.
[2] aij: approximate inverse Jacobian.
[3] asj: approximate sparse Jacobian.

Algorithm 1 Structure of the algorithm for advection problems.



Algorithm 2 Structure of the algorithm for level-set initialisation.



4. Results

In this section we will first investigate the performance of the discussed iterative methods to obtain a volume-conserving level-set function from a volume-of-fluid function. Thereafter we consider the experimental order of convergence and finally the use of the volume-correction to obtain an initial condition for the level-set function.

4.1. Iterative Methods

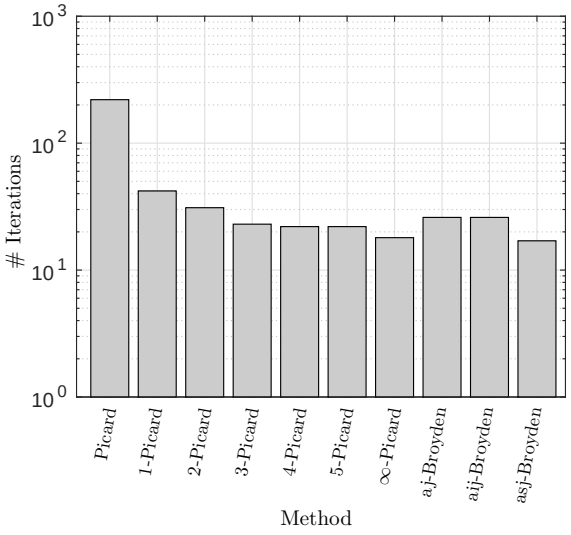
In Section 3.2 we have discussed the Picard method, the accelerated m-Picard methods and three Broyden methods. We have applied several of these methods to a volume-correction problem where the discrete volume-of-fluid function ψ_h is based on the piecewise-linear approximation of the exact level-set function

$$\phi(\vec{x}) = 0.5479 - \|\vec{x}\|_2 \,, \tag{27}$$

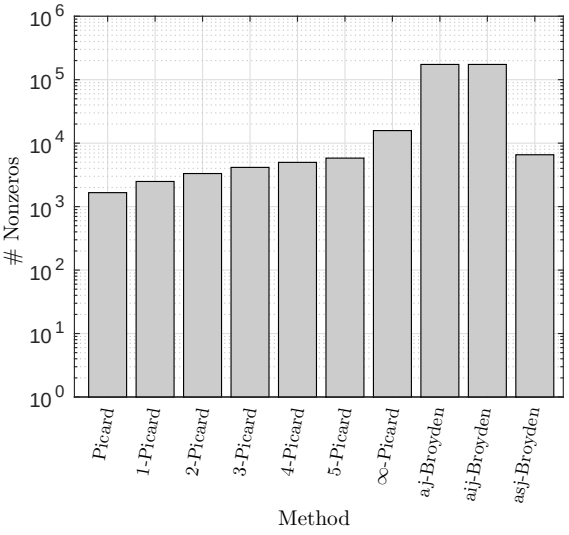
which represents a circle with radius 0.5479 centred around $(0, 0)$ within the domain $\Omega = [\text{-}1, 1]^2$. The discrete level-set function ϕ_h is obtained, for testing purposes, by the reinitialisation technique proposed within [13]. This although this reinitialisation technique gives exact values for the signed-distance to the piecewise-linear interface based on Equation 27, volume is not conserved as the reinitialisation is only performed in vertices of the primal mesh, which therefore causes a shift within the interface with respect to the discrete zero-contour of Equation 27. The primal mesh $\mathcal{T}$ for Ω is created by dividing Ω in two triangles over the south-west-to-north-east diagonal and then applying local mesh refinement with $n_{\min} = 0$ and $n_{\max} = 5$.

The methods we have used are the Picard method, the m-Picard method with $m \in \{1, 2, 3, 4, 5, \infty\}$ and all three Broyden methods. We have included the ∞-Picard method to see the performance and effect of acceleration with the use of as many iterates as possible. With each of the methods we have performed 100 simulations to circumvent temporary effects within the results. For each of the simulations we have measured the number of iterations, the maximum number of non-zeros needed for storage and the wall-clock time. Figure 6 shows the result obtained with the mentioned methods. Figure 6(a) and Figure 6(b) show, respectively, total number of iterations and the maximum number of non-zeros per method, which is the same for each simulation with each method. Figure 6(c) and Figure 6(d) show, respectively, the average wall clock time per iteration and the average total wall clock time per method, both scaled with respect to the Picard method for comparison.

