Moving mesh methods for solving parabolic partial differential equations

R. Marlow[a,*], M. E. Hubbard[a], P. K. Jimack[a]

[a] *School of Computing, University of Leeds, UK*

Abstract

A new adaptive method is described for solving non-linear parabolic partial differential equations with moving boundaries, using a moving mesh with continuous finite elements. The evolution of the mesh within the interior of the spatial domain is based upon conserving the distribution of a chosen monitor function across the domain throughout time, where the initial distribution is selected based upon the given initial data. The mesh movement at the boundary is governed by a second monitor function, which may or may not be the same as that used to drive the interior mesh movement. The method is described in detail and a selection of computational examples are presented using different monitor functions applied to the porous medium equation (PME) in one and two space dimensions.

Keywords:

Moving meshes, finite elements, moving boundaries, parabolic PDEs, porous medium equation

*Corresponding author.

Email address: corm@leeds.ac.uk (R. Marlow)

1. Introduction

Moving mesh methods have been shown to have great potential in solving problems with moving fronts and boundaries, problems involving phenomena such as blow-up and problems in a wide range of applications for which non-stationary features need to be tracked in time (see, for example, [1] and references therein). They are, however, not as widely used as other adaptive techniques in computational fluid dynamics (CFD), perhaps because there remain many outstanding questions over accuracy and reliability. In this context, we explore here a new method that uses the maintenance of a distribution of a monitor function in order to define the velocities of the nodes of the mesh in a Lagrangian co-ordinate system, and so drive the mesh movement. This technique is a natural generalization of [2], which is based on maintaining the distribution of a specific monitor function, namely the mass monitor.

Consider a parabolic partial differential equation (PDE) of the form

$$u_t = Lu, \tag{1}$$

whose solution lies in a time-dependent domain $\Omega(t)$. Here L is a purely spatial operator of second order. We focus upon the use of low order, continuous finite elements, on a moving mesh (with support $\tilde{\Omega}(t)$ say) which approximates $\Omega(t)$. The mesh movement in the interior of $\tilde{\Omega}(t)$ is to be driven by a monitor function $m(u)$, and our first aim in this study is to maintain the initial distribution of this monitor function as the domain, the solution and the mesh evolve. In the case where the initial distribution is an equidistribution of $m(u)$, the goal would therefore be to maintain an equidistribution for all

time. Starting with an initial condition, $u(x, t_0) = u_0(x)$, for the solution of (1) the initial monitor total may be evaluated:

$$\theta_0 = \int_{\tilde{\Omega}(t_0)} m(u_0)\, d\Omega\,, \tag{2}$$

on an initial mesh with support $\tilde{\Omega}(t_0)$. We may represent the relative monitor distribution across this mesh via the set $\{c_i\}$, defined by

$$c_i = \frac{1}{\theta_0} \int_{\tilde{\Omega}(t_0)} w_i m(u_0)\, d\Omega, \quad i = 1, 2 \ldots N, \tag{3}$$

where w_i are N piecewise linear finite element test functions, which form a partition of unity [2]. Hence, our first aim translates to keeping the right-hand side of (3) constant when evaluated at $t > t_0$ as the solution and the mesh, and so the spatial domain $\tilde{\Omega}(t)$, evolve with time. For the remainder of this paper we will always work with the discrete domain $\tilde{\Omega}(t)$ but will drop the tilde notation for convenience.

The desire for the c_i to remain constant provides us with a mechanism for determining the mesh velocities (i.e. the velocities of the N nodes of the mesh) as the solution evolves. In order to achieve this, we first define,

$$\theta(t) = \int_{\Omega(t)} m(u)\, d\Omega, \tag{4}$$

and then, based upon (3), we seek to impose that, for all time,

$$c_i \theta(t) = \int_{\Omega(t)} w_i m(u)\, d\Omega, \quad i = 1, 2 \ldots N. \tag{5}$$

If we assume each c_i does indeed remain constant in time then

$$c_i \dot{\theta}(t) = \frac{d}{dt} \int_{\Omega(t)} w_i m(u)\, d\Omega\,, \tag{6}$$

with the values of the c_i given by (3). Developing this equation by bringing the time derivative into the integral (using Reynolds' Transport Theorem),

$$c_i\dot{\theta}(t) = \int_{\Omega(t)} [\frac{\partial}{\partial t}(w_i m(u)) + \nabla \cdot (w_i m(u)\dot{\mathbf{x}})]\, d\Omega\ , \tag{7}$$

(where $\dot{\mathbf{x}}$ is the velocity of the mesh) and assuming that the piecewise linear test functions evolve with the mesh, that is

$$\frac{\partial w_i}{\partial t} + \dot{\mathbf{x}} \cdot \nabla w_i = 0\ , \tag{8}$$

one obtains the expression

$$\begin{aligned} c_i\dot{\theta}(t) \quad &+ \quad \int_{\Omega(t)} m(u)\nabla w_i \cdot \dot{\mathbf{x}}\, d\Omega = \\ \int_{\Omega(t)} w_i m'(u) Lu\, d\Omega \quad &+ \quad \int_{\partial\Omega(t)} w_i m(u)\, \dot{\mathbf{x}} \cdot \hat{\mathbf{n}}\, dS,\ i = 1, 2 \ldots N. \end{aligned} \tag{9}$$

Here $\hat{\mathbf{n}}$ is the outward pointing, unit-length normal at any point on the surface of Ω. Equation (9) provides us with a means to evaluate the interior mesh velocity $\dot{\mathbf{x}}$, from current values of u and $\mathbf{x}$, and the normal component of $\dot{\mathbf{x}}$ on the boundary.

