Computers and Mathematics with Applications 155 (2024) 110–125

Contents lists available at ScienceDirect

Computers and Mathematics with Applications

journal homepage: www.elsevier.com/locate/camwa

A velocity-based moving mesh virtual element method

H. Wells [a,*], M.E. Hubbard [a], A. Cangiani [b]

[a] School of Mathematical Sciences, University of Nottingham, University Park, Nottingham, NG7 2RD, United Kingdom
[b] Mathematics Area, SISSA, Via Bonomea 265, Trieste, 34136, Italy

ARTICLE INFO

Keywords:
Moving mesh method
Virtual element method
Arbitrary Lagrangian-Eulerian schemes
Polygonal meshes
Porous medium equation

ABSTRACT

We present a velocity-based moving mesh virtual element method for the numerical solution of PDEs involving boundaries which are free to move. The virtual element method is used for computing both the mesh velocity and a conservative Arbitrary Lagrangian-Eulerian solution transfer on general polygonal meshes. The approach extends the linear finite element method to polygonal mesh structures, achieving the same degree of accuracy. In the context of moving meshes, a major advantage of the virtual element approach is the ease with which nodes can be inserted on mesh edges. Demonstrations of node insertion techniques are presented to show that moving polygonal meshes can be simply adapted for situations where a boundary encounters a solid object or another moving boundary, without reduction in degree of accuracy.

1. Introduction

We propose a new numerical method for the simulation of free boundary problems over general polygonal meshes based on combining a moving mesh algorithm with the Virtual Element Method (VEM). The new approach is presented focusing on the solution of the free boundary problem for the porous medium equation as a model problem. To showcase its generality, we also briefly discuss an extension to a nonlinear fourth-order problem.

Moving mesh methods form part of a large class of adaptive mesh refinement techniques alongside h- and p-refinement strategies. The primary advantage of moving mesh methods is the ability to optimize mesh structures without requiring any change in the mesh connectivity, which can cause computational challenges, particularly when parallel implementations are considered. They also provide a natural framework for tracking physical features of a time-dependent PDE, such as blow-up problems [22], phase change modelling [10], fluid-structure interaction problems [45], and more general time-dependent PDEs [5,41,31].

Here we consider a *velocity-based* moving mesh algorithm [6,8,9,35,38,7] for the numerical solution of free boundary problems. The method is closely related to the Geometric Conservation Law method (GCL) [25] and also forms part of a larger family of adaptive moving mesh methods [33]. It uses a Lagrangian formulation of the given PDE to solve directly for the mesh velocities which are integrated over time to evolve the mesh. The solution computed on any given time-step is then transferred to the following time-step using an Arbitrary Lagrangian Eulerian (ALE) scheme [27] based on a weak distribution of a given monitor function.

We combine the velocity-based moving mesh algorithm with the virtual element method for spatial discretisation on general polygonal meshes [12]. The VEM was first introduced in [12] for the conforming discretisation of linear elliptic problems; cf. also [1,13,24,46] for extensions and implementation details. It provides a flexible discretization framework for the design of *compatible* general mesh methods that incorporate, at the discrete level, fundamental properties of the continuous problem at hand, such as topology, conservation, symmetry, and positivity. To achieve compatibility, discrete virtual element spaces are typically defined on individual polygonal elements implicitly through local boundary value problems. These are understood through a set of degrees of freedom instead of explicit basis functions. The use of local polynomial projections of the virtual discrete functions permits the definition of virtual element weak formulations which are computable at a cost on par with that of standard Finite Element Methods (FEMs). Here we consider the lowest-order VEM, which can be seen as a generalisation of the standard linear FEM to general polygonal/polyhedral meshes. The benefit of this generalisation is that typical issues such as node tangling, contact with obstacles, and topological changes of the moving domain [33,7] can be dealt with fully by local changes in the mesh topology. In particular, we highlight the possible issues with a remeshing strategy for complex problems with a FEM in which changes to mesh topology have to be carefully chosen and monitored to preserve shape-regular elements and avoid hanging nodes. The VEM provides a

* Corresponding author.
E-mail address: harry.wells@nottingham.ac.uk (H. Wells).

https://doi.org/10.1016/j.camwa.2023.12.005
Received 26 April 2023; Received in revised form 3 October 2023; Accepted 4 December 2023
Available online 12 December 2023
0898-1221/Crown Copyright © 2023 Published by Elsevier Ltd. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

simpler and more flexible approach to these, permitting hanging nodes and arbitrarily small edges.

To demonstrate this, we present a simple node insertion/removal algorithm for problems involving collision between a moving boundary and fixed obstacles. Just like linear FEM, the lowest-order VEM can be set up using only vertex information and in particular without the use of computationally expensive quadrature on polygons. As such, our approach can be interpreted as a generalisation of the linear finite element moving mesh method [7].

Numerical results demonstrate that optimal orders of convergence are achieved when applying a virtual element discretization on a diverse range of polygonal mesh structures. To simplify the presentation, the novel moving mesh VEM is developed for the two-dimensional porous medium equation (PME). The PME is a parabolic non-linear diffusion equation [51]. It is an ideal benchmark for a moving mesh method as it provides time-varying solutions with compact support and a moving boundary. However, the moving mesh VEM proposed here may be immediately generalised to other free boundary problems. Generalisations of the velocity-based method to other PDEs can be found in the literature [6,7] and the VEM has already been generalised to the solution of a wide range of PDEs [14,50,48], including flow in porous media [49,56,52]. Hence we expect our approach to be extendable to a range of problems. To hint to the generality of the approach, in the numerical examples section we demonstrate the application of the moving mesh VEM to a fourth-order diffusion problem with a moving boundary.

In recent years, several works have been published which involve time-dependent and moving polygonal domains. These developments include: ALE maps between two meshes using virtual element and discontinuous Galerkin (dG) methods [37], a cut-based dG scheme for fluid-structure interaction problems [2], an ALE scheme on time-dependent reconstructed Voronoi meshes [30], adaptive methods using elliptic reconstruction techniques on moving meshes [23], polygonal mesh quality measures for an anisotropic MMPDE scheme [34], and a VEM for long term simulations of moving landforms [40]. The benefits of using polygonal discretizations for moving mesh methods include representing moving boundaries and interfaces with a minimal number of degrees of freedom and localized mesh refinement when a change in mesh connectivity is required, such as in contact problems. To showcase the potential of the moving mesh VEM in this respect, we test the method for the numerical treatment of both obstacle and self-intersecting moving interface problems.

The layout of the remaining sections of this paper is as follows. In Section 2 we introduce the moving mesh algorithm, including the weak formulations necessary for the method. The components of the VEM discretisation are split into two sections. Section 3 reviews the fundamentals of the VEM and formulates a method for computing mesh velocities at a fixed point in time. Section 4 presents a framework for moving virtual elements along with a VEM for a conservative ALE update scheme. An overview of the algorithm and implementation details is given in Section 5. Node insertion/removal algorithms are discussed in Section 6. A set of numerical experiments is presented in Section 7 for the PME and a fourth-order diffusion problem before drawing some conclusions in Section 8.

2. The moving mesh framework

In this section we present the key components of the velocity-based moving mesh framework for a general time-dependent PDE of the form

$$\frac{\partial \rho}{\partial t} = \mathcal{L}\rho, \tag{1}$$

with $t \in (0,T]$, $T \in \mathbb{R}^+$, indicating time and $\mathcal{L}$ a generic spatial differential operator in $\mathbb{R}^d$. The choice of spatial operator will be made specific later on when we introduce the porous medium equation as the model problem. Further details on a finite element discretisation of this moving mesh method for general parabolic problems can be found in [6,7] and the references therein.

The time-dependent support of the evolving solution to Equation (1) will be denoted by $\Omega^t \subseteq \mathbb{R}^d$, with boundary $\partial\Omega^t$ which is allowed to move. The temporal superscript t will be swapped for n when considering discrete time levels t^n or, when no confusion arises, the superscript notation will be dropped entirely for ease of reading.

A moving coordinate system is defined by $\mathbf{x} = \mathbf{x}(\boldsymbol{\xi}, t)$ where $\mathbf{x}(\boldsymbol{\xi}, 0) = \boldsymbol{\xi}$ is the coordinate system on the initial domain Ω^0, which is used as the reference domain for simplicity. It is assumed that, for all $t \in [0,T]$, the map $\mathbf{x}(\cdot, t)$ is Lipschitz with Lipschitz inverse. The evolution of Ω^t is determined by the velocity field

$$\mathbf{v} = \frac{\partial \mathbf{x}}{\partial t}. \tag{2}$$

Following [18], we shall also refer to the space-time domain

$$\mathcal{Q}_T = \{(\mathbf{x}, t) \,:\, t \in [0,T], \mathbf{x} = \mathbf{x}(\boldsymbol{\xi}, t), \boldsymbol{\xi} \in \Omega^0\}.$$

2.1. The velocity field

The movement of the mesh is derived by specifying the time evolution of the distribution of the spatial integral of some solution-dependent monitor function, $\mathbb{M}(\rho)$. Specifically, the velocity field $\mathbf{v}$ is determined by requiring that the initial distribution of the monitor function $\mathbb{M}(\rho)$ is conserved as time progresses. Namely, we look for $\mathbf{v} = \frac{\partial \mathbf{x}}{\partial t}$ such that, for all $w \in L^2(\mathcal{Q}_T)$, the coordinates $\mathbf{x}(\boldsymbol{\xi}, t)$ satisfy

$$\frac{\int_{\Omega^t} w(\mathbf{x},t)\,\mathbb{M}(\rho(\mathbf{x},t))\,d\mathbf{x}}{\int_{\Omega^t} \mathbb{M}(\rho(\mathbf{x},t))\,d\mathbf{x}} = \frac{\int_{\Omega^0} w(\boldsymbol{\xi},0)\,\mathbb{M}(\rho(\boldsymbol{\xi},0))\,d\boldsymbol{\xi}}{\int_{\Omega^0} \mathbb{M}(\rho(\boldsymbol{\xi},0))\,d\boldsymbol{\xi}} \qquad \forall t \in [0,T]. \tag{3}$$

The derivation of the moving mesh method is based on application of the Reynolds transport theorem and on the assumption that the material derivatives of the weighting functions $w(\mathbf{x}, t)$ are zero with respect to the velocity field $\mathbf{v}$ (see Equation (2)). Namely,

$$\mathcal{D}_t w = \frac{\partial w}{\partial t} + \mathbf{v} \cdot \nabla w = 0. \tag{4}$$

This is a common assumption made in finite element approaches to moving mesh algorithms [37,18,17] and is equivalent to assuming that $w(\mathbf{x}(\boldsymbol{\xi}, \mathbf{t}), t) = w(\boldsymbol{\xi}, 0)$ in Equation (3).

Equidistribution-based mesh movement algorithms typically attempt to reduce the global approximation error without changing the number of degrees of freedom by choosing a finite set of weighting functions $w_i(\mathbf{x}, t)$, so that each one has a direct association with a mesh node or element. The monitor function $\mathbb{M}(\rho)$ is then selected to act as a local error indicator, and the mesh is adjusted in an attempt to equidistribute the values of the weighted monitor integrals

$$\mu^t(w_i) = \int_{\Omega^t} w_i\,\mathbb{M}(\rho)\,d\mathbf{x}, \tag{5}$$

and hence to equidistribute the local error across the mesh. Such an approach could be followed within our framework [38,7]. However, in this paper we adopt the following approach which is more akin to Lagrangian mesh movement algorithms, which attempt to move the mesh with the 'flow' velocity.

Choosing $\mathbb{M}(\rho) = \rho$ in Equation (5) naturally leads to a weak approximation of the Lagrangian 'flow' velocity of the PDE when $\mathcal{L}\rho = \nabla \cdot \mathbf{f}$ for some flux $\mathbf{f}$ in Equation (1). This has two benefits: (a) it allows us to predict the movement of free boundaries; (b) it reduces interpolation error of the computed mesh and solution between time-steps because the mesh (and hence the solution) is transported with the velocity field inherent to the PDE.

In many PDEs (including the PME), ρ represents a density, *i.e.* its integral in space is a mass, so we will refer to $\mathbb{M}(\rho) = \rho$ as the mass monitor. For clarity of presentation, we will assume use of this mass monitor from now on. Hence, each weight function $w(\mathbf{x}, t)$ is assigned its own 'mass' $\mu^t(w)$, which contributes towards the total mass θ^t:

$$\mu^t(w) = \int_{\Omega^t} w\rho \, d\mathbf{x} \qquad \text{where} \qquad \theta^t = \int_{\Omega^t} \rho \, d\mathbf{x}. \tag{6}$$

Having selected $\mathbb{M}(\rho) = \rho$, Equation (3) takes the form

$$\frac{\mu^t(w)}{\theta^t} = \frac{\mu^0}{\theta^0} := c(w), \tag{7}$$

in which we have defined distribution coefficients $c(w)$ which we can compute from the initial conditions and will assume remain constant in time when we derive our velocity field. It is this assumption that provides local mass conservation, *i.e.* each weight function w retains the same proportion $c(w)$ of the total mass θ^t as the solution evolves in time. The evolution of ρ is governed by Equation (1), so Equation (6) provides us with a way to prescribe the evolution of the coordinate system in a way which retains the initial 'mass' distribution.

Remark 1. The original moving mesh finite element method [6,7] chose the weight functions to be the standard linear Lagrange finite element test functions on meshes of simplices. This associates a fixed proportion of the total mass of the system with each node of the mesh, so the values of $\mu^t(w)$ depend not only on ρ, but also on the mesh node positions. In this situation, Equation (6) provides a way to compute mesh node velocities using standard finite element techniques, in a way which is consistent with local mass conservation when this is a property of the underlying PDE. Our results to follow demonstrate that the same principle can be applied on polygonal meshes within a virtual element framework.