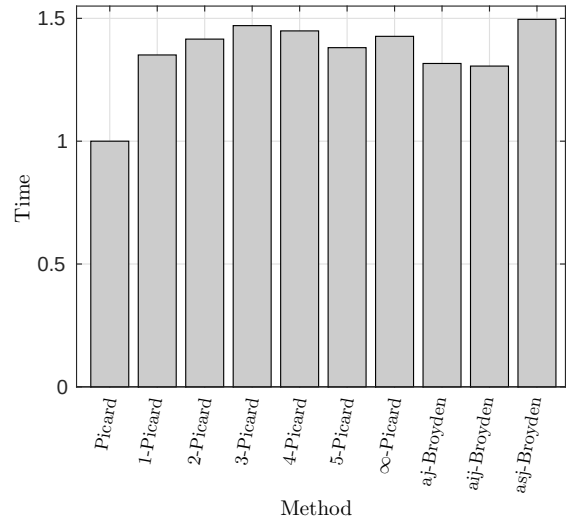
With reference to Figure 6(a), we see that the Picard method needs the largest number of iterations to achieve volume-conservation, which can be expected from the theoretical first-order convergence. All other



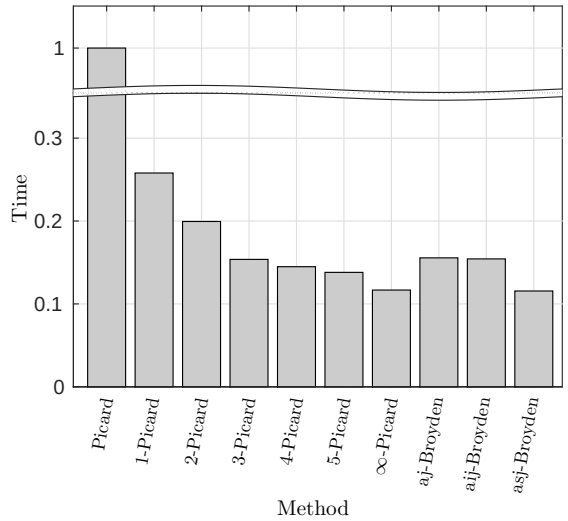
(a) Number of iterations.



(b) Number of non-zeros.



(c) Average time per iteration, relative to the Picard method.



(d) Total time per method, relative to the Picard method.

Figure 6: Comparison of test results for several iterative methods.

methods need significantly less iterations. For the m-Picard methods this is the desirable result and agrees with the discussion in [20] that the Anderson acceleration technique results in a quasi-Newton method, similar to the Broyden method, where at each iteration the updated Jacobian is based on the identity matrix, and thus a higher than first-order convergence is expected. Similarly is the Picard method out-performed by the three Broyden methods with respect to the number of iterations. The m-Picard methods decrease in the number of iterations for each time m is increase, however changing m from zero to one gives the largest direct decrease, where all other increments of m only lower the number of iterations slightly. Finally, we see that of the three Broyden methods, the aj-Broyden and the aij-Broyden method perform equal as both methods are equal by construction with the use of the Sherman-Morrison-Woodbury formula [2]. The asj-Broyden method performs slightly better in comparison to the other two Broyden methods, which agrees with the expectation that approximate Jacobian converges faster to the exact Jacobian due to the explicit use of the sparsity pattern.

With reference to Figure 6(b), we see that the maximum number of non-zeros within the Picard and m-Picard methods steadily increase with the each increment of m, agreeing with the larger amount of iterates stored at each each iteration for larger m. The aj-Broyden en aij-Broyden method again are equal with respect to the maximum number of non-zeros, agreeing with theory, and use significantly more non-zeros compared to any of the other methods. The asj-Broyden method uses, however, a drastically lower maximum number of non-zeros compared to the other two Broyden methods, and uses slightly more non-zeros compared to the 5-Picard method. For this test case the locally refined mesh has on average 5.8455 neighbours per vertex, resulting in at most 6.8455 per row of approximate Jacobian on average. This indicates that the amount of non-zeros within the asj-Broyden method is in worst case comparable to the 7-Picard method, which is supported by the experimental data.

With reference to Figures 6(c) and 6(d), it can clearly be seen that any of the methods is per iteration outperformed by the Picard method due to the amount of extra work needed in obtaining a new iterate. The large decrease in the number of iterations compared to the Picard method, however, decreases the total time significantly, ranging from 75 to 88 percent.Again each increment of m reduces the total time of the class of Picard methods, of which the ∞-Picard method performs best. From the Broyden methods the asj-Broyden method uses the least amount of time, irrespective of the larger amount of works needed per iteration in solving the sparse matrix-vector equation and the sparse Broyden-update Equation 26. The asj-Broyden method outperforms all other methods in total time, although the ∞-Picard method is a close second.