The value of the normal component of $\dot{\mathbf{x}}$ on the boundary, $\partial\Omega(t)$, may be determined in one of a number of possible ways. For example it may be known analytically for certain problems: either because the boundary is stationary or its motion is prescribed by some external condition. Alternatively, this boundary velocity may also be approximated numerically. For the sake of generality it is this latter approach that we adopt in this work, where the boundary motion is determined based upon the method of [2]. Assuming, for the simplicity of this discussion, that $u = 0$ on the boundary and that u is conserved, then selecting $m(u) \equiv u$ in (9) leads to the system

$$\int_{\Omega(t)} u\nabla w_i \cdot \dot{\mathbf{x}}\, d\Omega = \int_{\Omega(t)} w_i Lu\, d\Omega\ ,\quad i = 1, 2 \ldots N. \tag{10}$$

A finite element discretization of (10) allows $\dot{\mathbf{x}}$ to be approximated at all N nodes of the grid, including those on the boundary (see [2] for full details). Let the approximation obtained using $m(u) = u$, be given by $\dot{\boldsymbol{\xi}}$. This can now be used to fully prescribe the right-hand side of (9) by setting $\dot{\mathbf{x}} \cdot \hat{\mathbf{n}} = \dot{\boldsymbol{\xi}} \cdot \hat{\mathbf{n}}$ on the boundary:

$$\begin{aligned} c_i \dot{\theta}(t) \quad &+ \quad \int_{\Omega(t)} m(u) \nabla w_i \cdot \dot{\mathbf{x}} \, d\Omega = \\ \int_{\Omega(t)} w_i m'(u) Lu \, d\Omega \quad &+ \quad \int_{\partial\Omega(t)} w_i m(u) \, \dot{\boldsymbol{\xi}} \cdot \hat{\mathbf{n}} \, dS, \; i = 1, 2 \ldots N. \end{aligned} \tag{11}$$

This is a necessary additional step for general monitors $m(u)$ which, on their own, do not correctly predict the boundary movement in problems which do not have an explicit Stefan-type boundary condition. The monitor $m(u) = u$ is used to approximate the boundary movement in all of the examples presented in this paper.

The system (11) may be used to obtain a piecewise linear approximation to $\dot{\mathbf{x}}$. In one space dimension this may be achieved by solving directly for a finite element representation of $\dot{\mathbf{x}}$. However, in higher dimensions an intermediate step is required, as explained in [2], in order to ensure a unique solution. This intermediate step assumes that $\dot{\mathbf{x}} = \nabla\phi$ for some scalar potential function ϕ. Hence a piecewise linear approximation to ϕ is sought from

$$\begin{aligned} c_i \dot{\theta}(t) \quad &+ \quad \int_{\Omega(t)} m(u) \nabla w_i \cdot \nabla \phi \, d\Omega = \\ \int_{\Omega(t)} w_i m'(u) Lu \, d\Omega \quad &+ \quad \int_{\partial\Omega(t)} w_i m(u) \, \dot{\boldsymbol{\xi}} \cdot \hat{\mathbf{n}} \, dS, \; i = 1, 2 \ldots N. \end{aligned} \tag{12}$$

This system is augmented by an additional equation, found by differentiating (4) with respect to time and applying the Reynolds' Transport Theorem to

obtain

$$\dot{\theta}(t) \;=\; \int_{\Omega(t)} m'(u)\, Lu \; d\Omega + \int_{\partial\Omega(t)} m(u)\, \dot{\boldsymbol{\xi}} \cdot \hat{\mathbf{n}} \; dS\,. \tag{13}$$

Finally, since (12) is an equation in $\nabla\phi$, $\phi = 0$ is specified at a single, arbitrarily chosen, node in order to allow a unique solution, and the corresponding equation in (12) is ignored. Equations (12) and (13) can then be solved for $\dot{\theta}$ and the remaining nodal values of ϕ. A piecewise linear approximation to $\dot{\mathbf{x}}$ is then recovered in the interior of the domain by undertaking a standard L^2 projection of $\nabla\phi$ into the space of piecewise linears.

Once $\dot{\mathbf{x}}$ is known at an instant, a time-step may be taken and the value of the solution, u, on the new mesh at the end of the step may be recovered. Again, there are multiple options for achieving this and we compare two such options in this paper. The first of these (denoted "ALE") is based upon a standard arbitrary Lagrangian-Eulerian formulation, using the known mesh velocity to update the discretization of the original PDE (1) through the addition of a mesh advection term, *i.e.*

$$\int_{\Omega(t)} w_i\, \dot{u} \; d\Omega \;=\; \int_{\Omega(t)} w_i\, (Lu + \nabla u \cdot \dot{\mathbf{x}}) \; d\Omega,\; i = 1, 2 \ldots N. \tag{14}$$

This leads to nodal values for $\dot{u}$ which can then be used to update u at the same time as the nodal positions. Constant Dirichlet boundary conditions are specified by fixing $\dot{u} = 0$ on the boundary and ignoring the corresponding equations in (14). The second approach that we consider (denoted "Recovery") is a direct analogy of the method used in [2], where we take a time-step to update the mesh positions and then solve the nonlinear system (5), which is regarded as a set of algebraic equations for the nodal values of u (with the c_i constant). In this latter case we update θ and $\mathbf{x}$ (and so $\Omega(t)$) from our cal-

culated $\dot{\theta}(t)$ and $\dot{\mathbf{x}}$, using a standard time-stepping algorithm (forward Euler in this case), and solve (5) for u, with a Newton-Krylov method [3]. Dirichlet boundary conditions are imposed strongly by fixing u on the boundary and ignoring the corresponding equations in (5).

In summary, a single time-step for a general monitor function consists of the following stages:

1. Given mesh node positions and the distribution of the monitor $m(u) = u$ over these nodes, solve equation (10) in the manner described in [2] to find $\dot{\boldsymbol{\xi}}$.
2. Given the same mesh node positions and the distribution of the monitor chosen to govern the internal mesh movement, as in equation (5), solve equation (12) to find ϕ and $\dot{\theta}$.
3. Use a standard L^2 projection of $\nabla\phi$ to recover $\dot{\mathbf{x}}$.
4. Do one of the following:
   (a) Use equation (14) to find $\dot{u}$, then update the mesh node positions using $\dot{\mathbf{x}}$ and the nodal values of u using $\dot{u}$.
   (b) Update the mesh node positions using $\dot{\mathbf{x}}$ and the value of θ using $\dot{\theta}$, then recover the nodal values of u directly from equation (5).