Let $t \in [0, T]$ and consider all $w \in H^1(Q_T)$ such that $\mathcal{D}_t w(t) = 0$, cf. Equation (4). We differentiate the first equality in (6) with respect to time and apply the Reynolds transport theorem [53,42,18]:

Theorem 2.1 *(Reynolds transport theorem). For any $f \in W^{1,1}(Q_T)$ and $t \in [0, T]$ there holds*

$$\frac{d}{dt}\int_{\Omega_t} f \, d\mathbf{x} = \int_{\Omega_t} \frac{\partial f}{\partial t} + \nabla \cdot (f\mathbf{v}) \, d\mathbf{x}. \tag{8}$$

This immediately leads to

$$\dot{\mu}^t(w) = \int_{\Omega^t} w \left\{ \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) \right\} \, d\mathbf{x}. \tag{9}$$

Noting that (9) does not fully determine the velocity, we further require that each weighting function retains a fixed proportion of θ^t as the mesh evolves *i.e.* the distribution coefficients $c(w)$ defined in Equation (7) remain constant in time. Hence we impose the constraints

$$\dot{\mu}^t(w) = c(w)\dot{\theta}^t. \tag{10}$$

It is this choice which allows us to generate mesh velocities which approximate the Lagrangian flow velocities. Inserting (10) in (9), using (1), and integrating by parts results in

$$c(w)\dot{\theta}^t + \int_{\Omega^t} \rho \nabla w \cdot \mathbf{v} \, d\mathbf{x} = \int_{\Omega^t} w\mathcal{L}\rho \, d\mathbf{x} + \int_{\partial\Omega^t} w\rho\mathbf{v}\cdot\mathbf{n} \, ds, \tag{11}$$

with $\mathbf{n}$ the outward unit normal on $\partial\Omega^t$. Similarly, a direct application of the Reynolds transport theorem to the second equality in (6), yields

$$\dot{\theta}^t = \int_{\Omega^t} \mathcal{L}\rho \, d\mathbf{x} + \int_{\partial\Omega^t} \rho\mathbf{v}\cdot\mathbf{n} \, ds. \tag{12}$$

In fact, $\dot{\theta}^t$ is typically known explicitly because $\rho\mathbf{v}\cdot\mathbf{n}$ is known on the whole of $\partial\Omega^t$, and the boundary conditions provided with the PDE will enable evaluation of the integral of $\mathcal{L}\rho$. Furthermore, for mass-conservative problems, $\dot{\theta}^t = 0$. Note also that assuming that $c(w)$ remains constant preserves the initial distribution of the mass (or a more general monitor integral) so a monitor which is initially equidistributed between a set of weighting functions should remain equidistributed as the coordinate system evolves with this velocity field.

Equations (11) and (12) are used to compute an instantaneous velocity $\mathbf{v}$ which is consistent with conserving the proportion of mass associated with each weighting function. This provides a form of local mass conservation when $\dot{\theta}^t = 0$. However, it still does not uniquely define $\mathbf{v}$ in multiple space dimensions. To overcome this issue, the velocity field is written in terms of its Helmholtz decomposition

$$\mathbf{v} = \mathbf{q} + \nabla\phi, \tag{13}$$

where ϕ is a scalar potential and $\mathbf{q}$ must be specified. This constraint is equivalent to imposing the curl of the velocity field because $\nabla \times \mathbf{q} = \nabla \times \mathbf{v}$. Moreover, for simplicity, we may further assume that $\mathbf{q} = \mathbf{0}$. This is the natural choice for the porous medium equation, which is derived under the assumption of a curl-free flow velocity field. An example of the method applied with a rotational velocity field is presented in [7]. The problem for determining the velocity potential is therefore: find $\phi \in H^1(\Omega^t)$ such that

$$\int_{\Omega^t} \rho\nabla w \cdot \nabla\phi \, d\mathbf{x} = \int_{\Omega^t} w\mathcal{L}\rho \, d\mathbf{x} + \int_{\partial\Omega^t} w\rho\mathbf{v}\cdot\mathbf{n} \, ds - c(w)\dot{\theta}^t \quad \forall w \in H^1(\Omega^t), \tag{14}$$

where $\phi = 0$ is specified at one point in Ω^t to ensure uniqueness.

The velocity field is finally obtained as the solution of Equation (13) written in weak form with $\mathbf{q} = \mathbf{0}$ and ϕ given by (14). That is, we find $\mathbf{v} \in [H^1(\Omega^t)]^d$ such that

$$\int_{\Omega^t} z\mathbf{v} \, d\mathbf{x} = \int_{\Omega^t} z\nabla\phi \, d\mathbf{x} \qquad \forall z \in H^1(\Omega^t), \tag{15}$$

with $\mathbf{v}\cdot\mathbf{n}$ imposed on any part of the boundary where it is known. For instance, in case of contact with an obstacle, we would impose $\mathbf{v}\cdot\mathbf{n} = 0$ on the contact boundary. Note that (15) is just the component-wise L^2-projection.

Remark 2. The velocity field in the interior of Ω^t could be discarded at this stage, and replaced by one derived using a different approach which is constrained by the boundary velocity derived above. For example, interior mesh nodes could be moved using our approach with a different monitor function or a Laplacian smoothing could be applied [38,45].

2.2. Solution update

The velocity field $\mathbf{v}$ derived using Equations (14) and (15) can now be used in the update of the solution. Integration of Equation (9) by parts and substitution of Equation (1) results in

$$\dot{\mu}^t(w) = \int_{\Omega^t} w\mathcal{L}\rho \, d\mathbf{x} - \int_{\Omega^t} \rho\nabla w \cdot \mathbf{v} \, d\mathbf{x} + \int_{\partial\Omega^t} w\rho\mathbf{v}\cdot\mathbf{n} \, ds \quad \forall w \in H^1(\Omega^t). \tag{16}$$

This is a standard conservative ALE update. Along with Equation (2), it gives a system of ODEs governing the evolution of the coordinate system and the distribution of the mass monitor which can be approximated using standard solvers such as explicit Runge-Kutta methods [7,35]. With these at hand, the solution can be recovered by solving the problem: find $\rho \in H^1(\Omega^t)$ such that

$$\int_{\Omega^t} w\rho \, d\mathbf{x} = \mu^t(w) \qquad \forall w \in H^1(\Omega^t). \tag{17}$$

In the special case where $\dot{\theta}^t = 0$ and $c(w)$ is assumed constant in time, Equation (16) is redundant because $\dot{\mu}^t(w) \approx 0$ by design. In fact, *without* the velocity recovery from the potential through (14) and (15), we

would have $\dot{\mu}^t(w) \equiv 0$. However, the practical steps of the recovery procedure may introduce small perturbations. Alternatively, one may use the knowledge that $\dot{\mu}^t(w) \equiv 0$ and hence resort directly to the initial values $\mu^0(w)$, avoiding the need to calculate the ALE update (16) altogether: we refer to this as direct recovery. However, direct recovery can only be used with the specific choice of the mass monitor and with mass-conservative PDEs. In other situations the interior velocity field will not correspond to $\dot{\mu}^t(w) = 0$ and the ALE update is essential.

Remark 3. An alternative, non-conservative, ALE formulation can be derived which would give an equation for $\dot{\rho}$ instead of $\dot{\mu}$ (see [38]). This derivation does not require that the weight functions evolve with zero material derivative, but conservation of mass is lost as a result.

We complete this subsection by providing a high level summary of the main steps of the moving mesh algorithm, using a subscript $\cdot_h$ to indicate (in a very general sense) discrete approximations on a polygonal domain Ω_h. As stated above, the evolution of the mesh and solution can be expressed in the form of a system of ordinary differential equations in time:

$$\frac{d}{dt}\begin{pmatrix} \mathbf{x}_h \\ \mu(w_h) \end{pmatrix} = \mathbf{F}(\mathbf{x}_h, \mu(w_h)). \tag{18}$$

The following steps can be used to compute $\mathbf{F}(\mathbf{x}_h, \mu(w_h))$.

(1) Given $\mathbf{x}_h$ and $\mu(w_h)$, compute ρ_h by solving a discrete form of (17),

$$\int_{\Omega_h} w_h \rho_h \, d\mathbf{x} = \mu(w_h). \tag{19}$$

(2) Given ρ_h, compute $c(w_h)$ using a discrete form of (7),

$$c(w_h) = \int_{\Omega_h} w_h \rho_h \, d\mathbf{x} \Big/ \int_{\Omega_h} \rho_h \, d\mathbf{x}\,. \tag{20}$$

This step is omitted for the PME in the following sections because in that case $\dot{\theta} = 0$, which eliminates the requirement for $c(w_h)$ in step (3).

(3) Given ρ_h and $c(w_h)$, compute ϕ_h by solving discrete forms of (14) and (12),

$$c(w_h)\dot{\theta}_h + \int_{\Omega_h} \rho_h \nabla w_h \cdot \nabla \phi_h \, d\mathbf{x} \tag{21}$$

$$= \int_{\Omega_h} w_h (\mathcal{L}\rho)_h \, d\mathbf{x} - \int_{\Omega_h} \rho_h \nabla w_h \cdot \mathbf{q}_h \, d\mathbf{x} + \int_{\partial\Omega_h} w_h \rho_h \mathbf{v}_h \cdot \mathbf{n} \, ds,$$

$$\dot{\theta}_h = \int_{\Omega_h} (\mathcal{L}\rho)_h \, d\mathbf{x} + \int_{\partial\Omega_h} \rho_h \mathbf{v}_h \cdot \mathbf{n} \, ds. \tag{22}$$

For uniqueness, it is necessary to specify ϕ_h at one point in Ω_h. Furthermore, $\mathbf{q}_h$ and, on the boundary, $\rho_h \mathbf{v}_h \cdot \mathbf{n}$ must be provided. In this work we set $\phi_h = 0$ at an arbitrary mesh vertex and $\mathbf{q}_h = \mathbf{0}$ throughout the domain. For the PME, $\rho_h \mathbf{v}_h \cdot \mathbf{n}$ is provided in terms of the solution by the kinematic boundary condition and $\dot{\theta}_h = 0$.

(4) Given ϕ_h, compute $\dot{\mathbf{x}}_h$ by solving a discrete form of (15),

$$\int_{\Omega_h} z_h \dot{\mathbf{x}}_h \, d\mathbf{x} = \int_{\Omega_h} z_h (\nabla \phi_h + \mathbf{q}_h) \, d\mathbf{x}, \tag{23}$$

where in this work $\mathbf{q}_h = \mathbf{0}$. Note that $\dot{\mathbf{x}}_h$ is being used here instead of $\mathbf{v}_h$ to indicate the discrete velocity to make it clear that this is being used to provide the right-hand side of the ODE system (18).

(5) Given ρ_h and $\dot{\mathbf{x}}_h$, compute $\dot{\mu}(w_h)$ using a discrete form of (16),

$$\dot{\mu}(w_h) = \int_{\Omega_h} w_h (\mathcal{L}\rho)_h \, d\mathbf{x} - \int_{\Omega_h} \rho_h \nabla w_h \cdot \dot{\mathbf{x}}_h \, d\mathbf{x} + \int_{\partial\Omega_h} w_h \rho_h \mathbf{v}_h \cdot \mathbf{n} \, ds. \tag{24}$$

This is a standard conservative ALE update in which the boundary integral is computed using the appropriate boundary conditions.

Equations (23) and (24) provide the right-hand side $\mathbf{F}(\mathbf{x}_h, \mu(w_h))$ of the ODE system (18), which can be approximated by the user's method of choice.

2.3. Porous medium equation

On an open, bounded domain $\tilde{\Omega} \subset \mathbb{R}^d$, we consider the following initial-boundary value problem for the Porous Medium Equation (PME): find $\rho : \tilde{\Omega} \times \mathbb{R}^+ \to \mathbb{R}$ such that

$$\frac{\partial \rho}{\partial t}(\mathbf{x}, t) = \Delta \Phi(\rho(\mathbf{x}, t)) \qquad (\mathbf{x}, t) \in \tilde{\Omega} \times (0, \infty], \tag{25}$$

$$\rho(\mathbf{x}, t) = 0 \qquad (\mathbf{x}, t) \in \partial\tilde{\Omega} \times (0, \infty], \tag{26}$$

$$\rho(\mathbf{x}, 0) = \rho^0(\mathbf{x}) \qquad \mathbf{x} \in \tilde{\Omega}, \tag{27}$$

where $\Phi = \rho^{m+1}/(m+1)$, for some $m > 0$, and $\rho^0 \geq 0$ having compact support in $\tilde{\Omega}$. The PME belongs to the broader class of Generalized Porous Medium Equations (GPME), also known as filtration equations, obtained with $\Phi : \mathbb{R}^+ \to \mathbb{R}^+$ any increasing function. The mathematical analysis of the GPME is well developed; see the monograph [51] and the references therein. In particular, the notion of appropriate weak solutions is discussed in [51] where it is shown that, for non-negative $\rho^0 \in L^1(\tilde{\Omega})$ and for $\Psi(\rho^0) \in L^1(\tilde{\Omega})$, where Ψ is the anti-derivative of Φ, if $\Psi(\rho) > 0$ for $\rho > 0$, there exists a unique non-negative weak solution to the GPME globally in time.

The PME models a number of physical processes such as fluid flow, heat transfer, and diffusion. It exhibits several interesting properties, including the existence of a family of radially symmetric similarity solutions [42], which are used to test the numerical method in Section 7. Other properties of the PME are discussed in [51,7,43].

It is a fundamental example of a degenerate parabolic equation, stemming from the condition that Ψ is non-negative, rather than simply positive.

Solutions that exhibit an evolving compact support are ideal for the class of moving mesh methods considered herein because considering as unknown the support of the solution leads to a moving boundary problem that can be simulated over time without having to discretize the entire geometry of $\tilde{\Omega}$.

Introducing the time-dependent support of ρ as Ω^t, we define the time-dependent coordinate system $\mathbf{x}(\boldsymbol{\xi}, t)$ with a velocity field $\dot{\mathbf{x}} = \mathbf{v}(\mathbf{x}, t)$ that corresponds, at $\partial\Omega^t$, with the movement of the free boundary of Ω^t for all $t \in [0, \infty]$. Additionally, we consider the free boundary problem for the PME in which part of the boundary may be obstructed by a fixed object. To this end, the boundary is divided into a moving part $\partial\Omega^t_M$ and a fixed part $\partial\Omega^t_F$, such that $\partial\Omega^t = \partial\Omega^t_M \cup \partial\Omega^t_F$ and $\partial\Omega^t_M \cap \partial\Omega^t_F = \emptyset$. Thus, we arrive to the following classical free boundary problem for the PME, whose smooth solutions are weak solutions of (25)-(27), cf. [51].