If we compare all methods on all data available, the asj-Broyden method stands as being overall the better performing method. As of this we will use the asj-Broyden method in the remainder of this article.

4.2. Experimental Order-of-Convergence

For any test case, our method for volume correction should recover the same volume as the volume-conserving initial volume-of-fluid function, but should also preserve or recover the signed-distance function property

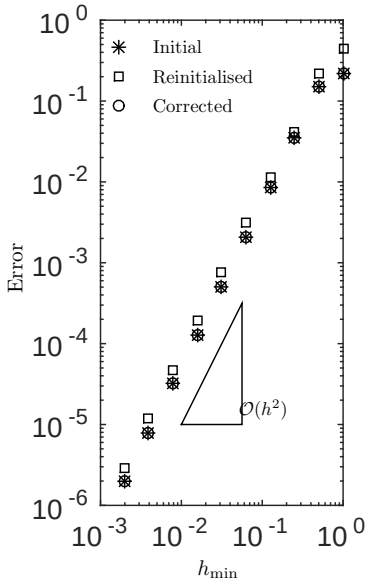
$$\|\nabla\phi\| = 1, \tag{28}$$

wherever the gradient of the level-set function is defined. To investigate this behaviour, we performed simulation for the same test case as in the previous section with locally refined meshes given $n_{\min} = 0$ and $n_{\max}$ ranging from 0 to 9, resulting in a minimum edge length $h_{\min}$ from 1 down to 2^{-9}, starting from the same base mesh as in the previous section. The obtained results can be seen in Figure 7.

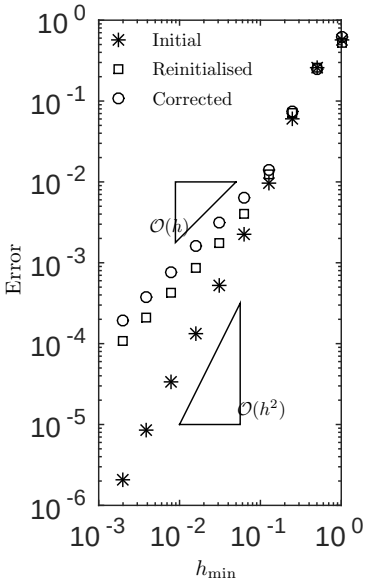
Figure 7(a) shows the absolute error in the volume before and after reinitialisation and after volume correction. It can clearly be seen that the although the reinitialisation causes volume loss, the second order convergence of the initial volume error is maintained. The volume correction procedure recovers the same volume as the initial level-set function and so maintains the same second order convergence.

Figures 7(b) and Figure 7(c) show the absolute error in the discrete 2-norm of the signed-distance property, respectively, in nodes next to the interface and in all nodes. The gradients are obtained using the super-convergent gradient-recovery from [21]. As the initial level-set function is exact, the gradients recovered will be of second order accuracy in regions where the gradient is defined and the mesh is regular and of lower order in regions where the mesh is non-regular, near the boundary of the domain and/or near the skeleton of the interface. This behaviour is observed exactly, as Figure 7(b) shows second order convergence, whereas Figure 7(c) shows initially first order convergence which drops to a convergence of $\mathcal{O}(h^{1/2})$ due to boundary nodes and the skeleton of the interface. For the reinitialised level-set function and the volume-corrected level-set function is a first order convergence achieved, whereas over the whole domain the same order of convergence as for the initial level-set function is recovered. It can also be seen that the volume correction only slightly increases the error in the signed-distance property near the interface, compared to the error for reinitialised level-set function.

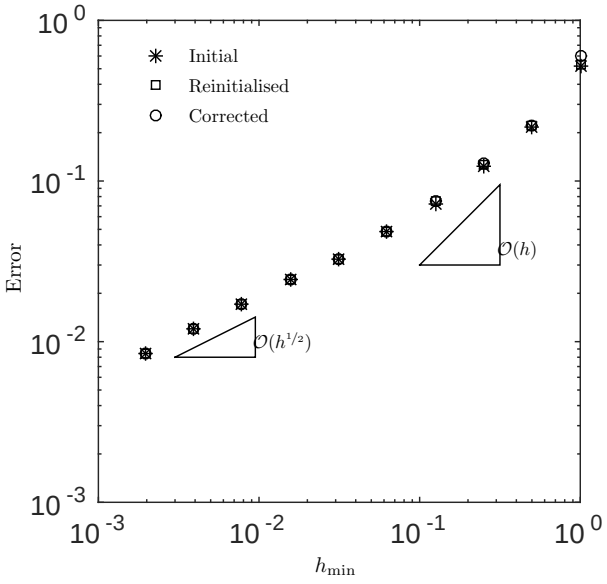
The results in Figure 7 indicate that the volume-correction is able to recover the volume correctly and will not destroy the signed-distance property after reinitialisation has been applied. To investigate this further, we will turn our attention to the problem of finding an initial level-set function for which no exact value is known.