2. Numerical Results

For this short paper we present numerical results for a single test problem, namely the porous medium equation (PME). However we do present results in both one and two space dimensions and consider different choices for the monitor function $m(u)$. The features of this equation which make it suitable for the tests that we undertake here include a moving boundary,

whose evolution must be determined as part of the solution procedure, and the existence of a family of known similarity solutions in the presence of radial symmetry.

In d space dimensions the PME is written as

$$\frac{\partial u}{\partial t} = \nabla \cdot (u^n \nabla u) \;\; (x \in \Omega(t), t > 0), n \in \mathbb{Z}^+; \; u \Big|_{t=0} = u_0(x); \; u \Big|_{\partial\Omega(t)} = 0 \; . \tag{15}$$

As indicated above, for suitable initial data this problem has a known solution of the form [4]:

$$u(r,t) = \begin{cases} \frac{1}{\lambda^d(t)}(1 - (\frac{r}{r_0\lambda(t)})^2)^{1/n} & \text{if } |r| \le r_0\lambda(t) \\ 0 & \text{if } |r| > r_0\lambda(t) \; , \end{cases} \tag{16}$$

where d is the space dimension, r the usual radial co-ordinate and

$$\lambda(t) = \left(\frac{t}{t_0}\right)^{\frac{1}{2+dn}} \qquad \text{with} \qquad t_0 = \frac{{r_0}^2 n}{2(2+dn)}. \tag{17}$$

Hence we are able to compare our computations against this known solution in order to obtain an empirical assessment of its accuracy. Furthermore, note that when $n > 1$ the similarity solution has an infinite gradient at the boundary, which makes the problem particularly challenging numerically. Indeed, since the above similarity solution is known to be a global attractor, even if the initial data does not have this property, the gradient of the solution grows unboundedly at the boundary when $n > 1$. In such situations it is often beneficial to construct an initial mesh with higher resolution near to the boundary. This is achieved in Section 2.2 by including a pre-processing step which adjusts the initial mesh to equidistribute the chosen monitor, *i.e.* arc-length.

2.1. An area monitor function

In [2] the monitor function $m(u) = u$ is used throughout. We refer to this as the "mass" monitor through the analogy of u with a physical density. Consequently, if the mass is equidistributed for the representation of the initial data on the initial mesh then the method will seek to maintain this equidistribution for all time. A simple alternative is to begin with a mesh in which each element has an approximately equal area, and then to seek to evolve the mesh with the solution in such a manner as to maintain approximately equal areas for all elements. As the total area of $\Omega(t)$ changes the area of each element will change but the relative area of each pair of elements should remain approximately constant. The relative areas may not be maintained exactly because, even though the mesh velocity potentials are derived under the assumption that the c_i remain constant, neither the subsequent recovery of the dependent variable values via an ALE approach, nor any other part of the algorithm, imposes any such constraint. As a result, although the mesh movement at a given instant in time is calculated in a manner designed to retain the monitor distribution, the recovery of the dependent variable on the moving mesh is governed by the behaviour of the PDE in the resulting Lagrangian frame, which does not locally preserve the monitor.

The natural monitor to use in equation (5), in order to preserve area, is simply $m(u) = k$ for some constant k. The fact that we have already approximated the values of the boundary velocities $\dot{\boldsymbol{\xi}}$ means that equation (12) can be used directly with this monitor, and the fact that the term involving the influence of the PDE (via Lu) disappears does not matter.

We have tested this approach for various values of k, and for the monitor $m(u) = u + k$ for large values of k (10^4, 10^6) and the results are virtually identical. The choice for the value of k has little effect on the condition number of the system and hence the speed of the algorithm. All of the results presented in this section use the ALE form of the method in order to recover the solution values once the mesh velocity has been determined.

Figure 1 shows an initial mesh and solution using the similarity solution for $n = 3$, and the meshes after $t = t_0 + T$, $T = 0.1$ are shown in Figure 2 for both the mass monitor and the area monitor. In the special case of the similarity solution the mass monitor also approximately preserves area. Convergence rates based upon the area monitor, when comparing to the exact solutions for $n = 1$ and $n = 3$, are shown in Figure 3. We can clearly see a convergence rate that is close to order 2 for $n = 1$ and to order 1 for $n = 3$. The reduced order in the latter case being due to the infinite normal slope at the boundary. This is consistent with the behaviour of the mass monitor, as seen in [2].

Finally in this subsection we consider non-similarity solutions for which $n = 3$ but the initial solution has a finite normal component of the gradient at the boundary (obtained using the same mesh as in Figure 1 but with u_0 given by (16) with $n = 1$ and $t = t_0$). Figure 4 shows one quadrant of the solution meshes at $T = 25$ for the mass and area monitors. The rest of the mesh shows complete four-fold symmetry in both cases. With the mass monitor we can see area compression near the boundary as the solution steepens there, whilst for the area monitor we see the initial equal area distribution of the elements being preserved, as directed by the choice of monitor function.

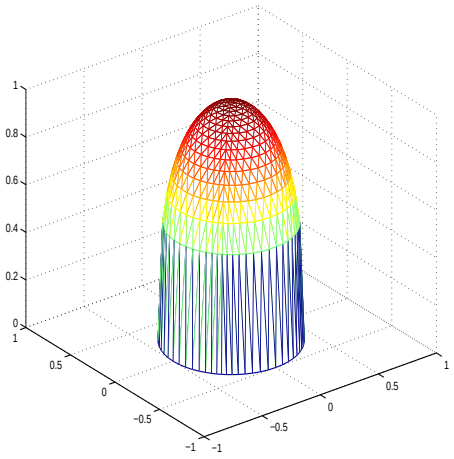

Figure 1: ALE/PME/2D: initial mesh profile, 545 nodes, $n = 3$.