Problem 2.1 *(The Porous Medium Equation (PME)). Let $T > 0$ and $m > 0$. Find $\rho = \rho(\mathbf{x}, t)$ such that $\rho(\mathbf{x}, 0) = \rho^0(\mathbf{x})$ for $\mathbf{x} \in \Omega^0$ and, for all $t \in (0, T]$,*

$$\frac{\partial \rho}{\partial t} = \nabla \cdot (\rho^m \nabla \rho) \qquad \mathbf{x} \in \Omega^t,$$

$$\rho = 0 \qquad \mathbf{x} \in \partial\Omega^t_M,$$

$$\rho^m \nabla \rho \cdot \mathbf{n} = 0 \qquad \mathbf{x} \in \partial\Omega^t_F.$$

Here, $\mathbf{n}$ is the outward pointing unit normal to the boundary $\partial\Omega^t$. Note that an additional (kinematic) boundary condition,

$$\rho \mathbf{v} \cdot \mathbf{n} = -\rho^m \nabla \rho \cdot \mathbf{n} \qquad \mathbf{x} \in \partial\Omega^t_M,$$

which imposes zero flux through the moving boundary, is required to determine the boundary velocity $\mathbf{v}$. *On* $\partial\Omega_F$, *the boundary velocity is specified and* $\rho \mathbf{v} \cdot \mathbf{n} = 0$.

Given a transformed coordinate system $\mathbf{x}$ and a distribution μ of the variable ρ, our method for solving the PME Problem 2.1 requires the solution of the following problems successively.

Problem 2.2 *(Solution reconstruction – equation (17)). Given* $\mu^t(w) \in \mathbb{R}$, *find* $\rho \in H^1(\Omega^t)$ *such that*

$$m(\rho, w) = \mu^t(w) \qquad \forall\, w \in H^1(\Omega^t), \tag{28}$$

where

$$m(\rho, w) = \int_{\Omega^t} w\rho \, d\mathbf{x}. \tag{29}$$

Problem 2.3 *(Velocity potential – equation (14)). Given* $\rho \in H^1(\Omega^t)$, *find* $\phi \in H^1(\Omega^t)/\mathbb{P}_0(\Omega^t)$, *where* $\mathbb{P}_0(\Omega^t)$ *is the space of constants, such that*

$$A(w, \phi) = d(w) \qquad \forall\, w \in H^1(\Omega^t)/\mathbb{P}_0(\Omega^t), \tag{30}$$

where

$$A(\phi, w) = \int_{\Omega^t} \rho \nabla\phi \cdot \nabla w \, d\mathbf{x}, \tag{31}$$

$$d(w) = -\int_{\Omega^t} \rho^m \nabla\rho \cdot \nabla w \, d\mathbf{x}. \tag{32}$$

This has been derived by substituting $\mathcal{L}\rho = \nabla \cdot (\rho^m \nabla \rho)$ in (14) and applying the boundary conditions from Problem 2.1. To reconstruct the velocity field we introduce the following space

$$\left[H^1_{\partial\Omega^t_F}(\Omega^t)\right]^d = \left\{ \mathbf{w} \in \left[H^1(\Omega^t)\right]^d \;:\; \mathbf{w} \cdot \mathbf{n}|_{\partial\Omega^t_F} = 0 \right\}, \tag{33}$$

in which we have the following reconstruction problem.

Problem 2.4 *(Velocity reconstruction – equation (15)). Given* $\phi \in H^1(\Omega^t)$, *find* $\mathbf{v} \in \left[H^1_{\partial\Omega^t_F}(\Omega^t)\right]^d$ *such that*

$$M(\mathbf{v}, z) = b(z) \qquad \forall\, z \in H^1(\Omega^t), \tag{34}$$

where

$$M(\mathbf{v}, z) = \int_{\Omega^t} z\mathbf{v} \, d\mathbf{x}, \tag{35}$$

$$b(z) = \int_{\Omega^t} z\nabla\phi \, d\mathbf{x}. \tag{36}$$

To compensate for the fact that the velocity is reconstructed from the potential (see the comments in Section 2.2), an *a posteriori* computation of the ALE update may be used [27]. Specifically, substituting $\mathcal{L}\rho = \nabla \cdot (\rho^m \nabla \rho)$ and the PME boundary conditions into (16) we arrive to the following.

Problem 2.5 *(ALE update – equation (16)). Given* $\rho \in H^1(\Omega^t)$ *and* $\mathbf{v} \in \left[H^1(\Omega^t)\right]^d$, *fix*

$$\dot{\mu}^t(w) = \int_{\Omega^t} -\rho \nabla w \cdot \left\{ \rho^{m-1} \nabla \rho + \mathbf{v} \right\} \, d\mathbf{x} \qquad \forall\, w \in H^1(\Omega^t). \tag{37}$$

Once again we note that, when $\dot{\mu}^t(w) = 0$ is assumed, Equation (37) is redundant and Equation (30) is equivalent to a weak form of the PDE

$$\nabla \cdot (\rho \mathbf{v}) = -\nabla \cdot (\rho^m \nabla \rho), \tag{38}$$

where $\nabla \times \mathbf{v} = \mathbf{0}$ has been assumed. In other words, the formulation recovers the Lagrangian flow field associated with the porous medium equation.

Equations (37) and $\dot{\mathbf{x}} = \mathbf{v}(\mathbf{x}, t)$ can now be combined to give a system of ODEs which determine the evolution of the coordinate system and the monitor distribution. These can be discretized in time using an appropriate solver for first-order ODEs. Specifically, the forward Euler method is used to perform the tests presented in Section 7. A first-order time-stepping approach is sufficient for this work because it considers only the lowest-order VEM, providing second-order accuracy in space, and the explicit nature of the method means that, for the PME, the time-step is required to scale in proportion with the square of the spatial mesh size to retain stability. Higher-order explicit time-stepping schemes have been presented in [35,7] for the FEM implementation of the moving mesh method.

We now proceed to describe how Problems 2.2–2.5 can be approximated in space on two-dimensional polygonal meshes using the virtual element method.

3. Virtual element method for the velocity

In this section we introduce the VEM and apply it for the discretisation of Problems 2.3 and 2.4 to determine the domain velocity field. We follow the framework introduced in [12,1,24] for the discretisation of linear elliptic problems. For simplicity, we consider the lowest-order (linear) VEM in two dimensions. Hence, for the rest of the paper, it is assumed that $d = 2$ and the solution space is of degree $k = 1$.

In the following we let $0 = t^0 < t^1 < \ldots t^{N_t} = T$ denote a given sequence of discrete time levels at which the PME is discretised.

3.1. A moving polygonal mesh

The polygonal representation of the moving domain Ω^t at the discrete time level t^n is denoted by Ω^n_h and the corresponding polygonal mesh partitioning Ω^n_h is denoted by $\mathcal{T}^n_h$. Polygonal elements and edges within the mesh are denoted by E and e, respectively. For a given polygonal element E, $|E|$ denotes the area, h_E the diameter, and (x_c, y_c) the centroid.

The following conditions are sufficient to guarantee convergence of the VEM and are assumed to hold true at each time level t^n, $n = 0, 1, \ldots, N_t$; see, for example, [12].

Assumption 1 *(Mesh partitioning).* The mesh $\mathcal{T}^n_h$ provides a partition of Ω^n_h into non-overlapping simple polygons.[1]

Assumption 2 *(Shape regularity).* Every $E \in \mathcal{T}^n_h$ is a star-shaped domain or a finite union of star shaped domains with respect to a ball of radius greater than γh_E for some uniform $\gamma > 0$. Additionally, for all edges $e \in \partial E$, the length of e is greater than δh_E for some uniform $\delta > 0$.

Remark 4. The shape regularity assumption on the edges required by Assumption 2 can be entirely removed following, for example, [19] or [15]. Small edges are relevant, for instance, for the various types of contact problems considered below.

[1] A polygon is *simple* if its boundary forms a closed graph with no intersecting edges other than at each vertex which intersects exactly two edges.

3.2. The Π^∇ and Π^0 projections

Given a subset $\omega \subset \mathbb{R}^d$, $d = 1, 2$, representing an element or an edge, with $\mathbb{P}_k(\omega)$ we denote the space of polynomials of degree k on ω. Furthermore, we use $(\cdot,\cdot)_\omega$ to denote the L^2 inner product over ω.

Critical to the construction of a VEM is the judicious choice of discrete spaces and degrees of freedom that permit the computation of local polynomial projections of virtual functions using only the degrees of freedom.

Specifically, the definition of the VEM space to be introduced below is based on an H^1-type projection operator Π_1^∇ onto $\mathbb{P}_1(E)$ given, for any $w \in H^1(E)$, by

$$\begin{cases} \left(\nabla\Pi_1^\nabla w, \nabla p\right)_{0,E} = (\nabla w, \nabla p)_{0,E} \quad \forall p \in \mathbb{P}_1(E) \\ \left(w - \Pi_1^\nabla w, 1\right)_{0,\partial E} = 0. \end{cases} \tag{39}$$

Further, the VEM requires the availability of the L^2-projection operator Π_1^0 onto $\mathbb{P}_1(E)$ defined, for $w \in L^2(E)$, by

$$\left(\Pi_1^0 w, p\right)_{0,E} = (w, p)_{0,E} \quad \forall p \in \mathbb{P}_1(E). \tag{40}$$

Additionally, the VEM requires the L^2-projection onto constants of ∇w which we denote by Π_0^0, thus

$$\left(\Pi_0^0 \nabla w, \mathbf{p}\right)_{0,E} = (\nabla w, \mathbf{p})_{0,E} \quad \forall \mathbf{p} \in [\mathbb{P}_0(E)]^2. \tag{41}$$

The same operator applied component-wise to vector-valued functions will be denoted by Π_1^0.

3.3. Virtual element spaces

Virtual element discrete functions are defined implicitly as solutions of element-wise boundary value problems. These functions are only accessed through their degrees of freedom with which specific projections can be computed; in this way, *no* evaluation of the (implicitly known) discrete functions is required. Here we consider the linear virtual element space of [1] which is a modification of the original space proposed in [12] for which both the H^1 and L^2 projections introduced in the previous section are computable from the degrees of freedom.

Given a polygonal element $E \in \mathcal{T}_h^n$, we first define the elemental boundary space as

$$\mathbb{B}(\partial E) = \{w_h \in C^0(\partial E) \ : \ w_h|_e \in \mathbb{P}_1(e) \quad \forall e \subset \partial E\}. \tag{42}$$

Then the local virtual element space of [12] is defined as

$$W(E) = \{w_h \in H^1(E) \ : \ w_h|_{\partial E} \in \mathbb{B}(\partial E),\ \Delta w_h|_E = 0\}. \tag{43}$$

Note that $\mathbb{P}_1(E) \subseteq W(E)$. In fact, it is immediate to check that the space $W(E)$ corresponds to the elemental linear finite element space whenever E is a triangle, that is $W(E) \equiv \mathbb{P}_1(E)$. In this respect, the VEM can be seen as a generalisation of the linear FEM to polygonal meshes. It is clear from the definition that the only degrees of freedom in the choice of a function $w_h \in W(E)$ are related to its values on the boundary and, given that w_h is linear on each edge, ultimately they depend only on the nodal values; a rigorous proof that these constitute a unisolvent set of degrees of freedom for $W(E)$ is presented in [12]. Hence, in analogy to classical continuous linear finite element spaces, we identify the VEM functions by their nodal values.

Additionally, the nodal values allow for the computation of the local Π_1^∇ projection of any $w_h \in W(E)$. Indeed, in order to compute the first equation in (39) we need to be able to evaluate, for any $p \in \mathbb{P}_1(E)$, the right-hand side term:

$$\left(\nabla w_h, \nabla p\right)_{0,E} = -\left(w_h, \Delta p\right)_{0,E} + \left(w_h, \mathbf{n}\cdot\nabla p\right)_{0,\partial E} = \sum_{e\in\partial E} \left(w_h, \mathbf{n}\cdot\nabla p\right)_{0,e}, \tag{44}$$

as $\Delta p = 0$. This shows that such terms only depend on the value of w_h on the boundary of E, which are known and are determined by the nodal values. The same is clearly true for the second equation in (39), hence the local Π_1^∇ projection is fully computable just by accessing the degrees of freedom. However, the L^2 projection is not computable for this space. Hence, following [1], we first consider the enriched space obtained by allowing the Laplacian of virtual functions to range in $\mathbb{P}_1(E)$,

$$\tilde{V}(E) = \{w_h \in H^1(E) \ : \ w_h|_{\partial E} \in \mathbb{B}(\partial E),\ \Delta w_h \in \mathbb{P}_1(E)\}, \tag{45}$$

from which a new *enhanced* local virtual element space can be defined as

$$V(E) = \{w_h \in \tilde{V}(E) \ : \ (w_h - \Pi_1^\nabla w_h, q)_{0,E} = 0 \quad \forall q \in \mathbb{P}_1(E)\}. \tag{46}$$

Notice that the space $V(E)$ is characterised by the following four properties: (i) we still have that $\mathbb{P}_1(E) \subseteq V(E)$; (ii) the dimension of $V(E)$ is equal to that of $W(E)$ and we can still use the nodal values as degrees of freedom (see [1]); (iii) the local Π_1^∇ projection is still computable; (iv) if $w_h \in V(E)$ then $\Pi_1^0 w_h = \Pi_1^\nabla w_h$ by construction, hence also the L^2-projection Π_1^0 is computable. The proof that the gradient projection into constants $\Pi_0^0 \nabla w_h$ is also computable is similar.

The global virtual element space is then defined for a given time t^n as

$$V_h^n = \{w_h \in H^1(\Omega_h^n) : w_h|_E \in V(E) \quad \forall E \in \mathcal{T}_h^n\}. \tag{47}$$

The dimension of this space is equal to the number of nodes in the mesh, which we shall denote by N^{dof}.

Remark 5. When required, homogeneous Dirichlet boundary conditions can be embedded in the virtual element space by fixing the relevant boundary nodes. Hence, given a $\Gamma \subseteq \partial\Omega_h^n$, we denote the constrained space by $V_{h,\Gamma}^n = \{w_h \in V_h^n \ : \ w_h|_\Gamma = 0\}$.

3.4. Discretisation of the velocity problems

Having defined the virtual element space in Section 3.3, we are ready to present the VEM for the solution of the velocity Problems 2.3 and 2.4. The VEM is based on the construction of approximate weak forms which are *computable* through the elemental projection operators. In particular, the VEM bilinear forms typically involve two terms. The *polynomial consistency* term acts on the projection of the discrete functions and is responsible for the accuracy of the method; the *stabilization term* is complementary to the consistency term and is required to ensure the coercivity of the VEM bilinear forms. See, for example, [12,1,24] for more details including proofs of stability and convergence.