(a) Absolute error in volume before and after reinitialisation and after volume correction.



(b) Absolute error in gradient-length before and after reinitialisation and after volume correction. Error near the interface.



(c) Absolute error in gradient-length before and after reinitialisation and after volume correction. Error over the whole domain.

Figure 7: Convergence study for circular test case with radius 0.5479 and center $(0, 0)$.

4.3. Use of the method for initial conditions

As mentioned, we can use the volume correction in combination with any reinitialisation method to obtain an initial level-set function from any arbitrary implicit function defining the initial interface. As an example, consider the interface $\Gamma(0)$ given by

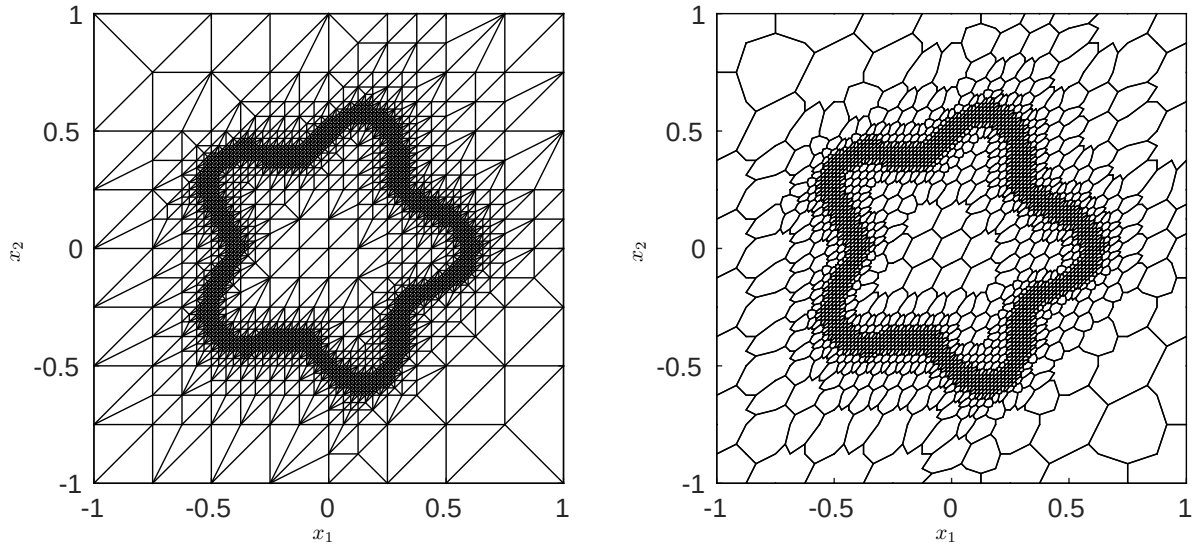
$$\vec{x}(\theta) = (R_0 + R_1 \cos(\nu\theta)) \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix} + \begin{bmatrix} x_1^c \\ x_2^c \end{bmatrix}, \tag{29}$$

for $\theta \in [0, 2\pi]$, which represents a circle with radius R_0 and centre $\vec{x}^c = (x_1^c, x_2^c)$ disturbed with by $R_1 \cos(\nu\theta)$. This interface will form a ν-pointed star with rounded corners. Using the technique described by [18], the implicit function defining $\Gamma(0)$ is given by

$$F(\vec{x}) = (R_0 + R_1 \cos(\nu \arg(\vec{x} - \vec{x}^c)))^2 - \|\vec{x} - \vec{x}^c\|^2, \tag{30}$$

with $\arg(\vec{y})$ the angle between $\vec{y}$ and $[1, 0]^T$. For this example, we take $R_0 = 1/2$, $R_1 = 1/10$, $\nu = 5$ and $\vec{x}^c = \vec{0}$.