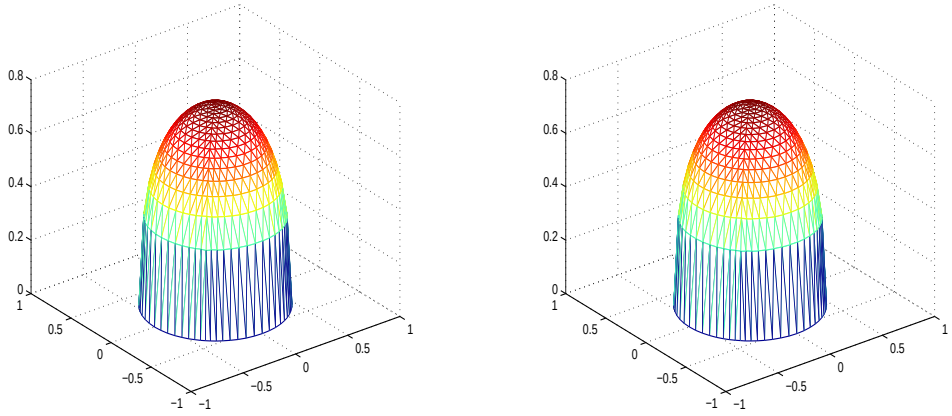

Figure 2: ALE/PME/2D: mesh profiles at $T = 0.1$ for mass monitor (left) and area monitor (right), both with $n = 3$.

Figure 5 shows the initial conditions for a test case which does not exhibit radial symmetry. Instead two initial peaks merge as they evolve and the solution tends towards a radially symmetric similarity solution. The solution

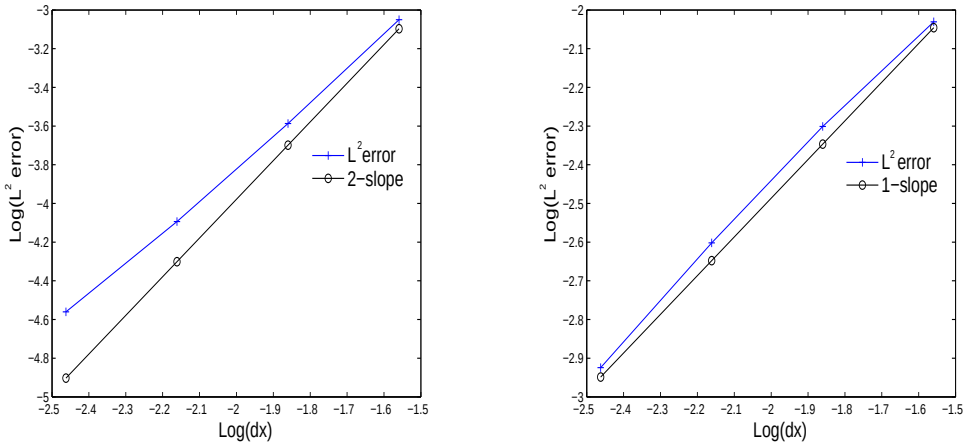

Figure 3: ALE/PME/2D: Convergence rates for area monitor at $T = 0.1$, for $n = 1$ (left) and $n = 3$ (right).

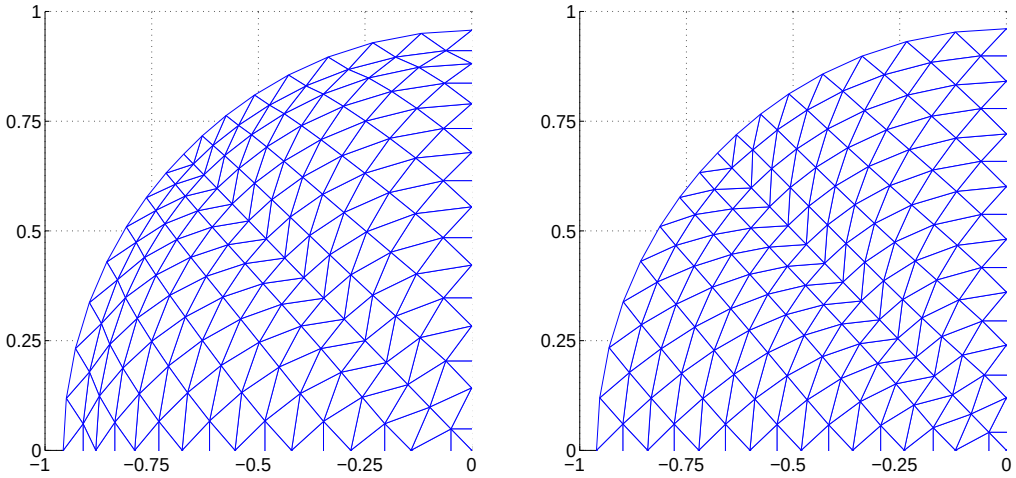

Figure 4: ALE/PME/2D: zoomed mesh planviews at $T = 25.0$ for mass monitor (left) and area monitor (right).

Initial conditions derived using $n = 1$ but evolution governed by $n = 3$.

profile and the mesh for $n = 2$ at $T = 1.0$ are shown for the mass and area monitors in Figures 6 and 7. In this case the area monitor clearly

preserves the shape of the initial mesh better than the mass monitor. This is particularly obvious at the centre of the domain where the cells are twisted by the mass monitor.

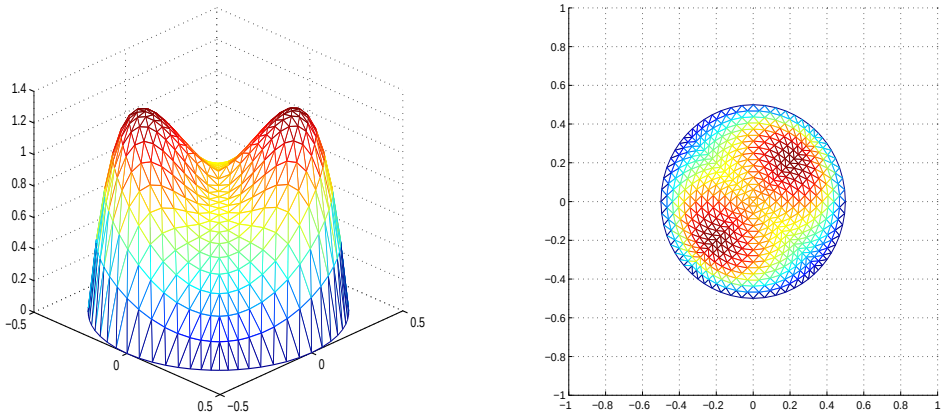

Figure 5: ALE/PME/2D: Non-radially-symmetric initial conditions with two peaks.