Assume that a discrete solution at time level t^n has been computed as $\rho_h \in V_h^n$ on the current mesh $\mathcal{T}_h^n$. When $n = 0$, ρ_h is defined as the virtual element interpolation of the initial condition of the PME Problem 2.1. Otherwise, ρ_h is the current time discrete solution.

The VEM discretisation of the velocity potential Problem 2.3 reads: given $\rho_h \in V_h^n$, find $\phi_h \in V_h^n$ such that

$$A_h(\phi_h, w_h) = d_h(w_h) \qquad \forall\, w_h \in V_h^n \tag{48}$$

with the approximate bilinear form A_h and linear form d_h built by summing element-wise contributions as typical of FEM, hence

$$A_h(\phi_h, w_h) = \sum_{E\in\mathcal{T}_h^n} A_h^E(\phi_h, w_h) \qquad \text{and} \qquad d_h(w_h) = \sum_{E\in\mathcal{T}_h^n} d_h^E(w_h).$$

As anticipated above however, the preceding definition of the local VEM forms necessitates the use of projections as follows:

$$\begin{aligned} A_h^E(\phi_h, w_h) = &\int_E (\Pi_1^0\rho_h)_0\ \Pi_0^0\nabla\phi_h \cdot \Pi_0^0\nabla w_h\ d\mathbf{x} \\ &+ (\Pi_1^0\rho_h)_0\ S_A^E(\phi_h - \Pi\phi_h, w_h - \Pi w_h), \end{aligned} \tag{49}$$

$$d_h^E(w_h) = -\int_E (\Pi_1^0\rho_h)_0^m\, \Pi_0^0\nabla\rho_h \cdot \Pi_0^0\nabla w_h \, d\mathbf{x}, \tag{50}$$

where $(\Pi_1^0\rho_h)_0$ is the constant component of $\Pi_1^0\rho_h$. Here, following [12], the stabilization form $S_A^E(\cdot,\cdot)$ is defined by

$$S_A^E(v_h, w_h) = \sum_{l=1}^{m_E} \mathrm{dof}_l(v_h)\cdot \mathrm{dof}_l(w_h), \tag{51}$$

with m_E denoting the dimension of $V(E)$, which for the space considered here is equal to the number of vertices of E, and with $\mathrm{dof}_l(w)$ representing the l-th degree of freedom of the function w. Hence, $\mathrm{dof}_l(v_h) = v_h(\mathbf{x}_l)$ with $\mathbf{x}_l$ denoting the l-th vertex of E. The integration constant is fixed by constraining a single vertex value of ϕ_h to zero. A variety of suitable stabilization choices are admissible [14,24] but we adopt the simplest choice in this paper. In particular, in the case when arbitrarily small edges appear in the mesh we refer to [19] for more appropriate stabilization terms.

The velocity reconstruction Problem 2.4 is a global L^2 projection. Its VEM discretisation reads: given $\phi_h \in V_h^n$, the solution of (48), find $\mathbf{v}_h \in \left[V_h^n\right]^2$ such that $\mathbf{v}_h\cdot\mathbf{n} = 0$ on the portion of Ω_h^n approximating $\partial\Omega_F^n$ and

$$M_h(\mathbf{v}_h, w_h) = b_h(w_h) \qquad \forall\, w_h \in V_h^n. \tag{52}$$

As before, the forms M_h and b_h are obtained summing the respective elemental forms

$$M_h^E(\mathbf{v}_h, w_h) = \int_E \Pi_1^0\mathbf{v}_h\, \Pi_1^0 w_h \, d\mathbf{x} \;+\; S_M^E(\mathbf{v}_h - \Pi_1^0\mathbf{v}_h, w_h - \Pi_1^0 w_h), \tag{53}$$

$$b_h^E(w_h) = \int_E \Pi_1^0 w_h\, \Pi_0^0\nabla\phi_h \, d\mathbf{x}, \tag{54}$$

with the stabilization term $S_M^E(\cdot,\cdot)$ given by [1]

$$S_M^E(\mathbf{v}_h, w_h) = |E| \sum_{l=1}^{m_E} \mathrm{dof}_l(\mathbf{v}_h)\cdot\mathrm{dof}_l(w_h). \tag{55}$$

3.5. Moving the mesh

The mesh is transferred between discrete time levels by displacing the nodes of the mesh and maintaining the mesh connectivity between $t = t^n$ and $t = t^{n+1}$. For each mesh node $\mathbf{x}^n$ the new position is obtained by the forward Euler method applied to $\dot{\mathbf{x}} = \mathbf{v}_h(\mathbf{x})$, yielding $\mathbf{x}^{n+1} = \mathbf{x}^n + (t^{n+1} - t^n)\mathbf{v}_h(\mathbf{x}^n)$. Thus, consistent with the VEM philosophy, only the values of $\mathbf{v}_h$ at the nodes, that is the degrees of freedom of $\mathbf{v}_h$, are required to compute the mesh movement.

Remark 6. We note that the mesh velocities computed in this way do not guarantee a priori that Assumptions 1 and 2 hold true indefinitely. For instance, when performing the numerical experiments for contact problems of Section 7, we observed a degradation of mesh quality near the contact boundary as a result of node tangling, which violates Assumption 1 as elements eventually overlap. The ease with which VEM allows local node insertion mitigates some of the mesh quality issues which would occur with FEM on triangles, but it does not completely prevent mesh tangling when the node velocities cause the mesh to become highly distorted. To compensate for this, a harmonic extension operator is used to regularise the velocity field and maintain element structure. Further details are provided in Section 7.

4. Virtual element method for the solution

Once the new mesh node positions have been computed, we consider the process of updating the solution. This is performed in two steps corresponding, respectively, to the conservative ALE update of the mass monitor Problem 2.5 and the actual solution update Problem 2.2. Before presenting the details on their VEM discretisation, a discussion on the hypothesis leading to such problems is in order. The original moving mesh method in [7] was based on the linear FEM for which the validity at the discrete level of the material derivatives assumption (4) leading to the ALE update (9) has been proven in [36]. In the VEM setting, instead, we exploit the fact that virtual element functions are *only accessed through their nodal values*: in close alignment with the work presented in [37], only the mesh skeleton velocity is known and used. Hence, in view of the solution update through the time step $[t^n, t^{n+1}]$, we can assume that (4) is satisfied by the space-time discrete basis which are then interpolated at the new time level, again, just by accessing the nodal values of the solution.

4.1. Discretisation of the solution problems

The initial condition ρ_h^0 is approximated by interpolating the degrees of freedom of ρ^0 into the VEM space V_h^0. Then, the initial mass monitor distribution is computed via

$$\mu_h^0(w_h^0) = \sum_{E\in\mathcal{T}_h^0}\int_E \Pi_1^0\rho_h^0\, \Pi_1^0 w_h^0 \, d\mathbf{x}. \tag{56}$$

The next task is the update of the mass monitor over time levels. Once again, this is performed via the forward Euler method: the new monitor is thus given by $\mu_h^{n+1}(w_h^{n+1}) = \mu_h^n(w_h^n) + (t^{n+1} - t^n)\dot{\mu}_h^n(w_h^n)$. This, in turn, requires the approximation of the ALE equation (37) which is performed still on the old time level (superscript omitted) by

$$\dot{\mu}_h(w_h) = -\sum_{E\in\mathcal{T}_h^n}\int_E \Pi_1^0\rho_h\Pi_0^0\nabla w_h \cdot \left\{ \left(\Pi_1^0\rho_h\right)_0^{m-1}\Pi_0^0\nabla\rho_h + \Pi_1^0\mathbf{v}_h \right\} \, d\mathbf{x}. \tag{57}$$

Once the monitor is updated, we set the time level to the new time and update the solution using a VEM discretisation of Problem 2.2: find $\rho_h^{n+1} \in V_h^{n+1}$ such that

$$m_h(\rho_h^{n+1}, w_h^{n+1}) = \mu_h(w_h^{n+1}) \qquad \forall\, w_h^{n+1} \in V_h^{n+1}. \tag{58}$$

Similarly to the velocity reconstruction, the discrete form $m_h(\cdot,\cdot)$ is computed on the new time level by summing over element contributions

$$m_h(\rho_h, w_h) = \sum_{E\in\mathcal{T}_h^n} m_h^E(\rho_h, w_h), \tag{59}$$

where

$$m_h^E(\rho_h, w_h) = \int_E \Pi_1^0\rho_h\, \Pi_1^0 w_h \, d\mathbf{x} \;+\; S_m^E(\rho_h, w_h), \tag{60}$$

with the stabilization term $S_m^E(\cdot,\cdot)$ being given by

$$S_m^E(\rho_h, w_h) = |E| \sum_{l=1}^{m_E} \mathrm{dof}_l(\rho_h)\cdot\mathrm{dof}_l(w_h). \tag{61}$$

4.2. Partition of unity and conservation

A beneficial property of the linear virtual element method is that the basis functions form a partition of unity on Ω_h^n at any discrete time level, i.e.

$$\sum_{i=1}^{N^{\mathrm{dof}}} \varphi_i = 1, \tag{62}$$

where the set $\{\varphi_i\}_{i=1}^{N^{\mathrm{dof}}}$ refers to the set of canonical VEM basis functions associated to the vertices of the mesh [12]; that is $\varphi_i(\mathbf{x}_j) = \delta_{ij}$ for $i = 1, \ldots, N^{\mathrm{dof}}$ where $\mathbf{x}_j$ is the j-th node in the mesh.

For a given $\rho_h \in W_h^n$ the monitor integral θ^n reads

$$\theta^n = \int_{\Omega^n} \rho_h \, d\mathbf{x}, \tag{63}$$

from which the polynomial consistency and partition of unity property of the VEM gives

$$\theta^n = \sum_{E\in\mathcal{T}_h^n} \int_E \Pi_1^0 \rho_h \, d\mathbf{x} \tag{64}$$

$$= \sum_{E\in\mathcal{T}_h^n} \int_E \Pi_1^0 \rho_h \sum_{j=1}^{m_E} \varphi_j \, d\mathbf{x} \tag{65}$$

$$= \sum_{i=1}^{N^{\text{dof}}} \sum_{E\in\mathcal{T}_h^n} \int_E \Pi_1^0 \rho_h \, \Pi_1^0 \varphi_i \, d\mathbf{x}. \tag{66}$$

Therefore the global conservation of the mass monitor is only dependent on the polynomial component of the discrete solution and weighting functions. Further, the exact value of the monitor can also be recovered via

$$\theta^n = \sum_{i=1}^{N^{\text{dof}}} \mu_h^n(\varphi_i). \tag{67}$$

Finally, considering the partition of unity property, the ALE update equation (16) and equation (67) for the PME, we get

$$\dot{\theta}^n = 0, \tag{68}$$

which agrees with the conservation of mass principle for this particular PDE. In fact, virtual elements preserving relevant global conservation laws in different contexts can be constructed, an example of which is given in [50] for the heat equation.

5. Implementation details

This section presents a complete overview of the moving mesh virtual element method. The construction of the required algebraic equations and imposition of boundary conditions are reviewed along with some practical remarks regarding the implementation of this method. The initial weak distribution of the monitor is stored in the vector $\boldsymbol{\mu}^0$ and can be computed using equation (56) whilst the mass matrix $\mathbf{M}^n$ is computed by assembling the contributions from equation (60). In order to compute the discrete potential $\phi_h \in V_h^n$ from equation (48), we solve the linear system $\mathbf{A}^n\boldsymbol{\phi} = \mathbf{d}^n$ with $\mathbf{A}^n$ and $\mathbf{d}^n$ computed using equations (49) and (50), respectively. The solution of the resulting linear system determines ϕ_h up to an additive constant which is inherited from the continuous formulation of the method; here we impose $\text{dof}_1(\phi_h) = 0$. Once ϕ_h is recovered, the velocity field is reconstructed solving $\mathbf{M}_R^n\mathbf{v} = \mathbf{b}^n$, with $\mathbf{M}_R^n$ and $\mathbf{b}^n$ given by equations (53) and (54), respectively. The ALE update of vector $\dot{\boldsymbol{\mu}}^n$ is then obtained using equation (57) and, along with the mesh nodal velocities $\dot{\mathbf{x}}_i^n = \mathbf{v}_h(\varphi_i)$, provides a system of ODEs which can be approximated using the forward Euler method. The main method is outlined in Algorithm 1.

In this work we strongly impose any Dirichlet boundary conditions on the solution in the standard manner. However, we note here that in doing so, the test functions we consider in our discretisation do not satisfy the partition of unity property (62). As a consequence, mass is not conserved exactly. This issue is investigated in detail in [35], where a methodology was proposed for the finite element method which allows exact mass conservation and strong imposition of Dirichlet boundary conditions to be achieved simultaneously. A near-identical approach can be applied to virtual elements on polygons: the test space is augmented by adding boundary-lying test functions to adjacent interior test functions and then modifying the test space accordingly (see Section 2.2 of [35]). In practice, this is achieved by lumping terms of $\mathbf{M}^n$ and of $\boldsymbol{\mu}^n$ before reconstructing the solution. The differences we observed in the numerical results were negligible, so we only present the standard approach in Section 7 and do not discuss the conservative version in any more detail.

Algorithm 1: Moving mesh VEM.

input : The initial condition $\rho_h^0 \in W_h^0$ and mesh $\mathcal{T}_h^0$, the final time T.
Set $n = 0$;
Compute $\boldsymbol{\mu}^0$ according to equation (56);
while $t^n < T$ **do**
  Construct and solve $\mathbf{A}^n\boldsymbol{\phi} = \mathbf{d}^n$ using equations (49) and (50) for $\phi_h \in W_h^n$;
  Reconstruct the velocity via $\mathbf{M}_R^n\mathbf{v} = \mathbf{b}^n$;
  Compute the ALE update $\dot{\boldsymbol{\mu}}^n$ from equation (57);
  Select Δt and set $t^{n+1} = t^n + \Delta t$;
  Update the mesh node by $\mathbf{x}^{n+1} = \mathbf{x}^n + \Delta t\mathbf{v}$;
  Update the monitor distribution by $\boldsymbol{\mu}^{n+1} = \boldsymbol{\mu}^n + \Delta t\dot{\boldsymbol{\mu}}^n$;
  Reconstruct and solve $\mathbf{M}^{n+1}\boldsymbol{\rho}^{n+1} = \boldsymbol{\mu}^{n+1}$ for $\rho_h^{n+1} \in W_h^{n+1}$;
  Update $n = n + 1$;
end
output: The final solution ρ_h^T, the final mesh $\mathcal{T}_h^T$

6. A contact algorithm

In this section we discuss two occurrences of contact and present corresponding basic node insertion algorithms that allow for localised and minimal changes to the mesh structure. In all mesh refinements considered, the change in mesh topology is only performed at the discrete time-levels. Hence, to ease notation, the superscript used to denote time-steps is omitted. Modified discrete functions, operators, and vectors are denoted using a hat symbol.