Examples of the primal and dual meshes for a maximum local refinement level of 7 can be found in Figure 8. These refinements are obtained by using the at each iteration the signed-distance to the discreet boundary on the current mesh, without volume correction being applied. Both the primal and dual mesh have a high density of points near the implicit interface, whereas further away the density decreases. After refinement, we apply volume correction to the reinitialised level-set function for the same refinement levels as in Section 4.2, resulting in the convergence results in Figure 9.



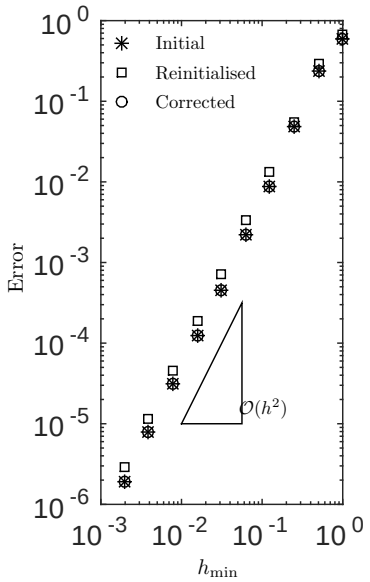
(a) Primal mesh.

(b) Dual Mesh.

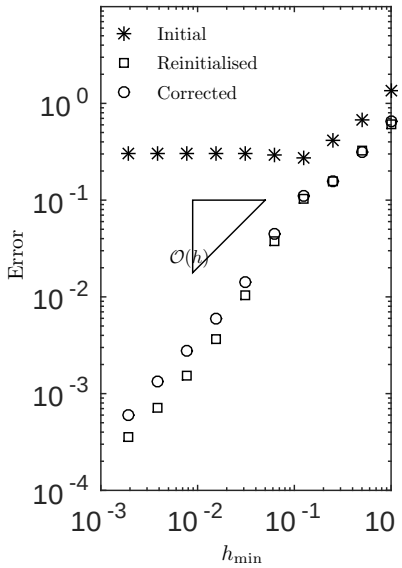
Figure 8: An example of the locally refined primal and dual mesh.

Similar to the results in Section 4.2, we obtain second-order convergence in the mesh-size for the volume and near-exact volume-conservation between the initial interface based on Equation 30 and the final volume-corrected level-set function, as can be seen in Figure 9(a). For the gradient-length we obtain, as expected, no convergence for the initial implicit function, but both the reinitialised and volume-corrected show first-order convergence near the interface and $\mathcal{O}(h^{1/2})$ convergence over the entire domain, all similar to the results in Figure 7.

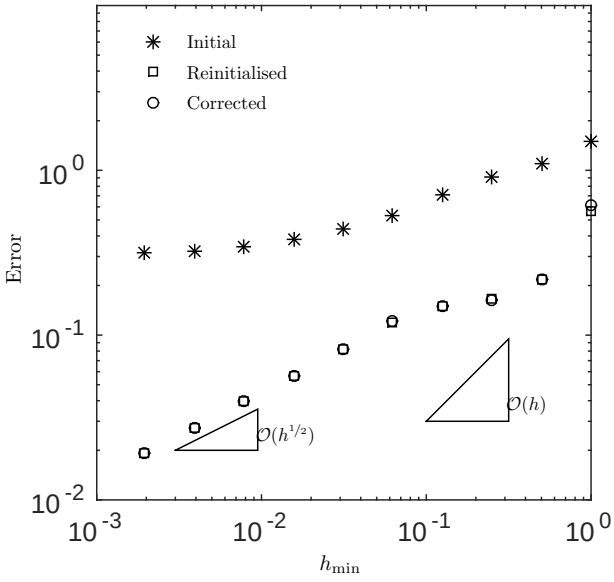
To show the results of the combined reinitialisation and volume correction, we have computed for the meshes in Figure 8 and for a globally refined mesh with the same level of refinement the contours of the initial function, the reinitialised level-set function and the volume-corrected level-set function. These contours can be seen in Figures 10 and 11. Subsequent contours differ by a value of $1/20$. The locally refined mesh captures the initial contours correctly in Figure 10(b) near the interface, but further away from the interface some numerical artefacts can be seen, compared to Figure 10(a). The globally refined mesh performs well over almost the entire domain, except near the origin, confirm Figure 10(c). The contours of the reinitialised and volume-corrected level-set functions, see Figure 11, do not differ much for the locally as well as the globally refined mesh. It can also be seen that the contours of the level-set functions near the interface are smooth and are similar in both meshes. It is also apparent that the distance between subsequent contours is indeed near $1/20$, indicating that the signed-distance property is recovered correctly for this test case.



(a) Absolute error in volume before and after reinitialisation and after volume correction.