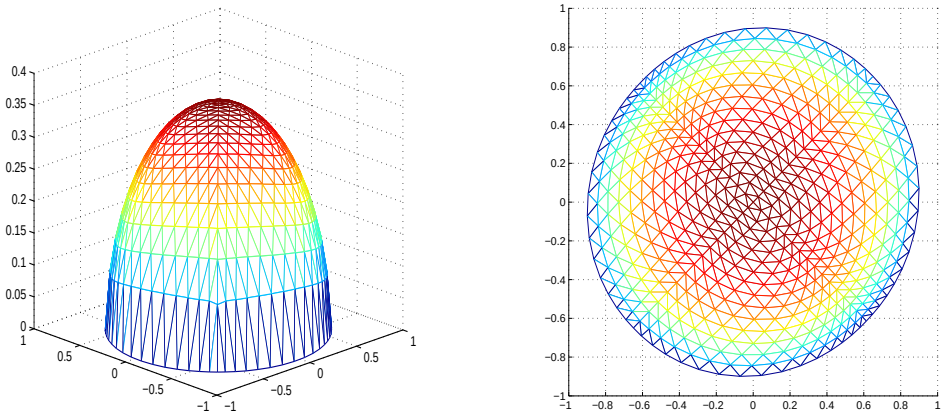

Figure 6: ALE/PME/mass monitor/2D: Snapshots of solution (left) and mesh (right) at $T = 1.0$ for the evolution of twin-peak initial conditions with $n = 2$.

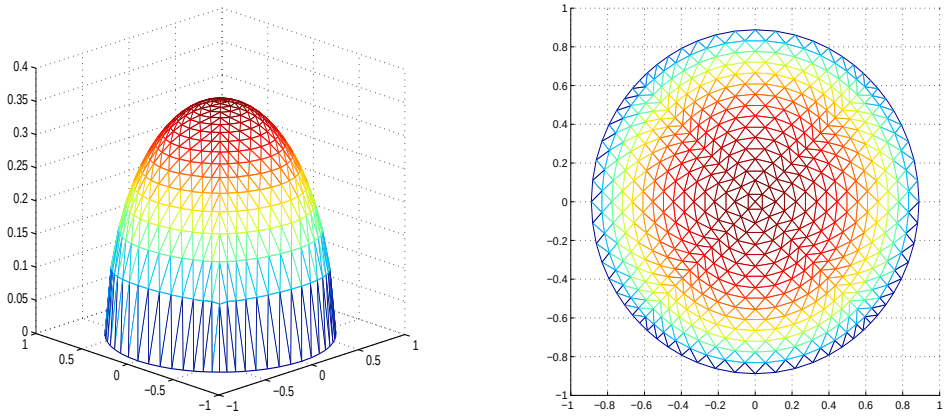

Figure 7: ALE/PME/area monitor/2D: Snapshots of solution (left) and mesh (right) at $T = 1.0$ for the evolution of twin-peak initial conditions with $n = 2$.

In addition to this, Figures 8 and 9 show the evolution of an initial profile in which the circular boundary has been perturbed in a sinusoidal manner to give concavities in the boundary. The results using the area monitor are shown: when the mass monitor was used the mesh tangled before this time was reached.

2.2. An arc-length monitor function

In this section we consider the monitor function $m = \sqrt{1 + u_x^2}$ to drive our moving mesh method for the solution of the PME in one space dimension. Note that this monitor is no longer a function of u but of $\frac{\partial u}{\partial x}$ and so the derivation of the moving mesh equations must be updated slightly. We begin by presenting this update for the one-dimensional case, however it should be noted that the generalization to multidimensions, where $m = m(\nabla u)$, is straightforward.

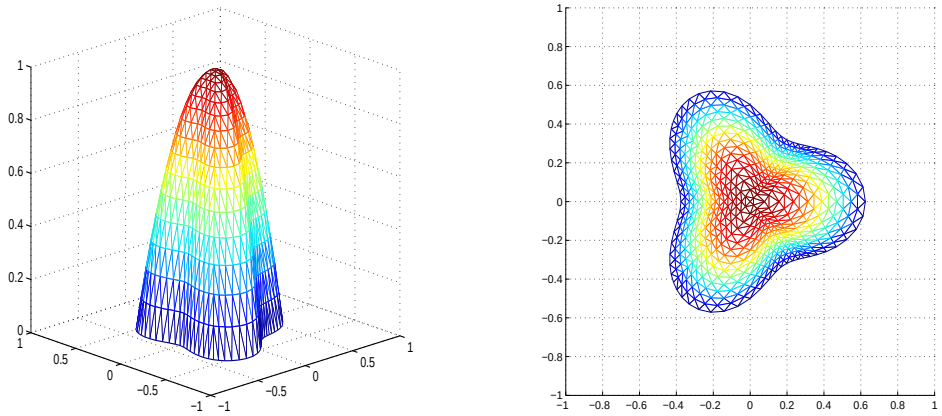

Figure 8: ALE/PME/2D: Non-radially-symmetric initial conditions with concave boundary.