6.1. Contact scenarios

Here we present two contact scenarios that are numerically investigated in Section 7. The first scenario concerns the collision of the moving boundary with itself (see Fig. 1) whereas the second situation involves collision with fixed geometric obstacles (see Fig. 2).

Self-intersection handles the situation where two parts of the moving boundary collide with each other. Typically, a remeshing is required in this instance. By using a VEM, the remeshing can be kept local and simple for colliding elements. A motivational case for a disconnected initial condition of the PME is given in Fig. 1. Moving mesh finite element simulations of this type of problem are presented and discussed in [43].

Obstacle contact is encountered when the evolution of Ω^t is obstructed by external obstacles. An example of this is the presence of a solid phase in porous media. Fig. 2 presents an example of collision with impermeable obstacles. By using a collision detection and node insertion algorithm, the moving mesh is capable of simulating the contact and the interaction between the moving mesh and a set of obstacles. As with the self-intersection problem, the VEM allows for this with minimal changes to the mesh topology. Additionally, the VEM is capable of performing local changes to polygonal elements such that the mesh boundary can move around the object boundaries without requiring additional mesh refinements.

We note that the use of the VEM for handling problems with colliding meshes was first studied in [55] to deal with elastic contact problems. Although we consider here a different model problem which is purely geometric, as in [55] a standard contact detection algorithm based on orthogonal projections between boundary nodes and boundary edges is used to determine contact between meshes. However, in our work, we use the mesh velocities to trace the exact contact time between meshes instead of projecting nodes to edges when a contact tolerance is met.

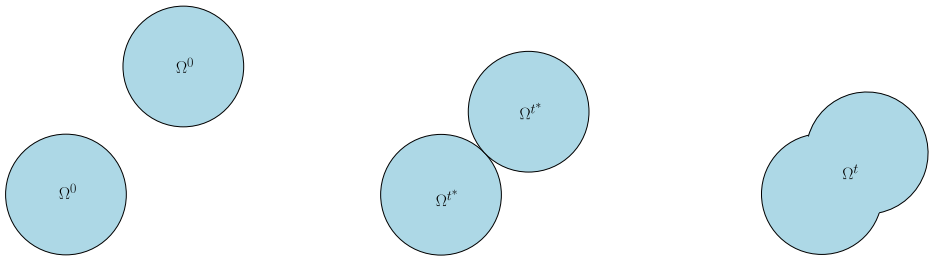

Fig. 1. A demonstration of the mesh self-intersection problem. The initial condition has disconnected support in $\tilde{\Omega}$ (left). The disconnected $\partial\Omega^t$ continues to move until some time t^* where the boundary collides with itself (centre). A new connected boundary is formed over time and the topology of Ω^t is now connected.

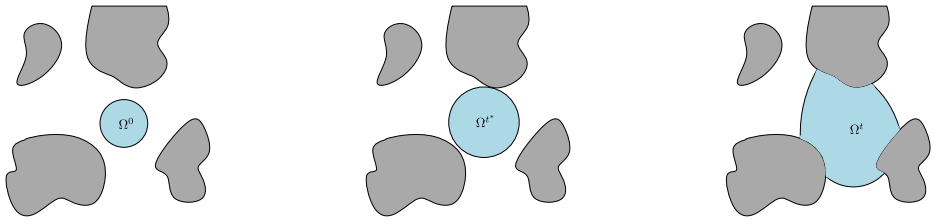

Fig. 2. A demonstration of the obstacle contact problem. The initial condition is contained in a region of obstacles (left). After some time t^* the moving boundary collides with the first obstacle (centre). A time-dependent interface now forms between the obstacles and $\partial\Omega^t$ (right).

6.2. Collision detection

For detecting mesh contact we use an adaptation of the classical *node-to-segment collision detection algorithm* [32,54,55]. We only consider boundary mesh and obstacle edges and nodes, thus ensuring that the additional computational cost is $O(N_B^2)$, where N_B is the number of boundary nodes. We consider triplets of points $\left(\mathbf{x}_1^t, \mathbf{x}_2^t, \mathbf{x}_3^t\right)$ where $\mathbf{x}_1^t$ and $\mathbf{x}_2^t$ form a time-dependent edge e^t and $\mathbf{x}_3^t$ is boundary node disconnected from e^t. This triplet is referred to as a node-to-edge pair. In our implementation, we choose to compare all boundary node and edge pairings where the node is not connected to the edge.

Given a set of linear velocities $\left(\dot{\mathbf{x}}_1, \dot{\mathbf{x}}_2, \dot{\mathbf{x}}_3\right)$, by defining the vectors

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}, \quad \dot{\mathbf{x}} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix}, \quad \dot{\mathbf{y}} = \begin{bmatrix} \dot{y}_1 \\ \dot{y}_2 \\ \dot{y}_3 \end{bmatrix}, \tag{69}$$

the contact time Δt^* between the line of e^t and $\mathbf{x}_3^t$ is given as the minimum positive root of the quadratic equation

$$0 = a(\Delta t^*)^2 + b(\Delta t^*) + c, \tag{70}$$

where the coefficients are defined by

$$a = \sum_{i=1}^{3} (\dot{\mathbf{y}} \times \dot{\mathbf{x}})_i \qquad b = \sum_{i=1}^{3} (\dot{\mathbf{y}} \times \mathbf{x} + \mathbf{y} \times \dot{\mathbf{x}})_i \qquad c = \sum_{i=1}^{3} (\mathbf{y} \times \mathbf{x})_i \,. \tag{71}$$

In practise, we choose the contact time that makes physical sense (e.g. we discard negative roots as infeasible contact time) and only admit a singular contact time value for Δt^*. In the case of no feasible contact times we set $\Delta t^* = \infty$ and when two feasible contact times are given by equation (70) we choose the minimum of the two. By solving equation (70) for a set of node-to-edge pairing, a set of contact times can be computed. In the context of the moving mesh method, each node-to-edge pair on the boundary of the mesh and (when given) obstacle mesh is considered. If any cases indicate contact, the time step is scaled down to the minimum contact time and the corresponding node-to-edge pair is marked for contact. The detection method is outlined in Algorithm 2 for a single node-to-edge pairing.

Algorithm 2: Contact Detection.

input : A node-to-edge pairing $\left(\mathbf{x}_1^t, \mathbf{x}_2^t, \mathbf{x}_3^t\right)$, a set of nodal velocities $\left(\dot{\mathbf{x}}_1^t, \dot{\mathbf{x}}_2^t, \dot{\mathbf{x}}_3^t\right)$, the current time step Δt

Solve equation (70) and set Δt^* to the minimum positive root;

for *each value of* $\Delta t^* \in \mathbb{R}$ **do**

  Compute $\mathbf{x}_3^{t+\Delta t^*}$ and $e^{t+\Delta t^*}$;

  if $\Delta t^* \in [0, \Delta t]$ *and* $\mathbf{x}_3^{t+\Delta t^*} \in e^{t+\Delta t^*}$ **then**

    Mark the node-to-edge pair for contact;

  else

    Set the contact pair to no contact;

  end

end

output: the node-to-edge contact pair, the contact time step Δt^*

6.3. Node insertion algorithm

Since the mesh allows for general polygonal element shapes, the insertion of a new node into a mesh edge can simply be performed by adding a vertex to the polygons sharing that edge. Then, a solution value associated with the new vertex must be introduced which requires an interpolation technique between the old and new global discrete spaces. In [23], an elliptic reconstruction operator is employed to preserve the quality of the discrete spatial derivative of the PDE. Instead, here we choose to preserve the polynomial component of the solution $\Pi_1^0 \rho_h$ between refinements through a redistribution of $\boldsymbol{\mu}^n$. The reason for this choice is that, by preserving the polynomial component of ρ_h, the global mass conservation of θ_h^n is maintained. This would *not* be the case if interpolation of the degrees of freedom was employed instead.

On a given element $E \in \mathcal{T}_h$ with an inserted node on the boundary, we impose that the polynomial component of the solution is preserved so that the reconstruction $\hat{\rho}_h \in \hat{V}(E)$ satisfies

$$\hat{\Pi}_1^0 \hat{\rho}_h = \Pi_1^0 \rho_h, \tag{72}$$

where $\hat{\Pi}_1^0$ denotes the projection operator constructed on the refined VEM space $\hat{V}(E)$.

The arguments in Section 4.2 can be easily modified to show that this approach conserves both locally and globally the mass of the discrete solution. When introducing a new node onto a mesh edge of a given element E under the assumption of equation (72), the local contribution to $\hat{\boldsymbol{\mu}}$ is given by,

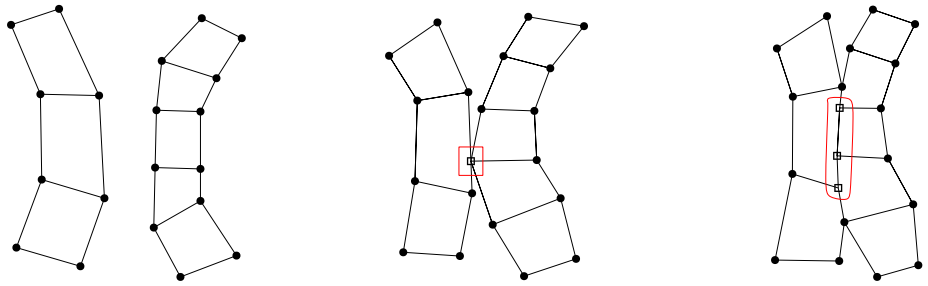

Fig. 3. A demonstration of the self intersection algorithm. A sample of elements is shown (left) where a collision is expected. Algorithm 4 is applied with contact nodes marked by a square (centre). Following subsequent mesh updates certain degrees of freedom no longer lie on the boundary (right). The highlighted degrees of freedom are now treated as internal degrees of freedom.

$$\hat{\mu}_h^E(\hat{\varphi}_i) = \int_E \Pi_1^0 \rho_h \hat{\Pi}_1^0 \hat{\varphi}_i \, d\mathbf{x} \quad i = 1, \ldots, N^{\text{dof}} + 1. \tag{73}$$

The algorithm for node insertion is given in Algorithm 3.

Algorithm 3: Node Insertion.

input : An element E, a position $\mathbf{x} \in \partial E$ to insert a node, the monitor distribution $\boldsymbol{\mu}$, the solution $\rho_h \in W_h$, the mesh $\mathcal{T}_h$
Compute $\Pi_1^0 \rho_h$ on E;
Compute the new mesh $\hat{\mathcal{T}}_h$ by inserting the node;
Compute the new monitor distribution $\hat{\boldsymbol{\mu}}$ using equation (73);
Reconstruct the new solution $\hat{\rho}_h \in \hat{W}_h$ by solving $\hat{\mathbf{M}}\hat{\boldsymbol{\rho}} = \hat{\boldsymbol{\mu}}$;
output: the new solution $\hat{\rho}_h$, the new monitor distribution $\hat{\boldsymbol{\mu}}$, the new mesh $\hat{\mathcal{T}}_h$

6.4. Self-intersection algorithm

Due once more to the VEM flexibility in element geometries, the self-intersection problem does not require any introduction of additional degrees of freedom. Instead, the local connectivity of the disconnected mesh is updated to include the new node-to-edge pairing. Then, the node insertion algorithm is applied to recompute the solution and monitor distribution. As the boundary node velocities are not arbitrarily set, small edges are likely to appear during node insertion. This has not presented any stability issues within the numerical experiments of section 7 and we expect that the method remains robust in the presence of degenerate edges [19]. The subsequent mesh velocity problem is then solved based on the updated mesh and corresponding virtual element space. A simple demonstration is provided in Fig. 3; the method at a given time step t^n is presented in Algorithm 4.

Algorithm 4: Self-intersection.

input : A mesh $\mathcal{T}_h^n$, a solution ρ_h^n, a velocity field $\mathbf{v}$, a time step size Δt.
Apply Algorithm 2 for boundary node-to-edge pairs in $\mathcal{T}_h^n$;
Update Δt;
Compute $\mathcal{T}_h^{n+1}$, $\boldsymbol{\mu}^{n+1}$, ρ_h^{n+1};
if *any contact pairs are marked* **then**
  find the element $E \in \mathcal{T}_h^{n+1}$ which contains the marked edge;
  Apply Algorithm 3;
end
Update the boundary conditions of ρ_h^{n+1};

6.5. Obstacle contact and pivot node algorithm

We denote a time-independent polygonal discretization of the set of obstacles by $\mathcal{O}_h$ and consider the mesh node-to-edge pairings of boundary nodes from $\mathcal{T}_h^n$ and edges of $\mathcal{O}_h$ and vice versa. In the case of obstacle-node to mesh-edge contact, the node insertion Algorithm 3 is applied to introduce an additional degree of freedom to the system; we refer to this new node, which is a fixed point on the obstacle geometry, as a "pivot node".

For contact between $\mathcal{T}_h^n$ and $\mathcal{O}_h$ a no-penetration condition on the nodal velocities is strongly imposed on the formulation of the potential Problem 2.3 and velocity reconstruction Problem 2.4; namely,

$$\mathbf{v} \cdot \mathbf{n} = \nabla\phi \cdot \mathbf{n} = 0. \tag{74}$$

Hence, movement tangential to the obstacle's boundary is allowed. This is except when a pivot node is introduced, in which case we constrain its velocity to zero to preserve the geometry of the interface between the domain and the obstacle. The obstacle contact algorithm is outlined in Algorithm 5.

Given that the pivot node mesh velocity is constrained to zero, it is possible for the other boundary nodes laying on the obstacle (which have experienced mesh-node to obstacle-edge contact) to pass through the pivot node. When this occurs, the connectivity of the mesh is updated to transfer the pivot node from one mesh edge to another as well as swapping the boundary node from one obstacle face to another.