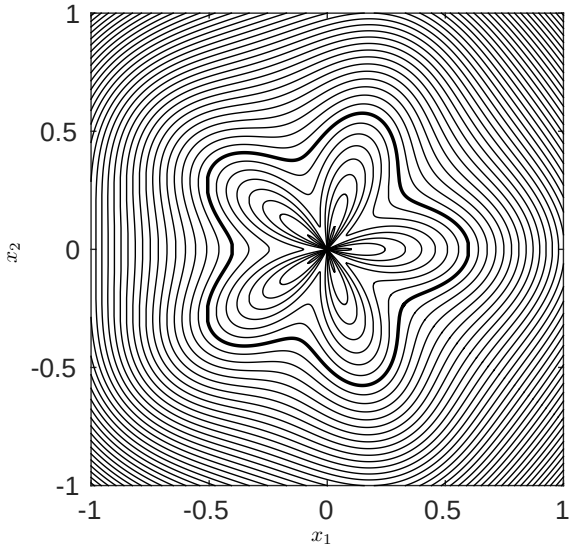


(b) Absolute error in gradient-length before and after reinitialisation and after volume correction. Error near the interface.

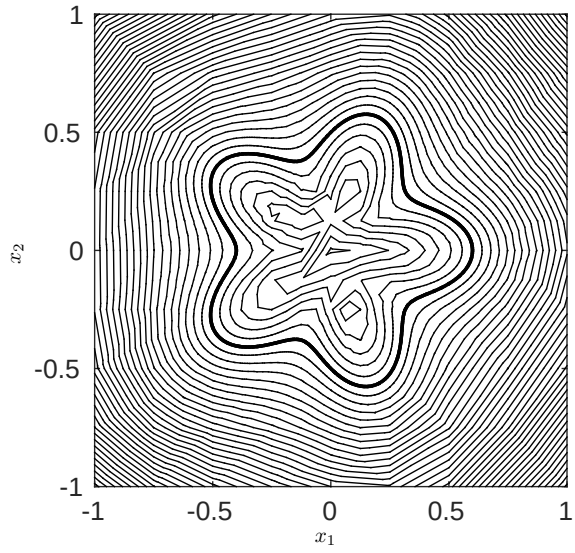


(c) Absolute error in gradient-length before and after reinitialisation and after volume correction. Error over the whole domain.

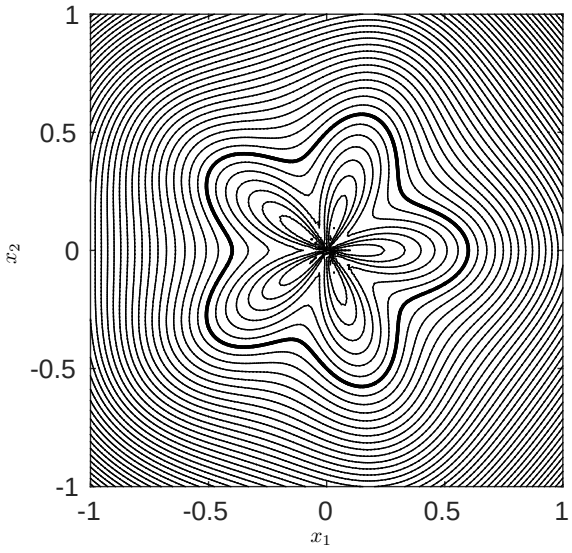
Figure 9: Convergence study for star test case.



(a) Exact contours.

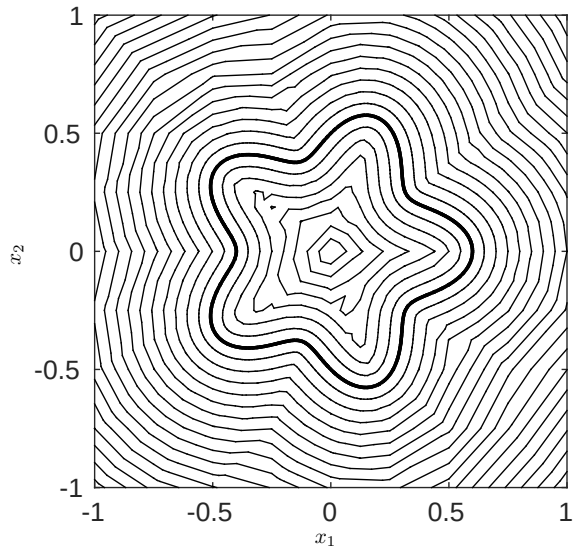


(b) Local refinement.