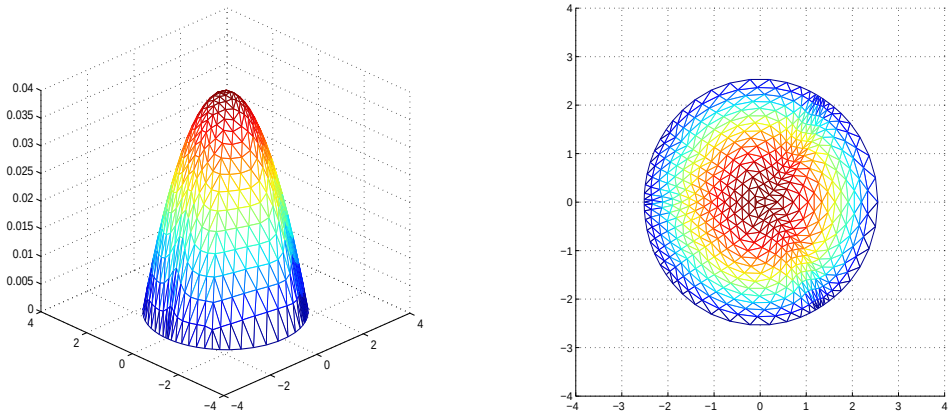

Figure 9: ALE/PME/area monitor/2D: Snapshots of solution (left) and mesh (right) at $T = 20.0$ for the evolution of concave-boundary initial conditions with $n = 1$.

First observe that in one dimension the domain $\Omega(t)$ becomes a moving interval $[a(t), b(t)]$ and so the finite element system for the mesh velocity

potential, (12), becomes

$$c_i\dot{\theta}(t) \; + \int_{a(t)}^{b(t)} m(u)\frac{\partial w_i}{\partial x}\frac{\partial \phi}{\partial x}\, dx \; = \int_{a(t)}^{b(t)} w_i m'(u) Lu \, dx \; + \; \Big[w_i m(u)\dot{\xi} \Big]_{a(t)}^{b(t)} \; , \tag{18}$$

for $i = 1, 2 \ldots N$. Noting that when $m(u)$ is replaced by $m(\frac{\partial u}{\partial x})$ we obtain

$$\frac{\partial m(\frac{\partial u}{\partial x})}{\partial t} \; = \; m'\left(\frac{\partial u}{\partial x}\right)\frac{\partial}{\partial t}\left(\frac{\partial u}{\partial x}\right) \; = \; m'\left(\frac{\partial u}{\partial x}\right)\frac{\partial}{\partial x}\left(\frac{\partial u}{\partial t}\right) \; = \; m'\left(\frac{\partial u}{\partial x}\right)\frac{\partial}{\partial x}(Lu) \; , \tag{19}$$

and writing $v = \frac{\partial u}{\partial x}$, equation (18) becomes

$$c_i\dot{\theta}(t) \; + \int_{a(t)}^{b(t)} m(v)\frac{\partial w_i}{\partial x}\frac{\partial \phi}{\partial x}\, dx \; = \; \int_{a(t)}^{b(t)} w_i m'(v)\frac{\partial}{\partial x} Lu \, dx \; + \; \Big[w_i m(v)\dot{\xi} \Big]_{a(t)}^{b(t)} \; , \tag{20}$$

for $i = 1, 2 \ldots N$.

Note that the first term on the right-hand side of (20) now involves a third derivative of u. When u is represented by a C^0 finite element approximation this may be dealt with by approximating Lu by a recovered piecewise linear function, q say. Let

$$q(x) = \sum_{j=1}^{N} q_j w_j \; , \tag{21}$$

then, following [5], a weak form of $q = Lu$ is given by

$$\int_{a(t)}^{b(t)} w_i q \, dx = \int_{a(t)}^{b(t)} w_i \, Lu \, dx, \quad i = 1, 2 \ldots N. \tag{22}$$

For the PME this becomes

$$\int_{a(t)}^{b(t)} w_i q \, dx = \int_{a(t)}^{b(t)} w_i \frac{\partial}{\partial x}(u^n \frac{\partial u}{\partial x}) \, dx, \quad i = 1, 2 \ldots N, \tag{23}$$

and integrating by parts gives:

$$\int_{a(t)}^{b(t)} w_i q \, dx = - \int_{a(t)}^{b(t)} \frac{\partial w_i}{\partial x}(u^n \frac{\partial u}{\partial x}) \, dx, \quad i = 1, 2 \ldots N, \tag{24}$$

since $u = 0$ on the boundary. Hence a simple mass-matrix system must be solved to recover $q \approx Lu$ in (20).

Figure 10 shows typical results obtained using the ALE formulation with $n = 1$ and $N = 21$. The $T = 0.0$ profile illustrates the initial solution and the mesh that has been used (for which the arc-length is equally distributed), whilst the $T = 1.0$ profile shows the computed solution and mesh at $t = t_0 + 1.0$ along with the exact solution at this time. Figure 11 (left) shows that the convergence rate for this method appears to be second order (as we would hope for $n = 1$), whilst Figure 11 (right) shows how the values of three of the c_i "constants" vary with time (the one at the centre, one at the boundary and one midway between). Recall that, as with the area monitor, although the desire to keep each c_i constant is used to drive the mesh evolution this constraint is never actually enforced in the ALE form of the method. Indeed, we see that the values of c_i for the points at the boundary (which are initially half that for each of the other points) are far from constant in these runs. This is also evident from inspection of Figure 10.

An alternative method (to the ALE approach) for recovering the solution values u, once the mesh velocity is known, is to update the mesh position over a time-step and then to recover values of u at the nodes so as to force the c_i values to remain unchanged. This effectively forces the distribution of the monitor function to remain static for all time and may be achieved by solving (5) for u. This is a highly nonlinear algebraic system of equations and is solved using a Newton-Krylov algorithm [3].