Detection for pivot node collision is performed using Algorithm 2 for connected mesh boundary nodes moving from one obstacle edge to another.

A node is considered to be on $\partial\Omega_F^n$ only if both boundary edges sharing that node are in contact with the obstacle. We define a node to be "connected" if it lies on an edge of the obstacles. Degrees of freedom associated to connected nodes are constrained by equation (74) in the mesh velocity computations. If a node is connected by mesh edges to other connected boundary nodes we consider this to be an "interface" node and consequently change the boundary conditions from Dirichlet to Neumann defined in Problem 2.1 (i.e. we change the degree of freedom from $\partial\Omega_M^n$ to $\partial\Omega_F^n$). If a connected node is not also an interface node, the homogeneous boundary condition and no-penetration condition are maintained. The structure of the boundary conditions are updated once every time step.

Algorithm 5: Obstacle contact.

input : A mesh $\mathcal{T}_h^n$, a solution ρ_h^n, a velocity field $\mathbf{v}$, a time step size Δt, an obstacle mesh $\mathcal{O}_h$.
Apply Algorithm 2 for boundary nodes in $\mathcal{T}_h^n$ and boundary edges in $\mathcal{O}_h$;
Apply Algorithm 2 for boundary nodes in $\mathcal{O}_h$ and boundary edges in $\mathcal{T}_h^n$;
Select the contact pair with the smallest Δt^* and set $\Delta t = \Delta t^*$;
Compute $\mathcal{T}_h^{n+1}$, $\boldsymbol{\mu}^{n+1}$, ρ_h^{n+1};
if *obstacle node to mesh edge is marked* **then**
  find the element $E \in \mathcal{T}_h^{n+1}$ which contains the marked edge;
  Apply Algorithm 3 to introduce a pivot node $\mathbf{x}_{pivot}$;
  set $\mathbf{v} = \mathbf{0}$ at $\mathbf{x}_{pivot}$;
end
if *Mesh node to obstacle edge is marked* **then**
  Set $\mathbf{v} \cdot \mathbf{n} = 0$ at mesh node;
  Update Neumann conditions for ρ_h^{n+1};
end

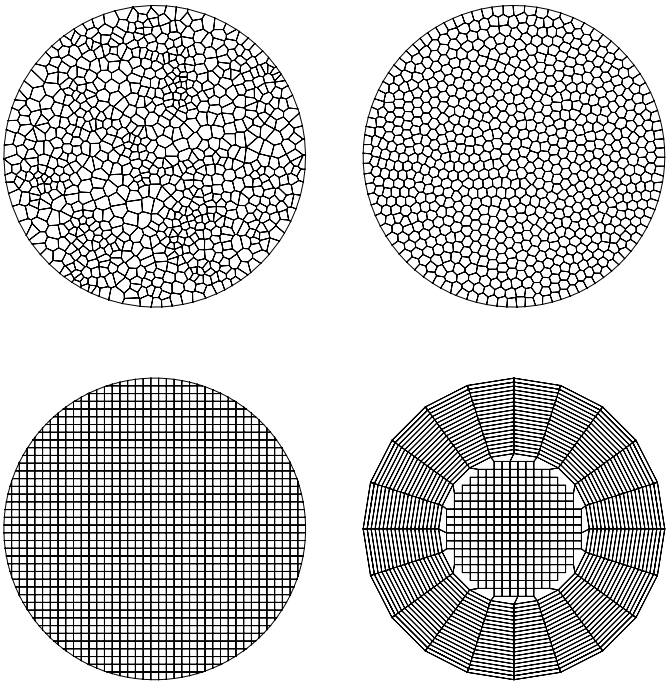

Fig. 4. Examples of each of the four mesh types used in numerical tests for a circular domain: the Voronoi Tessellation (top left), the CVT (top right), the grid mesh (bottom left) and the mixed mesh (bottom right).

When a mesh node $\mathbf{x}_i$ and a pivot node coincide (while the mesh node is moving along the obstacle boundary) the test function associated to a pivot node is chosen to satisfy $\varphi_{pivot} \equiv \varphi_i$. In other words, we duplicate the original VEM basis function and add it to the new discrete space.

Remark 7. The extension of the above algorithms to three-dimensional problems is by all means possible. We refer, for instance, to [16] for insights on the implementation of mesh intersection detection relevant to contact algorithms.

7. Numerical results

We report a series of numerical tests for the velocity-based moving mesh virtual element method proposed in the previous sections. Firstly, we present basic convergence test results using a known similarity solution of the PME Problem 2.1 for specific choices of velocity and solution recovery. Then, we investigate the effect on the solution of the node insertion algorithms described in Section 6. Finally, we present demonstrations of the contact algorithms of Section 6.

7.1. Sample meshes

We have tested four different mesh types used to subdivide the initial domain; representative examples of each are shown in Fig. 4. The first mesh is the Voronoi Tessellation produced by randomly sampling mesh seeds in the domain [44,4]. The second mesh is a Centroidal Voronoi Tessellation (CVT) produced by the Lloyd algorithm which smooths a given Voronoi tessellation such that the generator points are the barycentric coordinates for each polygon [28]. The MATLAB package PolyMesher [47] was used to produce these two mesh types. The third mesh is constructed by overlaying the domain with a grid of uniform squares and cutting the mesh along the boundary. The last mesh type is a mixture of uniform Cartesian and polar tessellations. Note that the first three initial mesh types may present arbitrarily small edges and, moreover, arbitrarily small elements may appear near the boundary in the grid mesh type, as such potentially contradicting the mesh regularity assumptions stated in Section 3.1. In this respect, we note that the VEM is known to be quite robust, as we have also witnessed. We refer to the mesh size in each case as the largest element diameter in $\mathcal{T}_h^0$.

7.2. The PME similarity solution

There exists a family of radially symmetric solutions on a given initial circular domain of radius r_0 for the Problem 2.1 defined in [42] and given by

$$\rho(r,t) = \begin{cases} \frac{1}{\lambda(t)^d}\left(1 - \left(\frac{r}{r_0\lambda(t)}\right)^2\right)^{\frac{1}{m}} & |r| \le r_0\lambda(t) \\ 0 & \text{otherwise} \end{cases}, \tag{75}$$

where d is the spatial dimension, r_0 is the initial radius, and

$$\lambda(t) = \left(\frac{t}{t_0}\right)^{\frac{1}{2+dm}}, \quad t_0 = \frac{r_0^2 m}{2(2+dm)}.$$

Because of the nature of Equation (75), the solution is expected to have finite slope normal to the moving boundary for $m \le 1$ whilst, for $m > 1$, the solution presents an infinite slope normal to the boundary. Further properties of this analytical solution are discussed in [43,7].

7.3. Error computation

The numerical error is computed for both the solution and mesh by generalising the discrete approximations from [7]. An l^1 solution error is given by,

$$\|\rho^n - \rho_h^n\|_{sol} = \frac{1}{N^{\text{dof}}} \sum_{i=1}^{N^{\text{dof}}} \left|\text{dof}_i(\rho^n) - \text{dof}_i(\rho_h^n)\right|, \tag{76}$$

while, for the mesh error, an l^1 norm is considered for the radial distance $r(t)$ from the boundary of the mesh to the origin; thus

$$\|r^n - r_h^n\|_{mesh} = \frac{1}{N^B} \sum_{i=1}^{N^B} \left|R_i^n - r_0\lambda(t^n)\right|. \tag{77}$$

Here, N^B denotes the number of boundary nodes in the mesh and R_i^n denotes the radial distance from the origin of the i-th boundary node at time t^n. A uniform Δt that is small enough to ensure numerical stability is set for each initial sample mesh. Note that the meshes used in these numerical tests are not hierarchical. Furthermore, each Voronoi mesh is generated independently from randomly generated seeds. For each reduction of initial mesh size by a factor of 2, the time-step Δt is reduced by a factor of 4 to ensure numerical stability. By reducing the time-step size by this factor we also expect that the temporal error to be $O(h^2)$ when using the Forward Euler method for the refinement path of the four mesh types.

7.4. Convergence test

In this first convergence test the solution of the similarity solution for $m = 1, d = 2$, and $r_0 = 0.5$ is compared against the numerical solution for $t = t_0 + T$. The method is tested on each of the four mesh types for a circular domain. The time step sizes for the coarsest meshes are chosen according to

$$\Delta t = \frac{1}{250} h_{mean}^2, \tag{78}$$

where h_{mean} is the average element diameter of the initial mesh $\mathcal{T}_h^0$. In each mesh case we observe that the initial time-step size is approximately 10^{-4}. From the coarse-mesh time-step sizes we reduce Δt by a factor of 4 each time the mesh is refined, which corresponds to the mesh size approximately halving with each refinement. In choosing the

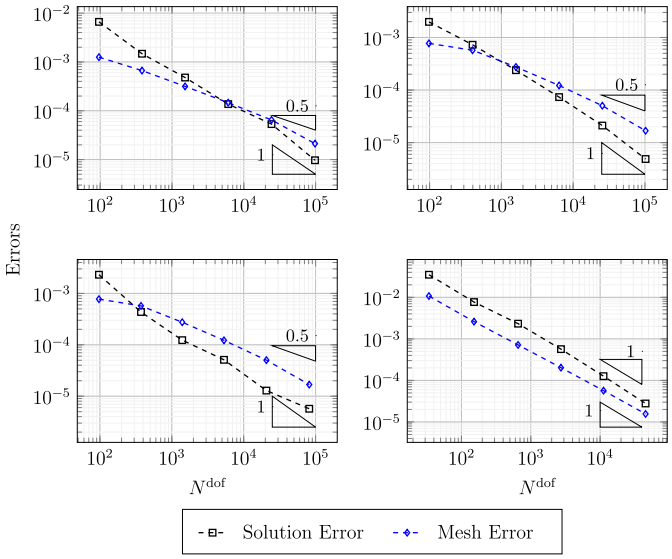

Fig. 5. PME similarity solution with $m = 1$: the l^1 solution and mesh errors (76) and (77), respectively, at time $T = 0.01$ for each mesh type: Voronoi (top left), CVT (top right), Grid (bottom left), Mixed (bottom right).

time step sizes we made conservative choices such that the numerical method was stable and presented the expected orders of convergence. A more robust approach would be to use adaptive time-stepping schemes but in this work we present convergence results for a uniform reduction in the time-step. As shown in Fig. 5, second order accuracy is observed for the solution error for all mesh cases when $T = 0.01$. In the case of the Voronoi and Grid meshes, the empirical order of convergence (EOC) is less smooth compared to the CVT and mixed mesh types. This is most likely due to the weaker shape regularity of elements in these mesh types, but further studies are required. The mesh error EOC appears to have a long pre-asymptotic regime: the EOC grows monotonically towards the expected rate in all cases with the final computed values ranging between 1.55 (Voronoi) and 1.83 (mixed). This is consistent with finite element approximations of Darcy flow which observed the order of convergence of the velocity field to be lower than that of the pressure field [20,39]. Conservation of mass in the numerical solution is observed up to machine precision in all test cases.

Setting $\boldsymbol{\mu} = \mathbf{0}$ produces similar results to those reported in Fig. 5. This is referred to as a "direct recovery" approach in the literature, see [7]. For brevity, the corresponding results are not presented here.

When $m > 1$, the solution of the PME presents a low regularity as the gradient is unbounded at the moving boundary [51]. Unsurprisingly, applying the moving mesh VEM to the PME with $m = 2$, we found that the method remains robust but only attains first order accuracy (results not shown), in line with what already observed for discretizations based on the FEM [6]. We expect that appropriately grading the mesh in the vicinity of the moving boundary can be used to improve the order of convergence as demonstrated in the FEM case in [6].

7.5. Node insertion convergence test

Our next numerical experiment considers the case of the one-dimensional PME extended in the x direction to a two-dimensional problem. This experiment has two interesting features. Firstly, the initial domain is geometrically exact ($\Omega^0 \equiv \Omega_h^0$) unlike the circular meshes. Secondly, it allows us to test the obstacle contact and node insertion algorithm numerically against a known analytical solution derived from the one-dimensional case of equation (75).

This is obtained by considering once again Equation (75) with the values $m = 1, d = 1, r_0 = 0.5$, and $r = y$ on the initial domain given by $\Omega^0 = [-0.5, 0.5]^2$ with initial condition

$$\rho(\mathbf{x}, 0) = 1 - 4y^2.$$

The mesh is connected to two vertical planes at $x = -0.5$ and $x = 0.5$ with a no-penetration condition strongly imposed in the x direction; namely,

$$\dot{x} = 0 \quad \text{when } |x| = 1/2.$$

Solution snapshots at time $T = 0.1$ are shown in Fig. 6 for the CVT mesh type. The mesh error is exclusively computed on the top and bottom faces of the rectangular domain.

In this test we only focus our attention on the CVT mesh type. We test the accuracy of the node insertion algorithm by including a discretization of the two planes into intervals in the y direction. In reference to the fixed domain PME (25) we define $\tilde{\Omega} := [-1, 1]^2$ and discretize the boundary $\partial\tilde{\Omega}$ into N uniformly spaced intervals to construct an N-gon. Vertices are then removed that intersect Ω^0. This results in uniformly discretized intervals along both contact planes.

For the discretization of the contact planes N is set to an initial value of 32 and is doubled with each mesh refinement. Convergence results are reported in Fig. 7. Here we observe second order accuracy in both the solution and mesh error. Also the mass is conserved, by design, up to machine precision throughout each test.

7.6. Contact demonstrations

We finally present two demonstrations of the node insertion algorithms for Problem 2.1 in challenging scenarios without known analytical solution.

To ensure the quality of the mesh is preserved we introduce an alternative velocity field on the interior of the moving mesh based on the ALE approach. First, the method outlined in Section 5 is applied to approximate the Lagrangian boundary velocity $\mathbf{v}$. The mesh velocity $\mathbf{v}$ for the internal node movement is then replaced by $\tilde{\mathbf{v}}$, the harmonic extension of the Lagrangian boundary velocity: as such it is the solution to the problem

$$-\Delta\tilde{\mathbf{v}} = \mathbf{0} \tag{79}$$

$$\tilde{\mathbf{v}}|_{\partial\Omega^t} = \mathbf{v}. \tag{80}$$

In a VEM framework this alternative mesh velocity is computed by solving a standard Poisson's equation with Dirichlet boundary conditions [13]. Other choices for the interior mesh velocities include a modified monitor function $\mathbb{M}(\rho)$ [38], pseudo-elastic, and biharmonic formulations [45]. We remark that the quality of the mesh will still deteriorate over time. The purpose of these examples is to demonstrate the application of the node insertion algorithms. Optimising the choice of ALE velocity is left for future investigation.