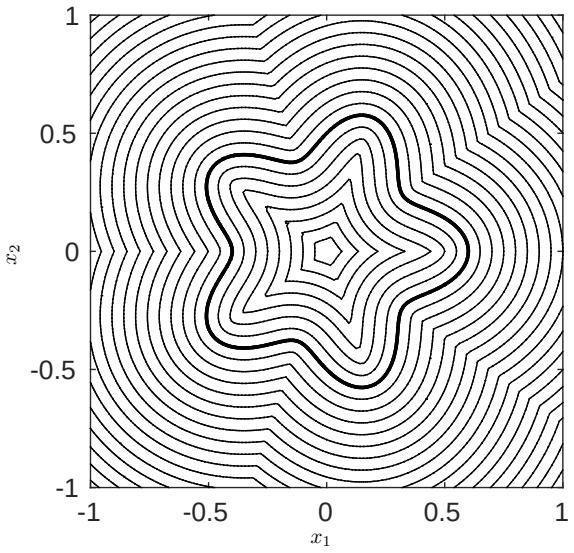


(c) Global refinement.

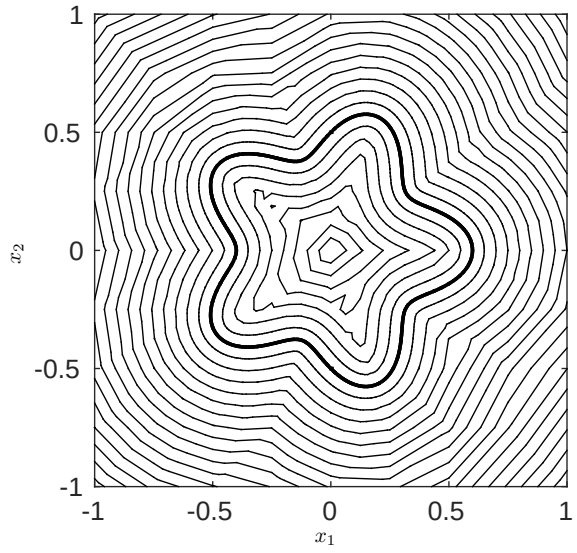
Figure 10: Initial contours.



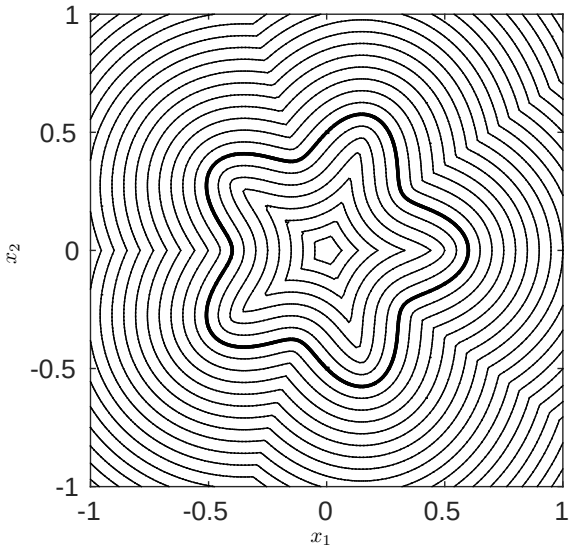
(a) Contours after (re)initialisation, local refinement.



(b) Contours after (re)initialisation, global refinement.



(c) Contours after correction, local refinement.



(d) Contours after correction, global refinement.

Figure 11: Contours after (re)initialisation and correction.

5. Discussion and Conclusions

We have introduced a coupling between the level-set method on arbitrary linear finite-element meshes and the volume-of-fluid method on star-shaped finite-volume meshes, where the star-shaped finite-volume mesh is the dual of the linear finite-element mesh. This coupling allowed us to find a volume-conserving level-set function given a non-conserving level-set function and a volume-conserving volume-of-fluid function by performing global updates of the level-set function using an iterative method. The global updates are obtained by performing local volume-conservation and mapping these to the global update variable.

Several choices for the iterative method to obtain global volume-conservation are available. We have analysed the Picard method without and with Anderson acceleration and the Broyden method without and with sparse updates of the approximate Jacobian. Each of these methods show convergence and the ability to attain volume-conservation, but the sparse updated Broyden method performed overall the best.

The volume-conservation technique has been applied to two test cases on locally refined meshes for several minimum edge lengths to investigate the behaviour. We have seen that volume-conservation is achieved with the same accuracy and order of convergence as the initial level-set function. We have also shown that the volume-conservation technique does not destroy the signed-distance property of a reinitialised level-set function.