Figure 12 shows typical results obtained using this Recovery approach,

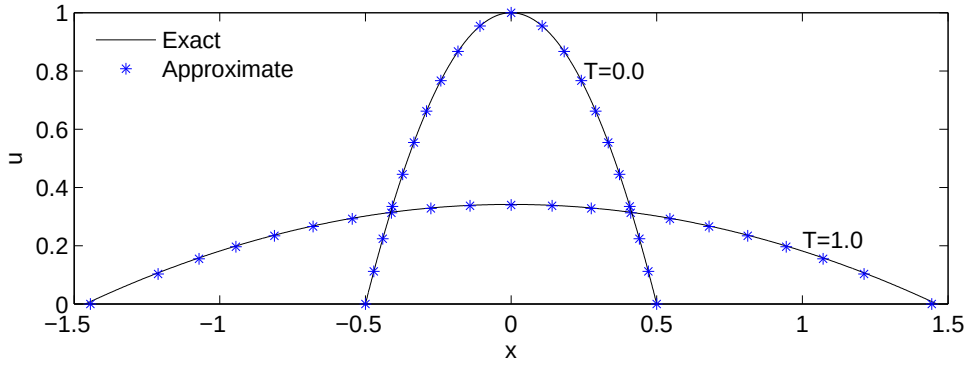

Figure 10: ALE/PME/arc-length monitor/1D: Grid evolution for an initial mesh which equidistributes arc-length, 21 nodes. Graphs compare the approximate solution at the mesh nodes with the exact solution at $T = 0.0$ and $T = 1.0$ for $n = 1$.

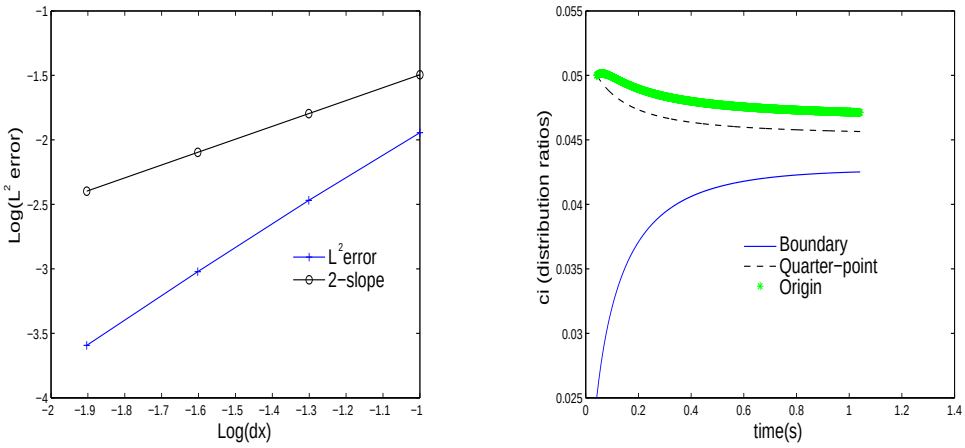

Figure 11: ALE/PME/arc-length monitor/1D: Convergence of the method at $T = 1.0$ for $n = 1$ (left), and evolution of the monitor distribution on the 21 node mesh which initially equidistributed arc-length (right).

again based upon $n = 1$ and $N = 21$. The $T = 0.0$ profile illustrates the same initial mesh and solution as in Figure 10, whilst the $T = 1.0$ profile shows the new computed mesh and solution at $t = t_0 + 1.0$ along with the corresponding

exact solution. Similarly, Figure 13 (left) shows the convergence behaviour for this version of the method whilst Figure 13 (right) demonstrates that the constants c_i do indeed remain unchanged for all time. Any small discrepancies are due to the fact that the c_i are node-based quantities, so retaining the initial values of the c_i doesn't necessarily mean retaining the precise initial distribution of arc-lengths across the cells.

Clearly, the strong imposition of the constraint that the distribution of the arc-length monitor function can never change has had an adverse effect on the accuracy of the solver, reducing it to just first order. This may be due to the piecewise constant nature of the discrete arc-length monitor which is now used directly in the recovery of the dependent variable. In contrast, the ALE approach updates u using a discrete form for the PDE on the moving mesh which is second order accurate whatever velocities are given to the internal mesh nodes. The difference is clearly visible when comparing Figures 10 and 12, and in Figure 14, which illustrates how the positions of the mesh nodes evolve over time for both approaches. Note also that, since it requires the inversion of a nonlinear system at each time-step, direct recovery is substantially more expensive per time-step than the ALE approach. The precise ratio depends on the number of mesh nodes, the tolerance used in the nonlinear solver, and the initial guess (taken here to be the values predicted by the ALE update).

3. Discussion

In this paper we have introduced a new moving mesh finite element method which generalizes earlier work of [2] by allowing the interior of the

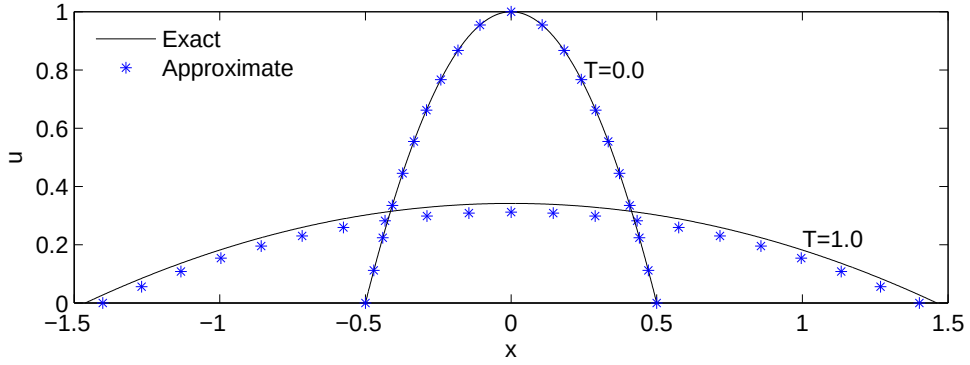

Figure 12: Recovery/PME/arc-length monitor/1D: Grid evolution for an initial mesh which equidistributes arc-length, 21 nodes. Graphs compare the approximate solution at the mesh nodes with the exact solution at $T = 0.0$ and $T = 1.0$ for $n = 1$.