In the first demonstration, we consider an initial condition of the PME that has a disconnected support such that self-intersection is expected to occur. The initial condition is given by

$$\rho(\mathbf{x}, 0) = \begin{cases} 1 - 4r_1^2 & r_1 = |\mathbf{x} - (-0.8, 0)|,\ r_1 \le 1/2, \\ 1 - 4r_2^2 & r_2 = |\mathbf{x} - (0.8, 0)|,\ r_2 \le 1/2, \\ 0 & \text{otherwise.} \end{cases} \tag{81}$$

An illustrative example of such initial condition is given in Fig. 8 (top-left plot). The standard method is applied to simulate the PME for $m = 1$ with the contact detection Algorithm 2 applied at every time level to check for collision between elements. When contact occurs, Algorithm 4 is used to update the monitor distribution whilst the Dirichlet boundary degrees of freedom are flagged as interior degrees of freedom as the mesh connectivity evolves. Snapshots of the solution evolving over time are reported in Fig. 8. The behaviour of the PME solution over time is in agreement with fixed mesh finite element approximations of this problem and similar benchmark tests performed for the PME in [43].

To demonstrate the obstacle contact algorithm we consider again the initial condition given by Equation (75). A set of obstacles are added to

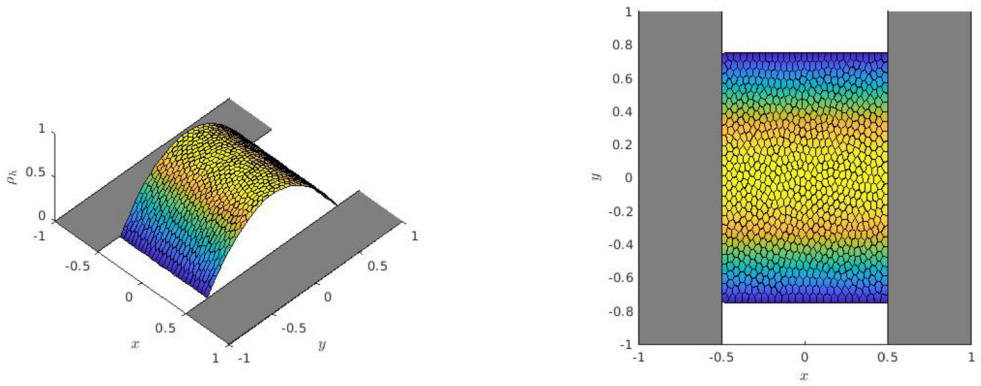

Fig. 6. A solution snapshot at time $T = 0.1$ for a CVT mesh with 800 elements (right).

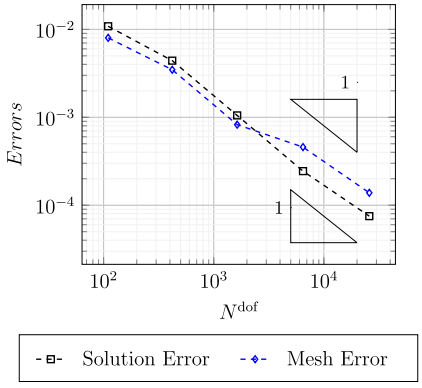

Fig. 7. Node insertion convergence test on a 1D-type PME similarity solution with $m = 1$: the l^1 solution and mesh errors (76) and (77), respectively, at time $T = 0.1$.

the computational domain in the shape of circles with random radii and centres. Each circle is approximated as a uniform polygon of comparable accuracy to the initial mesh. We tested the moving mesh VEM starting with a CVT type mesh made of 800 elements discretising the support of the initial solution. A few snapshots of the numerical solution are shown in Fig. 9. Pivot nodes are inserted and removed along the contact. The benefits of the VEM are highlighted in this case as the numerical simulation remains robust to elements with arbitrarily small edges (which appear, grow and shrink as the domain boundary extends over the surfaces of the obstacles) and elements that become strongly anisotropic around the contact interface. Furthermore, the degrees of freedom added to the problem are *only* the vertices required to define the contact interface between the moving mesh and obstacles, minimising the additional degrees of freedom required. In the case of a FEM, we would expect the method to be unstable with respect to vanishing edges and anisotropic elements. These may be generated, for example, as mesh nodes move along the obstacle boundary and pass over the vertices which define the obstacle geometry. A thorough numerical comparison of the stability of the FEM against the VEM is left for future studies. Mesh degeneration occurs after $T = 0.2$ and thus the simulation had to be terminated. In this case, and similar to finite element methods, a remeshing approach would rectify this issue.

7.7. A fourth-order problem

To demonstrate the extensibility of the moving mesh VEM we consider the following fourth-order nonlinear diffusion problem used as a benchmark for the original moving mesh method [7,6], for which the whole of the boundary $\partial\Omega^t$ is free to move, and the differential operator is given by $\mathcal{L}\rho = -\nabla\cdot(\rho^m\nabla\Delta\rho)$. In this work we choose $m = 1$, for which there is a simple similarity solution, defined below. The resulting time-dependent equation $\partial\rho/\partial t = \mathcal{L}\rho$ is complemented, at the free boundary, with two conditions on ρ, namely $\rho = \nabla\rho\cdot\mathbf{n} = 0$ plus the kinematic condition $\rho\mathbf{v}\cdot\mathbf{n} = \rho\nabla\Delta\rho\cdot\mathbf{n}$, which is used to determine the boundary velocity $\mathbf{v}$.

In view of its numerical solution, we re-write the fourth-order problem as a coupled system of second-order PDEs by introducing a pressure term $p = -\Delta\rho$. The problem then reads as: *find* $\rho = \rho(\mathbf{x}, t)$ *such that* $\rho(\mathbf{x}, 0) = \rho^0(\mathbf{x})$ *for* $\mathbf{x}\in\Omega^0$ *and, for all* $t\in(0,T]$,

$$\frac{\partial\rho}{\partial t} = \nabla\cdot(\rho\nabla p) \qquad \mathbf{x}\in\Omega^t, \tag{82}$$

$$p = -\Delta\rho \qquad \mathbf{x}\in\Omega^t, \tag{83}$$

$$\rho = \nabla\rho\cdot\mathbf{n} = 0 \qquad \mathbf{x}\in\partial\Omega^t, \tag{84}$$

$$\rho\mathbf{v}\cdot\mathbf{n} = -\rho\nabla p\cdot\mathbf{n} \qquad \mathbf{x}\in\partial\Omega^t. \tag{85}$$

This problem is structurally very similar to the porous medium equation problem and the moving mesh algorithm remains mostly unchanged. The main addition is an intermediate step which provides the pressure by discretising the weak form of (83), namely: *given* $\rho\in H^1(\Omega^t)$, *find* $p\in H^1(\Omega^t)$ *such that*

$$\int_{\Omega^t} pw\,d\mathbf{x} = \int_{\Omega^t}\nabla\rho\cdot\nabla w\,d\mathbf{x} \qquad \forall w\in H^1(\Omega^t). \tag{86}$$

This is discretised using once again the VEM applied to the problem within each time-step t^n: given $\rho_h\in V_h^n$ find $p_h\in V_h^n$ such that

$$m_h(p_h, w_h) = \sum_{E\in\mathcal{T}_h^n}\int_E \Pi_0^0\nabla\rho_h\cdot\Pi_0^0\nabla w_h\,d\mathbf{x} \qquad \forall w_h\in V_h^n, \tag{87}$$

where $m_h(\cdot,\cdot)$ is defined by Equations (60) and (61).

In Problems 2.3 and 2.5 the right-hand side terms are also modified for this problem to give, respectively,

$$d(w) = -\int_{\Omega^t}\rho\nabla p\cdot\nabla w\,d\mathbf{x}, \tag{88}$$

$$\dot\mu^t(w) = \int_{\Omega^t} -\rho\nabla w\cdot\{\nabla p + \mathbf{v}\}\,d\mathbf{x} \qquad \forall w\in H^1(\Omega^t). \tag{89}$$

These equations are approximated using the VEM discretizations (50) and (57), as described in Sections 3 and 4 for the approximation of the corresponding integrals for the porous medium equation. These are computed at time t^n as follows:

$$d_h(w_h) = -\sum_{E\in\mathcal{T}_h^n}\int_E (\Pi_1^0\rho_h)_0\,\Pi_0^0\nabla p_h\cdot\Pi_0^0\nabla w_h\,d\mathbf{x}, \tag{90}$$

$$\dot\mu_h(w_h) = -\sum_{E\in\mathcal{T}_h^n}\int_E \Pi_1^0\rho_h\,\Pi_0^0\nabla w_h\cdot\left\{\Pi_0^0\nabla p_h + \Pi_1^0\mathbf{v}_h\right\}\,d\mathbf{x}, \tag{91}$$

$\forall w_h\in V_h^n$.

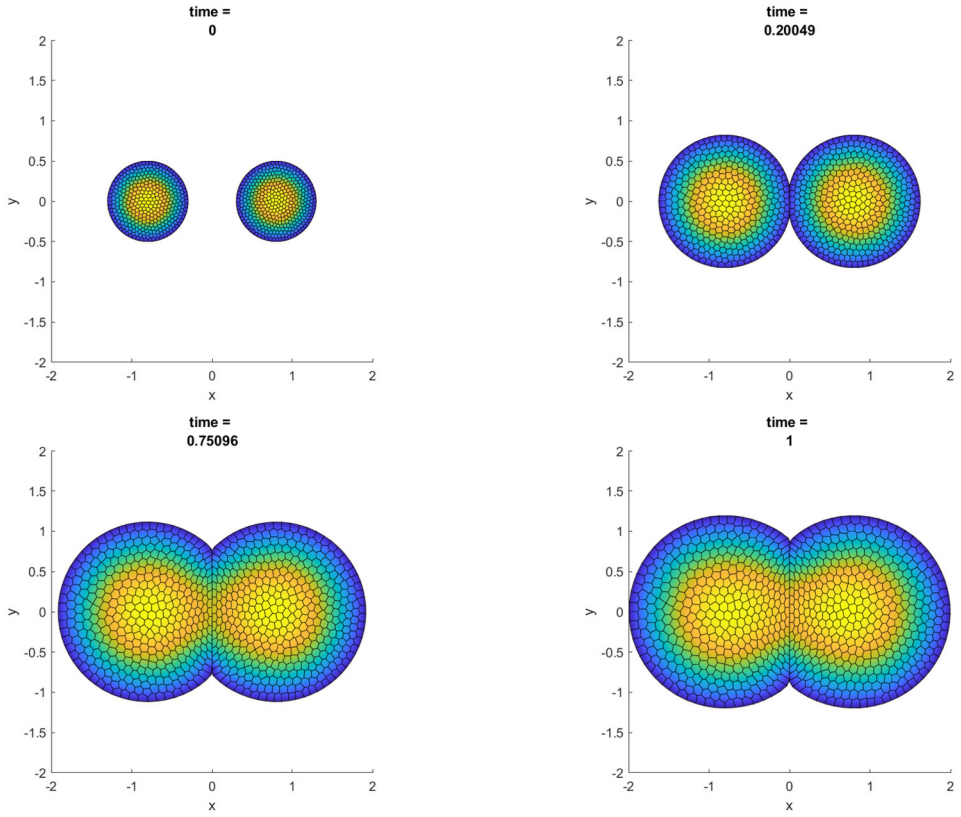

Fig. 8. Self-intersection demonstration: snapshots of the moving mesh VEM solution at $t = 0$ (top left), $t = 0.20049$ (top right), $t = 0.75096$ (bottom left), and $t = 1$ (bottom right). A CVT type mesh with 800 elements was used to initialise the mesh at $t = 0$.

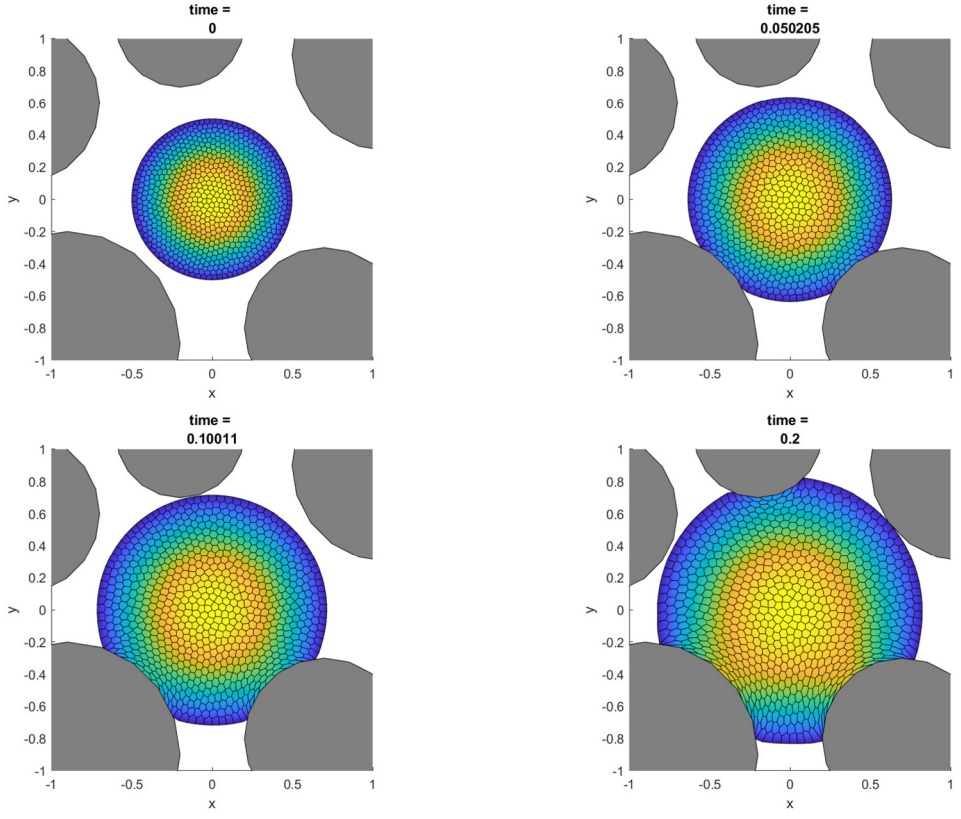

Fig. 9. Obstacle contact demonstration: snapshots of the moving mesh VEM solution at $T = 0$ (top left), $T = 0.050205$ (top right), $T = 0.10011$ (bottom left) and $T = 0.2$ (bottom right). A CVT type mesh with 800 elements was used to initialise the mesh at $t = 0$.