We have furthermore shown for both locally as well as globally refined meshes our volume-conservation technique can be used to obtain an initial level-set function from a known implicit function describing the initial interface which does not fulfil the signed-distance property.

We have not applied the volume-conservation technique to the advection of level-set functions and coupled volume-of-fluid functions, but as our technique only assumes that a coupling between the volume-of-fluid function and the level-set function exists and not in what manner the values of these function have been obtained. We do however recognise that the advection of the volume-of-fluid function on star-shaped meshes is crucial for the applicability of our volume-conservation technique, as the accuracy of the volume-of-fluid method is transferred to the accuracy of the volume-conservation. In a future study we will investigate the advection of the coupled level-set function and volume-of-fluid function in combination with volume-conservation.

References

[1] D.G. Anderson. Iterative procedures for nonlinear integral equations. *Journal of the ACM*, 12(4):547–560, 1965.

[2] M. S. Bartlett. An Inverse Matrix Adjustment Arising in Discriminant Analysis. *The Annals of Mathematical Statistics*, 22(1):107–111, 1951.

[3] C. Broyden. A Class of Methods for Solving Nonlinear Simultaneous Equations. *Mathematics of Computation*, 19:577–593, 1965.

[4] S. Chen, D.B. Johnson, and P.E. Raad. The Surface Marker Method. In L.C. Wrobel and C.A. Brebbia, editors, *Moving Boundaries: Fluid Flow 1st: Computational Modelling of Free and Moving Boundary Problems*. WIT Press, Southampton, United Kingdom of Great Britain and Northern Ireland, 1991.

[5] V. Dyadechko and M. Shashkov. Moment-of-fluid interface reconstruction. Technical report, Los Alamos National Laboratory, 2005.

[6] H. Fang and Y. Saad. Two classes of multisecant methods for nonlinear acceleration. *Numerical Linear Algebra with Applications*, 16(3):197–221, 2009.

[7] F.H. Harlow and J.E. Welch. Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with Free Surface. *Physics of Fluids*, 8(12):2182–2189, 1965.

[8] D. Hempel. Isotropic refinement and recoarsening in two dimensions. *Numerical Algorithms*, 13(1):33–43, 1996.

[9] C.W Hirt and B.D Nichols. Volume of fluid (VOF) method for the dynamics of free boundaries. *Journal of Computational Physics*, 39(1):201–225, 1981.

[10] E. Javierre. *Numerical methods for vector Stefan models of solid-state alloys*. PhD thesis, Delft University of Technology, The Netherlands, 2006.

[11] M. Möller. *Adaptive High-Resolution Finite Element Schemes*. PhD thesis, Technische Universität Dortmund, 2008.

[12] S.J. Osher and J.A. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics*, 79(1):12–49, 1988.

[13] D. den Ouden, A. Segal, F.J. Vermolen, L. Zhao, C. Vuik, and J. Sietsma. Application of the level-set method to a mixed-mode driven stefan problem in $2D$ and $3D$. *Computing*, 95(1):553–572, 2013.

[14] L.K. Schubert. Modification of a quasi-Newton method for nonlinear equations with a sparse Jacobian. *Mathematics of Computation*, 24(109):27–30, 1970.

[15] G. Segal, C. Vuik, and F.J. Vermolen. A conserving discretization for the free boundary in a two-dimensional Stefan problem. *Journal of Computational Physics*, 141(1):1–21, 1998.

[16] D. den Ouden. *Mathematical Modelling of Nucleating and Growing Precipitates: Distributions and Interfaces*. PhD thesis, Delft University of Technology, The Netherlands, 2015.

[17] S.P. van der Pijl. *Computation of bubbly flows with a mass-conserving level-set method*. PhD thesis, Delft University of Technology, The Netherlands, 2005.

[18] C. Ünsulan and A. Erçil. Conversions between parametric and implicit forms using polar/spherical coordinate representations. *Computer Vision and Image Understanding*, 81:1–25, 2001.

[19] F.J. Vermolen. On Similarity Solutions and Interface Reactions for a Vector-Valued Stefan Problem. *Nonlinear Analysis: Modelling and Control*, 12:269–288, 2007.

[20] H.F. Walker and P. Ni. Anderson acceleration for fixed-point iterations. *SIAM Journal on Numerical Analysis*, 49(4):1715–1735, 2011.

[21] Z. Zhang and A. Naga. A new finite element gradient recovery method: Superconvergence property. *SIAM Journal on Scientific Computing*, 26(4):1192–1213, 2005.