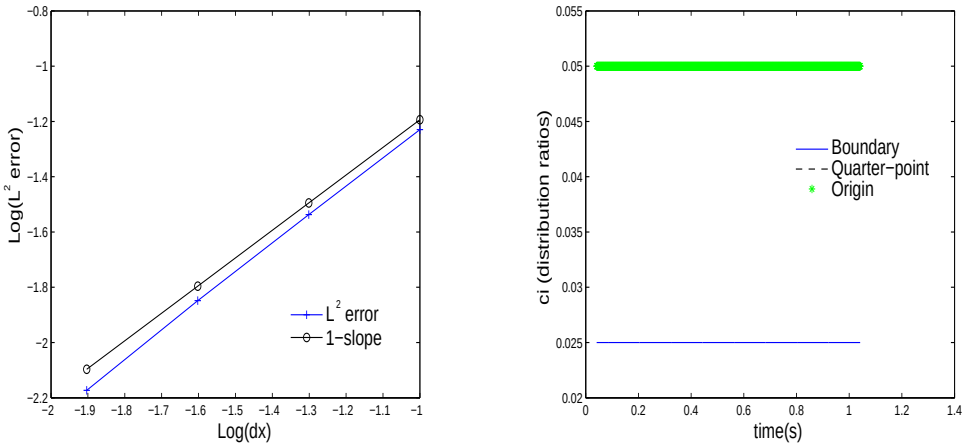

Figure 13: Recovery/PME/arc-length monitor/1D: Convergence of the method at $T = 1.0$ for $n = 1$ (left), and evolution of the monitor distribution on the 21 node mesh which initially equidistributed arc-length (right).

mesh to evolve based upon maintaining the distribution of an arbitrary monitor function $m(u)$ or $m(\nabla u)$. Note that whilst we may drive the interior mesh velocity with whichever monitor that we may wish to preserve, the mo-

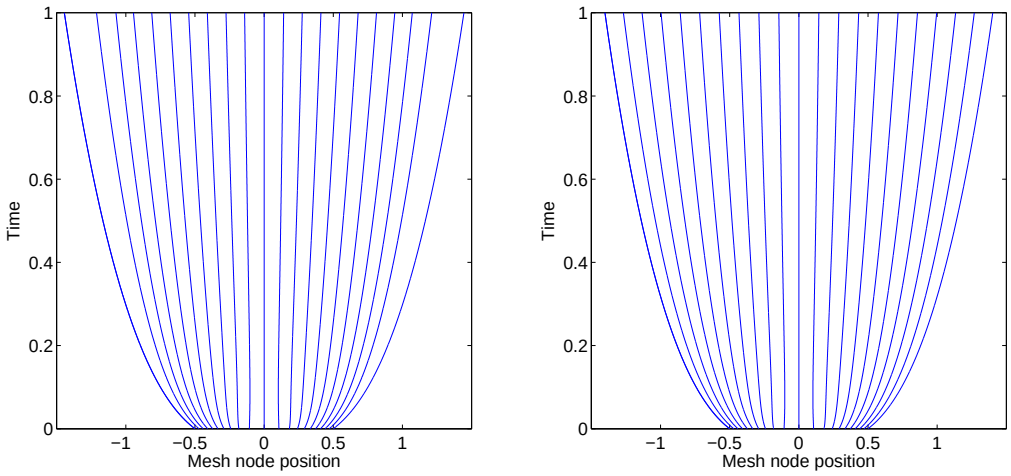

Figure 14: PME/arc-length monitor/1D: Evolution of the mesh node positions for $n = 1$ on the 21 node mesh using ALE (left) and direct recovery (right).

tion of the mesh points that lie on the domain boundary must be driven by physical constraints, such as known boundary velocities, conservation laws, etc.

Having derived the generalized version of the moving mesh algorithm it has then been tested for two monitor functions which have not previously been used with this approach: based upon the area of the finite elements and the arc-length of the solution on each element respectively. In both cases the ALE formulation of the method is shown to yield second order convergence for the porous medium equation with $n = 1$. As expected, a lower rate of convergence is observed for $n > 1$ due to the unbounded normal derivative of the solution at the moving boundary. In the latter case, where the arc-length monitor is used, the formulation of the method required a minor extension to permit the recovery of a second derivative of the solution as an intermediate step. In this case, a comparison of two different variants of the algorithm for

recovering the solution values u, once the mesh has been updated, showed that the ALE approach is advantageous even though this does not strongly enforce the preservation of the initial distribution of the monitor function.

Further work is required to improve the efficiency of these schemes, since the current explicit approach requires the time-step to be proportional to the square of the mesh size for stability, and to consider the application of these, and other, monitor functions for a wider range of PDEs in both one and two dimensions. In particular, we wish to consider blow-up problems for which the solution is known to grow unboundedly at certain points in the spatial domain in finite time, e.g. [6]. The underlying approach has already been applied with the mass monitor to a range of other multidimensional problems, including an oxygen absorption-diffusion problem (which includes a sink term) and phase change problems (which include an explicit Stefan condition on the moving boundary) [2].

Acknowledgements

This work was funded by EPSRC: grant number EP/P502578/1. We would like to thank Professor Mike Baines for his valuable suggestions throughout the course of this work.

References

[1] Budd, C. J. and Huang, W. and Russell, R. D., Adaptivity with moving grids, Acta Numerica (2009) 1–131.

[2] Baines, M. J., Hubbard, M. E. and Jimack, P. K., A moving mesh finite element algorithm for the adaptive solution of time-dependent partial

differential equations with moving boundaries, Applied Numerical Mathematics 54 (2005) 450–469.

[3] Brown, P. N., A local convergence theory for combined inexact-Newton/finite-difference projection methods, SIAM Journal on Numerical Analysis 24 (1987) 407–434.

[4] Murray, J. D., Mathematical Biology, Springer Verlag, 2003.

[5] Baines, M. J., Hubbard, M. E., Jimack, P. K. and Jones, A. C., Scale-invariant moving finite elements for nonlinear partial differential equations in two dimensions, Applied Numerical Mathematics 56 (2006) 230–252.

[6] Budd, C. J. and Williams, J. F., Parabolic Monge-Ampère methods for blow-up problems in several spatial dimensions, Journal of Physics A (2006) 5425–5444.