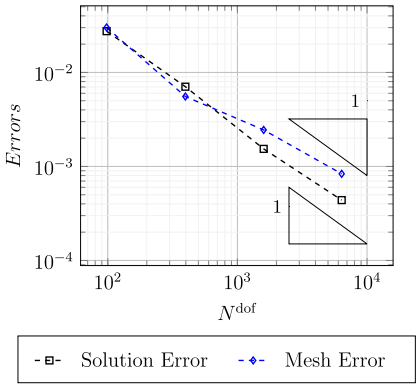

Fig. 10. Convergence test for the fourth-order diffusion problem: the l^1 solution (ρ) and mesh errors (76) and (77), respectively, at time $T = 0.01$ using the CVT mesh type.

In view of assessing numerically the resulting moving mesh VEM, we recall that this fourth-order nonlinear diffusion problem has a radially symmetric similarity solution given by

$$\rho(r,t) = \begin{cases} At^{\beta}U_0(1-\eta^2)^2 & |r| \le A^{\frac{1}{4}}t^{\delta} \\ 0 & \text{otherwise} \end{cases}, \tag{92}$$

where

$$\eta = \frac{r}{A^{\frac{1}{4}}t^{\delta}}, \qquad \delta = \frac{1}{4+d}, \qquad \beta = 4\delta - 1, \qquad A = U_0^{-4\delta}. \tag{93}$$

Setting $d = 2$, we fix $U_0 = 1/192$ so that $\rho(0,t_0) = 1$ and $t_0 = 1/192$ is specified so that the initial radius is equal to 1.

The VEM is tested on the same sequence of CVT-type meshes used in Section 7.4, with the same coarse-mesh time-step size of 10^{-4} and a reduction by a factor of 4 each time the mesh is refined. Fig. 10 shows that second-order accuracy is again attained for both the solution (ρ) and mesh errors. Similar to the PME, the mass is conserved exactly at each time step.

8. Conclusions

In this paper we have combined, for the first time, a velocity-based moving mesh method with a virtual element method. This was achieved by extending the FEM formulation of the moving mesh method to a linear virtual element scheme on polygonal meshes. Numerical tests for the porous medium equation and a fourth-order diffusion problem show that the proposed method obtains the same orders of accuracy as the original finite element approach. In fact, given that the linear VEM reduces to the linear FEM on triangular elements, our approach provides a natural extension of moving mesh FEM to polygonal mesh settings. Demonstrations of node insertion algorithms suggest that the virtual element method offers practical extensions to more complex problems. In particular, this work shows that it is straightforward to deal with situations where the moving boundary meets fixed obstacles or merges with other moving boundaries.

The VEM approach provides a flexible discretisation framework for adaptive moving mesh methods. For instance, VEM with curved elements are being developed with a level of generality out of reach for standard FEM, see e.g.. [11,37,3,29]. As such, the VEM is more suitable for the generalisation of moving mesh approaches in the higher-order setting, including for the solution of challenging problems such as phase-field, fluid-structure interaction, and moving surface PDEs [7], including in three-dimensions [21,37,26]. We remark that these are largely open problems also for the more standard moving mesh finite element method. Extensions of the moving mesh VEM in these directions will be the subject of future works.

Data availability

Data will be made available on request.

Acknowledgements

The corresponding author was supported by EPSRC doctoral training grants EP/N50970X/1 and EP/R513283/1.

References

[1] B. Ahmad, A. Alsaedi, F. Brezzi, L.D. Marini, A. Russo, Equivalent projectors for virtual element methods, Comput. Math. Appl. 66 (3) (2013) 376–391.
[2] P. Antonietti, M. Verani, C. Vergara, S. Zonca, Numerical solution of fluid-structure interaction problems by means of a high order discontinuous Galerkin method on polygonal grids, Finite Elem. Anal. Des. 159 (2019) 1–14.
[3] E. Artioli, L. Beirão da Veiga, M. Verani, An adaptive curved virtual element method for the statistical homogenization of random fibre-reinforced composites, Finite Elem. Anal. Des. 177 (2020) 103418.
[4] F. Aurenhammer, Voronoi diagrams—a survey of a fundamental geometric data structure, ACM Comput. Surv. 23 (3) (1991) 345–405.
[5] M. Baines, Moving Finite Elements, Oxford University Press, Inc., 1994.
[6] M. Baines, M. Hubbard, P. Jimack, A moving mesh finite element algorithm for the adaptive solution of time-dependent partial differential equations with moving boundaries, Appl. Numer. Math. 54 (3–4) (2005) 450–469.
[7] M. Baines, M. Hubbard, P. Jimack, Velocity-based moving mesh methods for nonlinear partial differential equations, Commun. Comput. Phys. 10 (3) (2011) 509–576.
[8] M. Baines, M. Hubbard, P. Jimack, A. Jones, Scale-invariant moving finite elements for nonlinear partial differential equations in two dimensions, Appl. Numer. Math. 56 (2) (2006) 230–252.
[9] M. Baines, M. Hubbard, P. Jimack, R. Mahmood, A moving-mesh finite element method and its application to the numerical solution of phase-change problems, Commun. Comput. Phys. 6 (3) (2009) 595–624.
[10] G. Beckett, J. Mackenzie, M. Robertson, A moving mesh finite element method for the solution of two-dimensional Stefan problems, J. Comput. Phys. 168 (2) (2001) 500–518.
[11] L. Beirão da Veiga, F. Brezzi, L. Marini, A. Russo, Polynomial preserving virtual elements with curved edges, Math. Models Methods Appl. Sci. 30 (8) (2020) 1555–1590.
[12] L. Beirão da Veiga, F. Brezzi, A. Cangiani, G. Manzini, L. Marini, A. Russo, Basic principles of virtual element methods, Math. Models Methods Appl. Sci. 23 (01) (2013) 199–214.
[13] L. Beirão da Veiga, F. Brezzi, L. Marini, A. Russo, The Hitchhiker's guide to the virtual element method, Math. Models Methods Appl. Sci. 24 (08) (2014) 1541–1573.
[14] L. Beirão da Veiga, F. Brezzi, L. Marini, A. Russo, Virtual element method for general second-order elliptic problems on polygonal meshes, Math. Models Methods Appl. Sci. 26 (04) (2016) 729–750.
[15] L. Beirão da Veiga, C. Lovadina, A. Russo, Stability analysis for the virtual element method, Math. Models Methods Appl. Sci. 27 (13) (2017) 2557–2594.
[16] D. Boffi, A. Cangiani, M. Feder, L. Gastaldi, L. Heltai, A comparison of non-matching techniques for the finite element approximation of interface problems, arXiv preprint arXiv:2304.11908, 2023.
[17] A. Bonito, I. Kyza, R. Nochetto, Time-discrete higher order ALE formulations: a priori error analysis, Numer. Math. 125 (2) (2013) 225–257.
[18] A. Bonito, I. Kyza, R. Nochetto, Time-discrete higher-order ale formulations: stability, SIAM J. Numer. Anal. 51 (1) (2013) 577–604.
[19] S. Brenner, L. Sung, Virtual element methods on meshes with small edges or faces, Math. Models Methods Appl. Sci. 28 (07) (2018) 1291–1336.
[20] F. Brezzi, T. Hughes, L. Marini, A. Masud, Mixed discontinuous Galerkin methods for Darcy flow, J. Sci. Comput. 22 (1) (2005) 119–145.
[21] F. Brezzi, K. Lipnikov, M. Shashkov, Convergence of mimetic finite difference method for diffusion problems on polyhedral meshes with curved faces, Math. Models Methods Appl. Sci. 16 (2) (2006) 275–297.
[22] C. Budd, W. Huang, R. Russell, Moving mesh methods for problems with blow-up, SIAM J. Sci. Comput. 17 (2) (1996) 305–327.
[23] A. Cangiani, E. Georgoulis, O. Sutton, Adaptive non-hierarchical Galerkin methods for parabolic problems with application to moving mesh and virtual element methods, Math. Models Methods Appl. Sci. 31 (4) (2021) 711–751.
[24] A. Cangiani, G. Manzini, O. Sutton, Conforming and nonconforming virtual element methods for elliptic problems, IMA J. Numer. Anal. 37 (3) (2017) 1317–1354.
[25] W. Cao, W. Huang, R. Russell, A moving mesh method based on the geometric conservation law, SIAM J. Sci. Comput. 24 (1) (2002) 118–142.
[26] F. Dassi, A. Fumagalli, A. Scotti, G. Vacca, Bend 3d mixed virtual element method for Darcy problems, Comput. Math. Appl. 119 (2022) 1–12.
[27] J. Donea, A. Huerta, J. Ponthot, A. Rodríguez-Ferran, Arbitrary Lagrangian-Eulerian methods, in: Encyclopedia of Computational Mechanics, second edition, 2017, pp. 1–23.
[28] Q. Du, M. Emelianenko, L. Ju, Convergence of the Lloyd algorithm for computing centroidal Voronoi tessellations, SIAM J. Numer. Anal. 44 (1) (2006) 102–119.

[29] J. Ferguson, J. Kópházi, M. Eaton, Nurbs enhanced virtual element methods for the spatial discretization of the multigroup neutron diffusion equation on curvilinear polygonal meshes, J. Comput. Theor. Transp. 51 (4) (2022) 145–204.
[30] E. Gaburro, W. Boscheri, S. Chiocchetti, C. Klingenberg, V. Springel, M. Dumbser, High order direct arbitrary-Lagrangian-Eulerian schemes on moving Voronoi meshes with topology changes, J. Comput. Phys. 407 (2020) 109167.
[31] R. Gelinas, S. Doss, K. Miller, The moving finite element method: applications to general partial differential equations with multiple large gradients, J. Comput. Phys. 40 (1) (1981) 202–249.
[32] J. Hallquist, G. Goudreau, D. Benson, Sliding interfaces with contact-impact in large-scale Lagrangian computations, Comput. Methods Appl. Mech. Eng. 51 (1–3) (1985) 107–137.
[33] W. Huang, R. Russell, Adaptive Moving Mesh Methods, vol. 174, Springer Science & Business Media, 2010.
[34] W. Huang, Y. Wang, Anisotropic mesh quality measures and adaptation for polygonal meshes, J. Comput. Phys. (2020) 109368.
[35] M. Hubbard, M. Baines, P. Jimack, Consistent Dirichlet boundary conditions for numerical solution of moving boundary problems, Appl. Numer. Math. 59 (6) (2009) 1337–1353.
[36] P. Jimack, A. Wathen, Temporal derivatives in the finite-element method on continuously deforming grids, SIAM J. Numer. Anal. 28 (4) (1991) 990–1003.
[37] K. Lipnikov, N. Morgan, A high-order conservative remap for discontinuous Galerkin schemes on curvilinear polygonal meshes, J. Comput. Phys. 399 (2019) 108931.
[38] R. Marlow, M. Hubbard, P. Jimack, Moving mesh methods for solving parabolic partial differential equations, Comput. Fluids 46 (1) (2011) 353–361.
[39] A. Masud, T. Hughes, A stabilized mixed finite element method for Darcy flow, Comput. Methods Appl. Mech. Eng. 191 (39–40) (2002) 4341–4370.
[40] A. Mazzia, M. Ferronato, P. Teatini, C. Zoccarato, Virtual element method for the numerical simulation of long-term dynamics of transitional environments, J. Comput. Phys. (2020) 109235.
[41] K. Miller, R. Miller, Moving finite elements. I, SIAM J. Numer. Anal. 18 (6) (1981) 1019–1032.
[42] J. Murray, Mathematical Biology: 1. An Introduction, 3rd edition, Springer-Verlag, New York, 2002.
[43] C. Ngo, W. Huang, A study on moving mesh finite element solution of the porous medium equation, J. Comput. Phys. 331 (2017) 357–380.
[44] A. Okabe, B. Boots, K. Sugihara, S. Chiu, Spatial Tessellations: Concepts and Applications of Voronoi Diagrams, vol. 501, John Wiley & Sons, 2009.
[45] T. Richter, Fluid-Structure Interactions, Lecture Notes in Computational Science and Engineering, vol. 118, Springer International Publishing, Cham, 2017.
[46] O. Sutton, The virtual element method in 50 lines of Matlab, Numer. Algorithms 75 (4) (2017) 1141–1159.
[47] C. Talischi, G. Paulino, A. Pereira, I. Menezes, Polymesher: a general-purpose mesh generator for polygonal elements written in Matlab, Struct. Multidiscip. Optim. 45 (3) (2012) 309–328.
[48] G. Vacca, Virtual element methods for hyperbolic problems on polygonal meshes, Comput. Math. Appl. 74 (5) (2017) 882–898.
[49] G. Vacca, An h-1-conforming virtual element for Darcy and Brinkman equations, Math. Models Methods Appl. Sci. 28 (1) (Jan 2018) 159–194.
[50] G. Vacca, L. Beirão da Veiga, Virtual element methods for parabolic problems on polygonal meshes, Numer. Methods Partial Differ. Equ. 31 (6) (2015) 2110–2134.
[51] J. Vázquez, The Porous Medium Equation: Mathematical Theory, Oxford University Press, 2007.
[52] G. Wang, F. Wang, L. Chen, Y. He, A divergence free weak virtual element method for the Stokes-Darcy problem on general meshes, Comput. Methods Appl. Mech. Eng. 344 (2019) 998.
[53] P. Wesseling, Principles of Computational Fluid Dynamics, vol. 29, Springer Science & Business Media, 2009.
[54] P. Wriggers, T. Laursen, Computational Contact Mechanics, vol. 2, Springer, 2006.
[55] P. Wriggers, W. Rust, B. Reddy, A virtual element method for contact, Comput. Mech. 58 (6) (2016) 1039–1050.
[56] J. Zhao, B. Zhang, S. Mao, S. Chen, The nonconforming virtual element method for the Darcy-Stokes problem, Comput. Methods Appl. Mech. Eng. 370 (2020).